Crowdsourcing lexical semantic judgements from bilingual dictionary users

A thesis presented by

Richard James Fothergill

http://orcid.org/0000-0002-8140-2743

for the degree of
Doctor of Philosophy

in
November 2017

to
the School of Computing and Information Systems
in total fulfillment of the requirements of the degree

The University of Melbourne
Melbourne, Australia
Crowdsourcing lexical semantic judgements from bilingual dictionary users

Abstract

Words can take on many meanings, and collecting and identifying example usages representative of the full variety of meanings words can take is a bottleneck to the study of lexical semantics using statistical approaches. To perform supervised word sense disambiguation (WSD), or to evaluate knowledge-based methods, a corpus of texts annotated with senses from a dictionary may be constructed by paid experts. However, the cost usually prohibits more than a small sample of words and senses being represented in the corpus. Crowdsourcing methods promise to acquire data more cheaply, albeit with a greater challenge for quality control. Most crowdsourcing to date has incentivised participation in the form of a payment or by gamification of the resource construction task. However, with paid crowdsourcing the cost of human labour scales linearly with the output size, and while game playing volunteers may be free, gamification studies must compete with a multi-billion dollar games industry for players. In this thesis we develop and evaluate resources for computational semantics, working towards a crowdsourcing method that extracts information from naturally occurring human activities.

A number of software products exist for glossing Japanese text with entries from a dictionary for English speaking students. However, the most popular ones have a tendency to either present an overwhelming amount of information containing every sense of every word or else hide too much information and risk removing senses with particular relevance to a specific text. By offering a glossing application with interactive features for exploring word senses, we create an opportunity to crowdsource human judgements about word senses and record human interaction with semantic NLP.
Declaration

This is to certify that:

(i) the thesis comprises only my original work towards the PhD except where indicated in the Preface;

(ii) due acknowledgement has been made in the text to all other material used;

(iii) the thesis is fewer than 100,000 words in length, exclusive of tables, maps, bibliographies and appendices.

Signed: _______________________________ Date: _______________________________

©2017 - Richard James Fothergill

All rights reserved.
Preface

This thesis includes material adapted from several publications and additional new material. Chapter 5 of this thesis has been adapted from the following two publications:


I carried out the research described in these papers, wrote the first draft of each paper and was the primary author, making changes based on consultation with the other listed author.

Parts of Chapter 6 were adapted from Sections 3.2 (Statistics) and 3.3 (Distribution) of the following publication:


I was the author of the adapted sections and carried out the work described therein independently. In particular, I was responsible for the word alignment and sense transfer between English *SemCor* and Japanese *Semcor*. The remaining sections of the paper — particularly the project to translate English *SemCor* into Japanese — is wholly the work of the other authors of the paper.

This work was made possible thanks to the funding provided by the Henry James Williams Scholarship administered by the University of Melbourne Scholarships Office.
Acknowledgements

I owe a great deal of thanks to many people for their roles in my life while I worked on this PhD. The list can never be complete but here I express my heartfelt gratitude to:

- My family — Mum, Dad, Joanne, Linda, Grandma and many more too numerous to list — for being there whenever I needed you but most of all for being understanding whenever I disappeared into my work. This goes for my all-but-family Lorand and Anna as well.

- Michael, Bernie, Marco and the rest of the team at Rome2rio for your support and friendship on the final stretches of this project in particular.

- Ele and Amelia likewise for being great housemates in that time; and before that Ned, Hannah and Bianca.

- Ned, Nikki, Florian and Mel for your solidarity in our long cafe fuelled writing sessions.

- Suganya for keeping me connected to a social life when I needed it the most.

- Many friends, including but not limited to Heather, Rob, Tim, Nyssa, Iain for the energy to keep going that I took from your warm companionship.

Special thanks goes to Etsuko Toyoda for facilitating demonstration of the glossing software essential to this thesis in Japanese classes at Melbourne University, and for your enthusiasm towards integrating technology with the curriculum.

I am deeply indebted to Jim Breen: your years of work on the *Japanese Multilingual Dictionary* project made this thesis possible and your WWWJDIC: Online Japanese Dictionary Service was essential to my study of the Japanese language and a major inspiration for the thesis.

Most of all I would like to thank my supervisor, Tim Baldwin, for his years of dedication beyond the call of duty and for sharing his truly formidable knowledge of the field and the research process.
# Contents

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Title Page</td>
<td>i</td>
</tr>
<tr>
<td>Abstract</td>
<td>ii</td>
</tr>
<tr>
<td>Table of Contents</td>
<td>vi</td>
</tr>
<tr>
<td>List of Tables</td>
<td>x</td>
</tr>
<tr>
<td>List of Figures</td>
<td>xii</td>
</tr>
<tr>
<td><strong>1 Background</strong></td>
<td>2</td>
</tr>
<tr>
<td>1.1 Introduction</td>
<td>3</td>
</tr>
<tr>
<td>1.2 Contributions</td>
<td>6</td>
</tr>
<tr>
<td>1.3 Outline</td>
<td>10</td>
</tr>
<tr>
<td><strong>2 Literature Review</strong></td>
<td>16</td>
</tr>
<tr>
<td>2.1 Computational lexical semantics</td>
<td>17</td>
</tr>
<tr>
<td>2.1.1 Introducing word sense disambiguation</td>
<td>17</td>
</tr>
<tr>
<td>2.1.2 Other approaches to computational lexical semantics</td>
<td>24</td>
</tr>
<tr>
<td>2.1.3 Multiword expressions</td>
<td>26</td>
</tr>
<tr>
<td>2.2 Word sense disambiguation</td>
<td>30</td>
</tr>
<tr>
<td>2.2.1 WSD research community</td>
<td>31</td>
</tr>
<tr>
<td>2.2.2 Classification of WSD research</td>
<td>33</td>
</tr>
<tr>
<td>2.2.3 Evaluation of WSD</td>
<td>36</td>
</tr>
<tr>
<td>2.2.4 Contemporary challenges for WSD research</td>
<td>39</td>
</tr>
<tr>
<td>2.2.5 Recent word sense research</td>
<td>42</td>
</tr>
<tr>
<td>2.2.6 A review of PageRank for word sense disambiguation</td>
<td>44</td>
</tr>
<tr>
<td>2.3 Multiword expressions</td>
<td>56</td>
</tr>
<tr>
<td>2.3.1 Type extraction</td>
<td>56</td>
</tr>
<tr>
<td>2.3.2 Compositionality of multiword expression types</td>
<td>58</td>
</tr>
<tr>
<td>2.3.3 Token identification</td>
<td>60</td>
</tr>
<tr>
<td>2.4 Human computation</td>
<td>67</td>
</tr>
<tr>
<td>2.4.1 Crowdsourcing lexical semantic annotations</td>
<td>69</td>
</tr>
<tr>
<td>2.4.2 Quality control in crowdsourcing</td>
<td>72</td>
</tr>
<tr>
<td>2.5 Summary</td>
<td>79</td>
</tr>
</tbody>
</table>
## Contents

### 3 Resources Review

<table>
<thead>
<tr>
<th>Section</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>3.1</td>
<td>Linguistic concepts useful for text processing</td>
<td>80</td>
</tr>
<tr>
<td>3.1.1</td>
<td>Part of Speech</td>
<td>81</td>
</tr>
<tr>
<td>3.1.2</td>
<td>Morphology</td>
<td>83</td>
</tr>
<tr>
<td>3.1.3</td>
<td>Word order</td>
<td>84</td>
</tr>
<tr>
<td>3.1.4</td>
<td>Dependency grammars and parsing</td>
<td>89</td>
</tr>
<tr>
<td>3.2</td>
<td>Japanese natural language processing</td>
<td>92</td>
</tr>
<tr>
<td>3.2.1</td>
<td>Available tokenisers and parsers</td>
<td>94</td>
</tr>
<tr>
<td>3.2.2</td>
<td>Dependency parsing and the Japanese language</td>
<td>96</td>
</tr>
<tr>
<td>3.3</td>
<td>Princeton WordNet</td>
<td>97</td>
</tr>
<tr>
<td>3.3.1</td>
<td>SemCor</td>
<td>102</td>
</tr>
<tr>
<td>3.3.2</td>
<td>Other WordNets</td>
<td>104</td>
</tr>
<tr>
<td>3.3.3</td>
<td>Translations of SemCor</td>
<td>106</td>
</tr>
<tr>
<td>3.4</td>
<td>Japanese MWE resources</td>
<td>108</td>
</tr>
<tr>
<td>3.4.1</td>
<td>The OpenMWE corpus</td>
<td>108</td>
</tr>
<tr>
<td>3.4.2</td>
<td>JDMWE</td>
<td>109</td>
</tr>
</tbody>
</table>

### II Glossing

<table>
<thead>
<tr>
<th>Section</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>The Wakaran glossing tool</td>
<td>111</td>
</tr>
<tr>
<td>4.1</td>
<td>Introduction to the Wakaran Glossing tool</td>
<td>112</td>
</tr>
<tr>
<td>4.2</td>
<td>Related software</td>
<td>116</td>
</tr>
<tr>
<td>4.2.1</td>
<td>WWWJDIC</td>
<td>117</td>
</tr>
<tr>
<td>4.2.2</td>
<td>Rikai.com and Rikai-chan</td>
<td>117</td>
</tr>
<tr>
<td>4.2.3</td>
<td>Furigana Inserter</td>
<td>119</td>
</tr>
<tr>
<td>4.2.4</td>
<td>The Sharp Intelligent Dictionary</td>
<td>120</td>
</tr>
<tr>
<td>4.3</td>
<td>User base</td>
<td>120</td>
</tr>
<tr>
<td>4.4</td>
<td>Design considerations</td>
<td>121</td>
</tr>
<tr>
<td>4.4.1</td>
<td>Word glossing in computer assisted language learning</td>
<td>122</td>
</tr>
<tr>
<td>4.5</td>
<td>System design</td>
<td>126</td>
</tr>
<tr>
<td>4.5.1</td>
<td>Data model</td>
<td>127</td>
</tr>
<tr>
<td>4.5.2</td>
<td>Architecture</td>
<td>130</td>
</tr>
<tr>
<td>4.5.3</td>
<td>Control flow</td>
<td>130</td>
</tr>
<tr>
<td>4.6</td>
<td>Evaluation</td>
<td>137</td>
</tr>
<tr>
<td>4.6.1</td>
<td>User feedback</td>
<td>137</td>
</tr>
<tr>
<td>4.6.2</td>
<td>Instrumentation</td>
<td>141</td>
</tr>
<tr>
<td>4.6.3</td>
<td>Usage by day</td>
<td>142</td>
</tr>
<tr>
<td>4.6.4</td>
<td>Usage across a session</td>
<td>145</td>
</tr>
<tr>
<td>4.6.5</td>
<td>Clustering usage patterns</td>
<td>149</td>
</tr>
<tr>
<td>4.7</td>
<td>Conclusion</td>
<td>162</td>
</tr>
</tbody>
</table>
## III Disambiguation

### 5 Multiword Expressions

5.1 Flavours of token candidate disambiguation

5.1.1 Qualities

5.1.2 Type specialised disambiguation

5.1.3 Type generalised disambiguation

5.1.4 Cross-type disambiguation

5.1.5 Additional resources

5.1.6 Related work

5.2 Feature extraction

5.2.1 Preprocessing

5.2.2 Idiom characteristic features

5.2.3 Context features

5.2.4 Type features

5.3 OpenMWE corpus experiments

5.3.1 Results

5.3.2 Discussion

5.4 JDMWE lexicon experiments

5.4.1 Results

5.4.2 Another look at WSD features

5.4.3 Discussion

5.5 Conclusion

5.5.1 Implications for Wakaran

### 6 Connecting lexical resources to free text

6.1 Related work

6.2 Re-tokenising text

6.2.1 Token candidate generation

6.2.2 Lexicon mapping

6.2.3 Candidate token disambiguation

6.3 Tokenisation of Japanese SemCor

6.3.1 Lexicon mapping for Japanese

6.3.2 Candidate token disambiguation for Japanese SemCor

6.3.3 WSD for Japanese SemCor

6.3.4 Results

6.3.5 Distribution

6.4 Retokenisation in the Wakaran Glosser

6.4.1 Retokenisation for JMDict

6.4.2 Eliminating over-generation and filtering

6.5 WSD with JWordNet

6.5.1 WSD for Japanese WordNet

6.5.2 Evaluation with JSemcor

6.6 Conclusion

6.A Permutation example for overlap p-value computation
IV Computational semantics and the Wakaran Glosser 270

7 Word Sense Disambiguation in Wakaran 271

7.1 Related work 272

7.1.1 WSD enabled glossing software 273

7.1.2 Crowdsourcing sense labels 279

7.2 Integrating WSD with Wakaran 281

7.2.1 PPR sense embedding for JMDict 281

7.2.2 Disambiguating JMDict words in context 283

7.3 A corpus of Wakaran submissions 287

7.3.1 Exporting and annotating the corpus 287

7.3.2 Near duplicate and subdocument detection 291

7.3.3 Distribution of user selections vs. automatic selections 302

7.4 Evaluation of word sense disambiguation algorithms 304

7.5 Discussion 308

7.5.1 Quality control implications 309

7.6 Conclusion 315

8 Conclusion 317

8.1 Contributions 318

8.2 Future work 321

8.3 Final remarks 327
List of Tables

4.1 Survey responses on the number of times the respondent had used *Wakaran*.................. 138
4.2 A sequence of survey questions rating usefulness of *Wakaran* on a Likert 1 to 5 scale...................... 138
4.3 A sequence of survey questions regarding *Wakaran* allowing selection of multiple responses each................................................................................................................... 140
5.1 Results of combining different features for cross-type classification. “Basic” results restrict the idiom and WSD features to those of Hashimoto and Kawahara (2009); “extended” results include our extensions......................................................................................... 190
5.2 Results for classification of MWE tokens of MWE types seen in the training corpus. As in Table 5.1 “Basic” results restrict the idiom and WSD features to those of Hashimoto and Kawahara (2009); “extended” results include our extensions......................................................................................... 191
5.3 Results of combining different features for type-specialised classifiers. “Basic” results restrict the idiom and WSD features to those of Hashimoto and Kawahara (2009); “extended” results include our extensions......................................................................................... 192
6.1 Corpus Size........................................................................................................................................ 240
6.2 Part of speech distribution for tokens without word alignment.............................................................................. 241
6.3 Performance of Personalised PageRank over JWordNet under several configurations and evaluation metrics. The best performing configuration under each metric is indicated in **bold**. The difference between any scores joined by snaking lines was not found to be statistically significant (by a t-test or paired t-test, LKB coverage permitting). All other differences in scores are statistically significant (*p* ≪ 0.05).................................................................................................................................................. 254
6.4 Summary of tokens for which WSD\textsubscript{DOC}\textsubscript{WN++} gives a better annotation than WSD\textsubscript{DOC}\textsubscript{BBN}.................................................................................................................................................. 258
6.5 Distinctive words in documents where WSD\textsubscript{DOC}\textsubscript{BBN} performs worse than WSD\textsubscript{DOC}\textsubscript{WN++}. In all cases words are ranked by *tf-idf*, with term frequencies counted across a single document or a subset of the corpus and document frequencies calculated across the whole corpus......................................................................................................................................... 259
6.6 Superiority overlap for various configuration parameters. Errors introduced by switching from WN++ to BBN are in much less consistent locations than switching from either BBN to WN+gloss or switching from DOC to PARA......................................................................................................................................... 265
7.1 Part of speech counts of the corpus after filtering out short contexts and the COS-WSD subcorpus............................................................................................................................................. 296
<table>
<thead>
<tr>
<th>Table</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>7.2</td>
<td>Part of speech counts at each stage of the deduplication procedure: before duplicate removal, after duplicate removal and after subdocument elimination.</td>
</tr>
<tr>
<td>7.3</td>
<td>Confusion table of senses selected by users while automatic WSD was not operational, and the sense that would have been selected had WSD been enabled at that time.</td>
</tr>
<tr>
<td>7.4</td>
<td>Confusion table of senses selected by users while automatic WSD was operational, and the sense that was automatically selected in the interface by the WSD algorithm.</td>
</tr>
<tr>
<td>7.5</td>
<td>Results for evaluation of all WSD configurations on the section of the Wakaran corpus for which automatic WSD was not enabled.</td>
</tr>
<tr>
<td>7.6</td>
<td>Results for evaluation of all WSD configurations (excluding the first sense baseline) on the section of the Wakaran corpus for which automatic WSD was not enabled.</td>
</tr>
<tr>
<td>7.7</td>
<td>Results for evaluation of all WSD configurations on the section of the Wakaran corpus for which automatic WSD was enabled.</td>
</tr>
</tbody>
</table>
List of Figures


2.1 Comparison of results using an evaluation metric. Result A and result B are worlds apart: one lies near a trivial baseline while the other approaches human performance at the task. However, without reference points they can only be said to differ by around 30%. .................................................. 38

3.1 A simple grammar for a very restricted subset of the English language (Manning and Schütze 1999). .................................................. 87

3.2 Derivation of *The children ate the cake* using the grammar given in Table 3.1 (Manning and Schütze 1999). .................................................. 88

3.3 The parse of the sentence *the children ate the cake* produced by the Stanford Parser [Klein and Manning 2002], generated using [http://nlp.stanford.edu:8080/parser](http://nlp.stanford.edu:8080/parser) ................................. 88

3.4 A tree diagram representing the parse shown in figure 3.3 .......................................................... 89

3.5 Two representations of an example dependency parse for *the rapidly tiring yellow cat ate the mouse*. The first representation emphasises the tree structure of the parse whereas the second captures the word order .......................................................... 90

3.6 Possible dependency parses for the sentence *What did you eat?* .......................................................... 92

3.7 Possible segmentations for the sentence Example (20). Pronunciations, translations and explanatory notes are a guide, not part of the output of the respective morphological analysers ................................. 93

3.8 An example KNP dependency analysis .......................................................... 95


3.10 An abbreviated example ConText file from *SemCor*. .......................................................... 104


4.2 This first view the user gets of a document submission contains only the original text with highlights on text where definitions are available. Text from [Wikipedia (2016)](https://en.wikipedia.org) .......................................................... 113

4.3 Hovering the mouse over a highlighted token shows glosses for all compounds that token takes part in. Text from [Wikipedia (2016)](https://en.wikipedia.org) .......................................................... 114
<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>4.8</td>
<td>UML class diagram of Wakaran data model. This figure shows only the entities and attributes supporting the main use case of user interaction with translational paraphrases.</td>
</tr>
<tr>
<td>4.9</td>
<td>UML component diagram of the system architecture of Wakaran.</td>
</tr>
<tr>
<td>4.10</td>
<td>UML sequence diagram of interactions between the Wakaran HTTP handlers and other components in the period of time between the submission of text and the first display of glossed text content for the user.</td>
</tr>
<tr>
<td>4.11</td>
<td>UML sequence diagram of interactions between the Wakaran HTTP handlers and other components in the period of time between the submission of text and the first display of glossed text content for the user.</td>
</tr>
<tr>
<td>4.12</td>
<td>UML sequence diagram of interactions between the Wakaran batch worker processes and other components in the period of time between the appearance of a new document on the AMQP queue and the persistence of the annotated document to the SQL database.</td>
</tr>
<tr>
<td>4.13</td>
<td>UML sequence diagram of interactions between the user’s web browser and the Wakaran HTTP handlers as the user selects and promotes paraphrases in dictionary entries linked to a submission.</td>
</tr>
<tr>
<td>4.14</td>
<td>Heatmap showing distribution of new document submissions throughout the year.</td>
</tr>
<tr>
<td>4.15</td>
<td>Heatmap showing distribution of new document submissions across the week during semester 1 2013.</td>
</tr>
<tr>
<td>4.16</td>
<td>Plotting document position vs time for interaction logs</td>
</tr>
<tr>
<td>4.17</td>
<td>Plotting position versus time for events logged on a variety of submissions.</td>
</tr>
<tr>
<td>4.18</td>
<td>Reading a document in two passes</td>
</tr>
<tr>
<td>4.19</td>
<td>Heatmap showing the number of documents with a given count for a given feature.</td>
</tr>
<tr>
<td>4.20</td>
<td>Feature distribution and sample cluster members for three clusters based on event counts features. Cluster members are represented by position-time plots using the same key for event type as Figure 4.18.</td>
</tr>
<tr>
<td>4.21</td>
<td>Comparing timeseries with the corresponding autocorrelation plots.</td>
</tr>
<tr>
<td>4.22</td>
<td>Heatmap showing the number of documents with a given count for a given autocorrelation feature.</td>
</tr>
<tr>
<td>4.23</td>
<td>Feature distribution and sample cluster members for three clusters based on autocorrelation features. Cluster members are represented by position-time plots using the same key for event type as Figure 4.18.</td>
</tr>
<tr>
<td>5.1</td>
<td>Illustration of cross validation partitioning strategies used to evaluate different classification schemes</td>
</tr>
<tr>
<td>Figure</td>
<td>Description</td>
</tr>
<tr>
<td>--------</td>
<td>--------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>5.2</td>
<td>English summary of the select parts of the output of <em>KNP</em> when run on the sentence in Example (23). Pronunciation and translation guides have been added for the reader and are not part of the output.</td>
</tr>
<tr>
<td>5.3</td>
<td>A sample of the WSD inspired features extracted for Example (25).</td>
</tr>
<tr>
<td>5.4</td>
<td>Accuracies for type-specialised classifiers with capped training set sizes.</td>
</tr>
<tr>
<td>5.5</td>
<td>Cross-type classification accuracy using <em>JDMWE</em> type-level features and lexical type-level features in combination with various token-level features</td>
</tr>
<tr>
<td>5.6</td>
<td>Looking separately at recall of idiomatic and literal instances in the corpus under various combinations of feature families. Classifiers with no features at all are the type specialised majority class baseline which is literal for some types and idiomatic for others.</td>
</tr>
<tr>
<td>6.1</td>
<td>Illustration of stages in <em>RETOKENISE</em> execution on an example from the start of sentence 22 in <em>Semcor</em> document br-f03.</td>
</tr>
<tr>
<td>6.2</td>
<td>Interface used by translators to develop Japanese translation of <em>SemCor</em> text (Bond et al. (2012) Figure 2, page 3).</td>
</tr>
<tr>
<td>6.3</td>
<td>Sample KAF record for <em>Scotty ha gakkō ni modora nakatâ</em>. The Japanese translation of English sentence <em>Scotty did not go back to school</em>.</td>
</tr>
<tr>
<td>7.1</td>
<td>Visualisation of one of the larger clusters of contexts in a containment relationship. Token alignment was performed with a recursive longest common substring method.</td>
</tr>
</tbody>
</table>
Chapter 1

Introduction

When we look up a word in a dictionary we often find that the dictionary author — the lexicographer — has laid out several distinct interpretations which are called senses of the word. For instance, here are several senses of the English word device given by the online dictionary Wiktionary (Wiktionary 2017a):

device$_1$ Any piece of equipment made for a particular purpose, especially a mechanical or electrical one.

device$_3$ A project or scheme, often designed to deceive; a stratagem; an artifice.

device$_6$ A motto, emblem, or other mark used to distinguish the bearer from others.

For brevity, we have extracted only the first, third and sixth senses which are the most general, and labelled them device$_1$, device$_3$ and device$_6$ accordingly. Certain senses may be more pertinent than others in a given context of usage of that word. For instance, in Example (1) from the entry for Mobile Phones in the online encyclopedia Wikipedia (Wikipedia 2017) the most relevant sense is device$_1$:

(1) The race to create truly portable telephone devices began after World War II, with developments taking place in many countries.
Chapter 1: Introduction

Whereas in Example (2) taken from a description of the official flags of the Red Cross and Red Crescent movement (CRW Flags 2017) (who do humanitarian work in war zones) is closest to sense device6:

(2) The red crescent or red lion and sun, in lieu of the red cross, will be recognised for use by countries that were already using those devices when the 1949 convention was adopted.

In computer programs that process human language — natural language processing (NLP) software — it can be desirable to narrow down the meaning of a word usage by identifying the most relevant sense to the situation. The act of identifying the meaning of a word with reference to a list of senses is generally called word sense disambiguation (WSD) (Navigli 2009).

There is not always only a single relevant sense. In Example (3) from a description of rules surrounding the use of the cross and crescent emblems (ICRC blog 2017) arguments could be made for the relevance of more than one sense:

(3) Misuse of the emblem as a protective device in time of war jeopardises the entire protective system established by humanitarian law.

This admits several interpretations with varying degrees of plausibility: (a) Misuse of the emblem as a protective artifice (device3) (b) Misuse of the emblem as a protective emblem (device6) and (c) Misuse of the emblem as a protective mechanism (device1, metaphorical). Although it is common for WSD research to focus on identifying a single sense for each usage of a word, there also exist graded variants of WSD aimed at representing degrees of relevance that assign a numeric similarity or probability to every sense for a single usage (Erk and McCarthy 2009).

Other complications arise for assigning senses to words when they appear in fixed phrases. For example, Wiktionary has a separate entry for the phrase plot device (Wiktionary 2017b):

plot_device1 An element introduced into a story, film, etc. to advance its plot.

Consider the Example (4) of device in the introduction to Anonymous (2014):
(4) The Arabian Nights vary greatly and some include elements of haunting, which is a common **plot device** used in gothic fiction and horror fiction.

The expression *plot device* requires a separate explanation in the dictionary because it has a particular meaning and occurs in a preferred form. For instance, in Example (4) we could potentially substitute *plot mechanism* but definitely not the seemingly equivalent *plot machine*. Such distinctive phrases are called **multiword expressions (MWEs)** and may need to be treated as distinct linguistic units in NLP systems (Sag *et al.* 2002; Ramisch 2015) and indeed as distinct words in a vocabulary (Bauer 1983). However, distinguishing them computationally from plain uses of their constituent words remains a challenging problem!

Bilingual dictionaries also commonly divide the meaning of a word into distinct senses, described in the language they are translated into. For instance, consider the following definitions for the Japanese word 仕掛け “shikake” given in the free Japanese English dictionary *JMDict* (Breen 2004):

- **shikake_1**: device; contrivance; mechanism; gadget;
- **shikake_2**: trick; trap; calculated manipulation; gambit;
- **shikake_3**: (small) scale; half finished;
- **shikake_4**: commencement; initiation;
- **shikake_5**: set up; laying out; preparation;
- **shikake_6**: challenge; attack;

We note that *device* is given as a translation of the first sense of *shikake* and that the second sense is very similar to device_3, so these two words appear to have meanings in common. However, the remaining four senses of *shikake* are not senses of the English *device* at all, being concerned with smallness, infancy and aggression.
In this thesis we study computational models of relatedness between dictionary senses and word usages in text. Our work covers monolingual and cross-lingual word sense disambiguation, and identification of usages of MWEs in arbitrary digital texts. Together these topics form a major part of the field of computational lexical semantics (although they are by no means exhaustive of it). A major theme of this thesis is resource usage, particular when it comes to large collections of text — corpus in the singular, corpora in the plural — and sources of word senses such as learner dictionaries and specialised linguistic ontologies. Over the course of the thesis we develop an intelligent dictionary application, the Wakaran Glosser (“Wakaran”), designed for students of Japanese as a second language (L2), which has interactive features for controlling the presentation of word senses. Wakaran is designed with two primary qualities in mind:

1. That usage of its interactive features should produce a data trail by which we can gain insight into human intuitions of the relevance of word senses to text.

2. That users should be motivated to use the application and its interactive features for their own information needs, without a dependency on additional costly motivation factors.

The confirmation of these two qualities constitutes the major findings of this thesis.

### 1.1 Motivation

A common and perhaps indispensable practice in the computational study of language is to compile a reference solution to the problem being tackled — a gold standard corpus — hand labelled by human annotators or by other means. When compared with other sub-fields of computational semantics, lexical semantics and WSD in particular carries a high cost in acquiring resources with sufficient data for statistical analysis. Classification in the field of computational semantics covers a broad spectrum of granularities of abstraction and word senses are one of the most finely divided. At one end of the spectrum lies sentiment analysis, which seeks to classify texts according to whether they convey a positive or negative attitude towards the subject matter. For instance, in
user reviews for a mobile phone one might want to automatically identify the users who liked the phone and the ones who did not. The simplest form of this task has just two classification labels for sentiment: positive and negative. A multiplicity of labels are found in other tasks such as named entity recognition (NER), where individual nouns in text are identified as representing people, locations, organisations and sometimes more such as date and time. In this case the number of labels is small but non-trivial and individual words are labelled, so construction of a gold standard corpus is relatively more costly. By comparison, word sense disambiguation, being faced with one or more senses for every word in the vocabulary, uses a label inventory potentially larger than the lexicon itself. Additionally, nearly every word in a piece of text will have sense labels applicable to it. The challenge of collecting examples of wordform usage that cover a representative variety of situations in which each sense is used is particularly high (Marquez et al. 2007; Artstein and Poesio 2008; Passonneau and Carpenter 2014; Lopez de Lacalle and Agirre 2015a). For example, a straightforward application of supervised machine learning to the task might model the probability distribution of words co-occurring with each distinct sense of a wordform. However, building an accurate model of the distribution would require many textual examples of usages for each sense of each word in a lexicon. The cost is a bottleneck for expansion of many studies beyond a small sample of words from a lexicon and to the realisation of WSD as a component in NLP applications. Some approaches to computational semantics bypass the corpus annotation bottleneck. For instance, unsupervised machine learning techniques can be used to discover patterns in word usage and map them onto existing sense inventories. A family of methods that have access to dictionaries or ontologies — collectively: lexical knowledge bases (LKBs) — are generally called knowledge-based methods. We explore a knowledge-based method, Personalised PageRank (PPR), in this thesis. Such methods have rarely been found to exceed supervised methods in their accuracy, although they tend to have the advantage of being able to handle larger vocabularies. However, evaluation of unsupervised and knowledge-based methods is generally best performed with at least a sample of gold standard data, so the problem of knowledge acquisition cannot be avoided entirely.
Chapter 1: Introduction

WSD has long been held to be AI complete — a necessary solution in the development of a true artificial intelligence — and extrinsic evaluations are long sought after but there are as yet few examples of WSD making a substantial difference to the performance of an application. Crowdsourcing methods promise to acquire data more cheaply than construction of semantically annotated corpora through professional annotation, albeit with a greater challenge for quality control. Most crowdsourcing efforts at gathering word sense annotations to date has incentivised participation in the form of a payment (Snow et al. 2008; Rumshisky et al. 2012; Jurgens 2013; Passonneau and Carpenter 2014) or gamification of the resource construction task (Venhuizen et al. 2013; Seemakurty et al. 2010; Jurgens and Navigli 2014). Crowdsourcing payments can be cheaper than paying professionals but data collection still scales linearly with cost. Turning the task into a game can attract free participants but the study then has to compete with a multi-billion dollar games industry for players. In this thesis we pursue a different approach: crowdsourcing human judgements on lexical semantics from naturally arising human activities.

The aim of this thesis is address the knowledge acquisition bottleneck by designing and evaluating apparatus and methodology for collecting data that addresses core research questions from the point where language students interact with a dictionary in pursuit of information about word meanings. The fulfilment of this aim will enable collection of data at only the cost of public promotion and software delivery, which are inherent costs of any crowdsourcing method. In specifics, the aim is to build a text glossing application which enables extrinsic evaluations of computational lexical semantic theory including WSD. A gloss is an explanatory note attached to a word in a piece of text. The text glossing application implemented for this thesis, Wakaran, is illustrated in Figure 1.1. It accepts submissions of arbitrary Japanese texts and attaches Japanese-English dictionary entries — glosses — to words in the text. The glosses are accessed by hovering the mouse pointer over a word in the text, which causes a pop-up to show containing the gloss(es) for that word. In the illustrated case two glosses are shown: one word matching the single Kanji character the mouse has hovered over and one matching a longer compound the character forms a part of. The first gloss is for the compound. It contains a pronunciation guide in Japanese, しゅとけん.
“shutoken”, and an English translation, *the capital city*. The second gloss has the same information for the shorter constituent. The senses — expressed as English translations — for the full word and its constituent are shown at their lowest visibility setting; the user can increase the visibility of any translation and even choose one translation to insert between the lines of the original text. This is the mechanism by which *Wakaran* is designed to capture user preferences for word senses in context. In this thesis we also explore the potential for automatic customisation of visibility using WSD and, to some extent, MWE token disambiguation as well.

This thesis forms one side of a coin: on the other lies future work which investigates the ability of automatic WSD to improve learning outcomes for users of glossing software. Learning outcomes are not studied in this thesis but the complementary aims do inform its scope and consequently some of the groundwork for future work is laid by the present investigation. As is always the case with crowdsourcing methods using untrained annotators, raw word sense judgements collected from dictionary users are not reliable enough to treat as a gold standard. Since this is a new
methodology for collecting data from naturally occurring human activities rather than a controlled
crowdsourcing task, we do not yet attempt to build a gold standard annotated corpus. We aim instead
to build a corpus annotated with student word sense choices indicated in user interface interactions
but limit this thesis to an investigation of the relationship between user choices and the output of
traditional methods of automated WSD.

This thesis uses the specific language pair and translation direction Japanese-English as
a case study. The results are expected to generalise to other language pairs with differences in
orthography. In addition, we expect the results to generalise to more closely related language pairs,
with the exception that density of data collection from dictionary consultation may decrease for
closely related languages with similar word roots.

A number of text glossing applications — that is, applications that take an arbitrary text
as input and annotate the words in it with dictionary entries — already exist for English speakers
reading Japanese as a second language. However, the most popular ones have a tendency to either
dump an overwhelming amount of information on the user giving descriptions of many word senses
for every word in the text or else hide too much information and risk removing senses with specific
relevance to a particular text. None offer interactive mechanisms for selectively showing and hiding
information on word senses, let alone automating the process of finding the most relevant informa-
tion. An opportunity exists to offer a broad selection of potentially relevant dictionary entries with
optional controls for consulting and selecting specific senses of words and MWEs. A consequence
of interaction with these features would be a trail of information about user judgements of word
senses relevant to their understanding of the text.

1.2 Contributions

The main contribution of this thesis is its novel crowdsourcing technique for addressing
the knowledge acquisition bottleneck for computational semantics. Part of this contribution is a
design for glossing software the Wakaran Glosser (“Wakaran”), which serves the joint purpose of
Chapter 1: Introduction

crowdsourcing human judgements on word senses whilst also providing a platform for delivery of NLP as a first class application feature serving user information needs. The major research questions we seek to answer with this apparatus are:

- whether students will use word sense visibility features in an interactive Japanese-English glosser to satisfy their information needs; and

- whether it is possible to draw sensible conclusions about lexical semantic models based on a record of sense choices made by students in that setting.

Components of this thesis develop theory on computational lexical semantics and contribute to the construction of annotated corpus resources useful for answering core questions in the field. The theoretical work is then applied in *Wakaran*’s ability to identify dictionary entries relevant to arbitrary Japanese texts. Modelling of data collected from user interaction events in *Wakaran* validates hypothesised usage and demonstrates the feasibility of the crowdsourcing method. Finally, an extrinsic evaluation for word sense disambiguation is presented that uses modelling of data collected specifically from user interactions with word senses in the application. Additionally, results are presented that demonstrate that integration of NLP features based on computational semantic theory will have a substantial impact on the information users of *Wakaran* focus their attention on, which has implications for future study of *Wakaran* within the field of computer assisted language learning (CALL), which has been excluded from the scope of this thesis.

The success in the identification of lexicalised multiword expression types by means of lexico-syntactic fixedness has lead to a number of attempts to identify their usages by means of local morpho-syntactic features (Fazly et al. 2009, Nasr et al. 2015, Morimoto et al. 2016). We show that morpho-syntactic variations are much less informative than contextual semantics for identifying usages of a lexicalised expression to the extent that:

1. an expression-agnostic context representation can be more informative than expression-specific syntactic rules; and
2. even expression-specific syntactic constraints encoded by a lexicographer are frequently broken by usages of the expression.

These conclusions have a direct consequence for word glossing software: the choice of whether to identify a phrase with a matching expression entry cannot be made accurately with a computationally inexpensive check against syntax constraints; to reliably make the choice a model of context semantics is needed.

A bottleneck to studying lexical semantics is the high cost of collecting data on the usage of word senses. We contribute a corpus of sense annotated Japanese text by transferring annotations from the English *SemCor* corpus to an existing sentence aligned Japanese translation by using word-type alignment suggestions provided by translators. We use the resulting corpus, *Japanese Semcor*, to test the effect that expanding a lexical knowledge base by orders of magnitude has on the results of the knowledge based method for word sense disambiguation, *Personalised PageRank*. We show initially that *WordNet++*, which expands *WordNet* with additional relations found in *Wikipedia*, substantially increases WSD performance compared to *WordNet* alone. However, we find that for Japanese WSD over *Japanese Semcor*, the *BabelNet* knowledge base, which expands *WordNet++* with concepts and relations from multilingual links in *Wikipedia*, results if anything in a decrease in performance relative to *WordNet++*. After a detailed error analysis we conclude that to leverage the information in *BabelNet for Personalised PageRank* it will be necessary to weight the relations in the network by quality and by kind.

Crowdsourcing has emerged as a popular new methodology for collecting data at scales that would be prohibitively resource intensive using professionally trained annotators. However, it introduces challenges for incentivising participation economically and for control over how participants engage with the crowdsourced task. In this thesis we describe the design of *Wakaran*, a dictionary-enabled web application, to fulfil an existing demand in second language learning. *Wakaran* captures and records human judgements about the association between dictionary entries and usages of words in text. We show through study of user interface interaction logs that *Wakaran* attracted and retained a base of users who engaged frequently with dictionary definitions while
reading systematically through Japanese texts.

The texts that users of Wakaran interacted with were user-selected excerpts from documents studied in intermediate tertiary studies in Japanese as a second language. Wakaran was used by many students taking the same unit of study and as such its records include user interaction data for many overlapping fragments of text from the same original source document. We describe a method for:

1. identification of fragments originating from the same original text;
2. cleaning noise introduced by text being excerpted at different times and by different methods; and
3. merging data collected from multiple sets of logs into a set of word sense judgement annotations on the cleaned unified text.

We then describe a method for automatic disambiguation of the same texts via a mapping of the glossing dictionary to the lexical knowledge bases we previously evaluated against Japanese Semcor. Our results show that users strongly focus their interactions on the first sense of the dictionary. However, senses selected using WordNet++ as the LKB are statistically more similar to those selected by users than senses selected using WordNet alone, which agrees with the comparable results of the Japanese Semcor evaluation. Additionally, we show that if our method for automatic sense selection is used to highlight senses in the user interface, user sense preference shifts to split between the first sense and the highlighted sense. We conclude that the quality of automatic sense selection will have a strong effect on the information gained by students using Wakaran. Furthermore our results suggest that there is a relationship between intrinsic measures of WSD performance and application specific performance in Wakaran. This presents a hypothesis for future study: that user sense preferences in Wakaran can inform improvements in automatic WSD performance in gold standard evaluation measures and, conversely, that improvements in intrinsic WSD performance can improve extrinsic measures in the Wakaran application and increase the value that users get from it.
1.3 Outline

This dissertation is divided into four main parts: I background material, II the crowdsourcing application, III computational lexical semantics and, finally, IV the synthesis of the application with lexical semantics, with conclusions. Part I comprises three chapters, including this introduction. Chapter 2 reviews related work in the field, initially situating this thesis within the broader field of computational semantics. The review then takes a more detailed look at related research into word sense disambiguation, multiword expressions and crowdsourcing. Chapter 3 gives an overview of resources used in the course of the research. Resources include dictionaries and publicly available annotated corpora, but also NLP applications that form part of the apparatus for the research presented in this thesis such as tools for identifying grammatical and morphological structure in digital text. Chapter 3 also includes a small amount of bridging linguistic theory necessary to understanding elements of the work that a reader with a general computer science background may not be familiar with.

Part II describes the crowdsourcing application. It comprises a single chapter, Chapter 4, in which we describe the design of the web application, the Wakaran Glosser, for crowdsourcing of human word sense judgements. In contrast to typical crowdsourcing methods, the application is designed to incentivise participation by filling a user need rather than by offering a reward structure. We use data collected from usage of the application to quantify its user acquisition and user behaviour, for the purposes of validating the software and confirming the feasibility of the crowdsourcing method.

Part III establishes theoretical results in lexical semantics relevant to the design of the glossing application, Wakaran, intended for use in the main crowdsourcing project. Chapter 5 describes our research into sources of information for identification of usages of idiomatic multiword expressions in arbitrary digital text. Chapter 5 is a broad investigation of considerations for computational lexical semantics. It starts with methods for high coverage identification of relevant dictionary entries for wordforms in arbitrary text. These methods have relevance to the effectiveness
of Wakaran as a glossing application but are also used in Chapter 6 as part of the construction of a sense annotated corpus in Japanese using an existing sentence and word-type aligned translation of an annotated English corpus. We use the resulting corpus to evaluate the effect of recent efforts in automatic cross lingual LKB enrichment on results for knowledge-based all words word sense disambiguation. The selection of WSD methods evaluated in this chapter is later subjected to extrinsic evaluation in Part IV using crowdsourced data.

Part IV describes the deployment of WSD methods within the Wakaran Glosser, evaluates them extrinsically with respect to the data collected from user interactions, and draws the final conclusions for the thesis. In Chapter 7 we show how the data collected from Wakaran usage can be aggregated into a corpus of sense selections. We then show that evaluation of automatic word sense disambiguation on this corpus predicts differences found by evaluation on the corpus built from transferred annotations in Chapter 6. Finally, we study data from usage of a version of Wakaran built with an automatic WSD assistant to demonstrate that improved automatic word sense disambiguation performance has significant scope to improve the usefulness of the crowdsourcing application to its users. The demonstrated impact of automatic WSD motivates future work studying Wakaran as a computer assisted language learning system with computational semantic features.

Chapter 8 draws together the results of the previous chapters to conclude that word sense disambiguation and computational semantics in general have the potential to fulfil real world applications in Wakaran and, in so doing, to improve our understanding of word senses. We outline how future work can utilise Wakaran to assess the impact on learning outcomes such as vocabulary retention for language learners of automatic WSD. Additionally, we outline the potential for research into improving qualities of computational semantics such as real time delivery of sense judgements that get little research attention but are important to its application in Wakaran.
Chapter 2

Literature Review

Lexical semantics — the meaning of words — is frequently studied today by means of reference to large corpora: collections of samples of human communication that are intended to represent the diversity and and distribution of language usage. This has not always been the case: there is a tradition in the academic study of language popularised by Chomsky (1957) whereby models of human language are developed by means of introspective examination of language cognition by experts. However, with the advent of electronic computers and the subsequent encoding of large corpora, statistical methods for linguistics became not only more feasible but hugely successful (Manning and Schütze 1999). Furthermore, the availability of machine readable dictionaries opened up scalable, data driven techniques for the study of the relationship between natural language and the concepts encoded in dictionary entries (Lesk 1986).

Data driven computational techniques for modelling the relationships between words, concepts and text fall broadly into the computer science fields of machine learning and data mining (Manning and Schütze 1999; Tan et al. 2006). To achieve success with these methods often plain natural language samples are not enough: additional expert annotations providing information targeted at the phenomena under study are required (Pustejovsky and Stubbs 2012). However, the acquisition of annotations for studying connections between dictionary entries and text can be a particularly costly and onerous task (Márquez et al. 2007). Recent developments in data acquisition use
a method termed human computation: channelling the activity of large numbers of non-experts into
the creation of annotated datasets suitable for statistical analysis (von Ahn 2005). This approach
holds promise for relieving the knowledge acquisition bottleneck of computational lexical seman-
tics (Venhuizen et al. 2013). To motivate participation in human computation, research projects
typically use one of monetary incentive, competitive game mechanics or a combination of both
(Morschheuser et al. 2016). However, there is an existing demand from students learning a second
language for applications retrieving dictionary entries from text (Whitelock and Edmonds 2000).
This thesis investigates leveraging that demand to construct annotated corpora suitable for study of
computational lexical semantics.

2.1 Computational lexical semantics

In this section we give a broad survey of techniques for data driven lexical semantics. The
following sections will dive deeper into selected topics.

2.1.1 Introducing word sense disambiguation

There are many words in English and other languages which can take on different mean-
ings with the same surface appearance of the word. Consider this hypothetical dictionary entry
listing definitions for the word *ash*: (a) a kind of tree, (b) the wood of that tree, and (c) the residue
of a fire. To computer software the string representation of all three meanings of *ash* appear the
same because they are encoded the same. For instance, in the sentence *a branch fell from the ash*,
there is no distinct rendering of *ash* to indicate which of the above meanings is closest to the in-
tended usage: to the computer program its meaning is ambiguous between the three. Some kind of
inference process must be used based on the content of the rest of the sentence. For many applica-
tions, especially downstream NLP applications such as machine translation, a representation of the
intended meaning of a particular usage of *ash* may be helpful or in fact essential to properly extract
information from the text. The task of identifying a particular dictionary sense relevant to a word
usage is called **word sense disambiguation (WSD)**.

Lesk (1986) introduced a data driven method for identifying the meanings of word usages, breaking from a tradition of painstakingly crafted artificial intelligence expert systems (Gale et al. 1992a). The Lesk algorithm takes unannotated text as input and looks up definitions of the words in a machine readable dictionary. Whenever the dictionary provides multiple alternate definitions for different senses of the word the algorithm must attempt to select the right one. For example, considering the definitions of *ash* given above, Lesk must select one definition to best describe the word as it is used in a given text: for *a branch fell from the ash* the best choice is sense [a]; for *he swept ash from the fireplace* it is [c]. This example epitomises the task of word sense disambiguation: the identification of the meaning of a word’s usage (its **token**) in terms of discrete definitions (senses) for the word in-vitro (its **type**). The identification of a meaning for a word token must be made based on knowledge of the context of the word’s usage. Sometimes a full understanding of an utterance can only be achieved with knowledge of who is speaking, who they are speaking to and events that have recently transpired. However, for machine readable documents it is often the case that the only thing known of the context of a word’s usage is the text of the surrounding paragraph. In this thesis the surrounding text is the only form of context we take into account for disambiguating the meaning of a word token. The Lesk algorithm disambiguates a token by consulting the dictionary definitions of words in the surrounding text. It selects the sense of the token that overlaps most with the definitions of other words in the context. Overlap is measured by the number of words two definitions have in common. In the examples above, a definition of *fireplace* might include the word *fire*, thus overlapping with definition [c]; a definition of *branch* may mention *tree*, overlapping with [a]. Lesk (1986) viewed this as a way to cheaply gain sense labels for word tokens to use in downstream applications without requiring the attention of human experts to individual word types.

In the years since, a number of direct derivatives of the Lesk algorithm have been proposed, including a simplified version that compares target word definitions to the context words instead of their definition text (Kilgarriff 2007), an extended version that includes definitions not just of target words and words in context but also follows thesaurus relations such as synonymy to
related words and includes their definitions as well (Banerjee and Pedersen 2003) and, naturally, the combination of those two modifications into a simplified extended Lesk algorithm (Ponzetto and Navigli 2010).

**Distributional word sense disambiguation**

Whereas Lesk (1986) and its variants used the content of dictionary definitions for sense disambiguation, another approach exists which models word senses purely on the contexts in which they are used. Bruce andWiebe (1994) manually labelled 2369 tokens of the word *interest* with senses from *Longman Dictionary of Contemporary English (LDOCE)* and fit a statistical model to the data. The statistical model required a representative sampling of the joint probability of all variables in the model, so the context was represented by a small number of categorical variables:

- **pluralisation** of *interest* as a Boolean,
- **part of speech (POS)** tag (Noun, Verb and so on) of the adjacent word,
- **word co-occurrence** in the sentence, as a Boolean, for each of five informative word types.

To keep the number of features small but informative, the five words least statistically independent of the sense labels were selected from amongst the 400 words appearing most commonly in sentences with *interest*. Those words were: *in, percent, short, pursue* and *rate*. All other aspects of the context were discarded. This highlights the main limitation of the method: it requires a lot of annotation effort to build a disambiguation model for just one word’s senses based on only a few word types in the context. Together, Lesk (1986) and Bruce and Wiebe (1994) highlight the two main strategies for solving WSD:

- **knowledge-based** using information encoded in lexical resources like dictionaries; and
- **supervised** using sense annotated corpora to learn a mathematical description of the contexts in which senses are used.
Supervised machine learning models which are more robust to sparse data make it possible to use much more detail from the context. Murata et al. (2001) extracted a wide variety of features from the textual context to use as features for Naive Bayes (Hilden 1984) and Support Vector Machine (SVM) (Boser et al. 1992) classifiers trained on a sense annotated corpus of a small selection of Japanese word types. Since standard SVM classification only discriminates two classes, they used a pairwise method to distinguish between more than two senses: the sense that is chosen over the most of its peers, pairwise, is the one selected. Features made use of a part of speech (POS) tagger and a sentence structure parser to represent:

- uninflected form and POS of adjacent words;
- uninflected form and POS of every word type appearing in the context;
- the immediate phrase type containing the target word (such as Noun-phrase, Verb-phrase);
- the particle of the target word (Japanese particles include prepositions such as at, on, in and so on);
- details of the word modified by the target token;
- details of words modifying the target token;
- and more besides.

Section 3.2 reviews details of Japanese POS taggers and parsers which enable these kinds of features. Lee and Ng (2002) performed a detailed comparison of such sources of features for English WSD for a variety of annotated corpus based methods including Naive Bayes and SVMs. They concluded that SVMs performed better than the other methods when provided with a wide variety of features, however other methods could perform better when an informative subset of the features were selected. In Chapter 5 we investigate the application of the word sense disambiguation features and learning algorithms described in this section to disambiguation of ambiguous expressions with potential idiomatic interpretations.
Knowledge-based word sense disambiguation

Lesk (1986) inferred connections between word senses through shared words in their definitions, but there are lexical knowledge bases (LKBs) which explicitly encode certain relationships between words. A popular one is Princeton WordNet ("WordNet") which encodes concepts as sets of synonymous words together with a definition explaining their shared meaning (Miller et al. 1990; Miller 1995). WordNet can be understood as an ordinary dictionary, where each word has one sense entry for each synonym set (synset) it is a member of. However, its real power rests in its network of relationships between synsets. In particular, it encodes the hypernymy or IS-A relationship. For example, the synset \{big cat, cat\} has, as a hypernym, the more general synset \{feline, felid\}, and among its hyponyms are found examples of specific kinds of cat such as synset \{leopard, Panthera pardus\} and synset \{tiger, Panthera tigris\}. Much work has been made of drawing measures of similarity between arbitrary pairs of words or concepts on the basis of paths following along these relationships. Rada et al. (1989) measure the distance between two concepts in a hypernymy graph by finding the minimum number steps needed to get from one to the other. In many cases this will mean ascending to a general concept such as animal or plant and the descending again to the other concept. For example, the distance between tiger and leopard in WordNet is 2: one step up from tiger to big cat followed by one step down again to leopard. Rada et al. (1989) found that distances described in this way correlated with semantic distances given by human judges. However, minimum distance did not work as well (unmodified) when the underlying graph was built from relationships other than hypernymy. Refinements and alternative network algorithms exist to the minimum distance metric. For example, Sussna (1993) uses multiple synset relation types in WordNet, giving different weights to each relation kind and additionally increasing the distance between abstract concepts and distances involving highly connected concepts. Resnik (1995) on the other hand, describes an algorithm that searches upwards through a taxonomy graph from two concept nodes to find their common subsuming nodes but instead of using path distance uses scores precomputed for the nodes as a similarity value. All concepts in the hierarchy are
assigned an information content score that decreases for more abstract concepts: the similarity between two concepts is given by the highest information content of any of their shared subsuming concepts. The information content of a concept is estimated from corpus statistics by measuring the frequency of words that could represent a concept or one of the concepts it subsumes and taking the negative log likelihood of it appearing. The information content measure had a substantially higher Pearson correlation (0.791) with a gold standard set of human judgements than a minimum distance similarity metric (0.667) over the same LKB and evaluation data. To measure how accurately a single human can predict the gold standard, Resnik (1995) performed a replication of the construction method for the word similarity corpus they used in their evaluation using 10 human similarity judgements per word pair. The combined replicated gold standard was highly correlated with the original human gold standard but individually human annotators had an average correlation with the reference of 0.885, which places an upper bound on algorithmic performance. Jiang and Conrath (1997) also use information content, but in a distance metric rather than a similarity measure. In the simplest form they take the distance between a concept and its hypernym to be the difference between their information content. This means the distance between two concepts simplifies to the sum of their information content minus twice the maximum information content of their subsuming concepts. A more complex form weights hypernymy edges on the product of three factors:

- the information content difference;
- the ratio of their depths raised to a parametrised power \( \left( \frac{d}{d+1} \right)^\alpha \); and
- the local network density.

Under evaluation against the same data as Resnik (1995) they find that their method yields a modest improvement to a correlation of 0.828, which is closing in on the human annotator average.

The path length and information content methods discussed so far all ultimately settle on a single path when measuring semantic distance or similarity; information about the length and number of alternate paths between concepts is discarded. By contrast, Kozima and Furugori (1993) perform WSD using a technique called spreading activation: starting at a single concept
and flooding activation outwards along semantic relations using mixing and damping functions. Collins and Loftus (1975) present evidence that spreading activation is an accurate model of human cognition of semantic relatedness. They argue that spreading activation simulates the time a person will spend answering concept association questions because the person starts at a primed concept, freely associating outwards to related concepts until reaching a target. The theory has a lot of qualifying detail to accurately reproduce experimental data but we highlight three elements of it here:

1. the spreading of activation must be damped over distance;

2. direct associations may carry different weights;

3. primary relations such as hypernymy are important, but any verb may carry a relation.

Page et al. (1998) proposed and tuned an algorithm, PageRank, based on extremely similar principles for an entirely different graph: that of the world wide web’s interconnected pages. PageRank is characterised by a random walks taken through the graph with uniform probability assigned to all outgoing edges from a node and a constant probability of restarting the walk at a node selected randomly from the entire network. A refinement, Personalised PageRank, restricts the random restart to a small selection of nodes and has proved effective as a network model of semantic relatedness. Agirre and Soroa (2009) show that selecting senses most strongly related to words in context by Personalised PageRank of WordNet can yield state of the art performance in WSD. For their network they use the standard WordNet relations augmented with additional relations between synsets and manually applied sense tags on the synset definitions. The edge weights in Personalised PageRank are typically dependent only on the number of outgoing edges, not on the relation type. In Section 2.2 we give a more detailed review of WSD literature with a particular focus on Personalised PageRank. In Chapter 6 we investigate in detail the effect of further augmentations of WordNet with new concepts and unweighted relations on Personalised PageRank WSD results.
2.1.2 Other approaches to computational lexical semantics

One thing we have not yet stopped to question is the validity of the sense inventory found in a specific lexicon. Does taking on the task of WSD inherently assume that the lexicon is complete and correct? Is there any basis to claim that the senses of words found in WordNet, or other machine readable dictionaries such as LDOCE, are the true senses of the words and, if not, can we truly hope to identify them in natural language? Kilgarriff (2007) describes the writing of a dictionary entry as a process of collecting examples of a word’s usage, clustering those examples into similar meanings and, finally, formulating a description of the common meaning in each cluster. This is ostensibly the job of a human lexicographer, but it may be modelled computationally: Schutze (1998) introduces an unsupervised clustering algorithm for inducing senses by grouping usages automatically. Furthermore, a recent SemEval task called for construction of descriptive entries that at least discriminate the main patterns of usage (Baisa et al. 2015). Researchers have met with some success when evaluating the ability of algorithms to reproduce sense distinctions found in existing inventories including WordNet and others (Pantel and Lin 2002; Manandhar et al. 2010). That being the case, it seems that the clusters do form a ground truth which a supervised method may learn to generalise, and sense definitions provided by the lexicographer could have sufficient descriptive power for a human annotator or knowledge-based algorithm to reconstruct the clusters from plain examples. So with reference to a specific dictionary word senses may exist and bear a relationship to clusters of usages. However, different lexicographers (and indeed different ordinary language users) may judge clusters of meaning on different criteria and thereby come to a different ground truth (Palmer et al. 2007a). As such, it makes sense for researchers to put aside an assumed sense inventory and investigate hypotheses that reflect on the nature of word usages and how their meanings differ from each other. Many data driven studies exist coming at this question in different ways:

Erk et al. (2013) developed two data sets using untrained annotators to study ordinary human intuition for distinctions in meaning:
**WSsim** collecting graded ratings for the similarity of word sense definitions to word usages; and

**Usim** collecting graded ratings for the similarity of word usages to other word usages.

They found that inter-annotator agreement on these tasks was good, meaning that the subjects did tend to agree on the similarity of meaning. **Erk et al. (2013)** also found that the graded sense assignments in **WSsim** correlated with single best sense assignments made by trained annotators. Thus, single best sense assignments could be considered correct, but less informative than grading all senses. In Chapter 6 and Chapter 7 we implement WSD algorithms capable of giving a graded score for each sense of a word in context and evaluate them against single best sense annotated resources.

Concern that typical sense inventories may split sense distinctions too finely has led to a number of studies for grouping senses into coarser subdivisions (**Palmer et al. 2007a**; **Navigli et al. 2007**). **Erk et al. (2013)** examined the implications of **WSsim** for groupings of sense inventories and found that whilst some of the fine grained senses of **WordNet** had highly correlated scores there were nevertheless specific usages that could justify finer distinctions by polarising two otherwise correlated senses. Additionally, specific examples existed to undermine a grouping by giving a high score to both of two otherwise negatively correlated senses.

**Usim** has the interesting property that it eliminates sense enumeration altogether, reducing the problem of semantic distinction to relationships between word usages. Work has been done on an annotation task called lexical substitution which takes a similar approach: instead of labelling word usages with dictionary senses, they are labelled with other words that could be substituted in without changing the meaning of the sentence (**McCarthy andNavigli 2007**; **Kremer et al. 2014**). **Erk et al. (2013)** compared **Usim** to lexical substitution data, finding that the degree of overlap between sets of substitutions suggested by human annotators for different usages correlated with human numeric ratings of similarity. Given that word usages do seem to have a form of objective similarity, it may yet be possible to describe objective criteria by which to group usages into self-similar clusters. This task is called called Word Sense Induction (or discrimination), and many
methods exist for it (see, for example, various SemEval tasks [Manandhar et al. (2010), Jurgens and Klapaftis (2013), Navigli and Vannella (2013)]) but the effectiveness of the results are difficult to compare on objective criteria. For example, Palmer et al. (2007a) use two methods:

1. Standard clustering performance metrics basing reference clusters on sense annotations from an existing lexical knowledge base.

2. Standard WSD performance metrics after a manual supervised mapping of induced senses to the best matching sense from an existing lexical knowledge base.

Either way, the results do not fully break the reliance on judgements made by the lexicographer(s) of the original lexical knowledge base. It may, however, be possible to do so: Schutze (1998) suggests that the utility of WSD for downstream tasks relies only on the ability to discriminate unrelated word usages, rather than on senses in a specific inventory. There is a lot to be learned from studies such as these that model semantics from the ground up, based only on unsupervised analysis of linguistic examples. However, in this dissertation we restrict ourselves to studying the relationship between free text and the entries in existing dictionaries.

2.1.3 Multiword expressions

A multiword expression (MWE) is an expression spanning multiple words that exhibits some kind of special or unusual behaviour beyond the normal compositional productivity of language [Sag et al. 2002]. In this thesis we focus on MWEs that have been lexicalised: that is, expressions with special characteristics that warrant explanation in a dedicated entry in the lexicon [Bauer 1983]. Take for example the English expression elephant in the room which means something that no-one could fail to be aware of but no-one has openly acknowledged. One could perhaps infer that an elephant in a room is a metaphor for a thing that demands attention but the matter of awkward silence on the situation makes this phrase unequivocally idiomatic. Other examples include call it a day, used to declare a halt to work, and let the cat out of the bag which means “to reveal a secret”.
Type extraction and classification

The absence of multiword expressions from a computational lexicon can have significant downstream effects on NLP applications (Sag et al. 2002). There has been considerable work into addressing this issue by mining corpus resources for MWEs. Ramisch (2015) in an extensive treatment of the topic, synthesised various levels of assumptions made by work in the field into a standardised processing pipeline with a particular emphasis on evaluation methods. The main stages in the pipeline are:

**Preprocessing** by external tools for identifying word boundaries, normalised token lemmas and parts of speech, and parse trees for corpus sentences.

**Indexing** of sequences in the corpus of each entity type identified in preprocessing including surface and lemma and POS n-grams and fragments of parse structure.

**Extraction of candidate** MWEs from the corpus by application of pattern searches to the index.

**Filtering of candidates** based on custom features extracted from the usages themselves.

This is followed by:

**Manual validation** of the candidates by a lexicographer or human computation task.

Or by:

**Learning** from a gold standard corpus to distinguish true MWEs from incidental candidates automatically, or evaluation of the extraction method via a gold standard corpus.

The pipeline is implemented in a software application called the mwetoolkit which should be a useful general purpose tool for a variety of MWE processing tasks.

Early methods for MWE extraction focused on lexico-syntactic inflexibility. For example Lin (1999) identified word pairs with unusually strong selectional preferences for each other in

grammatical dependency relationships, looking in particular for cases where otherwise valid lexical substitutions are not observed to occur. Bannard (2007), by contrast, identified MWEs by looking in corpora for phrases with morpho-syntactic inflexibility: specifically those that are not used in the grammatical passive form or that resist addition or subtraction of subordinate modifiers to their constituents.

It has been observed that MWEs are not all strictly idiomatic, some taking their meaning by degrees from their constituent words (Nunberg et al. 1994). On the one hand there are fully idiomatic MWEs such as *kick the bucket*, meaning “to die”, which is completely unrelated to the meaning of its constituents. On the other there are *traffic light* and *street light* which are both literal but have strictly distinct interpretations. There are also light verb constructions such as *take a nap* which get their literal meaning from *nap*, discarding the literal meaning of *take*. Fazly and Stevenson (2007) seek statistical measures to classify MWE types into categories including abstract combinations, light verb constructions and idiomatic combinations. They use a variety of criteria including lexico-syntactic fixedness — inflexibility to grammatical inflection and lexical substitution of constituents — and semantic compositionality. The latter is measured by modelling the distribution of contexts in which a word appears as a vector containing a count of each context word type. The cosine between the MWE’s context distribution and its constituents’ context distributions is used to represent similarity and therefore compositionality. Other data sources that have been mined for measuring compositionality of MWEs include monolingual and multilingual dictionaries (Salehi et al. 2014, Salehi and Cook 2013), and context word distribution vectors from unannotated corpora (Katz and Giesbrecht 2006) and annotated corpora built through crowdsourcing (Reddy et al. 2011).

In this thesis we work with such MWEs as have been defined in a dictionary and make use of type classifications where available. Specifically, in Chapter 5 we study a corpus of annotated MWE usage candidates for a set of MWEs found in Japanese idiom dictionaries and classified as ambiguously idiomatic. Chapter 6 works with lexical knowledge bases in their entirety and as such generally relies on the assumption that if an MWE has been included in the dictionary, it is there for
a reason and should be preferred, although we sometimes override this with certain heuristics. The enclosing chapters discuss an application for which it is preferable to label text with both MWEs and their constituents simultaneously.

Identification of tokens

Earlier, we motivated identification and classification of MWE types by the downstream impact naivety of MWEs has on NLP applications; it follows that we must be able to detect any usage of a MWE in text to realise the benefits of learning about its type. However, identification of tokens of a MWE comes with many challenges: not only might the constituent words’ surface forms be changed under inflection, but the constituents themselves might become separated by other words of the sentence. For example, the expression *to cut corners*, meaning “to finish a job cheaply by neglecting (potentially important) details”, can appear modified in both of these ways: *you are cutting some corners doing your own taxes like that*. As such, some means is required to identify the inflected constituents — *cutting*, from *cut* — and also to find the constituents despite their separation in the text. However, sometimes the constituents of a MWE might appear together as an ordinary literal expression: even divorced of any context the phrase *kicking the buckets* demands a literal interpretation, shirking the figurative sense that prefers the singular *bucket*. [Cook et al.](2007) develop a method for identifying MWE types by their syntactic restrictions and then use the restrictions to identify the outlying minority of apparent tokens of the MWEs that violate those restrictions as literal usages.

However, sometimes the clues that a particular expression is or is not a usage of a MWE are purely contextual: *after much hissing, yowling and scratching he finally let the cat out of the bag* is most likely talking about a literal cat and a literal bag, despite perfectly resembling the phrase *let the cat out of the bag* meaning “to reveal a secret”. [Katz and Giesbrecht (2006)] used distributional statistics from a labelled corpus to identify idiomatic and compositional usages of idioms with literal interpretations. They used a method called *latent semantic analysis (LSA)* for compressing the context representation from a vector of word type counts into a smaller vector space by performing
a singular value decomposition (SVD) of the word co-occurrence matrix (Deerwester et al. 1990; Landauer and Dumais 1997). Although the idiomatic token identification relied on an annotated corpus, the method inspired the second part of the study which performed type identification by comparing the LSA vector of an expression’s context distribution to the composition of the LSA vectors of its constituent’s distributions when appearing alone.

In Section 2.3 we review more details of MWE research with a particular focus on discrimination of MWE tokens from literal usages of their constituents. We show in Chapter 5 that syntactic restrictions can only take us so far and that ultimately semantic analysis is a more powerful approach to MWE token identification. In Chapter 6 we take a step back to look at the initial identification of potential usages with inflected wordforms by exploring several forms of the prioritised tiling algorithm of Whitelock and Edmonds (2000).

2.2 Word sense disambiguation

In this section we will cover in more detail the history and state of the art in word sense disambiguation. Both have been shaped by the organising influence of a series of collaborative evaluation events — named Senseval and more recently SemEval — and conferences in the Joint Conference on Lexical and Computational Semantics (*SEM) series. Contributions to the field span a variety of different objectives — from lexical sample and all words WSD through to sense induction and grouping — and use a variety of different resources including annotated corpora, bootstrapped corpora and both professional and crowdsourced lexical knowledge bases. The existence of standard reference points for evaluation made popular by the Senseval and SemEval shared tasks did a lot to ensure comparability of WSD research, but did lead to a gravitation of research towards the assumptions about word senses inherent in the exact match criteria (Navigli 2009). In recent years the variety of tasks and evaluation criteria standardised by the SemEval and *SEM events has diversified substantially. In this thesis we focus on knowledge based methods that provide graded values measuring the strength of a word sense’s relationship with a usage. In particular
Chapter 2: Literature Review

we investigate lexical knowledge base resources for Personalised PageRank methods for WSD.

2.2.1 WSD research community

The Senseval — now SemEval — shared task workshops were proposed by [Resnik and Yarowsky (1997)] as a means to standardise WSD results and focus the community’s efforts. These events make available shared tasks including datasets and evaluation criteria. The standardised results achieved by WSD systems entered into Senseval allow for quantitative comparison of competing methods and continue to provide benchmarks for research in the years following the competition. The Senseval events also help to co-ordinate the direction of the WSD research community: [Palmer et al. (2007b)] reports that after Senseval-3 in 2004 it was generally agreed that performance at WSD as a standalone task had hit a plateau and that it was time to expand the field in new directions. The next event in the Senseval series was renamed SemEval to reflect that it included a much wider array of semantic tasks, such as lexical substitution.

Although it began as a triennial workshop, since 2012 SemEval runs every year, as does the equally specialised *SEM conference which provides a dedicated venue for publication of work on computational lexical semantics. A broad range of topics is covered, including compositional distributional semantics [Gupta et al. 2015; Le and Zuidema 2015], readability [Mesgar and Strube 2015], negated phrase detection and category expression paraphrasing [Prabhakaran and Boguraev 2015; Sales et al. 2016], text segmentation on semantic boundaries [Glavaš et al. 2016] and many more. SemEval has always been about formalising task definitions to incentivise comparability of research; in recent times it has brought a variety of tasks such as sentiment analysis [Nakov et al. 2016] and its specialisation to particular aspects [Pontiki et al. 2016], complex word identification [Paetzold and Specia 2016] and semantic textual similarity [Agirre et al. 2016].

On lexical semantics specifically, the current diversity of topics featuring at SemEval and *SEM includes many interesting developments. The “DiMSUM” task [Schneider et al. 2016] combines MWE token identification with suppersense disambiguation — suppersenses being high level concepts from near the top of a concept taxonomy with a total cardinality only somewhat larger
than the entity types used in named entity recognition. Although supersenses may seem like a blunt instrument compared to a standard concept inventory closer to the size of the lexicon, semantic models built for them can be more accurate, their study is less inhibited by the knowledge acquisition bottleneck, and those savings are passed on to downstream applications which do not have to understand the full sense inventory. Lopez de Lacalle and Agirre (2015b) introduce a method for a more traditional study grouping senses of words into coarser granularities for WSD. Taking the data from a sense annotation crowdsourcing project (Passonneau and Carpenter 2014) they note that quality control concerns resulted in a higher number of annotators and a higher degree of disagreement on each instance than would be produced by professional annotators. Lopez de Lacalle and Agirre (2015b) used the annotator disagreement matrix on each single instance to build a hierarchical clustering of senses for each word type, which can be used to build a sense inventory at the desired granularity.

One topic being explored at recent SemEval events is the relationship between algebraic and vector space models of meaning. Kornai et al. (2015) give an interesting comparison of algebraic and vector space models from the perspective of competence, specifically by looking at how each paradigm is able to explain how the ability to understand words is acquired. They identify key elements of what it means to be competent in lexical semantics and conclude that both models are capable — to some degree — of encompassing a model of learning those competencies. Aletras and Stevenson (2015) explore models of lexical semantics that hybridise a LKB representation of lexical meanings with an existing distributional model representing words by a vector of co-occurrence features. Their first hybrid embeds concepts from the lexical knowledge base into the distributional vector space as the centroid of the vectors representing each word linked to the concept by the lexical knowledge base. The second hybrid embeds concepts in the vector space by taking the lemma represented by each vector dimension and seeding a Personalised PageRank calculation with it; that dimension of the vector is then assigned the Personalised PageRank score of the concept being modelled. These two distinct vector models of concepts then have their dimensionality reduced using latent semantic analysis and, finally, they are mutually transformed using canonical correlation.
Chapter 2: Literature Review

analysis (CCA). All up that makes six vector models for concepts in the LKB. Evaluated against word (type) similarity datasets, the CCA based models improved most overall on the standard co-occurrence vector model, although in a couple of instances the latent semantic analysis models were slightly better. However, when used for WSD via a cosine similarity with the context, only the second base model performed well. This suggests that although all six models nominally represent concepts rather than word types, only the vectors produced from the single-source Personalised PageRank distinguish word senses from each other well. For another perspective, we highlight Song et al. (2016) who use an adaptation of popular neural network methods (Mikolov et al. 2013) to learn word senses from a corpus whilst jointly learning vector space embeddings for both words and the induced senses. Specifically, they iterate over the corpus a number of times, visiting each word to calculate a current best sense from the context (in a variant, new senses have a probability of being created) and then performing a neural network update for predicting the context from the sense. The final vector representations of senses and context words are their first layer activation co-efficients in the neural network. Comparing to a variety of word sense induction systems on a historical SemEval task, they found that their method outperformed the others on supervised criteria (that is, relative to supervised sense boundaries) but were not clearly better on unsupervised clustering quality measures. Vector space embeddings of LKB and induced word senses are a promising area of development from both knowledge-based and distributional semantic model perspectives. For knowledge based approaches a distributional representation blurs the artificial bright lines that network models demarcate between closely related senses. On the other hand, for distributional models they eliminate the problem of finding a single point in a vector space to embed polysemous words. This is a promising area of research and hopefully its appearance in *SEM will lead to a targeted task formalisation featuring prominently at SemEval.

2.2.2 Classification of WSD research

Navigli (2009), in a comprehensive summary of the state of the art of WSD, provides a framework for classification of WSD research which will be useful to us in discussing the field.
In fact, the terms in this classification scheme have developed over a long period of time and have appeared earlier in this review but we give them a more formal treatment here in our description of the classification framework.

Firstly, word sense disambiguation research can be divided by the scope of the study:

**All words WSD**  Given a sense inventory with a wide coverage – such as a comprehensive dictionary – find the meaning of all words in an arbitrary text by assigning each word a sense from the inventory.

**Lexical sample WSD**  Given a sense inventory for a (usually small) selection of target word types and an arbitrary text disambiguate tokens only of the target words.

From a data miner’s perspective, this might mean the difference between collecting 1000 samples:

- all sharing a single target word type; versus

- each dedicated to specific word tokens from a 1000 word document.

Since the latter provides very little information about each individual word’s meaning, lexical sample WSD research can perform much better against evaluation metrics than all words disambiguation. However, Navigli (2009) reports vocal community sentiment going back to Senseval-3 that the lexical sample task is not particularly interesting given that systems are restricted to disambiguating a small set of words and so have limited potential utility in applications.

Navigli (2009) breaks the task down even further by making a distinction between word **type** disambiguation and word **token** disambiguation. A word token is considered to a specific word at a specific position within a body of text so token disambiguation assigns a sense to each position in the text. A word type on the other hand is an abstract conception of a word independent of its context: word type disambiguation picks one sense of a word and applies it to the whole target text. Note that type disambiguation is equivalent to a token disambiguation where one happens to choose the same sense for all tokens of the same word type. This is more commonly called the one sense
per discourse heuristic and it works well because for most consistent discourse only a single sense of any given word type is used (Gale et al. 1992b).

Another factor which influences the nature of a WSD study is the selection of resources used.

**Supervised WSD** involves the use of a corpus of examples with gold-standard sense annotations. That is, the disambiguation system has a selection of examples with known-correct sense annotations and makes use of this in deciding senses for words in unseen text.

**Unsupervised WSD** is the counterpoint to supervised WSD: this will typically describe research which uses a corpus of *unlabelled* examples to aid disambiguation; however it may also describe systems which use no corpus at all.

**Knowledge-based WSD** Makes use of digitised knowledge about words to perform WSD. This knowledge can come from machine readable dictionaries, thesauri or deeper ontologies such as the Princeton WordNet.

Supervised disambiguation typically uses naturally occurring language data but requires additional annotation data created specifically for the research study. Compared with unsupervised data mining, it is difficult to accumulate enough correctly annotated data to perform well at supervised disambiguation of word senses. As discussed previously, this difficulty is known as the knowledge acquisition bottleneck. Although various methods have been developed to overcome this bottleneck, it remains a significant challenge for supervised WSD (Lopez de Lacalle and Agirre 2015a). On the other hand, the much larger corpora researchers can amass for unsupervised WSD are much harder to make use of because they do not contain explicit information about senses. An unsupervised corpus may approximate a sense annotated corpus by clustering examples into groups of similar usages and assigning senses to clusters. However, because words tend to strongly prefer their most frequent sense the supervised *most frequent sense (MFS)* baseline tends to score very highly on evaluation metrics (compared to, for example, a random sense selection) and unsupervised WSD systems have difficulty performing better. Supervised systems on the other hand can
approach the accuracy achieved by a human annotator if working without consultation with other annotators (Navigli 2009). Since our goal in this thesis is to automate the human task of disambiguation, we will be primarily interested in knowledge based disambiguation methods, and in supervised methods where there is a possibility of generalising models beyond a lexical sample.

There also exists research into word senses which is neither supervised nor knowledge based, which focuses on identifying patterns in corpora and inferring the existence of word senses from those patterns: this is known as word sense discrimination (Schutze 1998). Although we have an interest in the nature of word senses, we will not study sense discrimination in this thesis because we are primarily interested in working with bilingual dictionaries for which senses have already been manually discriminated.

Finally, a salient characteristic of a WSD task is the nature and construction of the sense inventory used. In particular, a distinction can be drawn between fine-grained and coarse-grained sense disambiguation. A fine-grained sense inventory draws subtle distinctions between senses of each word, meaning that some senses will be closely related to each other. By comparison, a coarse grained inventory clumps together related usages into a smaller number of groups. Some implications of this are discussed in the next section.

2.2.3 Evaluation of WSD

WSD systems are typically evaluated on their ability to correctly select one sense for each target word (Resnik and Yarowsky 1999; Navigli 2009). Resnik and Yarowsky (1999) calls this the exact match approach and outlines several attractive alternatives Senseval might have explored. These include allowing WSD systems to produce a collection of senses with weights and score them accordingly, or taking into account relationships between similar senses and giving credit for answers which are almost correct. One decade on, Navigli (2009) noted that although interesting measures were investigated a little in the Senseval evaluation tasks, WSD was still primarily performed as an exact match task, and that remains the case for SemEval tasks to date (Moro and Navigli 2015) although semantic similarity measures do get attention, particularly between word
types or between larger blocks of text (Jurgens et al. 2014). The most common metrics used to measure performance are given by Navigli (2009) as:

**Coverage** The proportion of target words for which the system produced a response.

**Recall** The proportion of target words for which the system produced a *correct* response.

**Precision** The proportion of responses produced by the system which were correct.

**Accuracy** If coverage is complete, then precision and recall are the same and simply measure the accuracy of the system.

**F-score** This is the first harmonic mean of recall and precision: it can be used to compare systems which sacrifice either in order to do better at the other.

Although these metrics can be interpreted in isolation, a better picture is formed by comparison to reference results. Three kinds of reference are popular:

**Baselines** specify the performance of a trivially implemented disambiguation algorithm.

**Benchmarks** are competing systems which represent the state of the art. For this purpose, references to recent results from Senseval or SemEval events are often made.

**Upper Bounds** represent the best performance one could reasonably expect to achieve with the information given in the evaluation data.

Using baselines, benchmarks and upper bounds can give lead to a more realistic interpretation of results: Figure 2.1 illustrates how using reference points can put the difference between two annotators into perspective.

A common baseline is the *most frequent sense (MFS)* (also known as *majority class*) baseline: this is the performance we would achieve if we always selected the most common sense of the word. In practice the most frequent sense means the relative frequency of the dominant sense in a sense-labelled corpus, which is an approximation of the true frequency assuming the corpus is
balanced relative to natural language in general or at least the target language domain. Note that this means the most frequent sense baseline is a supervised disambiguation algorithm.

The inter-annotator agreement is a common upper bound used. This is a measure of the proportion of sense labels on which independent human annotators agree. It is considered to represent the performance of a human on the task and that it would be unreasonable to expect an automatic algorithm to do better (Palmer et al. 2007b). Palmer et al. (2007b) reports on a number of methods for calculating inter-annotator agreement, including exact matching, overlap and the kappa statistic.

Navigli (2006) investigates the implications of granularity of the sense inventory used for a WSD task. Figures reported from tasks in Senseval-3 give inter-annotator agreements of roughly 70% for fine grained sense inventories, but Navigli (2006) shows that grouping related senses into a coarser sense inventory significantly improved agreement between annotators. This raises the question of whether it is in fact appropriate to divide senses so finely that humans have such trouble distinguishing their usages. On the other hand, decreasing the number of senses — polysemy reduction — will naturally increase agreement, even for randomised sense labelling. However Navigli (2006) showed, by comparison of a benchmark annotator to a random one, that use of the coarser grained sense set results in a greater increase in performance than can be explained by polysemy reduction alone.

The work of this thesis involves presentation of dictionary entries to users so polysemy reduction would be impractical without a means of generating appropriate glosses for combined senses which would be a significant achievement in itself. We do however, wherever possible, try
to make use of evaluation schemes that move beyond the *exact match* criterion.

### 2.2.4 Contemporary challenges for WSD research

Navigli (2009) concludes his summary of the state of the art of WSD with the following future challenges:

**The representation of word senses** primarily focusing on the granularity of sense distinctions but also touching on how new senses are identified in changing language and also on models of meaning other than enumerating senses.

**The knowledge acquisition bottleneck** for both supervised and knowledge-based WSD. The knowledge-based side of this focuses on building or enriching ontologies by automatically learning new relationships between words from corpora.

**Domain specific WSD** is the specialisation of WSD to a specific domain of discourse. Most WSD research so far has been focused on generic language or has been limited to a specific domain (such as newspaper articles) incidentally due to the nature of the corpus used.

There has been a good deal of recent work towards the challenges outlined by Navigli (2009) in the subsequent years. Zhang and Heflin (2010) and Erk and McCarthy (2009) both investigate grading senses as an alternative to selecting the best sense; Agirre *et al.* (2010) showed WSD specialised to the biomedical domain; and Ponzetto and Navigli (2010) made a high impact contribution to knowledge-based resources for computational semantics.

Erk and McCarthy (2009) investigate the use of a corpus for which human annotators have provided sense-applicability ratings (or *grades*) for words in context on a five point scale. They explore evaluation methods for graded sense annotations including existing metrics and new metrics based on precision and recall. These metrics are applied to automatic sense grading algorithms, traditional best-match WSD algorithms and to the original human annotators who produced the gold standard. They found that human annotators varied significantly in terms of the average rating given, which had the effect of trading precision for recall or vice-versa. Grade producing algorithms did
better than best match algorithms, particularly when rating scales were normalised across available
senses. As with the human annotators, a trade-off between the measures of graded precision and
recall was evident.

Zhang and Heflin (2010) also pursue a sense grading approach but eschew arbitrary ratings
in favour of probabilistic output. They develop a probabilistic model for sense and outline methods
for estimating various parameters of it based on WordNet relations and sense frequency information.
The algorithm is applicable to any application domain for which a contextual bag of words can be
produced, but experiments were conducted on a specific domain: words in semantic web ontologies.
Context is defined by a network-based radius for traversing relations in the web ontology. Although
the aim was to produce a versatile probabilistic output the evaluation focuses on the accuracy of
the most probable sense when compared to best-match gold-standard annotations. The probabilistic
model achieved 84% accuracy compared to a first sense baseline of 64.1%.

The unusual nature of the word context considered by Zhang and Heflin (2010) makes
it, in a sense, a domain specific WSD application. However a better example of WSD applied to a
specific language domain is that of Agirre et al. (2010) who demonstrated disambiguation of words
with senses idiosyncratic to the medical domain. As an example, Agirre et al. (2010) gives the word
cold, which might refer to flu-like symptoms, body temperature or bodily sensations. A knowledge-
based approach is taken: specifically, Personalised PageRank is applied to an ontological graph
derived from a medical thesaurus. The resulting algorithm is applicable to all words in the thesaurus,
but is evaluated on a small number of words for which gold-standard data was available. A best
accuracy of around 68% is reported, which exceeded comparable benchmarks.

Chan and Ng (2006) work on adapting WSD models trained on one domain to classification on another. They adjust the outputs of a probabilistic classifier using an estimate of the sense distribution in the target domain. Their contribution is to the distribution estimation: they show that using a logistic regression to produce well calibrated sense probabilities for the target domain tokens works better than using a Naive Bayes classifier. The adjustments substantially increase the cross-domain classification accuracy.
Faralli and Navigli (2012) work on acquiring inventories of domain-specific senses. They take a higher level view of domain specific WSD, firstly identifying the relevant domain at the coarse level and then the sense of the word within that domain’s specific gloss inventory at the fine grained level. As such, their work is more general than treating domains as having additional senses or different sense priors. They extract glosses from web-pages for each domain by using heuristics to recognise formatted definitions and sourcing pages for a domain by searching for carefully selected terms unique to the domain. They construct an unsupervised semantic network between the extracted terms by linking headwords to each word in their glosses and perform Personalised PageRank WSD over the resulting graph. Evaluating with a purpose built gold standard dataset they find the precision and recall lag just behind a predominant domain classifier. They also test adjusting Personalised PageRank output with a domain relevance score which successfully boosts the accuracy above the predominant domain although not with statistical significance.

Ponzetto and Navigli (2010) contributed a substantial development in knowledge-based approaches for WSD which finally rivalled and in some cases exceeded the performance of supervised systems. This is achieved by augmenting the semantic relations encoded in WordNet with a much richer graph of interconnections between pages of the online collaborative encyclopedia Wikipedia. Wikipedia pages cover a single nominal topic and are aligned to WordNet noun synsets by a novel automatic approach. Wherever one page links to another and both are mapped to WordNet synsets a new edge is added to the WordNet ontology. Ponzetto and Navigli (2010) showed that even relatively simple knowledge based approaches – Extended Lesk and Degree – outperform comparable supervised systems on nouns and have comparable performance on all words. On domain specific testing corpora, these algorithms were able to significantly outperform supervised learners. Presumably this is because of the wide variety of domain specific knowledge encoded in Wikipedia.

As a final challenge for WSD, Navigli (2009) also highlights its lack of proven applications as a failing of the field and lists a number of potential applications which have been, or might be, tried. These include result disambiguation for information retrieval and extraction, machine translation, document classification, diacritic and spelling correction in word processing, lexicog-
raphy and the automatic construction of semantic links for the internet. One purpose of the word in this thesis is to take on this challenge by showing applications of WSD in disambiguation of word glossing for language learners. Furthermore, we will investigate ways to use user interface instrumentation and user feedback to extract knowledge resources for WSD, thereby tackling the knowledge acquisition bottleneck.

2.2.5 Recent word sense research

Many NLP tasks have see improvements recently through the use of a vector representation of word types learned by neural network models of word co-occurrence. However, when it comes to computational semantics two problems arise:

1. polysemy means that semantic relationships cannot be accurately represented by locality in the embedding space

2. composition of word vector representations into phrase and sentence representations remains an open problem.

Cheng and Kartsaklis (2015) tackle both of these problems simultaneously to show that sense disambiguation improves compositional models. In the basic form, sense embeddings are associated with context clusters and during training one sense is selected for each update by finding the nearest cluster centroid to the current context. A more general form selects senses using a softmax layer in the neural network itself. The compositional aspect is modelled on both word sequences using a recurrent neural architecture and alternatively on dependency syntax trees using a recursive neural architecture. In the end, the disambiguated models do show significantly better results on a number of tasks than direct equivalents using ambiguous word vectors.

Bhingardive et al. (2015) developed an unsupervised method for most frequent sense detection using neural word embeddings. A vector in the embedding space was constructed for word senses as the average embedding over a bag of words related to the sense — synset members, words in the sense gloss and words in example usages for the sense — and the sense closest to the word’s
embedding is taken to be its primary. Using the results for most frequent sense classification they found that they performed worse overall than a most frequent sense classifier trained on SemCor but that this was due mainly to less frequent words: on frequent words the unsupervised most frequent sense estimates were on average better than the supervised ones. Chen et al. (2015) also make use of sense glosses for the purpose of constructing sense embeddings. In this case they initialise the number of sense vectors for a word to the number of WordNet senses and initialise the vectors themselves by training embeddings that model the glosses of those senses in WordNet. The gloss-initialised embeddings are then used in the initialisation of a context clustering based sense embedding training procedure. Evaluation as a component in standard semantic similarity regression tasks shows improvement of the gloss-initialised sense embeddings over existing techniques but not under all evaluation metrics. Chen et al. (2015) hypothesise that their high level of polysemy in WordNet may be a disadvantage, given than their corpora do not attest the less common senses well.

Li and Jurafsky (2015) explore a “Chinese restaurant” method for building neural sense embeddings with a variable level of polysemy learned from corpus evidence. They focus in particular on the downstream effects of learning sense embeddings on the results in various NLP tasks including sentiment analysis, named entity recognition and part of speech tagging. In their analysis they take particular care to compare neural models with equal numbers of nodes in the base layer when mixing other signals in with their sense embedding dimensions. They find that sense embeddings improve the results in some tasks but not others. Understandably the sense embeddings do not cause a marked improvement in sentiment analysis, which answers a binary question about the general sentiment of a piece of text and is probably not often sensitive to ambiguity of single words. On the other hand, they do lead to an improvement in results for part of speech tagging and semantic similarity tasks. This study serves to highlight that the quality of learned sense representations is often subjective to the task for which they are to be used. Iacobacci et al. (2016) evaluated several strategies for use of word embeddings within the WSD task. They found that using word embeddings as features representing local context lead to substantial improvements to state of the art supervised models of WSD.
Whether for supervised learning or for evaluation of unsupervised systems, annotated corpora are essential to the study of word senses and disambiguation, and there has been interesting work recently into the annotation process itself. Sometimes in resource construction tasks professional annotators choose to assign more than one sense label to a usage. Jurgens et al. (2014) studied usage examples where professional annotators have chosen multiple senses to classify reasons for multiple assignment. The majority were cases of incomplete disambiguation due to insufficient context being included in the sample usage. However, many were found to be cases of a word performing a different semantic role for different dependants in the sentence. Jurgens et al. (2014) also examines the relationship of the multiple selected senses to each other, finding in the largest number of cases that the differences between the senses makes little difference to the meaning of the sentence — a coarser, more general sense could perhaps replace them — but also finding usages where two senses could be simultaneously true and cases where they are mutually contradictory. Where multiple humans create sense annotations for a corpus, annotator disagreement is often treated as something to be resolved by negotiation or majority vote, with measurements of disagreement used as a metric for the difficulty of the WSD task. However, Passonneau and Carpenter (2014) explore accounting for disagreement by building a joint statistical model of annotators — including individual accuracy and bias — and word sense annotations. As a result, they are able to produce a confidence level for annotations, and in some cases are able to place high confidence on annotations for which there was a low level of agreement. Jurgens et al. (2014), Passonneau and Carpenter (2014) and Lopez de Lacalle and Agirre (2015b) all look at multiple annotation of word senses from a different perspective, but taken together they highlight the ongoing problem of word usages that don’t quite line up with the assumption that they should be interpreted as having a single correct sense.

2.2.6 A review of PageRank for word sense disambiguation

The crowdsourcing strategy presented in this thesis involves collecting information from a text glossing application where human interaction with word senses naturally arises. A conse-
quence of the strategy is a lack of control over the specific documents and words which we collect information about. As such, the value of any resources or algorithms we use will be limited by their coverage or generalisability. Knowledge-based methods have a tendency to lack accuracy relative to supervised systems when it comes to WSD but have the advantage of extending coverage to all of the words in an encoded lexicon. In this thesis we focus in particular on Personalised PageRank (PPR) which has shown strong performance as a WSD method in recent years (Agirre et al. 2014). In this section we review background material and related work the Personalised PageRank algorithm. We then review its application to the task of WSD through the use of a lexical knowledge base. Finally, we compare the PPR approach other LKB methods for WSD.

The PageRank algorithm

PageRank started life as a component in the Google search engine (Brin and Page 1998). Its purpose was to improve the relevance of web pages returned in search results by using links to a page as a means of measuring the “authority” of that page. The method, detailed in Page et al. (1998), is defined in terms of assigning weight to web pages and then iteratively transferring weight along outgoing links until a steady state is found. More concretely, the algorithm terminates when on each iteration each page receives approximately the same weight from its in-links as it imparts on its out-links. The weight imparted is proportional to the node weight, so nodes with a high concentration of in-links end up with higher final weights overall, and this effect accumulates transitively. In order to prevent mass from concentrating in closed loops, a damping factor is applied to the transfer and a constant term is added to boost the weight of each page on each iteration. Although defined in terms of an iterative algorithm, Page et al. (1998) also formalise PageRank in statistical terms as the time spent on each page during a random walk. They describe the model in terms of a web surfer moving across the web by following a randomly selected link on each page visited. With the constant term added, the random walk includes a probability at each iteration of ignoring the link structure of the page and navigating directly to a randomly selected page on the web.
In the PageRank algorithm the standard choice for the constant boosting factor is to make it uniform across all pages. However, Page et al. (1998) also discuss the application of the constant term to select subsets of the pages, such as the page at the domain root, as a mechanism to mitigate abuse of the algorithm. In the random surfer conception of PageRank, this corresponds to random jumps being distributed according to the non-uniform weight assigned to pages. For example, if only domain roots receive weight, when a random surfer makes a random jump they can only land on a domain root. The idea of non-uniform constant term has carried forward into the concept of Personalised PageRank, which we will see soon has significant advantages over the standard PageRank when used with a semantic knowledge base for word sense disambiguation.

In one early application of a non-uniform constant term, Haveliwala (2002) produced rankings specialised to a set of predetermined topics. For each topic, pages related to that topic were selected to receive the constant mass. In the random walk interpretation this corresponds to the surfer making random jumps to the set of preselected topical pages, but still following links to anywhere on the web. It makes the algorithm give weight to links relevant to a search topic, whilst still getting information on page authority from the whole web hyperlink graph.

The standard PageRank algorithm uses a uniform distribution for contributing a page’s weight along its outgoing links. As with the random jump weight vector, non-uniform distributions across each page’s outgoing link structure have been investigated. Richardson and Domingos (2001) perform query-specific PageRank by weighting both the random jump term and the outgoing link weight distribution according to the target page’s relevance to a search query. They took the interesting approach of computing and storing a PageRank vector for each of approximately one hundred thousand search terms, to avoid costly computation at query time. Significant compression was achieved in storing the ranking for each term by only storing the PageRank score for pages containing the term. Time savings were also achieved in PageRank calculation by eliminating pages with a zero relevance score for the search term, since they can never receive mass. In practice this meant that only pages containing the current search term were included in the PageRank calculation. This is unfortunate as it means that authoritative survey and directory pages which happen
not to contain a specific query term are excluded from the computation and cannot convey their authority along outgoing links. It would be possible to avoid cutting out authoritative generalist pages completely by smoothing the outgoing-link distribution to assign a small non-zero minimum probability. However, given the large number of query terms for which PageRank vectors were calculated, the substantial reduction in graph size may well have been necessary to keep computation time in a feasible range. Richardson and Domingos (2001) performed user evaluations for search quality comparing their custom PageRank to a simple combination of standard PageRank with a relevance based ranking. The custom PageRank scored 20-35% better in user ratings.

Evaluating PageRank customisation in terms of user ratings has many pitfalls. For one, the reliance on human judgements makes evaluation data costly to produce and unreliable by virtue of its subjectivity. Secondly, PageRank itself was primarily designed to assess the level of deference the web community affords a page, not its relevance to a search query, so for any evaluation based on query results it must be combined with a relevance metric: the results may be significantly influenced by the choice of metric. Al-Saffar and Heileman (2007) describe an interesting way of evaluating PageRank customisation that evades these pitfalls. They evaluate how successfully a ranking has been customised by measuring the overlap of its top-ranking pages with the set of top-ranking pages from the standard PageRank. They also compare the overlap between top-ranking pages of different customisations. If the overlap is high, they conclude that customisation has been unsuccessful. They divide their experiment into two parts:

**Topic sensitive PageRank** where a substantial number of pages nominally covering a certain topic are given non-zero weights in their constant term.

**Personalised PageRank** where only a few pages receive non-zero weights in their constant term.

These may be sourced, for example, from a user’s web page bookmarks.

Note that, unlike the method of Richardson and Domingos (2001), all pages participate in both topic sensitive PageRank and Personalised PageRank. This is because the out-links retain the standard uniform weight distribution, so influential pages with a random jump constant of zero still receive —
and impart — weight, through the network. Al-Saffar and Heileman (2007) found that the damping factor was a strong determinant of the overlap in top results between the standard PageRank and custom PageRank results. For a damping factor $\alpha$ of 0.85, the overlap of the top-100 pages for a Personalised PageRank with the standard PageRank was up to 36%. For topic sensitive PageRank they found a lower level of overlap, but did find a high representation of the topic seed pages in the non-overlapping portion of the top ranked page list. Topic sensitive PageRank proved to be consistent in the sense that similar sets of seed pages produced similar final rankings. Their final conclusion that Personalised PageRank does “not perform too well in terms of the ability to discover new pages not already returned by its global variant” (Al-Saffar and Heileman (2007) p5) seems overly pessimistic — over 60% of the top 100 pages in a personalised rank are novel — but the results do establish bounds on the divergence between global PageRank and its customised variants.

**PageRank for WSD**

Mihalcea et al. (2004) make the observation that PageRank could be combined with a lexical knowledge base to find the central influential concepts for a piece of text. Their method requires a knowledge base of concepts linked by semantic relations. Given a piece of text, they identify all concepts in the lexical knowledge base linked to words in the text. They then build a PageRank graph out of those concepts, forming edges from the direct semantic relationships between them as found in the knowledge base. The PageRank result can be used to perform all words WSD by selecting the most influential sense for each word. Mihalcea et al. (2004) evaluated their PageRank WSD algorithm using WordNet as the knowledge base and both SemCor and Senseval-2 as the evaluation data. They found it performed better than its contemporary state of the art Lesk method and substantially better than the most frequent sense baseline. When extracting relations from the knowledge base to form a PageRank graph, Mihalcea et al. (2004) used two notable heuristics:

1. Where two related concepts in the knowledge base are also competing senses for a word in the text, the relation between them is ignored. This is to prevent them from mutually reinforcing each other in a closed isolated loop.
2. Synthetic edges were derived by composing the hypernym and hyponym relations to form a co-hyponym relation between siblings such as the primary senses of *dog* and *wolf*.

The inclusion of two-step edges introduces a small amount of information from the full graph of the knowledge base. However, ultimately this method limits itself almost completely to direct edges between concepts referenced in the text and excludes consideration of almost all longer paths that pass through unreferenced concepts.

Subsequently, methods were developed that build a larger knowledge base subgraph for the purposes of WSD, [Agirre and Soroa (2008)](#), for instance, find the shortest paths between all synsets of words in the text, and build a subgraph containing all the nodes and edges in those paths. [Agirre and Soroa (2009)](#), however, developed a method to make use of the full knowledge base graph to do all words WSD for a piece of text. Their answer was to use a Personalised PageRank, giving the random jump weight to only the synsets of words in the text, but leaving the full link structure of the knowledge base intact. Naturally PageRank on a whole graph is designed to identify the most globally influential nodes, so it pays to question whether a full graph Personalised PageRank will customise the ranking enough to re-order senses of a word from the global ordering. However, the results of [Richardson and Domingos (2001)](#) did show that personalisation of PageRank results in long distance re-orderings of the ranking. In fact, there was a tendency for the weighted nodes to rise into the top 100 overall, so there is reason to be confident that the discriminatory power of Personalised PageRank is high. In any case, should Personalised PageRank fail to diverge far from PageRank, the global result represents a measure of concept centrality which makes for a reasonable baseline heuristic, since the static PageRank correlates more highly with the most frequent sense baseline than Personalised PageRank does. To evaluate their Personalised PageRank method [Agirre and Soroa (2009)](#) used several versions of the WordNet knowledge base together with evaluation data from Senseval-2 and Senseval-3. When building the Personalised PageRank graph they did not only use the explicit semantic relations in WordNet but also extended relations derived from automatic and manual sense annotations on synset glosses. They did not compare the extended relation sets to Personalised PageRank over just the explicit relations so it is difficult to say exactly
what effect their inclusion had. However, sense annotations on synset glosses do represent a loose semantic relatedness between concepts so intuitively they should benefit the Personalised PageRank algorithm in terms of flowing weight to senses of words that are evoked by words in context. Unlike Mihalcea et al. (2004), Agirre and Soroa (2009) do not by default include a mechanism to prevent two related senses of a word from mutually reinforcing each other to gain dominance regardless of context. Instead they propose a strategy which they term Personalised PageRank-w2w whereby only a single word is disambiguated at a time and its senses alone are not given weight in the PageRank personalisation. In the final results, Agirre and Soroa (2009) found that their method performed below the most frequent sense baseline, which was all but universal for unsupervised methods at the time (Navigli 2009), but did perform better than previous knowledge based methods of WSD. Specifically, they found that Personalised PageRank performed as well as the subgraph PageRank of Agirre and Soroa (2008) with the advantage of not relying on a subgraph building heuristic. Personalised PageRank-w2w generally performed better, though the differences were not statistically significant. Agirre and Soroa (2009) also showed that their method was transferable to knowledge bases in other languages and made available a software package called UKB for Personalised PageRank over any WordNet based knowledge base.

**Comparison to other knowledge-based methods for WSD**

Ponzetto and Navigli (2010) contributed greatly to network methods for WSD with the construction of WordNet++, which extends the WordNet concept network using relations extracted from the collaboratively edited online encyclopedia Wikipedia. Ponzetto and Navigli (2010) unified WordNet senses with Wikipedia pages by matching web page titles to lemmas and page content to corresponding concepts. For polysemous words, a WordNet sense was selected for the Wikipedia page using a Lesk-like word overlap method. The WordNet relations between concepts are then augmented with additional relations based on the rich hyperlinking between Wikipedia pages and on other page metadata. Ponzetto and Navigli (2010) ran evaluations of some simple graph based

---

2See http://ixa2.si.ehu.es/ukb/
WSD algorithms and found that, using the enriched relations of WordNet++. The methods they used were:

**Simplified Extended Lesk** which represents each sense of a word by the bag of words in its gloss and the glosses of all concepts on hop away in the WordNet++ semantic graph. WSD is performed by maximising the overlap between the concept representation and the words in the disambiguation context.

**Degree centrality (Degree)** which extracts a subgraph from WordNet++ by the union of nodes up to a fixed distance from senses of words in the text. Degree centrality then selects senses with the maximum out-degree in the subgraph for each lemma.

For the degree centrality method, Ponzetto and Navigli (2010) found that the density of the WordNet++ relations was initially a liability, because of the existence of Wikipedia hyperlinks between weakly related concepts. We note that a Personalised PageRank algorithm should be robust to such links since it assigns importance based on a probability space of all paths and so is not overly influenced by a single short path. However, in this instance Ponzetto and Navigli (2010) mitigated the problem by filtering the relations derived from Wikipedia using a tuned cut-off parameter for a Lesk-like word overlap measure of page metadata. The final evaluation results rivalled those of supervised WSD at the time which is a significant achievement for a knowledge-based method!

Navigli and Ponzetto (2012) extend the methodology of WordNet++ to create a multilingual rich knowledge base called BabelNet. They leverage the fact that Wikipedia has editions in a great many languages and editors include links from pages to their equivalent in other languages. Although the pages are not direct translations of each other the title words may be considered to represent the same concept. In BabelNet, Wikipedia is treated as a concept graph where pages represent concepts and the page title is a word in the lexicon. Page titles in Wikipedia follow a convention whereby an ambiguous term is followed by a disambiguating term in parentheses. For example Java is used for the island in Indonesia of that name whereas Java (programming language) is used for the Object Oriented programming language. In BabelNet only the part outside the paren-
theses is used so words such as Java are polysemous, mapping to two separate concepts. Navigli and Ponzetto (2012) construct concepts in BabelNet by forming multilingual synsets containing WordNet lemmas and Wikipedia page titles from different languages. WordNet lemmas are linked to Wikipedia headwords using the method developed for WordNet++ with some improvements to the page-synset similarity measure. Multilingual Wikipedia titles are joined in by means of manual interlingual hyperlinking provided by Wikipedia’s editors. Additional multilingual members for synsets were discovered by use of automatic machine translation:

- Starting with a WordNet synset, Navigli and Ponzetto (2012) find 10 example sentences in SemCor and apply machine translation. If at least three instances translate to the same word then that word is added to the BabelNet synset.

- Similarly, starting with a Wikipedia page, they translate example sentences in Wikipedia which cross-link to that page and add the translated link text to the BabelNet synset for the page.

Thus BabelNet contains approximately 3 million synsets that originated as WordNet synsets, Wikipedia pages or both. The synsets contain words originating as WordNet lemmas and Wikipedia page titles in multiple languages, forming a total of around 26 million word-synset sense pairings. Relations are taken from the semantic relations of WordNet and from hyperlink structure and other metadata in Wikipedia and include around 350 thousand from WordNet itself, 617 thousand from sense labels on WordNet glosses, over 50 million from English Wikipedia and almost 20 million more from five other language editions of Wikipedia. Subsequently, BabelNet-2.0 has expanded to 50 languages and incorporates concepts and relations from additional resources (Ehrmann et al. 2014).

Navigli and Ponzetto (2012) use BabelNet-1 to perform WSD, evaluating several subgraph based algorithms using the SemEval-2007 coarse grained word sense disambiguation task data. Their method of extracting the subgraph is to identify all senses of words in a text context and perform a graph search out to a fixed distance $L$ from the contextual synsets. The final context subgraph consists of all nodes which lie on a path of length at most $L$ that connects two contextual synsets. However, for WSD, they filter out paths that connect senses of the same word and filter
out ‘weak’ relations according to a similar heuristic to that used by Ponzetto and Navigli (2010) for their degree centrality WSD implementation. They then perform WSD by applying several different network algorithms to the subgraph:

**degree centrality (degree)** as in Ponzetto and Navigli (2010)

**Inverse path length sum** finds all paths $p$ from a sense to other senses in the context and sums

$$\sum_p \frac{1}{e^{\text{length}(p)} - 1}$$

The highest scoring sense of each word wins.

**Path probability sum** is like the inverse path length sum but uses the same weights that filtered out ‘weak’ relations to scale path lengths. The score for each path is the product of its edge weights.

**Static PageRank** performed over the context subgraph is the final method used.

Navigli and Ponzetto (2012) compare the results of their evaluation to the same algorithms run using WordNet alone and to several supervised and unsupervised submissions to the SemEval–2007 coarse grained all words task. Their algorithms using BabelNet did substantially better than WordNet and significantly better than competing systems. Ponzetto and Navigli (2010) previously showed an f1-score of 79.4 on nouns for this task using the degree centrality method over WordNet++. Navigli and Ponzetto (2012) show that using BabelNet improves this result to 82.5. A semisupervised variation using a most frequent sense backoff with the path length method achieved their best result with an f1-score of 85.4. They also showed that a static PageRank over the subgraph performed worse than any of their other network methods at 80.7 over nouns. However, Agirre et al. (2014) subsequently showed an f1-score of 83.6 on nouns using Personalised PageRank-w2w over WordNet alone on the same dataset. They hypothesise that applying Personalised PageRank-w2w to WordNet++ would yield even better results.
Agirre et al. (2014) also include an interesting exploration of subgraph methods and Personalised PageRank methods on their WordNet based knowledge base. Using the subgraph construction of Ponzetto andNavigli (2010) and Navigli and Ponzetto (2012), they found that Personalised PageRank and Personalised PageRank-w2w over a subgraph performed better on a variety of datasets than degree centrality and a selection of other network algorithms. Interestingly they also found that global Personalised PageRank-w2w only sometimes performed better than subgraph Personalised PageRank methods; on some datasets it performed numerically worse (though not with a statistically significant difference). Putting aside the question of whether or not to extract a subgraph however, it seems that Personalised PageRank methods are superior to path methods such as degree centrality when performed on the same graph.

Moro et al. (2014) used BabelNet to create a joint solution to the tasks of WSD and entity linking which they call Babelfy3. Entity linking is the task of identifying an entry in an encyclopedic resource referenced by a potentially ambiguous term in text such as a person or place name. They use a concept network derived from BabelNet1.1.1 as a basis for a contextual-subgraph disambiguation algorithm. The concept network is a dense graph between BabelNet synsets designed to reduce the impact of missing relations and weak relations. It is built by performing a Personalised PageRank for each concept individually and linking it to the concepts whose final weight exceeds a threshold. The edge transition probabilities for the Personalised PageRank random walks are made proportional to the number of 3-cycles the edge participates in to bias the walk to remain within clusters of concepts. To achieve high coverage in arbitrary text, Moro et al. (2014) use an interesting multiword token identification algorithm. To identify BabelNet entries in text they first tokenise and part of speech tag, extracting 1- to 5-grams that contain a noun. They then retrieve all BabelNet entries which contain one of the n-grams as a substring. This process allows identification of candidate interpretations as pairings of BabelNet concepts with text fragments. For example, BabelNet’s Mario Gomez would be paired with the 1-gram Gomez in a text. Candidate interpretations are then linked in a graph using the edges between concepts in the constructed concept network. This is

3http://babelfy.org
similar to a contextual subgraph extraction method except that concepts can appear more than once if they are linked to more than one text fragment. Finally, they use a network density algorithm to score concept relevance for each text fragment. Considering the example sentence *Thomas and Mario are strikers playing in Munich*, by performing WSD and entity linking together [Moro et al. (2014)] are able to correctly link the identities of the players and the senses of *striker* and *play* via their common connections through the sport *soccer*. Due to *BabelNet's* coverage of both concepts and named entities in many languages, this algorithm can be applied to many existing task datasets for WSD and entity linking. [Moro et al. (2014)] evaluated their algorithm against the state of the art on many tasks including *SemEval-2013* multilingual WSD and *SemEval-2007* fine- and coarse-grained all words English WSD. On most tasks *Babelfy* performed better than, or had no statistically significant difference from, existing state of the art systems on the same task. This included supervised systems and, for multilingual WSD, an implementation of *Personalised PageRank* running on the complete *BabelNet1.1.1* semantic graph, which is richer in concepts and relations than *WordNet++*. On the *SemEval-2007* coarse grained task *Babelfy* achieved an *f1-score* on nouns of 84.6 coming between the previously mentioned results of *BabelNet* subgraph *degree centrality* with *most frequent sense* backoff (85.4, [Ponzetto and Navigli 2010]) and *Personalised PageRank*-*w2w* over *WordNet* (83.6, [Agirre et al. 2014]), though not with a statistically significant difference. On the entity linking task *Personalised PageRank*-**w2w** performed much worse than *Babelfy*, even on the full *BabelNet* graph.

An error analysis by [Moro et al. (2014)] comparing *Personalised PageRank*-*w2w* to *Babelfy* identified *Babelfy’s* emphasis on finding sense and entity annotations which support each other as the reason for its success. We note that, by contrast, in the output of the standard *Personalised PageRank* and *Personalised PageRank*-*w2w* methods for WSD the low ranked senses have made an equal contribution by way of random jump weighting as the high ranked senses. In essence, *Babelfy* uses the *Personalised PageRank* precalculation to isolate the contribution each potential sense makes to each other potential sense and then searches for highly cohesive subset. This is suggestive of a more general strategy: given the isolated *Personalised PageRank* for each concept in context,
find a subset of those concepts which produce an ‘optimal’ full-context *Personalised PageRank*. The definition of ‘optimal’ could be an interesting topic for further research.

### 2.3 Multiword expressions

In Section 2.1.3 we introduced multiword expressions as lexicalised expressions comprising multiple shorter words. In this thesis, our treatment of MWEs is focused on identification of usages in text of MWE types found encoded in a variety of lexical resources. As such we implicitly assume that the placement of the expression in the resource, generally by a human editor, can be taken as a judgement that the expression has been lexicalised. In this section we review in more detail corpus and lexical knowledge base studies of MWEs including identification of types and tokens.

#### 2.3.1 Type extraction

A longstanding problem being tackled in computational MWE research has been the identification of expression types from corpus evidence. If an expression has indeed been lexicalised or has some kind of idiosyncratic behaviour then it should be distinguishable in its usages from ordinary compositional language. A large variety of methods and information sources have been explored for the identification of MWE types, a few of which were already discussed in Section 2.1.3. In this section we give a few recent examples.

Motivated by the need for construction of computational lexicons of MWEs, Tsvetkov and Wintner (2014) pull together a variety of linguistically motivated features of two word sequences to build a probabilistic graphical model of MWEs and their usage distributions. They fit the model to a reference lexicon of MWE and non-MWE bigrams built with an unsupervised method using evidence from word alignment successes and failures in a small bilingual corpus. The model is a *Bayesian network* of variables describing features aggregated over instances of a bigram in a monolingual corpus. Specifically, Tsvetkov and Wintner (2014) define a number of distributional
lexico-syntactic and context-semantic features such as the shape of the sorted histogram of inflection
patterns seen on a candidate MWE’s usages and a similar representation of the diversity of words
seen in the local context. The Bayesian network is then defined with the MWE/non-MWE status
label as an underlying variable for all features, but with additional hand crafted dependencies be-
tween specific feature pairs. The results of this method were compared via 10-fold cross validation
to the results of using a SVM model and to the results of using a Bayesian network with a learned
rather than hand-crafted structure. The model with the manual structure performed significantly and
substantially better than the other methods, managing to distinguish MWE types from non-MWE
bigrams with an accuracy of around 80%. It should be noted that this accuracy is relative to positive
and negative test labels generated using an unsupervised method from bilingual aligned corpora. It
would be interesting to see the accuracy on a true gold standard of positive and negative examples
of MWEs.

Riedl and Biemann (2015) take a purely distributional semantic approach to MWE type
extraction. They start with a distributional thesaurus — a mapping from \( n \)-grams to the 200 most
similar \( m \)-grams (of mixed lengths) as measured by the distribution of adjacent words. MWE
candidates are then scored according to the proportion of similar \( m \)-grams that are single words.
The idea is that expressions representing a single concept will be more similar to single words and
that compositional expressions will be more similar to longer \( n \)-grams. A second measure is used
to penalise incomplete expressions: after building a shortlist of words found adjacent to a MWE
candidate, the proportion of tokens taken up by the most common adjacent word is computed.
If that proportion is high, it is an indication that the candidate forms part of a longer expression
comprising the additional word. These two measures are combined into a ranking function, and the
ranks are scored using average precision against a gold standard list of MWEs annotated in labelled
corpora. The method was tested using a POS based MWE candidate filter and without. With POS
filtering the performance on a small test corpus was very high, with average precision over 90%.
The method didn’t fare as well without POS filtering or on the larger corpora, but in all cases over
50% (and in a majority over 60%) of the top 500 ranked MWE candidates were deemed correct.
Lossio-Ventura et al. (2014) stack two different ranking methods for MWE term extraction, in this case for discovery of multiwords specific to the medical domain. Candidates are extracted using a POS tagger and a set list of POS patterns. They then parse the head MWEs in a medical domain dictionary to learn the 200 most common syntactic structures for terms in the domain. With these they associate a probability with candidate MWEs according to how often they match the expected structures in text. They additionally use a measure called the C-value for the candidate MWE (Frantzi and Ananiadou 1999), which is proportional to its frequency but discounted by the mean frequency of larger terms in which it is contained, and \( \text{tf-idf} \) as a measure of distinctiveness. The first ranking is based on the product of these three measures. A re-ranking is then applied after taking the top-\( k \) terms. The re-ranking is based on a term co-occurrence graph where edges are made between terms when their mutual Dice co-efficient exceeds a minimum threshold. Terms are re-ranked to higher positions if they have fewer terms within a distance of two edges in the graph (technically, all neighbours’ neighbours are counted, including duplicates). The intuition is that terms co-occurring with too many other terms are too general to be considered domain specific. As such, this measure would not necessarily apply if extracting MWEs for a general purpose lexicon. The first, more general ranking was fairly successful, with a precision over 80% on the top 100 results and an improvement of over 10 percentage points over comparable baselines. The graph based re-ranking also proved successful, though requires tuning of the cut-off for the Dice co-efficient and is specialised to the domain-specific gold standard.

2.3.2 Compositionality of multiword expression types

Although we define a MWE as a lexicalised unit, it is not a contradiction to describe some MWE types as having compositional semantics: an expression may be lexico-syntactically rigid while still having a meaning that can be described in terms of its constituents literal (or, under some definitions, figurative) meanings. Reddy et al. (2011) and Cordeiro et al. (2016) study the properties of crowdsourced numerical ratings of compositionality of MWEs. Both found that the mean compositionality scores given by human annotators to the selected expressions were fairly evenly
distributed across the available range, indicating that MWEs do exist on a continuum of compositionality or literality. Reddy et al. (2011) explore two models of measuring compositionality from co-occurrence distributions:

- calculating the similarity between each constituent’s distribution and the expression’s distribution to represent the constituent’s literality, then composing the constituent’s literality figures to predict the expression literality.
- composing the co-occurrence distributions of the constituents and comparing the composition to the expression’s actual distribution.

A number of composition functions were tried, but linear composition of the co-occurrence distributions was found to best approximate the human compositionality judgements.

In a recent approach to distributional modelling of MWE compositionality, Yazdani et al. (2015) explore composition functions for vector space models of lexical semantics and then measure MWE compositionality via their conformance with the models. They use a neural word embedding representation of both words and multiwords (trained as if they were single tokens) and fit a variety of regression models to learn a composition function between the vectors of two constituents and the vector of the MWE. The composition functions include linear projection (that is, a parameter per input-output dimension pair rather than the weighted vector sum of Reddy et al. (2011)) and quadratic projection, both with and without regularisation. They also trained a neural network composition model with one hidden layer. Predicting compositionality of specific MWEs by model error, the regularised polynomial regressions performed best, however almost all the learned models performed substantially better than a vector sum benchmark. A small increase in performance is then achieved by training models with parameters for composition and subsequent decomposition back into constituent embedding vectors. Finally, they repeated the model training with an iterative algorithm that predicts a fixed number of non-compositional MWEs after each regression and leaves them out of the next regression iteration. The idea is that non-compositional MWEs are not appropriate for use in training a model of compositionality. Again, the results improve somewhat
A variety of methods have been developed for prediction of compositionality of MWE types. Salehi and Cook (2013) describe a method for measuring compositionality of MWEs by examining translational equivalents in the large phrase translation inventory PanLex (Baldwin et al. 2010; Kamholz et al. 2014). Their basic idea is that MWEs translated word-for-word are more likely to be semantically compositional. They compare translation entries for MWEs as a whole to translations of the individual words, accounting for morphological variation by using a variety of string difference metrics rather than strict word equality. They found that difference metrics which boost longer contiguous matches were more effective than character alignment metrics such as Levenshtein difference. The overall extraction efficacy was competitive with and complementary to distributional techniques.

In the crowdsourcing study mentioned at the beginning of this section, Cordeiro et al. (2016) performed a detailed analysis of inter-annotator variation. They found that the quality of annotations is best improved by filtering out outlier annotations made by the annotators rather than annotators wholly. Additionally they found that most outlier annotations appear for MWEs in the middle of the compositionality scale (accounting for the ends of the scale not permitting a two-tailed distribution) suggesting that although a spectrum of compositionality exists, the MWE types that are not wholly literal or figurative are difficult for humans to rate consistently.

2.3.3 Token identification

Having identified MWE types we can start to find instances of them in text. As an intermediate step we can look for what we call an MWE token candidate. This is a sentence in which the constituent words of the MWE appear in a syntactic arrangement consistent with its type. For example, for the expression kick the bucket, he kicked the bucket is a candidate but the bucket kicked him is not. It is not uncommon to simplify the identification of candidates by looking only for unbroken sequences of words: for example, Tsvetkov and Wintner (2014) considered only instances of contiguous word bigrams for MWE identification. However, many studies seek candidate MWE
tokens with intervening non-constituents and morpho-syntactic inflections. In their search for MWE tokens with intervening words, Morimoto et al. (2016) approach the problem of candidate identification by inferring a canonical syntactic structure for each MWE from its fully contiguous instances. The canonical structure may include non-MWE constituents when a MWE always appears with an argument. Such non-constituent branches of a parse are represented by the POS of their headword such as NN in the expression a number of NN. Morimoto et al. (2016) then consider any occurrence of the constituents of the MWE where the dependency parse yields the same relationships between constituents as the canonical structure to be a token of the MWE.

For many MWE types such a candidate is unambiguously a token. For example, the expression drive (someone) up the wall only ever means to agitate a person: in practice it does not mean to physically transport someone up a wall. A type classification process may already have told us that this is the case: that all instances of these words in this syntactic configuration have a certain unambiguous meaning. However, some MWE types can be said to be ambiguous in that their token candidates are not always instances of the MWE. For example, he kicked the bucket of water is not an example of the the idiom kick the bucket meaning “to die”, it is a literal sentence which happens to share the same words and their syntactic form. In these cases a disambiguation step is required to validate the token identification.

**Disambiguation for token candidates**

MWE token candidate disambiguation starts with a sentence containing the words of an ambiguous MWE in their expected syntactic configuration and asks the question: is this a token of the MWE type, or are these words simply playing their ordinary literal roles in the sentence? In this section we will look at the data and methods used to solve this task.

The OpenMWE corpus is a collection of annotated token candidates of ambiguous MWE types (Hashimoto and Kawahara 2009). To our knowledge it is by far the largest freely available gold-standard corpus of ambiguous MWE types in any language, comprising 146 MWE type with 102,856 annotated MWE token candidates in total. More detail about the OpenMWE corpus appears
in Section 3.4.1.

Apart from the enormous task of constructing the *OpenMWE* corpus, Hashimoto and Kawahara (2009) also used it to perform some experiments with supervised MWE token classification. Noting similarities between this task and WSD, they employed the most effective WSD features and machine learning algorithm surveyed by Lee and Ng (2002). They also included linguistic features explored by Hashimoto et al. (2006) designed to capture the relative fixedness of Japanese idioms, which we will refer to as idiom features. The machine learning algorithm used was Support Vector Machines and models were trained on the WSD features with various combinations of the idiom features. Classifiers were trained for each of the 90 MWE types which were deemed to have sufficient idiomatic and literal examples in the corpus. The model trained on WSD features was found to improve greatly on the type level first sense baseline, with some additional performance added by one of the idiom features.

Li and Sporleder (2010) conducted a thorough investigation of features used for supervised MWE token identification. Context features similar to the WSD features of Hashimoto and Kawahara (2009) were used, as were a number of linguistically motivated features. Interestingly, classifiers were trained on candidates for a subset of the MWE types in the corpus and then tested on candidates for types that did not appear in the training set. The motivation for this was a low number of instances available for each MWE type: per-type classifiers would not have had access to enough training instances to sample the feature distribution sufficiently for the supervised method to work. However, in Chapter 5 we examine in more depth the application of a cross-type partitioning to evaluate the ability of a MWE semantic disambiguation algorithm to operate on previously unseen expressions. Unfortunately many of the features were still too sparse to have a significant effect and only the collocational context features produced significant results. This may have been due to the relatively small size of the corpus used, which comprised around 4000 MWE tokens across 13 MWE types. In our experiments on the *OpenMWE* corpus in Chapter 5 the distributional context features still dominate, but the effects of idiom based features can be seen.

Diab and Bhutada (2009) described a novel supervised MWE token identification system
based on a sequence labelling model. Unlike the methods discussed so far, their model also learns to identify token candidates in the text as part of the classification process. Like Li and Sporleder (2010) the size of the corpus is small (2500 MWE tokens of 53 MWE types), and classifiers were trained on collections of MWE types. Anecdotally, the classifiers were able to pick some MWEs out of running text without even knowing their constituents beforehand, however their performance at this was not tested. A major finding of Diab and Bhutada (2009) was that reducing the feature space of context features by replacing word lemmas with their named-entity category had a significant positive effect on classification performance, a finding that is consolidated by Diab and Krishna (2009).

Fazly et al. (2009) observed that idiomatic uses of MWEs tend to occur in one of a small set of canonical morpho-syntactic forms and developed an unsupervised method for learning these canonical forms, based on a set of linguistically-motivated features which built on the work of Cook et al. (2007). They applied the learned set of canonical forms to the task of MWE token identification with remarkable success. The identification of canonical syntactic and morphological forms for MWEs has been studied in a variety of settings (Shudo et al. 2011; Al-Sabbagh et al. 2014; Morimoto et al. 2016). Fazly et al. (2009) showed that such information can be used to classify ambiguous MWE token candidates. Nasr et al. (2015) takes the interesting approach of solving MWE token identification as part of dependency parsing itself by introducing a dependency relation between constituents, MORPH, which is named after morpheme constituency due to the MWE conceptually representing a single lexical unit. The study covers two specific functional French multiwords — a conjugation and a determiner — that can be ambiguous with uses of the constituents together in their usual functions. They test a modification of the parser dictionary to mark verbs with a feature representing whether they take one of the constituents as a dependent. The test set is small, but their parsers achieve 100% recall of the MORPH dependency and 70-80% precision. Interestingly the syntactic argument features are not helpful, resulting in a decrease in recall. Overall the parser was reasonably successful at delineating the MWEs tested, though it is not clear the approach would generalise to expressions that are not as intrinsically bound to a syntactic function.
as conjugations and determiners. In Chapter 5 we compare in detail information sources on MWE morpho-syntactic flexibility with information gained from distributional representations of context semantics.

Like Diab and Bhutada (2009), Riedl and Biemann (2015) use a sequence labelling model for joint recognition and classification of MWEs tokens and named entities. They compare the use of different MWE lexicons including a variety of manually developed lists and a selection developed automatically by taking a range of cut-off ranks from two different ranking methods for MWE extraction. Each MWE token was classified into one of several different kinds including Noun-Noun compounds and idiomatic expressions. In their results they found that for the most frequent class of MWEs, Noun-Noun compounds, the manually generated resources resulted in better identification performance. However, on the less frequent classes of MWE tested the automatically constructed lexicon results in the best performance on at least one of the rank cut-off levels.

Schneider and Smith (2015) introduce a task and gold standard corpus for identifying MWE tokens in text and labelling them with supersenses: the set of top level hypernyms in the WordNet concept hierarchy. Thus this task goes some way further than the MWE candidate disambiguation task we have discussed so far (such as that explored by Hashimoto and Kawahara (2009)) where the only distinction drawn is whether or not the MWE semantics apply. Instead, MWE tokens are both identified and have a (very general) meaning label assigned to them from the same inventory as the rest of the lexicon. The argument for doing this is that for expressions such as low brow meaning not intellectual an annotation of brow as a body part would be inappropriate (and likely unhelpful to most downstream applications). To construct the corpus they took a corpus previously annotated for multiword expressions and had annotators label each token (including MWE and single word tokens) with the WordNet supersenses. Schneider and Smith (2015) run supervised MWE token identification and supersense tagging experiments on their dataset using MWE identification features from the literature and features identifying known supersenses for a token and for certain neighbours in its local context. They find that the inclusion of supersense features on top of MWE identification features improves MWE token identification $f_1$-score by a small margin and
supersense tagging by a substantial amount.

Gharbieh et al. (2016) use a neural word embedding representation for disambiguation of idiomatic Verb-Noun compound token candidates. They use a vector averaging composition function to represent an expression in terms of its constituents’ embeddings and a similar composition to represent the local context of each (potentially separated) constituents in a candidate usage of the expression. These two averaged vectors are then combined by subtracting the MWE type vector from the context vector to form an instance feature set used in a supervised classification algorithm. One additional feature dimension is appended: a Boolean feature indicating whether the MWE appears in its canonical form as computed by Fazly et al. (2009). Thus the classifier has two kinds of information: a feature representing lexico-syntactic variation of the expression and a vector representing the difference between a compositional reading of the MWE and the context semantics. The supervised method is evaluated on each MWE type by using $n$-fold cross validation where $n$ is the number of instances of the MWE type in the corpus (that is, a leave-one-out evaluation). A number of different word embedding parametrisations are tested and interestingly the one trained with the smallest context window — of just one word — lead to the best performance. This is attributed to a hypothesis that the immediate lexico-syntactic relations of a MWE token candidate are more important than a model of the local distributional semantics for disambiguation. All configurations easily outperform the supervised and unsupervised benchmarks set by Fazly et al. (2009). The method’s performance on unseen types is also tested by cross-validation with one type left out at a time. In this case the performance exceeds the majority class baseline but does not exceed the performance of the unsupervised benchmark. Finally, Gharbieh et al. (2016) also test an unsupervised algorithm in which the embedding feature set is clustered and the clusters are each assigned a label of idiomatic or literal based on the majority value of the lexico-syntactic feature for the cluster. Varying the number of clusters they find that the method is able to achieve similar results to the unsupervised benchmark for the highest number of clusters only. This makes sense since the method must assign a single label to all instances in a cluster whereas the benchmark labels each instance individually. However, Gharbieh et al. (2016) did find that a perfect labelling of clusters by
an oracle annotator would achieve a much higher performance, so there is room for improvement by applying majority sense learning techniques to clusters of word embedding features for MWE token candidate disambiguation.

In Chapter 5 we identify and formalise a number of subcategories of token disambiguation including methods such as those used by Gharbieh et al. (2016) for testing the performance of MWE token disambiguation models on unseen types.

**Relation to Word Sense Disambiguation**

MWE token candidate disambiguation can be approached as if it were a word sense disambiguation (WSD) task where the MWE types correspond to word types and MWE token candidates to word tokens (Hashimoto and Kawahara 2009). In this conception, the analogue of word senses are the idiomatic and literal classes for an MWE type.

In WSD, supervised methods are by far the most successful but large amounts of data are required for each word type to be disambiguated (Navigli 2009). However, unlike in WSD, we can expect to find some linguistic commonality between the idiomatic senses of distinct MWE types. This leads us to hope for more general cross-type classification algorithms, which might alleviate the knowledge acquisition bottleneck. In this paper we will refer to WSD (more specifically “word expert”) style classification of ambiguous MWE tokens as type-specialised classification.

The first sense or majority class baseline, in which a word is always labelled with its predominant sense, is known to perform very well at WSD due to a Zipfian distribution of senses (Preiss et al. 2009). We expect no different for the MWE token disambiguation task where the two-class problem is virtually guaranteed a majority class baseline accuracy of over 50%. There has been work in unsupervised first sense learning for WSD using lexical resources (McCarthy et al. 2007), but the de facto baseline in WSD is a supervised first sense baseline (Navigli 2009). We make use of a type-specialised baseline, which is a supervised majority class baseline modelled on the WSD first sense baseline. We also introduce a corpus baseline which, calculated based on idiomatic and literal counts of a collection of MWE types, is the type-specialised baseline’s cross-type analogue.
2.4 Human computation

The main aim of this thesis is to acquire an electronic record of knowledge about word senses from the activities of users of an electronic dictionary. The term Human Computation was coined by von Ahn (2005) to describe a family of applications in which humans perform part of a computation that computers are as-yet unable to perform. An example is the ESP game — so named for the concept of Extra Sensory Perception — wherein two isolated users are shown the same image and invited to guess which words the other player will use to describe the image. Points are awarded as soon both players have entered one word in common, which incentivises quick and relevant descriptions. Although players participate for enjoyment of the game, they are simultaneously performing a difficult computation task: assigning descriptive text labels to images which are suitable for search indexing. The ESP game serves as a prime example for a large family of applications involving human computation to support research. In particular, note the following two factors:

1. Users are self-motivated to participate (for entertainment) rather than volunteering as research subjects.

2. Human computation (image labelling) occurs as a side-effect of the users pursuing their own goals (scoring points for ESP).

Many subsequent projects have pursued the same strategy of gamification for human computation. Foldit is an online multiplayer game in which players leverage human visual problem solving and their ability to intelligently adapt search strategies to find ways to fold proteins into low energy forms (Cooper et al. 2010). The game was designed to attract players with several factors:

- In game score.
- Persistent leaderboards.
- Social interaction in forums.
• Contribution to science.

In addition, players reported being motivated by compelling gameplay. The Foldit player base have been successful at previously intractable steps in retroviral research (Khatib et al. 2011). In a more recent example, Lieberoth et al. (2014) designed the game Quantum Moves to leverage human problem solving for computing efficient plans for atom displacement in quantum computer construction. They paid particular care to designing the game’s difficulty gradient and incentive structure to build a dedicated and skilled user base, which was required for the computation goals of the project to be achievable. In fact, designing a gaming incentive structure suitable for a specific human computation task is a real challenge for the methodology (Choi et al. 2014).

Gamification is not the only means available to leverage human computation; it fits into a larger family of strategies called crowdsourcing, which started off not strictly as a source of computation but as a source of commercial labour likened to the practice of outsourcing (Howe 2006). It has since been adopted by research communities as a new research methodology (Ranard et al. 2014). A popular platform for research as been Amazon’s Mechanical Turk service which provides a marketplace for crowdsourcing where workers are incentivised by payment for tasks completed (Chen et al. 2011). Crowdsourcing has been used extensively in natural language processing research in recent years. Snow et al. (2008) evaluated performance of Mechanical Turk annotators relative to professional annotators on a variety of resource production tasks including emotion labelling, semantic similarity, entailment, temporal relation extraction and word sense disambiguation. The results for WSD are particularly interesting: using a majority of 10 vote the Mechanical Turk annotators collectively achieved an accuracy of 99.4%, until it was determined that it was actually the gold standard annotations that were in error and the result bumped up to 100%. Gamification has been applied to resource generation for a variety of NLP tasks, including, for example, the anaphora resolution game Phrase Detectives (Poesio et al. 2013).
2.4.1 Crowdsourcing lexical semantic annotations

Venhuizen et al. (2013) use a gamification methodology for the construction of resources for word sense disambiguation. Their application, Wordrobe, has modes of operation for a variety of annotation tasks. The WSD task presents a sample sentence with a highlighted word as the question, and lists sense glosses as choices in a multiple choice question system. Users are invited to select the sense that they think the most other users will select, and place a bet. The higher the bet, the higher the reward if correct and the higher the penalty for failure. In this way, the application collects information both about user intuition for word senses and for their confidence levels. The game design is very close to a traditional word sense labelling method but with points awarded for agreement with other annotators. Annotation quality was evaluated relative to a gold standard produced by professional annotators. Taking the sense selected by the most players for each target word and leaving ties unannotated achieved an f1-score of 85.7%. Taking only senses where players were unanimous in their choice achieved a precision of 97.5% with a recall of 34.7%. Although these results are not as high in accuracy as those of Snow et al. (2008), they do suggest that gamification can produce a reliable sparse gold standard for WSD. Additionally, although the higher coverage relative majority annotations may not be sufficiently accurate to count as a gold standard, they may still be useful for uses that involve aggregation of statistics over a corpus.

Seemakurty et al. (2010) also target gamification methods at WSD, but consider multiple choice from a closed set of dictionary glosses to be flawed as a gameplay mechanic. Specifically, they contend that having to read and understand many potentially long and subtly different dictionary senses is tedious and demotivates players. Additionally, they are concerned that offering a multiple choice answer entices players to simply guess rather than go to the trouble of comparing each sense carefully to a context to make their decision. Instead, they design the game as a lexical substitution task: players are shown a sentence and a target word, and are asked to enter potential alternatives to substitute. As with the ESP game, players are awarded points for typing the same word as an anonymous partner, which incentivises entering words that are good semantic substitutes.
for the original. In 54% of cases the lexical substitution selected by both players uniquely matched a single synset of the target word. The rest of the cases either matched multiple synsets (11%) or were not an exact match to any synset of the target word in the lexicon. Regardless of the direct utility to WSD, lexical substitutions themselves are a valuable resource (Kremer et al. 2014).

Parasca et al. (2016) created a novel game for eliciting definitions of words from players. They contend that representing lexical semantics by definitions has the capacity to preserve information lost in the distributional representations commonly used in lexical semantic research. The game, Word Sheriff, assigns one player the role of narrator and gives them a word to communicate to the other players. The narrator and the players then take turns with the narrator giving a single word clue and the players replying with one guess each. The scoring incentivises narrators to describe the target word in as few descriptive words as possible. The result does not resemble a dictionary entry per-se, but is still an organically generated description of the word. Parasca et al. (2016) analysed the strategies that narrators used and highlighted some common themes that might not be well captured by distributional representations of words: hypernymy, antonymy and world knowledge.

Jurgens and Navigli (2014) aim for a more radical approach than simply gamifying traditional annotation tasks, instead designing graphical games in a traditional computer game genre and building crowdsourcing features into them. The first is a car racing game called Puzzle Racer in which users must attempt to drive through gates decorated with images according to a theme described at the start of the race. The research data produced by Puzzle Racer is a set of links of word senses to images. The incentive to drive through images that match the theme comes from booby trapped distractor images inserted into the game that are known not to be related ahead of time. The players are given no indication which images are distractors and which are candidates for a target word sense, so to perform well at the game they must select the best image to describe the sense. Distractor images, along with a designated correct alternative to go alongside them, must be populated with gold standard images for word senses designated by professional human annotators ahead of time. However, new distractors can later be drawn from the results of gameplay. To attract
a player base, Jurgens and Navigli (2014) advertised a contest with a modest monetary prize by email to university students. They received 126 entrants who completed 7199 races. Professional annotators compared the results of the gamified annotation to the results of an equivalent multiple choice question task crowdsourcing platform CrowdFlower, finding the top three images from each source to be equally appropriate on a substantial majority of senses. In the cases where Puzzle Racer and CrowdFlower disagreed the gamified images were sometimes preferred, but the CrowdFlower images were preferred more often. On the other hand, taking into account the prize money, the gamified annotations worked out 27% cheaper to produce.

The second game tested by Jurgens and Navigli (2014), Ka-boom!, is designed to annotate a selected word in a text sentence with one of its senses. Senses are represented in the game by images, selected from a sense tagged image collection such as is generated by gameplay in Puzzle Racer. In Ka-boom!, players are shown a sample sentence with a target word highlighted. The game is one of speed, reflexes and judgement: images are thrown across the screen and those unrelated to the meaning of the word in the sample sentence must be targeted and destroyed by the player. The game throws in images known to represent senses of the word and additional images known not to represent the word at all. The player is penalised for missing the additional, known-unrelated, images that have been mixed in and the researchers infer the sense of the word from the word sense images the player does not destroy. The WSD annotation performance for players of Ka-boom! was compared to the results of the most similar gamified word sense annotation project Wordrobe (Venhuizen et al. 2013) and outperformed it significantly in f1-score.

In Chapter 4 we outline the design of a software application Wakaran for crowdsourced sense annotation which attracts users by filling a use case in second language learning and explore the extent to which users will organically engage with its features. Then, in Chapter 7 we wrap up the thesis by exploring methods to leverage the data collected from user interactions with Wakaran for NLP research, and propose a means by which the research and the application usage can improve each other symbiotically. In this way, our methodology promises to provide ongoing value to attract participants without need for monetary prizes or to compete for attention in gamers’ leisure time.
with the video games industry, which had estimated yearly revenues of $67 billion as of 2012 (Marchand and Hennig-Thurau 2013). In particular, the potential for research results to feed back into and improve the incentive mechanisms of the application give it the potential to stand out from the crowd of applications filling a similar space.

2.4.2 Quality control in crowdsourcing

One of the promises of crowdsourcing is that it can produce work at a reduced cost compared to contracting professional services, but the tradeoff is that the work may not be performed at the same standard by non-professionals. To deal with this, crowdsourcing projects have an increased dependency on quality control for the work produced. However, quality control comes with costs of its own. Iren and Bilgen (2014) as part of developing a formula for estimating the costs of ensuring quality in variety of crowdsourcing strategies, highlight two tradeoffs with respect to cost and quality:

1. the cost savings of crowdsourcing work versus the cost of ensuring quality; and

2. the cost of ensuring quality — “conformance costs” — versus the costs of poor quality — “non-conformance costs”.

To manage these tradeoffs careful selection of quality control methods is required. It should be noted that quality is not a universal property, but rather a suitability to the requirements of a project (Kern 2014). In this section we review broad strategies for quality control in crowdsourcing and their relationship to the quality requirements of our crowdsourcing project. The crowdsourcing system produced for this thesis is the Wakaran Glosser, which is designed to be useful to students of Japanese as a second language while engaged in reading comprehension tasks in class. It is described in detail in Chapter 4. Here, we review factors in crowdsourcing that affect quality and the methods available to achieve quality, and discuss each with relevance to two goals of our research:

1. Successful collection of a record of word translation glosses users of Wakaran have selected to assist in understanding L2 texts.
2. A proposal on the basis of our research for the appropriate forms of quality control for learning sense labels from selected glosses in future work.

The primary output of the crowdsourcing project in this thesis is a record of naturally arising interactions with word senses. Thus conformance with quality goals is dependent on:

1. attracting users into the crowdsourcing system
2. the system being used for reading comprehension tasks; and
3. the system successfully making a record of senses used versus senses rejected.

The cost of conformance is the cost of the steps we take to ensure these goals are met and to verify that the goals are met. The main costs lie in the effort of developing and evaluating NLP subsystems for the glosser (Chapter 5 and Chapter 6), careful design of the glosser and detailed monitoring of the usage of the glosser (Chapter 4). We explore how those activities fit into the bigger picture of quality control in crowdsourcing in the remainder of this section. Non-conformance means:

1. not attracting users, or
2. attracting users who only look up one word at a time, or
3. attracting users who read but don’t interact with word sense interface elements, or
4. collecting only sense interactions that do not imply restricted interpretation of sense.

The outcome of non-conformance could be either of two things:

1. a negative result for the viability of the crowdsourcing method; or
2. a false negative result, in which the quality of the implementation lets down the method.

Fundamentally the cost of Item 1 is zero — a negative result is still a result — but the cost of Item 2 is a failure of the project. The broader view is to learn how we might develop a high quality sense labelled corpus from this natural interaction and we make recommendations in Chapter 7.
based on the results of this thesis. In this review we survey general principles and methods of quality control in crowdsourcing as they pertain to both of these quality goals.

There are two main areas to look at when deciding on quality control for a crowdsourcing project: task factors and worker factors (Allahbakhsh et al. 2013). The properties of the task include the kind of work it involves but also the infrastructure built around it. In general the nature of the work may not have much room to change for quality control purposes but it has an impact on quality control decisions, particularly on leveraging the crowd to verify the output quality. Infrastructure, including task description and tooling, incurs an up-front cost expended to prevent non-conformance with quality (Iren and Bilgen 2014). Worker factors include the level of qualification of workers and their motivation for participating.

The main impact task work has on quality control is the nature of the outputs produced by workers. Iren and Bilgen (2014) classify tasks as either finite or non-finite in response type. For example, a finite task may involve giving a yes or no answer, or selecting from multiple possibilities, whereas writing free form text according to certain requirements may have effectively non-finite valid responses. Iren and Bilgen (2014) also talks about tasks in terms of subjective and objective correctness, where a subjective response may be a personal opinion or perspective or have culturally varying relevance. Kern (2014) takes a slightly different approach, classifying tasks into deterministic — where there is one correct answer — and non-deterministic — where parallel or repeated performances of the task may produce different but acceptable outcomes. Finite objective tasks should in general be deterministic and admit to certain kinds of quality control such as gold standard validation. Other task kinds however may need structured crowdsourced methods for validation, with the relevance of the validation method depending very much on the task type — that is, on the kinds of responses workers give. For our crowdsourcing project the selection of relevant glosses is a finite but potentially somewhat subjective task. We expect it to trend closer to the deterministic side of the spectrum. The collection of a gold standard corpus of sense labels however is a more strictly deterministic task and we make findings in Chapter 4 on how to proceed from the former to the latter.
Design of the infrastructure and delivery of our crowdsourcing task is a major focus of this thesis. Allahbakhsh et al. (2013) identify four major factors in the design of a crowdsourcing task: definition, user interface, granularity and compensation policy. Kern (2014) similarly discusses “application characteristics” specific to a work task and “platform characteristics” specific to a crowdsourcing service, but treats compensation policy as a workforce characteristic. The task definition (Allahbakhsh et al. 2013) or application characteristics (Kern 2014) include how the task flow is designed, how the task’s requirements are communicated to workers and the setting of eligibility requirements for workers. Our crowdsourcing project has a novel set-up in this respect, since we do not seek to set a task ourselves but rather to capture an audience performing an existing task. In Chapter 7 we will discuss how worker eligibility could come into play in future work with our crowdsourcing application. The user interface — “ergonomics” in the case of Kern (2014) — is important for two reasons: firstly ease of use affects retention of users, and secondly it impacts the difficulty gap between performing the task as intended and giving thoughtless responses. In Chapter 4 we describe how we designed our glossing system defensively to mitigate the risk of user interface issues interfering with the goals of the project. Finally, granularity refers to whether a decomposable task is given as a monolithic one to a single user. In the case of our project the users choose the granularity of their own tasks because Wakaran is designed to accept user-submitted documents. In Chapter 7 we see that they make a variety of choices regarding the size of portions of a document to submit to our system for reading.

The other major category of factors affecting quality is worker characteristics. Iren and Bilgen (2014) look at organisation-internal versus external crowds where the difference is in how much access and influence the task setters have. The purpose of our method design is to make external crowds accessible but for this project we have bypassed the advertising costs of acquiring an external crowd by starting with classes at our own institution to whom we have a more direct access. However, as described in Chapter 4 beyond advertising the existence of our platform itself, we otherwise conducted our study as if the glosser’s users were an external crowd. This implies less control over working times and hours, but that is already an intrinsic part of our strategy of mea-
suring naturally occurring activity. Allahbakhsh et al. (2013) discusses a worker’s profile in terms of reputation and expertise. Reputation concerns a worker’s history on a generic crowdsourcing platform but could apply to our special purpose system one day depending on the nature of future projects. Expertise concerns the credentials and experience that qualify a worker for the particular task. The advertising process for our project selects for students of Japanese at an intermediate level, but for a wider distribution a quality control method will need to account for expertise. Kern (2014) also discusses the importance of worker qualification for a crowdsourcing task, including familiarity with the subject matter and familiarity with the tools through which the crowdsourcing task is being run. Finally, the compensation policy can also affect quality outcomes (Allahbakhsh et al. 2013). It has generally been found that offering a monetary reward does not necessarily increase quality but may increase the speed at which results can be collected by virtue of increasing the pool of workers willing to participate, whereas intrinsic motivation factors are more likely to increase quality (Kern 2014). In our crowdsourcing task the motivation is entirely intrinsic, since the users are performing the task for their own needs. For future work we could seek to amplify the effect by emphasising the contribution to research that participation entails.

The project characteristics discussed so far affect decisions on what quality control methods to employ, which can be broadly broken down into design time methods and runtime methods (Allahbakhsh et al. 2013). Defensive design should be employed to reduce or mitigate any major risks from task design factors such as ergonomics and task description. In our project this means ensuring that the glosser user interface is designed to support its main use case for students and that its interactive features collect the information we need. Chapter 4 covers our system design in detail. Additionally, Chapter 5 and Chapter 6 develop and assess the reliability of NLP methods which is an ergonomic consideration essential for trust in a glossing application (Nerbonne et al. 1998). The worker selection method is also important. Allahbakhsh et al. (2013) list three main strategies:

- Open to all
- Reputation based
Chapter 2: Literature Review

- Credential based

Our project is designed to be compatible with an open to all strategy. That is, the system is specifically designed as an open web application so that it may attract users from anywhere in the world. A credential based approach would typically involve requiring certain qualifications or administering a qualification test \cite{Kern2014}. In this thesis we have trialled our system by advertising it to intermediate students of Japanese as a second language which contains an implicit qualification test. We discuss how a more explicit test may be administered to an open field of users in Chapter 7 when we make recommendations supporting the future of our system. Reputation based methods reflect on past worker performance in the crowdsourcing system and do not apply at this stage in our thesis, but may come to apply in its maturity.

The most widely deployed family of runtime methods is output-based quality control \cite{Kern2014}. In general this involves using one or more of several methods for validating the output produced by crowdsourced workers:

- giving multiple workers the same instance of a task;
- having worker output checked or rated by other workers; and
- validating worker output against a gold standard result.

The subcategories of output-based quality control methods are diverse. For example:

- The gold standard or ground truth pattern evaluates worker quality by evaluating their output on a subset of task instances with a known correct output \cite{IrenBilgen2014,Allahbakhsh2013,Kern2014}.

- The redundancy, output agreement or voting pattern involves giving multiple workers the same instance of a finite deterministic task and choosing the most popular answer \cite{IrenBilgen2014,Allahbakhsh2013,Kern2014}.
The control group, expert review, validation and comparison patterns all broadly have one or more workers perform the task and then either a professional or a group of additional workers verify the output (Iren and Bilgen 2014; Allahbakhsh et al. 2013; Kern 2014).

In the iteration pattern the output of a complex task is passed from each worker to the next for further refinement, until quality goals are met (Kern 2014).

There are a great many variations on these patterns, and selection of a specific output based quality control method should be based carefully on the characteristics of the task and the output produced by workers (Kern 2014).

Another category of quality control methods identified by Kern (2014) is called execution process monitoring. This entails keeping a record of the actions taken by a worker while performing a task to track how they are working and to infer quality or the lack thereof in their output. For example a system may take regular screenshots or snapshots of work progress to send back to task setters. Rzeszotarski and Kittur (2011) take a sophisticated approach to execution process monitoring. They instrument a web interface with logging that records events such as scrolling the page, navigating between text input components, keys pressed, mouse movements and mouse clicks to build a predictive model of task performance based on user behaviour. From the raw stream of events they produced summary features such as total time, idle time, counts of fields accessed and mouse clicks. The summary features were fed into SVM and decision tree machine learning models and had reasonable success at detecting acceptance rates for workers output on a number of trial tasks. In Chapter 4 we describe how we implemented similar execution process monitoring into our glossing interface. We used less detailed but more application specific events such as interaction with glossed words and with their glosses, leaving out mouse movements and key presses. We also reduced our event logs to summary counts but additionally investigated several other ways of turning event position timeseries into a summary fingerprint. Rather than perform a supervised regression, we perform unsupervised clustering of behaviours to give insight into how the glosser is used and in particular to verify that the quality goal of capturing reading comprehension usage and gloss sense
consultation is met. Our execution process monitoring records will be useful to future work with our glosser involving prediction of measurable quality goals such as the production of gold standard sense annotations.

In this section we have reviewed the main features of crowdsourcing projects that affect quality, the broad categories of quality control methods used and discussed how they relate to this thesis. At the end of Chapter 7 we make a more detailed discussion of output based quality control methods in relation to the results of our research.

2.5 Summary

This chapter has covered three main topics under the umbrella of computational lexical semantics: word sense disambiguation, multiword expression token identification, and crowdsourcing semantic annotations. For both WSD and MWE research we highlighted the difficulty of developing semantically labelled corpus resources with a distribution of meanings for a large coverage of words. In the case of WSD we narrowed the discussion to knowledge-based methods which can develop high coverage models without using annotated text, with a particular focus on Personalised PageRank. We also reviewed evaluation methods for such models which may use such annotated corpora as are available. Our review of MWE identification explored the complexity of identifying usages of dictionary expressions in the face of idiosyncratic but not fully fixed behaviour with respect to syntactic and morphological variations. Finally, we surveyed research projects aimed at collecting semantic annotations for text from non-experts through crowdsourcing methods. Implicit annotation through game design and explicit paid annotation were found to be the primary strategies employed in the case of word sense annotation crowdsourcing.
Chapter 3

Resources Review

This chapter supplements the review of related work (Chapter 2) with additional material describing annotated corpora, software packages, linguistic theory and machine learning algorithms used in the course of the research described in the remainder of this thesis.

In Section 3.1, we introduce common linguistic concepts such as part of speech (POS) and syntax that provide symbolic and structured representations of text useful for any computational analysis. Section 3.2 narrows the focus to the Japanese language including challenges particular to the language and software tools for generating POS tags and parse trees from Japanese text. Section 3.3 describes one of the most widely used lexical knowledge base resources, Princeton WordNet (“WordNet”), and its equivalents in many languages including Japanese. It additionally describes the sense annotated corpus SemCor that is closely tied with WordNet and the process of translating that corpus into Japanese. Finally, Section 3.4 describes a rich corpus and interesting dictionary useful for MWE token identification in Japanese.

3.1 Linguistic concepts useful for text processing

Language finds its way into computational systems in a number of ways: a encoded waveform audio recording; an image of someone’s handwriting; and text entered by keyboard as encoded
character data. Digital systems can copy and transmit language in these forms with little understanding of their content. However, we can do more if we recognise patterns in language and teach the computer to manipulate our data on a higher level. The linguistic concepts introduced in this section are powerful tools for working computationally with language.

In this section we take a broad view of the relationship between computing and human languages, particularly written text. The primary purpose is to provide an introduction to basic linguistic ideas that are useful in text processing but may not be familiar to a reader with a general computer science background. The concepts introduced in this section will be fundamental assumed knowledge for the greater part of the dissertation. Linguistic concepts such as part of speech tagging and dependency parsing will be covered, with a focus on how they are realised computationally. For a reader familiar with these topics this section may be safely skipped. A number of popular tools for natural language processing will be introduced.

### 3.1.1 Part of Speech

One of the most basic patterns identified in human languages is the tendency for words to fall into groups exhibiting similar behaviour, the most common groups being the nouns and the verbs. Manning and Schütze (1999) demonstrates the substitution test for whether two words fall in the same group as follows:

\[
\begin{array}{c}
\text{sad} \\
\text{intelligent} \\
\text{green} \\
\text{fat} \\
\ldots
\end{array}
\]

Substituting any of these words — though changing the meaning — yields a grammatical sentence. On the other hand, we cannot substitute pie, dog, house or intelligence into the same position:
(5) * The pie one is in the corner.

Similarly we cannot substitute sad or intelligent for corner but we can substitute pie or house.

(6) * The sad one is in the intelligent.

(7) The sad one is in the house.

The accumulation of evidence from the substitution test can lead us to divide words into groups called word classes, categories or parts of speech (POSs).

One thing to note, however, is that words can belong to more than one POS. In the following examples, fat is used as a noun and adjective respectively:

(8) He trimmed the fat from his steak.

(9) The fat one is in the corner.

We can perform a grammatical but nonsensical substitution and a substitution that is not allowed:

(10) # The sad one is in the intelligence.

(11) * The sad one is in the intelligent.

This can be explained by noting that intelligence, like corner, is a noun; whereas the word intelligent, being an adjective, is not a valid substitution. Note that for linguistic examples in this thesis that exhibit semantic incoherence or infelicity we use a # mark rather than the * reserved for examples that would not occur in ordinary language.

Finally, we can broadly divide the word classes into two kinds termed open and closed (Manning and Schütze 1999). The open, or lexical, word classes — including verbs, nouns and adjectives — have a large number of members and admit new words into the class over time. The closed, or functional, word classes are small in number (of words) and tend not to change. One closed class is the prepositions, which includes the words at and in.
3.1.2 Morphology

Say we have classified a word, run, as a member of the verb class and we wish to find uses of it in a collection of text documents. Then we may simply look for all instances where the character r is followed by u and n in succession. However, what of the words ran, runs and running? We know these are all closely related: is there some way to identify them with each other?

The relationships between these words are called morphological processes. Manning and Schütze (1999) identify three main kinds of morphological process: inflection, derivation and compounding.

Inflection refers to functional changes in a word such as pluralisation (dog → dogs) or tense (walk → walked). Delightfully, word classes — though distinguishable by the substitution test alone — also distinguish themselves with systematic inflection rules. In English inflections are typically suffixes, including -ing, -s and -ed for verbs and -s and ’s for nouns. In other languages prefixes attaching to the start of the word and even infixes inserting themselves into the word are observed (Yu 2003). We call the group of all inflected forms of a verb the lexeme (Manning and Schütze 1999). Lexeme and the related term lemma are used to distinguish two related concepts in a cognitive account of sentence construction (Kempen and Huijbers 1983; Kempen and Hoenkamp 1987). In this account, the lemma of a word is accessed first for its syntactic properties and to determine requirements for number and inflection; in a second step the lexeme is accessed to perform the required morphological and phonological rendering of the word. In NLP we commonly make use of the term lemma to describe a morphologically “basic” representative form for a word. For example we may represent the lexeme \{walk, walks, walking, walked\} by the lemma walk. Since dictionaries usually list morphologically unmarked headwords the lemma may also be called the dictionary form or citation form of the word. To lemmatise a morphologically marked word is to find its lemma.

Manning and Schütze (1999) use derivation to describe morphological processes which form new words, usually of a different class, such as the suffix -ly which turns some nouns like
heaven into adverbs (*heavenly). They note that these processes are not as systematic as inflections, being very selective about which words allow the transformation (consider *computerly, *catly).

Finally we have compounding, in which two separate words bind closely to each other and begin to act as one, such as street+light and school+yard. The rules governing compounding are, perhaps, even more opaque than derivation. However, Manning and Schütze (1999) observe that, should we compile a list of known words and their properties — a lexicon — we will inevitably want to include some compounds as entries in their own right.

In this thesis we make extensive use of Japanese morphological processing tools, which we review in more detail in Section 3.2.

3.1.3 Word order

We have seen that word classes account for varying sentence validity under substitution, but valid sentences may also become invalid with a simple re-arrangement of words:

(12) The cat sat on the mat.
(13) * Sat cat mat the the on.

To account for this we will need a model of word order.

In this section we start with a simple account of word order called an n-gram model. This model does a poor job accounting for sentence validity but it, or the ideas behind it, are commonly used in NLP. The section goes on to a more advanced account of word order called phrase structure grammars (also known as context free grammars).

n-gram language models

Consider an English sentence in progress, say Yesterday I went to the ..., and try to predict the word that comes next. You might think of words describing locations someone would go to: nouns like park or bank but also adjectives describing them like new or nearby. To answer the question more generally we are going to need a model of how valid sentences are constructed. If we
had access to a complete canonical list of all valid English sentences then we could predict our next word based on all sentences which begin with *Yesterday I went to the ...* If we knew the frequency with which each sentence appeared we could even assign a probability to each following word. However, building a list of potentially infinite possible English sentences and their frequency of use is impossible, so instead we build symbolic and probabilistic models of the language.

As a simple example, let us try to make a record of all valid three word sequences and their frequencies of use. If we limit ourselves — quite reasonably — to a finite vocabulary such a model of language is enumerable. To populate the model we might, for example, collate every three word sequence observed in a very large collection of books and newspapers. If we count the number of occurrences of each unique three word sequence we can store this corpus relatively compactly: each entry is three words and an integer. Omitting the counts, we end up with a collection that looks something like this:

```
(instead of making), (of making a), (making a record), (a record of), (record of all),
(of all possible), (all possible sentences), ...
```

Now we can estimate the possible or probable next words in our sentence by looking for all three word sequences in the collection of the form *to the ....* The relative counts of each match allows us to assign a probability to each potential next word. This approach to predicting valid sentences surely lacks accuracy but it is certainly computable. In fact this technique, called an \textit{n}-gram language model, is a common technique that has been considerably successful at next-word prediction in a wide variety of natural language processing tasks such as speech recognition (Manning and Schütze 1999). The \( n \) is the length of the word sequences used, which in the case of our example is three. Manning and Schütze (1999) observe that 2- and 3-grams are common called bigrams and trigrams respectively, a mixing of Latin prefix with Greek suffix that they simultaneously note and condone.

An \( n \)-gram model is called generative in that it can be used to generate sentences from scratch. One starts by picking a likely \( n \)-gram, then uses the model to pick the \( i + 1 \)th word, the \( i + 2 \)nd word and so on, always using the final \( n - 1 \) words of the sentence so far to estimate the
next. The following text generated from a three-gram model of the complete words of William Shakespeare appears in [Jurafsky and Martin (2000)](#)

> Sweet prince, Falstaff shall die. Harry of Monmouth’s grave.  
> This shall forbid it should be branded, if renown made it empty.  
> What is’t that cried?  
> Indeed the duke; and had a very good friend.  
> Fly, and will rid me these news of price. Therefore the sadness of parting, as they say,  
> ’tis done.  
> The sweet! How many then shall posthumus end his miseries.

To the author’s native speaker intuitions this text makes very little sense but reads with a reasonable fluency and is strongly reminiscent of the writing style of Shakespeare.

**Syntax**

In Section [3.1.1](#) we saw the substitution test for single words. If a family of words could be substituted for each other without changing the validity of a sentence then it was counted as evidence that the words fall into the same category. However, we can expand this idea to cover more than a single word. Phrases within a sentence can be assigned categories as well.

For example, consider the following sentences:

14. The *cat* sat on the mat.

15. The *yellow flea bitten cat* sat on the mat.

16. I saw a *cat*.

17. I saw a *yellow flea bitten cat*.

18. Peter submitted his *documents* to the court.

19. Peter submitted his *yellow flea bitten cat* to the court.

As we go on we find an abundance of sentences in which *yellow flea bitten cat* can be substituted for *cat* — or indeed any noun — without afflicting the validity of the sentence. We use this to argue for a new category: the noun phrase. Similar arguments can be made for the existence of verb phrases.
and a great many more phrasal categories. The existence of these substitutable phrasal categories motivates the idea of phrase structure grammars.

The key observation is that a phrase, say a noun phrase, is composed of a sequence of shorter phrases of differing categories and that noun phrases in general only decompose into other categories in a limited number of ways. For example, the verb phrase *quietly gave the tall stranger the locked briefcase* decomposes into verb phrase (*quietly gave*) noun phrase (*the tall stranger*) noun phrase (*the locked briefcase*); however you would be hard pressed to find a verb phrase which decomposed into five small noun phrases in a row.

Manning and Schütze (1999) demonstrate how a sentence can be built up systematically from phrasal categories by allowable substitutions. They use the substitution rules seen in Figure 3.1. The substitution of a single symbol for multiple symbols according to a grammar rule may be alternatively referred to as an expansion.

Consider the rule for NP: this is the noun phrase rule. What this rule means is that in any position where a noun phrase can be substituted, we can substitute any of the following three patterns:

1. A determiner (AT, a or the) followed by a plural noun (NNS).
2. A determiner followed by a singular noun (NN).
### Current sentence | Next rules to apply
---|---
S | S → NP VP
NP VP | NP → AT NNS
VP → VBD NP
AT NNS VBD NP | NP → AT NN
AT NNS VBD AT NN | (lexical category substitutions)

Figure 3.2: Derivation of *The children ate the cake* using the grammar given in Table 3.1 (Manning and Schütze 1999).

Tagging

```
The/DT children/NNS ate/VBD the/DT cake/NN ./.
```

Parse

```
(ROOT
 (S
   (NP (DT The) (NNS children))
   (VP (VBD ate)
     (NP (DT the) (NN cake)))
   (. .))
)
```

Figure 3.3: The parse of the sentence *the children ate the cake* produced by the Stanford Parser (Klein and Manning 2002), generated using [http://nlp.stanford.edu:8080/parser](http://nlp.stanford.edu:8080/parser).

3. A nested noun phrase followed by a prepositional phrase (PP, for example *at ...* or *on ...*).

S is a pseudo-phrasal category representing the whole sentence. Manning and Schütze (1999) demonstrate the derivation of *The children ate the cake* by repeated substitution of rules from Figure 3.1. The steps of the derivation can be seen in Figure 3.2. To begin with, the sentence category S is expanded using its only rule to NP VP; in this grammar all sentences are a noun phrase followed by a verb phrase. Phrasal categories may then be expanded freely according to the grammar rules, but in this case they are expanded with a particular target sentence in mind. A sequence of expansions is chosen that leaves no remaining phrasal categories in the sentence — only lexical categories remain. Finally, words are substituted for their lexical category.

This is all very well as a system for generating grammatical sentences, but it doesn’t (yet) help us to analyse sentences stored as data in a computer; nor to process those sentences. For
that we need a process that is called parsing. In fact, many algorithms exist for parsing natural language. Here is the output of the Stanford Parser on our running example: Figure 3.3 shows a sample output of the Stanford Parser (Klein and Manning 2002). Careful examination reveals that this parse uses the same categories and substitutions as the derivation in Figure 3.2 (with some slight symbolic differences, such as DT for determiners instead of AT). Each pair of parentheses represents a substitution from the grammar: the first thing after an open parenthesis is one of the category symbols; after that comes the sub-phrases it has been analysed into. Each sub-phrase in turn may be analysed further or may be a single word. We can also represent the parse of a sentence diagrammatically by suspending the expanded elements of a category below it in a parse tree. A tree representation of the parse is shown in Figure 3.4.

3.1.4 Dependency grammars and parsing

A dependency grammar is an alternative to constituency grammars (see Section 3.1.3) for giving a tree structure analysis to a sentence. A node is created for each word in the sentence, and branches are assigned between words. Each branch is called a dependency link, with the one word’s node assigned as the dependent and the other as the head or governor.

An example dependency parse appears in Figure 3.5. Notice that the dependents (lower down the tree) typically modify or provide additional information about the heads: yellow modifies
Figure 3.5: Two representations of an example dependency parse for the rapidly tiring yellow cat ate the mouse. The first representation emphasises the tree structure of the parse whereas the second captures the word order.

cat; rapidly modifies tiring and so on. Furthermore, the dependent only features in the sentence by virtue of its dependency link relationship; whereas the head may or may not require the presence of the dependent. For example, yellow would be meaningless in this sentence without cat, but cat could surely do without yellow.

Useful dependency grammar properties

Dependency parsing is a useful step in an NLP pipeline. The minimalistic word-to-word tree representation is easy to work with while still capturing salient features of sentence structure. Here we discuss some typical features for dependency grammars which do not necessarily exist for all languages but are either common or very useful where they appear.

Covington (2001) surveys a range of dependency grammars and parsers and draws out common properties. We are interested in particular in the property of Projectivity (adjacency). He uses the following definition: the subordinates of head word are its dependents, the dependents’ dependents and so on until no more dependents are found. Projectivity then means that “if word A
depends on word B, then all words between A and B in the sentence are subordinate to B.” Note
that the intervening words may or may not be subordinate to A as well. Covington (2001) gives
the following example of projectivity violation, where B violates projectivity between A and C (it
would not if it were headed by A or C instead):

Note the new style of dependency parse illustration which emphasises both the tree struc-
ture and the word order of the sentence. The tree is raised above the original sentence and the node
for each word is tied to the word by a dotted line. If one of the solid dependency lines “crosses” a
dotted line, a projectivity violation has occurred. There is redundancy in the diagram in that each
word is displayed twice: once in the sentence and once on the node. Therefore sometimes the nodes
in this kind of dependency diagram are used to display additional information such as part of speech
for the word.

Clearly projectivity does not hold in languages with a totally free word order. However it
permits a certain amount of freedom of word order for phrase arguments so is a nice abstraction for
weakly ordered languages. Projectivity proves useful in the work described in Chapter 5.

One other property highlighted by Covington (2001) is that dependency links are very
close to semantic relations. For example, in the example in Figure 3.5 the dependents of eat are
the the eater (cat) and the eaten (mouse). These relationships, when correctly identified, mean
potentially much more to us than direct adjacency relationships between words. However Osborne
(2011) investigates a tension between this property and the projectivity property. One case covered
is wh-fronting in English. Consider the dependency parses for the sentence what did you eat?.
Figure 3.6a shows a dependency parse for the sentence which preserves the relationship eat →
what, but violates projectivity. On the other hand, the parse shown in Figure 3.6b is a common
Chapter 3: Resources Review

Figure 3.6: Possible dependency parses for the sentence What did you eat?

(a) This parse demonstrates a projectivity violation, but it captures the relationship eat $\rightarrow$ what.  
(b) Though projectivity is preserved this parse does not capture the relationship eat $\rightarrow$ what.

simplification preferred by dependency parsers restoring projectivity at the cost of instead coding a relationship did $\rightarrow$ what.

Osborne (2011) examines many more situations in which, in his terminology, a word’s head dependency rises so that it is no longer subordinate to its governor, thus drawing a distinction between the true governor and the head to which the dependency is attached in the parse structure. His treatment also includes cases where rising occurs due to a dependency grammar (or parser) for reasons other than projectivity violation. We make no further comment on this phenomena, except to state that we make use dependency parsers relying on both projectivity and on discovering governor/governee relationships and that we accept that trade-offs may have been made.

3.2 Japanese natural language processing

Japanese is a non-segmenting language, meaning that word boundaries are not explicitly marked by spaces as they are in English. This poses a problem for working computationally with Japanese; not the least being that our discussion of parsing so far has simply assumed that a sentence starts as a sequence of words. Segmentation of Japanese turns out to be a non-trivial problem to solve, especially since there is no definitive answer to what constitutes a word in Japanese. Several competing analyses exist for explaining Japanese sentences in terms of morphology, chunking and dependency parsing. We will focus on a small selection and on the features of each which informed
Figure 3.7: Possible segmentations for the sentence Example (20). Pronunciations, translations and explanatory notes are a guide, not part of the output of the respective morphological analysers.

To help us in understanding methods of Japanese NLP, let us consider the example sentence:

(20) ワイキペディアはオーブンコンテンツの百科事典です。

“Wikipedia is an open content encyclopedia.”

We begin by examining a few potential segmentations for this sentence. Figure 3.7a has been segmented by the Juman morphological analyser (Kurohashi and Kawahara 2017). We see of the tokens Juman has identified that are a number we understand as words — wikipedia, ṃpuncontento “Open Content”, hyakkajiten “Encyclopedia” and desu “is” — and a number are functional, typically marking case. Figure 3.7b shows the segmentation produced by another morphological analyser, MeCab. An interesting difference is that MeCab has segmented two words
where Juman identified only one: hyakkajiten “Encyclopedia” has been segmented into hyakka “panoply of subjects” and jiten “dictionary”. From the perspective given by the English translations this appears to be a failure to identify a lexical item. However, it is sometimes advantageous to identify parts of a compounded word when those parts fit neatly into classes of morphological variation.

Maekawa (2009) in an analysis of the Corpus of Spontaneous Japanese, describes how usages can be segmented on long and short units, noting that short unit words (SUWs) may be divisible into smaller short unit words, and drawing attention to the potential for short unit words to be compounds themselves of atomic constituents termed simplex SUWs. Two commonly used morpheme inventories for segmentation of Japanese are the IPAdic (Asahara and Matsumoto 2003) and unidic (Den et al. 2007) dictionaries. The latter is newer, has a finer granularity and can be considered to better represent the simplex short unit words of Japanese, thus leading to segmentations with tokens of a more uniform size (Den et al. 2008).

However, neither of the fine grained word segmentations in Figure 3.7 are particularly desirable for dependency parsing. In particular we are likely to find that most dependencies arrows have a functional word at one end. In practice, Japanese dependency parsers use a morphological tokenisation as a preprocessing step and then regroup tokens for the purpose of establishing a dependency parse between chunks or rather bunsetsu, which are a maximal word unit combined with any attached prepositions, case-marking particles or auxiliary verbs. Figure 3.8 shows the output of the KNP dependency parser for the sentence Example (20).

3.2.1 Available tokenisers and parsers

KNP is a dependency parser (see Section 3.1.4) for Japanese (Kurohashi and Nagao 1994). It identifies a tree of dependency relationships between bunsetsu, which are formed as groups of morphemes using the output of morphological analyser Juman (Kurohashi and Kawahara 2017). It was developed with the aim of handling of co-ordinations such as A, B and C and A, B or C.
well, where the co-ordination has ambiguous scope because the co-ordinates $A$, $B$ and $C$ might be phrasal types. To help resolve this ambiguity $KNP$ uses a Japanese thesaurus to attempt to assign co-ordinated phrases which have a higher semantic similarity to each other. The full algorithm looks at similarities between POS composition of single bunsetsu and uses a dynamic programming method to calculate similarities between potentially co-ordinated sequences of bunsetsu of differing lengths (Kurohashi and Nagao 1992). It references other lexical resources as well, including a case frame dictionary that lists the argument types of 861 common Japanese verbs. After constraining the parse tree using the co-ordinated structure similarity measures and case frame information a number of additional Japanese specific rules are applied to resolve remaining ambiguity. In the original implementation of Kurohashi and Nagao (1994) dependency ambiguity is resolved in part by a scoring system for the available case frames. Recent versions use a generative probabilistic model of case frame assignments based on corpus statistics (Kawahara and Kurohashi 2006b).

The $Juman$ tokeniser used by $KNP$ has its own morpheme dictionary, but to use the newer dictionaries $IPAdic$ and $unidic$ a popular choice today is $MeCab$ which uses a conditional random field (CRF) model (Lafferty et al. 2001) to segment and tag Japanese text (Kudo et al. 2004). $MeCab$ uses a network representation of a Japanese sentence called a lattice where the nodes are
known dictionary words that have a potential surface form matching a substring of the sentence and the edges represent adjacency in the text. It learns and solves for full paths through the dictionary matches connecting the start of a sentence to the end. As such, the conditional random field model has advantages over previous models of Japanese segmentation. In particular:

- Features extracted for edges can look forwards and backwards in the lattice;
- Full sentence training and optimisation eliminates bias on the basis of word and path length.

Additionally, the conditional random field model can be used with any dictionary and can simultaneously learn an arbitrary number of morpheme tags such as POS. In testing on standard Japanese tokenised corpora MeCab achieved a higher f1-score than Juman on segmentation and fine grained POS tagging and a substantially higher f1-score on a labelling task that included multiple levels of POS and inflection labelling.

A popular dependency parser for Japanese today that can be used with either IPAdic or unidic tokenisations output by MeCab is CaboCha [Kudo and Matsumoto 2002]. CaboCha uses a method called cascaded chunking for building up a dependency tree leaf-first. The algorithm iteratively processes the sentence deciding whether any bunsetsu is dependent on the word directly following it (taking advantage of a head-last assumption which is valid for Japanese). All bunsetsu which are marked as dependent are then removed from the sentence before the next iteration. Iteration stops when only one bunsetsu is left. The decision regarding whether a bunsetsu is dependent on its neighbour is made by a SVM classifier based on features of the bunsetsu themselves and the parse constructed so far. The classifier is trained by simulating the parsing algorithm on gold standard parsed sentences to build up example gold standard decision results.

### 3.2.2 Dependency parsing and the Japanese language

CaboCha and KNP both make particular assumptions about the Japanese language that simultaneously make dependency parsing more suitable than constituent parsing and also help to guide interpretation of parse trees. Specifically, they assume:
Chapter 3: Resources Review

- projectivity (see Section 3.1.4), and
- head-finality (Kurohashi and Nagao 1994; Kudo and Matsumoto 2002).
- (Relatively) free word order.

The first assumption means that an intervening word in a phrase must be a modifier of one of the phrase’s constituents. The second assumption means that in Japanese dependencies are unidirectional — parent nodes in the dependency tree always come later in the sentence than their children. In Chapter 5 we use these assumptions to simplify feature extraction for words in a syntactic relationship with constituents of MWEs in an idiom token candidate disambiguation task.

The third assumption means that the dependent children of a head word do not have a strict ordering. A word’s relationship to its head is explicitly denoted by a functional particle rather than by its position in the sentence. Although the conventional Subject-Object-Verb order is usually used, order of indirect objects is less fixed, and it is beneficial to have a parser that can handle unusual-but-not-illegal orderings. An additional assumption that dependency parsing handles well is that ellipsis of words is quite commonly allowable in Japanese. For example, the subject of the sentence can be freely elided if it is the same as the previous sentence. Ellipsis is also frequently used in place of personal pronouns where the context makes the person clear. Phrase structure grammars are not well suited to representation of free word order or ellipsis; a separate rule is required for each permutation, elided word and combinations thereof. Therefore, in this thesis (and Chapter 5) in particular, we chose to use dependency parsing when requiring syntactic analysis of Japanese.

3.3 Princeton WordNet

One of the most important — and certainly widely used — resources in computational lexical semantics is the Princeton WordNet (“WordNet”) lexical knowledge base. WordNet is, at its heart, a dictionary but it was created with computational processing rather than a human audience.
Chapter 3: Resources Review

Noun:

- sleep, slumber
  (a natural and periodic state of rest during which consciousness of the world is suspended)
  *he didn’t get enough sleep last night; calm as a child in dreamless slumber*

- sleep, sopor
  (a torpid state resembling deep sleep)

- sleep, nap
  (a period of time spent sleeping)
  *he felt better after a little sleep; there wasn’t time for a nap*

- rest, eternal rest, sleep, eternal sleep, quietus
  (euphemisms for death (based on an analogy between lying in a bed and in a tomb))
  *she was laid to rest beside her husband; they had to put their family pet to sleep*

Verb:

- sleep, kip, slumber, log Z’s, catch some Z’s
  (be asleep)

- sleep
  (be able to accommodate for sleeping)
  *This tent sleeps six people*

Figure 3.9: The synsets of the word *sleep* in WordNet 3.1. Retrieved from [http://wordnetweb.princeton.edu/perl/webwn](http://wordnetweb.princeton.edu/perl/webwn).

in mind (Miller 1995). As such, it includes not just a lexicon and POS information but also rich information on semantic relations between words. WordNet is used and referenced extensively in the work of this thesis, so in this section we give a detailed review of the construction of WordNet with a particular focus on terminology. Any reader familiar with WordNet can safely skip this section.

Synonymy — the phenomena of two words having the same meaning — is fundamental to the organisation of WordNet. Each word is a member of one or more synsets. A synset can be thought of as a concept represented by the collection of words that can signify that concept. Definition glosses in WordNet are given for synsets, not words, and the semantic relations encoded in WordNet are between synsets as well. The senses of a word are the synsets of which it is a member; the synonyms of a word are the other words in its synsets. The WordNet senses of the
word *sleep* are shown in Figure 3.9.

*WordNet* encodes the following relationships between synsets (Miller 1995):

**Antonymy** connecting concepts of opposite meaning;

**Hyponymy and hypernymy** connecting concepts with more specific or general meanings, respectively;

**Meronymy and holonymy** connecting concepts that are considered constituent parts of a respective whole;

**Troponymy** for verbs regarding the same action performed in a different manner; and

**Entailment between verbs** for occurrences that necessarily involve the occurrence of a related event.

The inclusion of the hyponymy relation makes *WordNet* a *taxonomy* of the English language; the suite of semantic relationships makes it an *ontology*.

*WordNet* is distributed with software for querying its database. This software encodes information on inflectional morphology: a search for *went* will return the entry for *go*. *WordNet* did not originally encode any information about derivational or compound morphology and so, for example, *interpret* was not linked to *interpreter* in any way (Miller 1995). However, there is now a “standoff file”[^3] for morpho-semantic links between word senses including the new semantic role of the morphologically derived form (Fellbaum *et al.* 2007). For example for *interpret* the derived form *interpreter* is assigned the role agent.

**Guide to terminology**

As a dataset and software package, *WordNet* is quite complicated and has a sometimes-cryptic vocabulary for describing its parts. Here, we introduce some of *WordNet’s* terminology and[^3]

[^3]: a separately distributed dataset
structure as a beginner’s guide to its use and as an ongoing reference for later chapters. In particular, we focus on the origins of the terminology and composition of synset and sense identifiers, to emphasise that they are tied to specific WordNet versions. More detailed information and WordNet downloads can be found online (WordNet 2013).

Some of the key concepts of WordNet are:

**Synset type** or **ss.type** is a part of speech (or “category”) for a synset. There are four main POSs and one special case. They are represented in several different ways in different parts of WordNet:

<table>
<thead>
<tr>
<th>Part of speech</th>
<th>numeric code</th>
<th>character</th>
<th>filename</th>
</tr>
</thead>
<tbody>
<tr>
<td>Noun</td>
<td>1</td>
<td>n</td>
<td>noun</td>
</tr>
<tr>
<td>Verb</td>
<td>2</td>
<td>v</td>
<td>verb</td>
</tr>
<tr>
<td>Adjective</td>
<td>3</td>
<td>a</td>
<td>adj</td>
</tr>
<tr>
<td>Adverb</td>
<td>4</td>
<td>r</td>
<td>adv</td>
</tr>
<tr>
<td>Adjective Satellite</td>
<td>5</td>
<td>s</td>
<td>adj</td>
</tr>
</tbody>
</table>

A definition of satellite adjectives is given below.

**Lexicographer files** are the source files from which everything else is built. They are not typically needed for applications but their design affects the terminology and usage of many of WordNet’s features.

**lex_filenum** or **lexicographer file numbers** identify lexicographer files. Each file is strictly limited to one POS and many of the files are dedicated to subcategories, for example 13 is **noun.food** and 30 **verb.change**.

**lex_id** is the name given to a two digit identifier assigned by the lexicographer to identify a sense of a word: combined with a **lemma** it identifies a synset the word is a member of.

These numbers are used in sense keys and **lex_sense** fields as defined below to identify word senses.
Data files contain entries for synsets. One file exists for each POS: data.noun, data.verb, data.adj and data.adv. These files are generated from the lexicographer files and are more machine readable.

Each line of each data file represents a single synset and as such synsets are often referred to by their byte offset into the file. See also: synset offset.

The following information is recorded about each synset:

- The lemma of each word in the synset and the sense number of (lex_id) each word.
- Pointers to other synsets — and optionally specific words — representing lexical and semantic relations such as hypernymy.
- Additional information such as the source lexicographer file number for the synset, the POS as an ss_type number and sentence frame information for verbs.
- Finally, a gloss which consists of a definition and zero or more example sentences.

Synset offsets (in the data file) combined with ss_type (POS) characters are often used to identify synsets. The offset represent a byte offset in the data file for the ss_type as an eight digit zero padded decimal integer. For example, the synset for the common house cat is 02124272-n in WordNet 3.1, but the offset changes from version to version.

Index files are indexes for data files. They are sorted by lemma and provide, amongst other things, a list of synsets for which each word is a member. The synsets are expressed as offsets into the corresponding data file, and are ordered by frequency of usage in a reference corpus.

Sense keys are identifiers for word senses in the lexicographer files. They comprise the word’s lemma joined by a % character to a sense identifier called a lex_sense, which identifies a sense entry in the lexicographer files and has the following format:

ss_type:lex_filenum:lex_id:head_word:head_id
A typical sense key looks like `interpret%2:31:00::` or `interpret%2:31:02::`. The final two fields are usually empty: they are used to link satellite adjectives and adverbs to their head lemma and lex_id.

The sense index file, `index.sense`, maps sense keys to offsets in the data file for the sense key’s part of speech.

Satellite adjectives can be thought of as nested synsets. They are defined inside other synsets in the lexicographer files and have very closely related meanings. For example, the first sense of `preceding` has a number of satellite synsets: one for `above`, one for `above-mentioned` and `above-named`, one for `foregoing`, one for `introductory` and so on.

Princeton WordNet has undergone several updates and each time it is updated the lexicographer files have changed, as do all the data and index files generated from it. This means that sense keys and synset offsets may change between WordNet versions. For this reason, users of WordNet annotated corpora must be very careful as to which versions of WordNet were used for annotation. Sense mappings between versions are available alongside the main WordNet distribution (WordNet 2013).

3.3.1 SemCor

SemCor is abbreviated from the term semantic concordance, which is a collection of text for which words have been labelled with senses from a dictionary or, to put it another way, a sense-annotated gold standard corpus. SemCor is a specific — and very popular — concordance with sense labels from Princeton WordNet. In fact, WordNet word senses are ordered by frequency based on their counts in SemCor annotations. SemCor’s construction is detailed in Landes et al. (1998) which we summarise briefly here.

The original semantic concordance for WordNet was assembled using 103 texts from the Brown Corpus (Francis 1965), which are each approximately 2000 words in length, coming from a wide variety of US English sources. It also included chapters from a novella on the American Civil
war. The most recent version of *SemCor* distributes only *Brown Corpus* text but has been expanded by 83 additional fully annotated documents and 166 more with sense annotations on verbs.

Each document in *SemCor* was first segmented into words contained in *WordNet* and tagged for part of speech. Human annotators then selected sense tags for each word sequentially using a software application called *ConText* which was developed concurrently with the annotation task. The application displayed text from a document with a single word highlighted and the user selected from relevant *WordNet* senses displayed below the text. In the first pass, annotators were given sole responsibility for their annotations. Accuracy was verified by a manual check of every 11th word. Problematic words and documents were noted and fixed, then the check was re-run on every 12th word.

**ConText files**

The resulting semantic concordance was released in the *ConText* file format, which is an *Extensible Markup Language (XML)*-like text format. Specifically, it conforms to an *XML* precursor, *Standard Generalized Markup Language (SGML)*, notably allowing unquoted tag attributes. The root node(s) are *(contextfile)* tags. Nested within comes one (or more) *(context)* tags which contain documents. Nested within that come *(p)* for paragraph, then *(s)* for sentence and finally *(wf)* for words alongside *(punc)* for punctuation. The *(wf)* has many attributes, including *lemma* for the word’s base form as it appears in *WordNet* and *lexsn* which is a *WordNet* *lex*-*sense*; together they represent a sense key: see Section 3.3 for key structure and usage. Other notable *(wf)* attributes include *pos* containing the words part of speech and *cmd* indicating the word’s annotation status (*done* for sense tagged words). A complete example appears in Figure 3.10.

According to documentation supplied with *WordNet*, *SemCor* is no longer maintained or distributed. However, it can still be downloaded from [Mihalcea (2017)]
Figure 3.10: An abbreviated example ConText file from SemCor.

3.3.2 Other WordNets

The Princeton WordNet design has since been emulated for a number of languages other than English. Here we discuss a few which have made a particular effort to link synsets from the new WordNet back into Princeton WordNet wherever they represent the same — or at least very similar — concepts. Pianta et al. (2002) outline two methods for constructing an aligned WordNet:

- Build the new WordNet from scratch, including semantic relations, and then make crosslingual links between synsets where possible.

Pianta et al. (2002) attribute this approach to EuroWordNet, which originally covered English, Dutch, Italian and Spanish (Vossen 1997) and has expanded to more languages since. EuroWordNet can be licensed for a fee.
• Bootstrap the new WordNet by examining synsets from an existing WordNet to determine whether the concepts can be copied directly. All relevant target language words are added to the new synset. Any semantic relations between copied concepts are assumed to exist between the words in new language WordNet. MultiWordNet (Pianta et al. 2002) and Japanese WordNet (JWordNet) (Isahara et al. 2008) are examples of this approach.

MultiWordNet is an Italian WordNet linked back to Princeton WordNet and is available free of charge; a license can be obtained from multiwordnet.fbk.eu/english/license.

JWordNet is similarly linked back to Princeton WordNet and lives at nlpwww.nict.go.jp/wn-ja/index.en.html.

Bond et al. (2008) described an automatic process for mapping Japanese words into Princeton WordNet synsets. It was done using a combination of translation dictionaries and existing non-English WordNets with mappings into Princeton WordNet. The basic idea was that if a Japanese word is a translation equivalent of an English one it should share some, but not all, of the English word’s synsets. Bond et al. (2008) observe that if a single English word translates to two distinct Japanese words with different meanings then it may translate to different words in another language as well. For example, one synset contained the English bat and the French chauve-souris and chiropteran. Two of those three words had translation dictionary entries for the Japanese k¯omori “bat (mammal)”, making it a strong candidate for inclusion in the synset. Similarly, another synset contained English bat and French gourdin and batte. Again, two of three translated to Japanese batto “bat (club)” making another strong candidate.

To formalise this idea, Bond et al. (2008) used linked WordNets for French, Spanish and German to include words from those languages in Princeton WordNet synsets. Then, for each mixed language synset $S$ they used translation dictionaries to link the English, French, Spanish and German words to Japanese words. They scored Japanese words for membership in $S$ according to how many words in $S$ it had translation links from. A bonus was given to Japanese words which
had translations from multiple languages within $S$. Manual analysis of the result found that the use of multiple language WordNets lead to higher precision and lower recall for the task of identifying Japanese words for inclusion in Princeton WordNet synsets.

*JWordNet* is used extensively in many chapters of this thesis.

### 3.3.3 Translations of SemCor

The existence of multilingual synset-linked WordNets presents an opportunity for sense tags on texts in one language to be transferred to words in a translation of that text. *MultiSemCor* is one such translation of *SemCor*, which is annotated with Princeton WordNet synsets, into Italian text annotated with *MultiWordNet* synsets (Bentivogli et al. 2004).

Bentivogli et al. (2004) started construction of *MultiSemCor* by having professional translators translate *SemCor* into Italian. The translators were given a reference translation dictionary and asked to use the word translations provided by the dictionary in their text translations. The same dictionary was used by a word alignment system called *KNOWA* (Pianta and Bentivogli 2004) to link words in the original *SemCor* to their direct translations in *MultiSemCor*. Word alignment is an imperfect process but asking the translators to make use of the translation dictionary was found to improve alignment precision from 83.5% to 88.4% and recall from 57.9% to 67.5%. Finally, for all sense annotated words in *SemCor* which were aligned to Italian words with a linked synset, the annotation was transferred to *MultiSemCor*.

Not all of *SemCor* was translated: *MultiSemCor* consists of 116 texts translated from the *Brown Corpus* texts in *SemCor* (Bentivogli et al. 2005). Nevertheless, this amounted to 258,499 English word tokens and 267,607 Italian ones. *MultiSemCor* has 92,820 Italian tokens with sense annotates transferred from the 119,802 sense-annotated English tokens in the translated subset of *SemCor*. The corpus can be browsed at multisemcor.fbk.eu/index.php.

Pianta and Bentivogli (2004) gives more detail on how the word alignment is performed by *KNOWA*. The input to *KNOWA* is a pair of sentences to be word-aligned. The sentences are first run through morphological analysers for their respective languages to get a list of potential
lemmas for each word. Words are then allowed to align if their potential lemmas are equivalents in the supplied translation dictionary. KNOWA’s algorithm looks first in the English-Italian direction, preferring to align English words to Italian words in a similar position in the Italian sentence. It makes any alignments it can, then repeats the process in the other direction for Italian-English word translations. In a final step unaligned words are considered for alignment based on graphemic similarity.

**Japanese SemCor**

*Japanese Semcor* was constructed as a translation of the original *SemCor* on the same sentences as *MultiSemCor* (Bond *et al.* 2012). By contrast to Bentivogli *et al.* (2004), word alignment data was collected as part of the translation process.

Translators were originally asked to translate single sentences in isolation. However, due to concerns about the coherence of the translations the task was modified to give the translators more context for the sentence of translation and also to assign all sentences from each document to a single translator.

The word alignment data was collected indirectly via click-through data from the user interface used for translation. Words which were sense tagged in the English *SemCor* had potential translations drawn from *Japanese WordNet* for the tagged sense. These potential translations were generated via the mapping between *Japanese WordNet* and *Princeton WordNet* synsets. In the sentence translation interface, translators were supplied with the lists of the candidate word translations. Clicking on a suggested Japanese word would insert it into the translation text. Translators were encouraged to use this feature wherever possible, without sacrificing the fluency of the translation. By recording these clicks, the software was able to record a partial word alignment between the source text and the translation. However, only the lemma of the inserted Japanese word was captured, not its position in the translated text, nor any later edits for inflection.

The word alignment click-through data collected from the translation interface recorded alignments between lexical entries within the scope of a sentence rather than at the word token level.
Chapter 3: Resources Review

The process of converting this data into a word alignment and using that alignment for sense transfer and annotation of the Japanese Semcor is detailed in Chapter 6.

3.4 Japanese MWE resources

The distinction between short unit words, long unit words and bunsetsu presents a problem much like MWE token identification in itself, but Japanese additionally has multi-bunsetsu expressions which take on idiomatic meanings much like MWEs in English. In this section we review two significant resources available for Japanese MWEs of the multi-bunsetsu kind.

3.4.1 The OpenMWE corpus

The OpenMWE corpus (Hashimoto and Kawahara 2009) is a collection of sample usages of potentially ambiguous Japanese MWEs. MWE types were collected from five Japanese expression dictionaries. Rare or dubious expressions were culled by only selecting expressions appearing in a majority (that is, three or more) of the dictionaries. For many expressions, their particular arrangement of constituent words could only ever mean one thing. However for a subset of the expressions the words could be interpreted with an idiomatic expression meaning or compositionally with words’ literal meanings. These MWEs are termed ambiguous. Two native speakers filtered the list of the MWE types gathered from the idiom dictionaries, selecting ambiguous expressions and rejecting expressions with only one meaning. Their work was validated by calculation of agreement with two additional native speakers on a subset of the idioms. The end result was that 146 idiom types were classified as potentially conflatable with a literal usage. The idioms were tokenised with Juman and parsed with KNP to get a canonical structure.

Next the Japanese Web Corpus (Kawahara and Kurohashi 2006a) was searched for appearances of the ambiguous expressions. Example sentences were found in which an ambiguous expression’s constituent words appeared in the canonical syntactic arrangement found for the expression. These sentences were collated and manually annotated I if they were in fact an idiomatic
token of the MWE, and $L$ if they were literal and only happened to share the form of the idiom. The result is a corpus suitable for use in supervised machine learning identification of idiom tokens. [Hashimoto and Kawahara 2009] performed some initial experiments of this kind but left many questions open.

The final corpus comprised 102,856 sentences each containing one MWE token candidate. Of these, 68,239 were idiomatic MWE tokens and 34,617 were literal usages. 68 idioms had more than 1000 token candidates annotated. For the remaining idioms fewer than 1000 candidates were found in the *Japanese Web Corpus*. Overall it is one of the largest idiom corpora available, especially for a gold-standard annotated resource.

### 3.4.2 JDMWE

The recently-published *Japanese dictionary of multiword expressions* (*JDMWE*) encodes type-level information on thousands of Japanese MWEs [Tanabe et al. 2014]. Parts of the dictionary available for download at [http://jefi.info/](http://jefi.info/) *JDMWE* encodes information about lexico-syntactic variations allowed by each MWE type it contains. For example, the expression *hana wo motaseru* — literally “to have [someone] hold flowers” but figuratively “to let [someone] take the credit” — has the syntactic form entry \([N \text{ wo}] \star V30\). The asterisk indicates modifiability, telling us that the head [V]erb *motaseru “cause to hold” allows modification by non-constituent dependants — such as adverbs — but the dependent [N]oun *hana “flowers” does not. A subset of the dictionary was released some time ago [Shudo et al. 2011], and overlaps to some extent with the MWE types in the *OpenMWE* corpus. The *OpenMWE* corpus and the early subset of *JDMWE* are central resources used in Chapter 5 of this thesis.
Part II

Glossing
Chapter 4

The Wakaran glossing tool

A common bottleneck in the computational study of semantics is the acquisition of appropriate annotated corpora. Manual annotation with the abstract phenomena we wish to study such as word sense can be labour intensive to achieve coverage of a large number of word types. In Chapter 6 we will describe one shortcut to this process: the construction of a sense annotated corpus of Japanese text translated from English by transfer of annotations from the translation source corpus. However, that method is limited to existing manually annotated corpora and made use of the direct links between concepts in the Japanese lexical knowledge base (LKB) JWordNet and the original English WordNet. In this chapter we describe the development of an end-user application which will naturally produce links from text to dictionary senses as a side-effect of its use. The annotations will not be expected to carry the same level of rigour as an annotator with specialist linguistic knowledge could provide, but does have the qualitative advantage of being based on a user’s information needs rather than an abstract annotation guideline. This chapter will give details of how the application was designed to cater to users’ needs whilst also producing a rich source of data for research. It will conclude with an analysis of some general purpose usage signals captured by the software to model how users chose to interact with it. Chapter 7 will go into detail of the collected annotations and make use of them for evaluation of WSD algorithms that attempt to reproduce the semantic interpretations made by users.
Chapter 4: The Wakaran glossing tool

4.1 Introduction to the Wakaran Glossing tool

The Wakaran Glosser (“Wakaran”) is an interactive web site built for intermediate learners of Japanese as a second language (L2) to a first language (L1) of English. It provides reading assistance by means of a vocabulary reference. Specifically, Wakaran curates a full vocabulary list for a piece of Japanese text by retrieving entries from a Japanese-English (JE) dictionary for every Japanese word found in that text. The full vocabulary list is hidden by default, with dictionary entries for specific words available in a floating window that pops up when the user hovers their mouse over the Japanese word. However, Wakaran offers features to expand abbreviated definitions for additional information and to promote specific definitions to be permanently visible inline above the text. Figure 4.1 shows a number of these features in action at once. In this section we go into full detail of the features and intended usage of Wakaran.

The primary use case intended for the Wakaran Glosser is to assist students with reading
tasks prescribed by courses in intermediate Japanese as a second language. The first page a user sees when navigating to the Wakaran URL is a text submission box into which they may type Japanese text but are primarily expected to paste full text of a reading task which has previously been copied to their system’s clipboard. Wakaran assigns a unique identifier to every occurrence of a submission to this form and the full text is stored for processing. We refer to each copy of text submitted in this manner as a document submission (or, more briefly, a submission) to distinguish it from the source document’s content which is expected to be submitted multiple times by a number of students taking the same course simultaneously. Once text is submitted, Wakaran identifies tokens of words in the Japanese Multilingual Dictionary (JMDict) before displaying for the user a copy of the text with interactive features for accessing the dictionary entries. The first view a user gets of a document is illustrated in Figure 4.2. None of the retrieved dictionary entries are visible as yet: the only text is that of the original document. Wakaran allows the student to read uninterrupted until they choose to seek additional help by hovering the mouse over a word to show its translations. Parts of the text for which translations are available have been highlighted with a grey background. Before they can read the dictionary entries the user will have to take specific action to trigger their display. This quality of Wakaran is beneficial to both its research goals and application use case. For research it provides a means to keep an objective record of the dictionary entries the user has consulted. The benefits to the application use case of hiding the vocabulary aid until it has been explicitly requested are discussed in Section 4.4.

When a user hovers their mouse pointer over one of the highlighted segments of text a floating window appears showing dictionary entries related to that piece of text. Figure 4.3 shows
Figure 4.3: Hovering the mouse over a highlighted token shows glosses for all compounds that token takes part in. Text from Wikipedia (2016).
an example of the pop-up when it has just been triggered for a specific token ken. The window shows a compound shutoken “capital city area” in which the token participates and also shows two possible translations of the word ken itself: sphere/circle and category. There are many cases in which Wakaran will show multiple entries for a token. As seen in this example a token may be a constituent of one or more compounds. In some cases a token may have an exact match to an entry but also be a potential inflection of a different entry. In a non-segmenting synthetic language like Japanese, identifying all compounds and uninflected forms relevant to a piece of text is a non-trivial task. A general procedure for identifying dictionary entries matching free Japanese text is developed in Chapter 6. Its specialised implementation for JMDict in Wakaran is discussed in more detail in Section 6.4.

All of the translations shown in Figure 4.3 are abbreviated and slightly faded but have a pair of control buttons at the far right. To view a translation more clearly and in more detail the user can select it by clicking the star icon. Moreover, a translation can be promoted out of the gloss to be inserted inline in the text by clicking the other icon, which is a Japanese character for the word insert. In Figure 4.1 which shows a gloss for the token hō, the translation method is expanded and shown in high contrast because it has been selected; the translation law is highlighted and appears above the original token in the body of the text because it has been promoted. The promoted text law will remain in place even when the gloss pop-up has been dismissed by moving the mouse away. In this way the user may have a more visible and constant reminder of a translation of their choosing. Information about which translations are selected and promoted are logged by Wakaran for the purpose of research outcomes. Note that the glosses in Figure 4.1 and Figure 4.3 also have Japanese elements with controls for selection and promotion: these are phonetic readings for the words which are additional information provided by the dictionary that may be useful to a reader.

In Japanese, a phonetic reading that has been inserted above a word in text is called furigana. The inclusion of optional furigana injection as a feature is not included in the research focus of Wakaran but is an obvious extension of the translation injection functionality and implements a common information need for students.
Finally, Wakaran provides a link on the page containing the submission text to a wordlist page which shows any entries for which students have selected or promoted translations or phonetic readings. The full entry is shown but is formatted the same as the gloss pop-up to indicate which senses have been marked as selected or promoted. Figure 4.4 shows a wordlist for the running example that has been used in this section (Wikipedia 2016).

In this chapter we will explore the design of the Wakaran application and give a detailed account of user engagement with its user interface. In Chapter 7 we will look more closely at what can be learned from user interactions with the JMDict lexicon through the Wakaran interface. In addition, Chapter 7 will outline the implementation, deployment and evaluation of a WSD algorithm in Wakaran to automatically pre-select translations of polysemous words.

### 4.2 Related software

Wakaran shares features with a number of similar Japanese-English software applications for multilingual users. In this section we review a selection of the most similar applications that are
freely available. None has exactly the same set of features as the others and in a sense Wakaran unifies the main features of each of the applications reviewed here.

4.2.1 WWWJDIC

The closest related software to Wakaran is WWWJDIC: Online Japanese Dictionary Service (Breen 2016). Like Wakaran, WWWJDIC is a website with a free text glossing feature which accepts submissions of Japanese text as input and looks up entries in the Japanese-English dictionary JMDict, which was developed as part of the same project (Breen 2004). Figure 4.5 illustrates a small portion of the output of WWWJDIC’s text glossing feature on the same paragraph of text used extensively in the Wakaran examples in Section 4.1. Each line of text has its complete list of extracted entries displayed in full beneath it. As a result, the entries for the line of text in Figure 4.5 take up a significant proportion of a page. Although this provides maximal access to dictionary information it does hide the greater part of the source text at any one time. If Wakaran had taken this approach it would have made collection of linguistic data less feasible because there would not be a recordable interaction at the point when a user accesses a dictionary entry. In Section 4.4 we discuss the potential benefits to a language learner of Wakaran’s approach of displaying the complete source text and holding back the dictionary aid for optional consultation.

4.2.2 Rikai.com and Rikai-chan

The website www.rikai.com (Rudick 2016) and the derived Firefox browser plugin Rikai-chan (ffjon 2016) both extract JMDict entries for Japanese free text and hide dictionary entries behind pop-up windows in a similar manner to Wakaran. As shown in Figure 4.6 the pop-up dictionary entries of the browser extension Rikai-chan have a relatively concise presentation when compared to Wakaran, and can be applied to Japanese text in any HTML web page viewed in the browser. However, they do lack features for marking specific translations by selection and, more significantly, for promoting translations and furigana readings inline above the text.
Figure 4.5: An example of the presentation of JMDict entries retrieved by WWWJDIC (Breen 2016) for free text. Text from Wikipedia (2016).
4.2.3 Furigana Inserter

Another browser extension, Furigana Inserter is designed to insert furigana reading aids inline above Japanese text in any web page. To insert furigana above a piece of text the user highlights that text with their mouse and then activates Furigana Inserter from a right-click context menu. Furigana Inserter also integrates the pop-up window dictionary entry abilities of Rikai-chan which can be accessed by hovering the mouse over any word for which furigana has been inserted. Figure 4.7 shows Furigana Inserter’s features in action. Of the systems reviewed in this section, this one comes closest to feature parity with Wakaran, but with a proviso: only furigana readings can appear permanently shown above the original text; the translations are non-interactive and cannot
be selected, promoted or made permanently visible in any way.

4.2.4 The Sharp Intelligent Dictionary

Whitelock and Edmonds (2000) describes the Sharp Intelligent Dictionary (SID), which is a computer application that glosses English text with Japanese. The Sharp Intelligent Dictionary is presented as a reading and learning aid on the principle that breaking away from a text to consult a dictionary disrupts the learning processes. It attempts to choose the best glosses by first disambiguating the part of speech of the English words. It then seeks to segment the sentence into the largest possible MWEs with dictionary entries that exactly match the POS tags extracted. Tied MWE lengths are broken first by the number of intervening tokens and then by the probability estimates output by the POS tagger for the tags matched on the constituent tokens. Together these steps perform a rudimentary coarse grained WSD and MWE token identification in a manner similar to the RETOKENISE algorithm of Chapter 6. Interestingly, the Sharp Intelligent Dictionary then allows the user to make corrections to the segmentation by accessing rejected MWEs in a context menu and re-calculates the selection of MWE tokens based on any corrections made. The Sharp Intelligent Dictionary shows the selected translations inserted below the lines of the original text by default. It does offer the ability to hide translations below a certain reading difficulty level. However, as we will discuss in Section 4.4 the automatic appearance of all glosses (above a student’s selected reading level) removes incentive for the student to first try to infer the meaning of an unknown word, which is an activity that improves learning outcomes (Fraser 1999).

4.3 User base

The Wakaran application was presented to Japanese as a second language undergraduate students by means of a 5-10 minute guest appearance in timetabled classes giving a demonstration of the software’s features. A link to the web interface was also posted by the subject co-ordinator to the subject’s online materials, alongside a link to WWWJDIC. Students were free to use Wakaran,
any similar tool of their choice or even just a paper dictionary as an aid to understanding Japanese language texts from the subject’s assigned readings.

On several occasions — in particular, after deployment of significant changes to Wakaran—we stayed in the classroom for an additional 30 minutes after an introductory presentation to provide assistance in the case of technical problems. In this time we made the following observations of the context in which Wakaran was being used:

- Students worked in small groups, typically of 2-4 people, sharing a computer to co-operatively read and understand a Japanese text.

- Reading tasks ranged from under one page to many pages long; longer texts were split over more than one week with a portion of the text studied in a given week.

- The instructor fielded questions from students as they arose.

- Occasionally the instructor would initiate a whole-class discussion on a subtlety in a particular sentence.

The classes in which Wakaran was promoted were all second and third year university courses. The complexity of the texts and relative independence of students’ engagement with the material also indicates that the potential user base of Wakaran was working at an intermediate to advanced level of Japanese as a second language. This gives us confidence that although the users are not native speakers or linguists they will still be making informed choices about how to interpret the Japanese texts they are studying.

4.4 Design considerations

It is important for our research that the design choices for Wakaran be consistent with results of research into design of software for computer assisted language learning (CALL). We want to study the usage of Wakaran by students who are engaged with figuring out the meanings of words and not with users who are engaged with figuring out an unhelpful user interface. Moreover,
we want usage of Wakaran to be motivated by naturally arising problems for which students are seeking a practical solution rather than motivated by an artificial goal set for the purposes of the research. In this section we examine how Wakaran’s design is influenced by the findings of recent CALL literature. We will not be investigating the learning outcomes for users of Wakaran itself, rather we seek to use the findings of CALL research to increase the potential for Wakaran to provide a valuable user experience. Since usage of Wakaran is motivated by the users own needs, we will use active engagement with Wakaran as a proxy measure of how well it achieves its goals. In Section 4.6 we will examine whether actual usage of the website is consistent with an actively engaged user base.

4.4.1 Word glossing in computer assisted language learning

To date there have been numerous studies into the effects of providing contextual information for a second language reading task. They have measured, variously:

- the effects of glosses on incidental vocabulary acquisition (Kulkarni et al. 2008).
- the effects of glosses on reading comprehension (Nerbonne et al. 1998).
- the effects of varying the amount of information provided in glosses (Lomicka 1998).
- the effects of fully disambiguating sense in glosses, as opposed to forcing the learner to infer meaning from a selection of senses (Lin and Huang 2008).
- the strategies language learners use when encountering an unknown word while reading (Fraser 1999).

There are a number of published reports on the inclusion of natural language processing technology into glossing software such as Nerbonne et al. (1998) Whitelock and Edmonds (2000), which both use morphological analysis to improve identification of dictionary entries in text. Kulkarni et al. (2008) made a promising step towards using WSD in this domain by showing in controlled
user studies that reordering sense glosses using a state of the art supervised WSD method does indeed improve vocabulary acquisition for students using the system. In this section we explore the implications of these studies, and others, for the design of Wakaran.

The major use case intended for Wakaran is to provide assistance in reading comprehension. Lomicka (1998) describes an observational user study on twelve users performing reading comprehension tasks. Noting a lack of research on the effects of glosses on reading comprehension, Lomicka (1998) performed a think-aloud reading comprehension study, varying the level of detail given in the glosses. Subjects were observed to make more inferences from the text the more gloss material they were given. This result supports the decision for Wakaran to include all definitions of words from the JMDict dictionary and also to include all possible compounds each morpheme of text could participate in.

Fraser (1999) was a more comprehensive long term observational study aimed at getting a deep understanding of how language learners deal with unknown words in text and the effects this has on vocabulary acquisition. Strategies for dealing with unknown words were broken down into three parts: ignore, infer and consult. It was found that the worst strategy to take was to ignore an unknown word. Best overall was to infer a meaning but to then consult a dictionary for reference. These results have implications for glossing software such as Wakaran: it is important to be aware that any characteristic of the interface encouraging users to ignore, infer or consult on the meaning of unknown words might affect learning outcomes. In particular, designing Wakaran to hide dictionary entries until the user requests them supports the most constructive learning strategy because a user has a chance to infer the meaning of an unknown word before they have laid eyes on a translation.

Lin and Huang (2008) target an implicit challenge to glossing as an educational aid: namely that it is thought that the process of inferring meaning helps with incidental vocabulary learning and providing glosses takes away the need to infer meaning. Lin and Huang (2008) give a succinct summary of the literature relevant to this concern, covering the following key points:

1. Inferring meaning in context is thought to help with vocabulary acquisition.
2. However, making mistakes while reading is thought to hinder acquisition.

3. Therefore, student skill is considered to be a factor in the usefulness of glosses.

These results mean that Wakaran may be most helpful with less advanced users who are more likely to make mistakes inferring the meaning of unknown words. However, the results of Fraser (1999) found that checking a dictionary following an attempt at inferring meaning was more effective than inference alone. As such, the support of Wakaran for that use case should be sufficient to allow students to make use of the benefits of inference.

Lin and Huang (2008) conducted a study with 175 language students investigating two questions:

1. What effect does student skill have on the usefulness of glosses?

2. Which leads to better vocabulary acquisition: glosses which fully disambiguate sense (meaning-given glosses) or multiple choice glosses which require a degree of inference (meaning-inferred glosses).

They found that the students given meaning-inferred glosses gained more vocabulary immediately and retained it better in tests two weeks later. Results for student skill were less clear: the students classified with higher skill scored higher as expected, but differences in retention were not statistically significant. In fact the higher skill students with the meaning-given glosses retained the least of all! Lin and Huang (2008) put the lack of variability in results for students of different skill down to the homogeneity of subjects – they were all taken from the same school. Nevertheless, these results do indicate that the design choice for Wakaran to present a full list of available translations and MWEs is a good one for vocabulary acquisition because it requires some degree of meaning inference.

Though the meaning-given glosses provided to subjects in the study of Lin and Huang (2008) were manually disambiguated, it raises interesting questions regarding the purpose and nature of automatic disambiguation. Although researchers in WSD have been mostly content to find
the *correct* sense, perhaps it would also be interesting to identify senses as definitively *incorrect*.

In Chapter 7 we investigate the inclusion of automatic WSD for *Wakaran* by using a graded sense annotator to pre-select a *single best sense* which is the more traditional setup for WSD. We leave investigation of a worst-sense annotator for future work.

_Nerbonne et al. (1998)_ make the case that natural language processing technology can improve CALL systems by demonstrating the application of morphological analysis in the system *GLOSSER*. *GLOSSER* provides reading tasks and vocabulary tests on the principle that vocabulary learning is benefited by exposure to example uses of a word. The system allows the user to look up glosses including dictionary senses and information derived from the morphological analysis for words in the reading tasks. Accuracy was seen by [Nerbonne et al. (1998)] to be the biggest hurdle to the success of natural language processing in *GLOSSER*, for fear of losing the trust the user holds in the system. As such a preliminary study was conducted testing the accuracy of the morphological analysis subsystem of *GLOSSER* directly. This was followed by a user study assessing the tool as a whole. The preliminary study turned up an error rate higher than was hoped, with errors fitting into a few main categories. However these errors were not found to show up during the user study.

The user study described by [Nerbonne et al. (1998)] tested reading comprehension rather than the vocabulary acquisition of the human subjects. [Nerbonne et al. (1998)] point out that a cross lingual glossing tool is useful not only as a learning aid but also as a reading aid. Ultimately the results showed that students using *GLOSSER* did slightly better than students manually referring to hand-held dictionaries and were significantly more confident about their understanding. [Nerbonne et al. (1998)] claim that this success shows natural language processing is ready to improve CALL. However no direct comparison to a CALL system not using natural language processing is shown. Like *GLOSSER*, *Wakaran* makes use of a morphological analyser to aid the dictionary lookup process.

In the case of *Wakaran* the analysis allows us to present dictionary entries for a wider variety of inflections and compounds which has two positive benefits: (a) it is a guard against poor coverage which might reduce confidence in *Wakaran*; and (b) it means that a student consulting a gloss will still have a small meaning inference task to perform which can aid in learning outcomes.
Tabata-Sandom (2016) note the growing popularity of online Japanese glossing dictionaries, particularly for students engaging in wider reading on the web. They performed a qualitative study with 11 participants with a detailed analysis of a think-aloud procedure performed by the students to describe what they are doing as they perform a reading task. Vocabulary acquisition and reading comprehension were also tested though the small number of subjects mean the derived scores are not generalisable. Students from a large range of learning levels participated and Tabata-Sandom (2016) found a large variety of interaction levels with the pop-up glosses. The more advanced students engaged with the sentence top-down, meaning they engaged with the syntax first and only consulted glosses where they thought the meaning was important to the text. At the other end, the least advanced students consulted a large proportion of the glosses and built their understanding bottom up. Students at the intermediate level performed a mixture of both approaches. We highlight that in one of the quotes from the think aloud exercise a student sounds ready to give up on understanding a word because “there are too many different definitions... I can’t get my head around it...”. Tabata-Sandom (2016) conclude that when the text is too demanding the pop-up glosses do not help students to engage deeply with the Japanese text. They also conclude that students benefit better from reading when the language of naturally occurring texts is modified to suit pedagogical purposes. We advertised the existence of Wakaran to intermediate classes in Japanese which inflates the chances of seeing intermediate students in our study and also makes it more likely that the texts studied will be those prepared by a teacher. In Section 4.6 we analyse data collected from usage of Wakaran to establish that Wakaran was used in the context and manner we intended. However, future work may require a means of detecting student reading level, the nature of student engagement with glosses, or both.

4.5 System design

In this section we will give an outline of the design of the Wakaran application software to meet the requirements outlined in Section 4.1 and Section 4.4. Additionally, the design includes
provision for the automatic WSD features of \textit{Wakaran} described in Chapter\textsuperscript{7} to be integrated into \textit{Wakaran’s} control flow.

\textit{Wakaran} was designed as web application so as to have a low barrier to entry for its use. Since the quantity and quality of research data collected is dependent on students choosing to operate \textit{Wakaran} for their own purposes we did not want operating system compatibility nor software installation and configuration to be an obstacle or disincentive. However this choice of platform comes with a significant challenge: potentially long running NLP pipeline steps including morphological analysis, retrieval of dictionary entries and WSD (see Chapter\textsuperscript{7}) must be triggered by time-constrained \textit{HTTP} POST requests. This means that data related to document processing jobs must be shared between a series of requests. Thus a major element of the design of \textit{Wakaran} is a subsystem for spawning long running background tasks from \textit{HTTP} requests and tracking their progress in order to display the results upon task completion.

The second main architectural challenge for \textit{Wakaran} is to enable the collection of research data from user interactions with the web page front end. In this instance the decision to use a web-enabled application architecture is an advantage because it presupposes a communication channel between the application frontend and a centralised application server.

In this section, we will describe how data is stored and shared between \textit{Wakaran HTTP} server processes and how research data is collected. We will also describe the parallel job processing model used to handle \textit{Wakaran’s} NLP pipeline.

\subsection*{4.5.1 Data model}

The main elements in the \textit{Wakaran} data model for a document submission are the dictionary entries to be displayed and the spans of text to which they will be linked. Text is decomposed into paragraphs and sentences for formatting purposes and further into tokens identified by a morphological analyser. \textit{Wakaran} uses the algorithms described in Section\textsuperscript{6.4} to link dictionary entries to sequences of one or more morphemes: in the \textit{Wakaran} data model such links are called gloss entities. Dictionary entries themselves are modelled as a headword linked to a collection of read-
Figure 4.8: *UML* class diagram of *Wakaran* data model. This figure shows only the entities and attributes supporting the main use case of user interaction with translational paraphrases.

ings and paraphrases grouped into senses. Since *Wakaran* uses the Japanese-English portion of *JMDict* the headwords are in English, the readings are in the Japanese phonetic alphabet hiragana and the paraphrases are in English. To enable updating the version of the *JMDict* dictionary used in *Wakaran* whilst still enabling interpretation of research data, dictionary entries are grouped under a versioned resource identifier in the data model.

Figure 4.8 gives a *Unified Modeling Language (UML)* class diagram showing the objects and fields used to implement interactive text glosses in *Wakaran*. Each document submission is represented as a *Document* object and is assigned a uniquely identifying key. Submissions are also grouped into mini-corpora unique to the browser session. *Gloss* objects assign dictionary headwords to sequences of tokens in the document. Where a *Token* participates in multiple overlapping headword matches it will be related to more than one *Gloss*. Each *Gloss* contains a sequence of *Paraphrase* objects representing its readings and senses. *Paraphrase* is the only object class with mutable state: the *selected* and *promoted* members denote whether
the entry is shown highlighted and expanded or inserted between the lines of the source text as shown earlier in Figure [4.1]. Due to space considerations imposed by how Wakaran displays content, each Token is constrained to have at most one Paraphrase promoted to appear inserted above it. Whilst Paraphrase objects record state for a specific submission, they are proxies for concepts and readings found in a dictionary Resource which are represented by objects of the Entity class. The Entity.kind member distinguishes reading entries from true meaning paraphrases. The reading and paraphrase text itself is stored as an EntityAttribute component of an Entity. A JMDict word sense may have several paraphrases which are modelled as separate EntityAttribute objects on the Entity for that sense. A meaning Entity can also have attributes denoting POS which are differentiated from the paraphrases by the kind member of EntityAttribute.

The model as presented has one significant omission: the static Entity objects are not grouped under their JMDict headwords; grouping of paraphrases under headwords is only represented by the per-submission Gloss objects. Information about how headwords are mapped to concepts in JMDict is stored in a separate index. The links are built by the NLP process that identifies entries for linking to the text which is described in Section 6.4.

This data model is replicated in three components of Wakaran’s architecture:

1. The SQL database layer which persists the document state between the reading view and the wordlist view.

2. The server application layer where the document goes through the NLP pipeline and is rendered to HTML for the frontend.

3. The HTML and Javascript frontend which implements the user interface and communicates changes to paraphrase selected and promoted state back to the server for persistence.

In the next two sections we examine the major components of Wakaran and how the control flow of Wakaran moves data between them.
Chapter 4: The Wakaran glossing tool

4.5.2 Architecture

Wakaran is a dynamic web application meaning that users interact with it using a web browser to view pages generated at request time by handlers in an HTTP server. A HTTP handler process in Wakaran is supported by a number of other long-lived server processes used to manage persistence, batch jobs and data synchronisation. A diagram of the system architecture can be seen in Figure 4.9. Computationally intensive requests are handled asynchronously in background worker processes managed by the Celery framework. Tasks for the background workers are submitted to a message queuing service implementing the AMQP standard and task state is synchronised between processes using the memory key-value store Redis. Processed submissions are stored in an SQL database representation of the model described in Section 4.5.1.

4.5.3 Control flow

In a basic use case for Wakaran a user uploads Japanese text to the website and in response receives a copy of their text with interactive features for showing definitions of each word. The user can then select certain parts of each definition to be highlighted or pencilled between the lines of the
original text and those changes are persisted so that they will be represented on the wordlist page and when the submission view page is reloaded. As the user interacts with various features of the Wakaran user interface, events are recorded in a log on the server for research purposes.

The HTTP frontend is implemented in python as a Web Server Gateway Interface (WSGI) handler served by the uWSGI HTTP server and handles HTML generation and session management tasks. It receives new document submissions as HTTP POST requests with the text to be annotated in form data. The submission is immediately assigned a unique key based on a current timestamp.

http://uwsgi-docs.readthedocs.io/
Figure 4.11: UML sequence diagram of interactions between the Wakaran HTTP handlers and other components in the period of time between the submission of text and the first display of glossed text content for the user.
Figure 4.10 shows the sequence of events that follow from the submission of a new document. The annotation of a submission with dictionary glosses involves some potentially long running NLP tasks — especially once automatic WSD is introduced in Chapter 7 — so new submissions are placed on a task queue hosted on an Advanced Message Queuing Protocol (AMQP) server (specifically, RabbitMQ). After a new annotation task has been submitted to the queue the WSGI handler returns a HTTP redirect notice to a URL for pending tasks that can be safely refreshed in a browser without re-submission of the original POST form data. The handler for the pending task URL waits for the completion of the task to be registered in the Redis key-value store using a blocking call with a timeout of one minute. The completion of the annotation tasks is handled by asynchronous worker processes not show in Figure 4.10; the steps involved will be described in the next paragraph and are illustrated in Figure 4.12. If the task does not finish within the timeout period, the page renders as a message informing the user that the task is taking a long time and that they may refresh the page to try again. The extended wait sequence including a page refresh is shown in Figure 4.11. Otherwise, when the handler detects that the task is complete, it sends a redirect response to a URL based on the submission’s unique key from which the annotated document can be viewed using the interface described in Section 4.1 and shown in Figure 4.1.

Python worker processes spawned by a Celery framework worker daemon consume document annotation tasks off the AMQP queue and process them, periodically updating the task’s status in a shared Redis key-value store. Figure 4.12 shows the sequence of events that take place from when a worker takes a task off the AMQP message queue until the task has been fully persisted to the database. Initially, the worker preprocesses submitted text to detect sentence and paragraph breaks. The sentences are then tokenised and dictionary entries relevant to the text are identified using the algorithm described in Section 6.4. When the process of identifying dictionary entries for the text has completed, a copy of the glossed document is sent to the Redis store to be immediately accessible to HTTP requests and a flag is set. After this point, HTTP request sequences such as

---

1. https://www.rabbitmq.com/

Figure 4.12: UML sequence diagram of interactions between the Wakaran batch worker processes and other components in the period of time between the appearance of a new document on the AMQP queue and the persistence of the annotated document to the SQL database.
those shown in Figure 4.10 and Figure 4.11 can proceed from the submission pending URL past the block on the content being ready to the document view URL. However, the submission still needs to be persisted to the database before changes to the selected and promoted states of glosses can be reflected on the server. The worker persists a copy of the result to the SQL database which uses a schema representation of the model described in Section 4.5.1. Once the submission has been persisted its new state is flagged in the Redis store so that the HTTP handlers can start accepting changes to gloss state. The Redis copy of the document is removed: subsequent requests to the document’s URL will fetch the document from the SQL database. Finally, the Celery worker saves a static copy of the document to the database as a record of its initial gloss state. In the basic version of Wakaran the glosses all start in the same state so this snapshot is redundant but in Chapter 7 we introduce automatic WSD features of Wakaran that set some paraphrases to be preselected in the initial version of the submission.

After a complete annotated document has been returned to a user, the user may interact with it using the features described in Section 4.1. A number of user interface events are communicated back to the server. The most important are the paraphrase selection and promotion events which are sent to the server and reflected in the wordlist page if the user navigates to it. Section 4.6.2 gives full details of other user interface events that are logged by Wakaran. Figure 4.13 gives a simple example of the sequence of calls between components of Wakaran as the user interacts with a paraphrase in a gloss. Any user interface event that changes the selected or promoted state of a paraphrase is immediately forwarded to the HTTP handler. The handler then checks the submission’s state in Redis in case the worker that processed it has not yet finished persisting it to the database (see Figure 4.12). Once the HTTP handler can be certain the submission has been persisted it updates the paraphrase in the database. Thus the change to the paraphrase will be reflected in the wordlist page and will persist if the submission view page is reloaded. Note that, in the example, the frontend queues up miscellaneous user interface event logs and only sends them to the server when an event occurs that changes the state of one of the paraphrases. However, during general usage queued event logs are set to be flushed to the server on a 5 second timer if no paraphrase state
Figure 4.13: UML sequence diagram of interactions between the user’s web browser and the Wakaran HTTP handlers as the user selects and promotes paraphrases in dictionary entries linked to a submission.
mutation occurs before then.

In this section we have described details of Wakaran’s architecture with special attention to the way Wakaran delivers the results of potentially long running NLP glossing algorithms to users of its web interface. In the next section we develop a picture the collected data gives of the Wakaran web application in action.

4.6 Evaluation

In this section we present results of data gathered from usage of Wakaran. We use data logged from the usage of Wakaran to discover whether users are engaging with the Japanese language through its features. Based on the casual observations made of our intended classroom use case described in Section 4.3 we expect users to have been progressing through a document in a linear fashion, with the focus of their interaction moving forwards through the document over time. In addition to this, we received a few responses to Wakaran’s user feedback submission form which are indicative that students do find the gloss pop-up and inline insertion features useful. However, the observations and feedback do not tell us the prevalence of this kind of usage nor the level of interaction with Wakaran’s interactive features. In this section we apply time series analysis and clustering methods on logs of Wakaran’s usage to reveal the level and nature of interaction that has occurred. In this chapter we do not examine the content of document submissions but instead seek to describe as fully as we can the interaction that occurred between users and Wakaran. Chapter 7 will then describe how we used the record of those interactions for research into word sense disambiguation resources and methods.

4.6.1 User feedback

Wakaran includes a user feedback form running through Google Docs4. The goal of the feedback form was to verify qualitatively that students are using Wakaran for its intended purpose

---

4Using https://docs.google.com/forms
The survey was broken into four parts. The first part asked for the most general feedback on the user’s experience with *Wakaran* with just two questions. The first question asked: *How satisfied are you overall with *Wakaran*?* The response average was 3.857 on a 1 – 5 Likert scale.

The other question was *Approximately how many times have you used *Wakaran*?* The responses are shown in Table 4.1. These results indicate that the respondents were mostly return users to *Wakaran* who were moderately happy with the application overall.

The next section was a group of questions asking the user to respond to certain statements about *Wakaran* by rating them from 1 – Strongly disagree to 5 – Strongly Agree. It read as follows:

Please rate the following statements according to how strongly you agree. Base your answers on the relative utility of *Wakaran* as compared to looking up words in a hard-copy dictionary.

The questions and average responses are shown in Table 4.2. The results indicate that the respondents see *Wakaran* primarily as a tool for reading comprehension and only see vocabulary acquisition as a secondary usage.
The following section allowed the user to select from multiple options per question. The section was introduced with the text:

Could you give us more detail about how you use Wakaran? Please check the boxes that apply to you.

Table 4.3 shows the responses. The first question asked what situations the respondents had used *Wakaran* in. All respondents selected that they used *Wakaran* in class or in preparation for class (or both). These results show that the respondents only used *Wakaran* for their course materials. Since the users giving feedback are most likely self selected to be amongst the most active users, this makes it likely that *Wakaran’s* usage was limited to the classroom setting, despite being available at a public *URL*.

The next multiple choice question asked which features the respondents found most useful, with the clear winners being insertion of translations and of readings inline in the text. Since insertion of translations is one of the most explicit interactions with word meaning performed by a user its popularity as a feature is a good sign that the research purpose of *Wakaran* is in agreement with its usefulness as an application. In other words, the feature which has the most usefulness to research data generation is also the one that the respondents find most useful.

The remaining three questions in the multiple choice section were designed to elicit priorities for potential improvements to *Wakaran* including better handling for MWEs and improvements to page load times and responsiveness. The most popular choices were stacking of inserted translations with furigana, long term access to word lists, and more aggressive filtering of the word list (to remove unselected senses completely). This feedback may be used to guide future work on *Wakaran*.

The final section asked for comparisons to other dictionary tools that might be used for the same purpose as *Wakaran*. Respondents were able to select one or more tools from the following list:

- Rikai (rikai.com)
- Rikai-chan (browser extension)
### Table 4.3: A sequence of survey questions regarding Wakaran allowing selection of multiple responses each.

<table>
<thead>
<tr>
<th>Question</th>
<th>Response</th>
<th>Count</th>
</tr>
</thead>
<tbody>
<tr>
<td>What contexts have you used Wakaran in?</td>
<td>In class</td>
<td>6</td>
</tr>
<tr>
<td></td>
<td>Preparation for class</td>
<td>5</td>
</tr>
<tr>
<td></td>
<td>Revision</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>Browsing the web</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>Other (please specify)</td>
<td>0</td>
</tr>
<tr>
<td>What features of Wakaran do you find useful?</td>
<td>Selection of one or more translations</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>Insertion of translation into text</td>
<td>5</td>
</tr>
<tr>
<td></td>
<td>Selection of one or more readings</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>Insertion of reading into text (furigana)</td>
<td>5</td>
</tr>
<tr>
<td></td>
<td>The word list page</td>
<td>0</td>
</tr>
<tr>
<td>When wanting Wakaran to translate a word, have you ever found:</td>
<td>No translations for the word?</td>
<td>3</td>
</tr>
<tr>
<td></td>
<td>All translations are inappropriate?</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>The word is part of an untranslated longer expression?</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>A longer expression is translated but is not relevant?</td>
<td>1</td>
</tr>
<tr>
<td>What features would you like to see added to Wakaran?</td>
<td>Reduced word list displaying only selected translations</td>
<td>2</td>
</tr>
<tr>
<td></td>
<td>Reordering translations for a word</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>Ability to delete translations</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>Automatic selection of relevant translations</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>Automatic removal of irrelevant translations</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>Concurrent insertion of translation and reading (furigana) into text</td>
<td>3</td>
</tr>
<tr>
<td></td>
<td>Ability to save selected translations and word lists online</td>
<td>3</td>
</tr>
<tr>
<td></td>
<td>Other (please specify)</td>
<td>0</td>
</tr>
<tr>
<td>Have you had any of the following issues when using Wakaran?</td>
<td>Difficulty opening and closing translation popups</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>Long lists of translations are difficult to read</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>Page becomes unresponsive</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>Page takes too long to load</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>Other (please specify)</td>
<td>0</td>
</tr>
</tbody>
</table>

- WWWJDIC (Jim Breen’s website)

- Electronic handheld dictionary

- Paper dictionary

- None

Of those, only the Rikai-chan browser extension and the WWWJDIC website were selected (although one user indicated in a later comment that they were in fact using an offline application for
Respondents were asked to comment on their choice and all provided some comments. The common themes were:

- The Rikai-chan browser plugin is easier to use for general web browsing because it doesn’t require going to a specific web page and copying text.
- The inline translation and furigana insertion features of Wakaran are its main strength.
- The compactness of Wakaran’s display format makes it better for longer texts than WWWJDIC.

Overall, these results are indicative that Wakaran will suit its initial target audience of students in classroom activity: in this situation it is less important that the tool be useful for general purpose web browsing. However, to fully realise the potential for data collection Wakaran would need to be useful for general web browsing so a browser plugin implementation is considered a high priority to enable future research.

The user feedback discussed in this section gives a qualitative indication that Wakaran is well suited to its design purpose as a user application for students of Japanese as a second language, and additionally that the features most relevant to the research goals are the ones that users will find most useful. However, having had only seven responses these results do not indicate the overall level of usage of or interaction with Wakaran. In the following sections we give statistics and visualisations of Wakaran usage based on data from user interface instrumentation.

### 4.6.2 Instrumentation

To measure interaction we instrumented the frontend to log the following user interface interactions which are given in increasing order of the significance we attach to them:

**Token mouseover (Mouseover)**

The mouse pointer passes over a token in the document.

**Gloss show (Show)**

The mouse pointer hovers over a token long enough to show its gloss.
Paraphrase selection change (Selection)

The user selects or deselects one of the definitions or readings in a displayed gloss.

Paraphrase promotion change (Promotion)

The user chooses to promote a definition or reading to insert it inline, or removes a previously inserted gloss.

When a definition or reading is inserted inline it may force the removal of a previously inserted gloss. For this reason, we additionally logged separate events for:

Insert ruby (Insert)

Insertion of ruby text.

Remove ruby (Remove)

Removal of ruby text

These latter two events are not separate user interactions but rather occur in response to a gloss promotion change.

Usage data logs are tied to a single document ID in the Wakaran data model (see Section 4.5.1). In addition to this, the token mouseover and gloss show events are linked to a specific token in the document. Likewise, the paraphrase selection change and paraphrase promotion change events are linked to a specific gloss. Figure 4.13 and the discussion in Section 4.5.3 describe how these events are communicated back to the Wakaran HTTP server and persisted to the SQL database.

4.6.3 Usage by day

Wakaran was introduced to users by means of demonstrations during timetabled undergraduate classes (see Section 4.3). Use of the software was at the student’s discretion and all of the respondents to the feedback questionnaire were aware of competing applications (see Section 4.6.2). Demonstrations of the software were given during the early weeks of semesters one and two of 2013.
In this section we examine the days on which Wakaran was used and which days it received its highest usage. The dates and times of Wakaran usage begin to give us an idea of the purposes it was used for.

Figure 4.14 shows days in the period 2013 and 2014 in which Wakaran received some usage. Overall, 2286 submissions were made to Wakaran, though only 1902 showed activity in the logs and it is the latter set that we show in our results in order to filter out submissions that were made but never used. We observe that Wakaran received its highest usage:

- In the semesters it was actively demonstrated.
- Only during semester.
- Mainly on working days (concomitant with class times).

Note that although usage is highest in 2013 there is a substantial amount of usage in semesters in which Wakaran was not actively promoted by us in student classes. We cannot discount the possibility that teaching staff of the subject may have introduced Wakaran of their own accord. However, we nevertheless draw two conclusions:
Figure 4.15: Heatmap showing distribution of new document submissions across the week during semester 1 2013.

- Direct promotion of the application by means of demonstration had a substantial positive impact on usage levels.

- The application was sufficiently appealing to sustain some level of usage without active promotion from the developer.

We justify the second conclusion on the basis that neither teaching staff nor students were given any incentive or requirement to promote or use the application outside of the benefits of the application itself.

The pattern of usage shows clearly that Wakaran was mainly used during weeks that classes were running at the University, but was it used during classes themselves or for self-study? Figure 4.15 is a weekly time of day histogram showing the days and times at which Wakaran received the most usage during semester 1 2013. There is scattered usage throughout the week of
times when one or two submissions were made during the semester but the heaviest usage was on Monday evening and Tuesday morning. As was suggested by the content of the feedback survey (see Section 4.6.1) these usage times most likely correspond to preparation for and participation in classes scheduled for Tuesday morning.

Usage dates and times alone are enough to determine that *Wakaran* was used chiefly for the purpose of study of class materials. However, we would like to know more about what users actually did while they were using the website. In the next sections we look first a few specific examples of how *Wakaran* was used, and then generalise into broader patterns of usage to get the big picture of what *Wakaran* was used for.

### 4.6.4 Usage across a session

So far we have looked at the days and times at which *Wakaran* was used and they indicate that it was used mainly for coursework but we don’t yet have any details of the extent to which students used the features of *Wakaran*. In this section we build a picture of what the real-world usage of *Wakaran* might look like by examining the details of a few selected usage sessions. We will posit that these examples demonstrate determined and considered usage of the software. In the following sections we look at ways to mine the full set of submissions to see how much of *Wakaran*’s usage demonstrates the same characteristics.

The logs attached to a submission track user interface events from the initial submission of the document into *Wakaran* to the end of the browser session. In this section we examine the logs collected for single submissions to demonstrate that *Wakaran* is being used for its intended purpose (by at least some users).

Since the logs include an entry for each time the mouse passes over a token in the text and each time the user interacts with a gloss, we can get a picture of whether or not a submission was used or sat idle by plotting the location of logged interaction events against time. Two such plots are shown in Figure [4.16a](#) and Figure [4.16b](#). They have been shaded to show interactions with glosses most prominently but the density of the data means that they show the overall patterns of interaction.
(a) In this submission interaction with the glosses occurs in more or less monotonic progression through the document but the user frequently sweeps the mouse across the full length of the document. This behaviour could indicate parking the mouse in a position away from the text where it would not accidentally trigger a gloss.

(b) In this submission the user makes a single steady sweep of the document with very little deviation of mouse position from that sweep.

Figure 4.16: Plotting document position versus time for logs of interaction with two example document submissions.
Figure 4.17: Plotting position versus time for events logged on a variety of submissions.
rather than the specific details. In both sessions, the gloss interaction appears to occur in a more or less monotonic line that progresses steadily from the start of the document to the end. Each of these sessions also exhibits a distinctive behaviour that is consistent throughout the lifetime of the session: In Figure 4.16a the mouse frequently sweeps between the focus of gloss interaction and the end of the document (or, sometimes, the start of the document); in Figure 4.16b mouse movements appear to be clustered in the near vicinity of the gloss interaction. It is likely that in the case of Figure 4.16a, the user is positioning the mouse outside the text area to avoid accidentally triggering pop up glosses. The mouse position events in this plot appear roughly in regularly spaced bands mirroring the shape of the gloss interaction curve. Document position is recorded as an offset in the document text so this phenomena is most likely due to the wrapping of the text into lines. This text can be inferred to have been wrapped into seven or eight lines with a little over 50 characters in each. It should be noted that Figure 4.16b represents a much longer submission and so positioning the mouse past the end of the text may have been impractical. Additionally, the user spent much less time interacting with the document: of the order of 25 minutes as opposed to almost an hour and a half. In this shorter amount of time, and working with a longer text, this second user has consulted glosses consistently throughout but has not stopped to promote many glosses for insertion inline. Conversely, Figure 4.16a show extensive use of gloss promotion features. Together these two sessions both show users working their way through a document methodically, but with some distinctly different behaviours in each case. The mouse hover logs for several more submissions are shown in Figure 4.17. They show that Wakaran saw a wide variety of levels of usage and usage patterns. In the next section we discuss a method to study the prevalence of different patterns of usage. However, before we proceed, we will examine some aspects of the logs for single submissions in a little more detail.

The submissions we have seen so far showed users progressing monotonically through the text. For some documents, such as the one whose gloss show events are shown in Figure 4.18, two-pass piecewise monotonic patterns emerge. This means that the user has triggered the display of glosses for glosses in one pass of the document and then returned near to the start and repeated...
Figure 4.18: Plotting document position versus time for interaction logs. In this submission the user makes two passes over the document. The user appears to have promoted glosses inline in first pass, but only consulted glosses in the second.

the process. Noting that the second pass takes more time (has a smaller slope) we might suspect that the user had a different purpose in each pass. Indeed, in the first pass many paraphrases were promoted to display inline, whereas in the second pass the glosses are shown but few meanings are selected. A likely interpretation is that the user made word meaning selections in advance before attempting to read and understand the document as a whole.

These single examples don’t show the extent to which features of *Wakaran* were used overall. They also don’t show whether the monotonic or two-pass behaviour observed with respect to showing glosses is widespread. In the next section we look at statistics across all logs taken and cluster submissions according to various features of the logs taken for them.

### 4.6.5 Clustering usage patterns

In order to discover patterns in the usage of *Wakaran* across collections of submissions we extracted several kinds of feature from the logs, mapping each submission to a point in a high dimensional real vector space. We then employed clustering techniques to examine the set of all
submissions for the characteristics we observed in individual usage sessions in Section 4.6.4.

**Event type frequencies**

One factor in usage patterns we wish to explore is differences in usage of different features of *Wakaran*. For example, in the previous section we saw plots of logs for two submissions in Figure 4.16a and Figure 4.16b where one user slowly promoted translations for display inline whereas the other moved relatively quickly through the document viewing glosses but not interacting with them. For each submission we extract the count of user interface events separately for each type of event (see Section 4.6.2 for the event types). These counts represent the extent to which each user interface feature was used in a session.

Using raw counts means that the feature vectors will distinguish more active or long running submissions from shorter ones by means of the substantial difference in raw event counts. However, we are also interested in the weight assigned to different kinds of events. Generally speaking the less frequent event types — selection of definitions and promotion inline — are the ones of greater interest to us, but occur less frequently. The small magnitude of these features limits the influence they can have on clustering: for example, any algorithm using the $l_2$ metric will care less about features with small absolute variances. Therefore, we scaled each feature by its standard deviation as measured across all documents. Note that the sparse features (that is, those that are frequently zero) will have low variance and therefore will scale to relatively large values when they do occur. Since the rare features are the ones we are most interested in we do not consider this side effect of standardisation undesirable.

Figure 4.19 shows the distribution of event type features across all submissions collected. It is a heatmap with the dimensions of the feature vector representation on the x-axis: each has a dimension number and a feature name. The y-axis represents the feature value: this is the normalised count of events on a single submission. The luminance of each box represents a count of submissions within its corresponding range of y-axis values. Thus each column in the plot represents a histogram of values for a feature.
Each feature type is dependent on the one to its right to occur. For instance, a paraphrase cannot be promoted before being selected first, and a gloss cannot be shown without a token mouseover to precede it. However, due to the normalisation used the distribution of features forms a peak at the gloss show event before tailing off for the more rare events. Note that the scaling has no effect on documents with zero logs for every event type. For the documents that did see a non trivial amount of activity, normalisation means that the event count features all fall within roughly the same order of magnitude and should contribute equally to a clustering, though perhaps with the degree of gloss show events dominating the results somewhat.

We applied the $k$-means clustering method to the featurespace of event counts to form $k = 10$ clusters. The number of clusters was chosen to be a reasonable number of clusters for manual interpretation whilst still being large enough to have a moderate discriminatory power. The largest cluster, of 861 documents, saw relatively little activity and so is of little interest. The remaining clusters each fit into one of three groups:

1. showing glosses but not interacting with paraphrases,
Chapter 4: The Wakaran glossing tool

(a) A cluster of 211 submissions with little interaction other than showing glosses.

(b) A cluster of 33 submissions with moderate usage of interactive features.

(c) A cluster of 25 submissions with heavy use of inline gloss promotion.

Figure 4.20: Feature distribution and sample cluster members for three clusters based on event counts features. Cluster members are represented by position-time plots using the same key for event type as Figure 4.18
2. selecting many paraphrases but little to moderate inline insertion.

3. high use of inline insertion.

One cluster from each group is shown in Figure 4.20.

The first group comprises the second to fifth largest clusters and the seventh largest. Each has a similar feature plot weighted towards the token mouseover and gloss show events with a drop-off for events arising from interactions with specific paraphrases. The feature heatmap for the fourth ranked cluster is shown in Figure 4.20a, along with position time plots for a number of submissions sampled from the cluster. Some submissions in this cluster do have a small amount of interaction with inline promotion of gloss content, but it is small in proportion to the distribution of the gloss show feature. The other clusters in this group look very similar but the total level of interaction with the submissions increases as the cluster size decreases. The cluster sizes are 315, 306, 211, 102 and 31 for a total of 965 submissions.

The sixth largest cluster, of 33 submissions, is shown in Figure 4.20b. This cluster forms a bridge between the gloss-only clusters and the clusters with heavy interaction with individual paraphrases. Its members show a distinctively high proportion of paraphrase selection but mainly do not show high counts of promotion of paraphrases inline in the text. Finally, the three smallest clusters, with 25, 17 and 1 submissions for 43 total, show a high level of usage of inline insertion of gloss content. Figure 4.20c shows the middle of these three clusters. As with the first group discussed above, the level of activity increases as the cluster size decreases.

This clustering on event counts gives a general picture of the level of interaction with document submissions. It shows that there were just under 900 submissions with very little activity and just under 1000 submissions for which many glosses were viewed but relatively few paraphrases were inserted inline. The remaining 76 submissions showed a very high level of interaction with paraphrase selection and inline insertion features. These high interaction submissions will be the richest source of information about user interaction with definitions but in Chapter 7 we will discuss a method for combining paraphrase interaction information from the submissions with lower counts
of interaction events as well.

In the previous section we saw an example of a session in which the user read through the submission in one pass in Figure 4.16b but also a session where the user took two passes through the document in Figure 4.18. The sample event plots drawn from clusters for Figure 4.20 show that there are in fact a number of instances of both reading patterns in the history of Wakaran’s usage. However, in this clustering different patterns are mixed in together into the same clusters. In the next two sections we look at timeseries analysis methods to estimate the prevalence of each pattern.

**Autocorrelation**

In this section we develop a clustering of document submissions based on a standard time-series statistic called autocorrelation. Autocorrelation is a measure of how predictable a timeseries’ future values are from the values that come before it. It is defined as a cross correlation of the timeseries values when paired with values from a version of the timeseries that has been offset in time by a fixed time lag. In order to get a complete pairing, the forward copy of the timeseries has to omit samples from the start for a time period equal to the length of the lag. Similarly, the lagged copy of the timeseries omits samples from the end for a time period equal to the lag. Autocorrelation is simplest to define in terms of regularly spaced time series – that is, series with a constant sampling time period. In this section we adopt a definition of autocorrelation for time series with arbitrary length times between samples. We then use a predefined set of lag times to produce a vector of autocorrelation features for clustering submissions.

For a regularly spaced timeseries \( x_t \) with \( N \) samples, the formula for autocorrelation with a lag of \( n \) timesteps is:

\[
\sum_{t=n+1}^{N} (u_t - E(u))(v_t - E(v))/\sigma(u)\sigma(v)
\]

where \( u \) is the forward samples:

\[
u := (u_t = x_t | t = n \ldots N)
\]
$v$ is the lagged samples:

$$v := (v_t = x_{t-n}|t = n \ldots N)$$

and $E$ is the expected value (arithmetic mean) and $\sigma$ is the standard deviation. However, in our dataset, logs can come anywhere from a few microseconds to several minutes apart so it is undesirable to treat the time spacing as regular. It is also impractical to aggregate logs into time buckets since the burstiness of the data will inevitably leave holes in the regularised series while also smoothing over potentially large variations in position within a single bucket. Instead, we choose a lag time $T$ in seconds and pair each sample in our timeseries by going back $T$ seconds and finding the first sample at or before that time. That is, for $X = ((x_i, t_i)|1 \leq i \leq N)$ we define:

$$u := (x_i|\min_j(t_j) + T \leq t_i \leq \max_j(t_j))$$

As a helper, we define an interpolation function $I : [\min_j(t_j), \infty) \rightarrow \mathbb{R}$ where:

$$I(s) := x_j \text{ where } j = \max(i \mid t_i \leq s)$$

Then we can define the lagged timeseries as:

$$v := (I(t - T)|\min_j(t_j) + T \leq t_i \leq \max_j(t_j))$$

We design an autocorrelation feature space by choosing a set of quadratically increasing time lags of 1 second, 4 seconds, 9 seconds and so on through to 3600 seconds (that is, one hour). The quadratic stepping allows us to get detailed information at the sub-one minute level whilst still getting a reasonable 1-2 minute stepping at the half hour to hour level. For each submission we calculate autocorrelation at each of these lags but only for lags up to half the total session time. Our reason for stopping at half the session length is that to use longer lags would leave the centre of the timeseries out of the analysis completely.
Chapter 4: The Wakaran glossing tool

(a) For this submission, the extremely linear and compact time series results in high correlation for all lags.

(b) A two-pass reading of the document gives a negative correlation for lag distances that put the passes out of phase with each other.

(c) The sweeping motions of the mouse in this document cause the autocorrelation to be low, even at very short lag timespans.

Figure 4.21: Comparing timeseries with the corresponding autocorrelation plots.
Figure 4.21 shows a number of submission position-time plots with their corresponding autocorrelation plots (also known as a correlogram). We can see that for the relatively linear usage session shown in Figure 4.21a, the corresponding autocorrelation is high for all lag amounts. This makes sense as we would expect any offset version of this timeseries to look very much like the original (once offset by its mean).

The two-pass reading session shown in Figure 4.21b does something quite different. For lag times that align the bottom of the first pass with the top of the second, the correlation becomes negative. The autocorrelation calculation trims the start off one side of the comparison and the end off the other. Thus, for certain lags, the ‘forward’ copy of the timeseries is trimmed to start near the end of the document, backtracks to the start then returns to the end again. Conversely, the ‘lagged’ copy starts at the beginning of the document, tracks to the end and then returns to the beginning. Of course, once the lag is sufficient to skip the entire first pass, the comparison is now between two timeseries that start near the beginning of the document and end at the end, so the correlation returns positive for larger lags. In fact, for any periodic timeseries autocorrelation tends to be highly positive at offsets that are multiples of the period and negative when the lagged series is “out of phase” with the original. In this way, our two-pass document readings could be seen as a special case of the periodic phenomena. This pattern suggests that clustering on autocorrelation features should be able to reveal the subset of documents with two passes.

Some issues do arise in using autocorrelation values as clustering features across our complete set of submissions. Figure 4.21c shows a case where the user has been sweeping the mouse away from the main focus of reading. The autocorrelation drops off much faster than either of the other two examples, which is the effect of having a lot of rapid down and up sweeps of the mouse. This suggests that submissions with active mouse activity between gloss consultation could be detected via autocorrelation of token mouseover logs with lags under about thirty seconds. However, in this section our purpose is to measure longer range trends in gloss interaction. Therefore for the purposes of clustering we removed the token mouseover events from the timeseries and additionally used only lag values of 36 seconds or more. To ensure that the gloss interaction events occurred in
sufficient number to represent a reading curve, we also limited this clustering to submissions with at least 10 events remaining after filtering.

To make autocorrelation features work well for clustering the overall trend in reading focus, additional restrictions were necessary. Note that the session shown in Figure 4.21b lasted for a longer time than the one shown in Figure 4.21a and consequentially has autocorrelation features for longer time lags. Our chosen clustering algorithm, $k$-means, does not handle sparse features so the maximum lag usable for an autocorrelation feature is effectively capped by the shortest duration submission in the collection. On the other hand, we are trying to detect longer term trends in reading style so we do not want to eliminate longer lags too aggressively. In particular, the strongest indicator of the two pass reading session shown in Figure 4.21b is the negative correlation values that all occur at lags larger the the maximum lag of the shorter session. As a compromise, we limited autocorrelation clustering to the set of submissions with event series duration at least 1252 seconds and consequently capped the maximum lag at 625 seconds. Relative to a one hour class room reading activity these seem to be reasonable timespans to study.

After filtering for sessions meeting our criteria for minimum duration and number of gloss interaction logs we were left with 411 submissions to cluster. We extracted autocorrelation features for a series of lags on a quadratic scale, starting at 36 seconds and ending at 625 seconds. The quadratic scale was chosen to capture more detail in movements over a short timespan while still getting more detail at the longer timespans than an exponential scale would have yielded. Each feature was normalised by its standard deviation as we did for the event count features in the previous section. The final feature distribution appears in Figure 4.22. Since normalisation scaled features without translating them the distribution of positive and negative autocorrelations can be clearly seen. It is evident that a large number of submissions maintain a high level of autocorrelation even for very long lag distances but there is also a significant population of submissions which drift into negative correlations for larger lags.

We again used a $k$-means clustering, this time with only $k = 5$ clusters since we are mainly looking for two specific phenomena. In particular, we are seeking to divide single pass from
Figure 4.22: Heatmap showing the number of documents with a given count for a given autocorrelation feature.
Chapter 4: The Wakaran glossing tool

(a) A cluster of 139 submissions with high correlation across all timeseries lags and monotonic time series.

(b) A cluster of 66 submissions with low or negative correlation at many timeseries lags and non-linear timeseries plots for gloss interactions.

(c) A cluster of 59 submissions with high correlation for short timeseries lags trailing off to zero or negative correlation for longer offsets. Many of these timeseries show two pass reading behaviour.

Figure 4.23: Feature distribution and sample cluster members for three clusters based on autocorrelation features. Cluster members are represented by position-time plots using the same key for event type as Figure 4.18.
multi pass readings of the submission. A small buffer above two clusters was allowed in case of unexpected behaviour. To understand the results we once again inspected feature distributions and sampled timeseries plots from each cluster: three clusters are shown in Figure 4.23. The two largest clusters exhibited high autocorrelation for all time lags and both consist of submissions showing monotonic single pass reads of the text. The larger of the two is shown in Figure 4.23a; the other had a slightly lower level of correlation in the tail and its timeseries showed a few gaps or oddities. These clusters contained 139 and 80 members for a total of 219 monotonic submissions. The third largest cluster, of 67 submissions, had a feature distribution with a similar downward trend and a broader spread that reached down to zero at the largest lags. Its members showed substantial monotonic segments but had multiple discontinuities or segments with unfocused activity on glosses. The fourth largest cluster had 66 members and is illustrated in Figure 4.23b. This cluster comprises submissions with low and negative correlations at relatively short time lags and the event plots all show quite mobile interaction across the length of the text. Finally, the smallest cluster had 59 members. Illustrated in Figure 4.23c it is a good candidate for a cluster of two pass readings of the text. Around half of the samples show two or more clear linear passes of the text in the gloss show features and some in the gloss promotion features. Some of the rest show an interesting V or inverted V shape including one pass of reading the text proceeding backwards from token to token. However, the samples show that this cluster includes a few submissions that do not fit into the desired category of periodic sweeps of the text. The autocorrelation feature values are distinguished in this cluster by being high valued for smaller time lags but dipping at or below zero for the longer time lags. Although a periodic sweep of the text does produce this pattern of features, there are other causes of low correlation at longer time lags.

Clustering on autocorrelation features gives some insight into the distribution of user behaviour in our collection of document submissions but has its limitations. This method proved effective at identifying monotonic single pass readings of the text and so could be used as a cleaning step were that the desired result. Highly uncorrelated timeseries were also successfully identified which has a similar benefit. One limitation was that the method could only be performed on a subset
of the collection due to constraints on both session duration and gloss interaction count. Although the parameters chosen for those constraints were not overly severe it would still be preferable to gain insight into a greater subset of the full collection of submissions. The other limitation of this method is that it did not cleanly separate periodic timeseries from the moderately uncorrelated ones. On the other hand, the cluster with higher correlation on shorter time lags and negative correlation at higher time lags did serve to highlight some unexpected but interesting behaviours involving piecewise monotonic reading sessions. Overall this clustering gives interesting insight into the distribution of user behaviours in the subset of submissions that had a high level of interaction over a longer period of time.

4.7 Conclusion

In this chapter we established the design of Wakaran as a novel text glossing software application capable of simultaneously serving a naturally arising use case whilst also collecting information about human interpretation of word senses. Wakaran’s novelty stems from linking a user information need to a NLP research question via the design of interactive paraphrase visibility controls. We established the suitability of Wakaran for its second language learning use case in three ways: comparison to existing software, user feedback surveys and analysis of usage logs. Finally, we studied detailed logs collected from usage of Wakaran to verify that users engaged with its interactive features for controlling paraphrase visibility in order to help understand the meaning of the text.

The design of Wakaran specified in this chapter solves several challenges related to its application and research goals:

- Making potentially long running NLP services available in a user friendly way over a HTTP interface.

- Improving user experience by hiding information whilst designing visibility controls to correspond to desired research data.
Collating logs of user interface events from remote clients to the research database.

In Chapter 7, we will utilise Wakaran’s architectural design qualities more fully by introducing a new longer running NLP component. The major elements of this design will be reusable in future work in NLP resource construction. The improved user experience by hiding information is partly validated by user feedback regarding screen space for the submitted text but most importantly the visibility enable the infer+consult method for approaching unfamiliar words as determined by Fraser (1999) to be the most effective strategy for vocabulary acquisition.

Comparison to similar applications indicates that there is user demand for the functionality offered by Wakaran. There exist a number of popular web applications with the same intended usage as Wakaran with features that overlap to varying degrees. Wakaran is notable mainly for its interactivity allowing the user to control the extent to which translation information is exposed or hidden away, at a paraphrase granularity. The evaluation in this chapter shows that Wakaran garnered a user base who made substantial use of those features.

User feedback confirmed for a small number of users that Wakaran was found useful in its intended use case for second language class work and preparation. Feedback indicated that in contexts outside class work other similar applications with browser plugins and the ability to annotate text in arbitrary web pages were more useful than Wakaran. This information will have bearing on future work expanding the range of usage and data collection with Wakaran.

Finally, we measured the extent and nature of interaction with Wakaran's features in the full user population in a variety of ways. Firstly we established from counts and timestamps of text submissions made to Wakaran that Wakaran was used primarily on semester days during specific contact hour timeslots and on the nights before those timeslots. We concluded that Wakaran was primarily used for its intended use case of language learning reading activities. Next we looked at the distribution of feature usage intensity across document submissions. Although many submissions were barely used or not used at all, around half saw active use of Wakaran's interactive features. Finally, we looked at submissions with active feature use and moderate session duration to identify categories of reading behaviour. We found that around half of the chosen submissions
Chapter 4: The Wakaran glossing tool

showed users reading monotonically through the document but a non trivial number showed users reading the document in multiple passes, sometimes with both forwards and backwards passes. We conclude that users were engaging with Wakaran’s interactive features for paraphrase visibility and were actively using them to help understand the meanings of words in the text. On the basis of this we are satisfied that Wakaran’s usage logs are a valid source of information about user interaction with Japanese paraphrases of English words.

With the design of Wakaran for its second language learning use case established, we can turn to its purpose as a means of collecting research data on natural language. User interaction with visibility controls for paraphrases in Wakaran provides a coarse record of level of interest in specific translational paraphrases of Japanese words. In Chapter 7 we will investigate the data collected about word senses and discuss the implementation of WSD for Wakaran glosses. However, we first investigate some theoretical topics with applications in gloss selection. Chapter 5 looks at identification of multiword expressions and how to distinguish them from ordinary usages of their component words. Chapter 6 will then go into detail of how entries from a lexical resource can be identified in text of a non-segmenting language like Japanese. It concludes with experiments in WSD over a particular lexical resource, JWordNet, which leads into the work of Chapter 7.
Part III

Disambiguation
Chapter 5

Multiword Expressions

A multiword expression (MWE) is a specific combination of lexicalised words which, when used or seen together, act in some marked way as a separate lexicalised unit. In this thesis we deal exclusively with MWEs which have already been encoded in a dictionary and thereby take it as assumed that they are lexicalised. This chapter covers the identification of usages of MWEs in text. MWEs can be a challenge both for NLP (Sag et al. 2002) and for second language learners (Irujo 1986; Laufer 2000) alike. As a first step to dealing with MWEs in an application like the Wakaran Glosser introduced in Chapter 4 we must be able to identify their usages in text. Generally speaking, a canonical arrangement of particular lexemes and syntactic structure can be identified for each MWE (Morimoto et al. 2016). However, matches to the canonical structure of a MWE are not always unambiguously a usage of the idiomatic expression: in some contexts the particular combination of the idioms words takes its standard literal compositional meaning (Hashimoto and Kawahara 2009). For such ambiguous expressions we call a usage of its constituents in their canonical structure a multiword expression token candidate (token candidate). Many studies have focused on resolution of token candidate ambiguity by identifying fixedness of constituent inflection and syntactic dependencies in the surrounding text (Fazly et al. 2009; Nasr et al. 2015). Success of such a model would mean that idiom token identification can be solved cheaply and easily in glossing software with a lemmatiser and dependency parser, which...
Chapter 5: Multiword Expressions

would be a great asset to Wakaran because it would allow either the gloss of a MWE or the
glosses of its constituents to be cheaply discarded as appropriate. Others have sought instead
to use semantic models to distinguish idiomatic and literal usages [Schneider and Smith 2015; Gharbieh et al. 2016], which in general will introduce more latency to an end user application than a dependency parsing approach. In this chapter, we explore both approaches using a large annotated corpus of Japanese idioms, OpenMWE, and an idiom dictionary that encodes idiom constraints on syntactic relationships with the surrounding text, the Japanese dictionary of multiword expressions (JDMWE).

Even under the category of multiword expression token candidate disambiguation (candidate disambiguation) we see a range of different properties and nuances to research projects. This chapter begins by presenting some of the motivations for candidate disambiguation strategies and properties seen in candidate disambiguation research. We then describe three computational tasks which all fit the description of candidate disambiguation but which address different motivations and facilitate different properties of the task solution. We introduce our terminology for these subtasks and briefly describe existing research from Section 2.3.3 in its terms.

The remainder of the chapter describes our supervised machine learning solutions to the tasks defined in Section 5.1. We make use of the OpenMWE corpus of Japanese idioms as it is the largest freely available corpus of ambiguous token candidates. As such all of our work is with the Japanese language, but we make notes about generalisation to other languages as we go along. We begin in Section 5.2 by detailing feature extraction for instances in the OpenMWE corpus. To help manage the huge variety of features explored, we organise them into a taxonomy. Additionally we motivate feature classes in terms of the tasks and desirable properties explored in Section 5.1.

Having introduced the features being used, we move on to tackling each of the tasks in Section 5.1 and testing our hypotheses about existing and proposed features. We divide the experiments into two main parts based on the datasets used, for pragmatic reasons which will become clear later. Our investigation of features is more complete and robust than previous work. In part this is because the size of the OpenMWE dataset allows us to work with a meaningful sample of
many low frequency features. The work categorising features into a taxonomy in Section 5.2 also helps us to identify holes in previous research for further investigation. Our development of new features takes inspiration from work in languages other than Japanese and from resources peculiarly available for the Japanese language. We focus in particular on which features are specific to identification of tokens of a single MWE type and which features could be used to disambiguate independent of type.

In our conclusions we confirm our hypotheses for several feature classes. However, some of the feature classes disappoint our expectations and others produce unintuitive results which raise questions that warrant further investigation.

### 5.1 Flavours of token candidate disambiguation

In this section we develop a nomenclature for several different flavours of token candidate disambiguation. This subcategorisation is somewhat reminiscent of the division of word sense disambiguation research into all words and lexical selection disambiguation in that it describes how the relative valuation of goals such as coverage and precision entails different modelling and research strategies. We define a family of related computational tasks and organise them into a small taxonomy under the token candidate disambiguation root. In subsections, we go into more detail for each disambiguation task, exploring the advantages and disadvantages of pursuing a solution to it.

Computational MWE research has developed a healthy array of task types. Until recently it could mainly be categorised into one of three main sub-fields: “MWE extraction”, “[type] disambiguation” or “token disambiguation”. However, to avoid certain technical inaccuracies we find it preferable to follow the recent trend and discard “token disambiguation” as a term in favour of multiword expression token identification and candidate disambiguation. As we saw in Section 2.3 multiword expression token identification consists of identification of token candidates, in some cases followed by a necessary multiword expression token candidate disambiguation step. Candidate dis-
ambiguation has seen very little attention until recent years, but is mature enough now for consistent themes to emerge. We identify and formalise three clear subtasks of candidate disambiguation—type-specialised, type-generalised and cross-type—and explore the characteristics of each with reference to each other.

We begin with type-specialised classification, in which MWEs are modelled one type at a time independently of each other. If the investigation uses supervised classification, the corpus token candidates of a single MWE type are collected to train a machine learning classifier which is tested on held-out candidates for that same type. A cross-validation partitioning for testing type-specialised classification is illustrated in Figure 5.1a.

In contrast to this we define two kinds of multi-type aware disambiguation which we call type-generalised classification and cross-type classification. Cross validation partitionings for these are illustrated in Figure 5.1b and Figure 5.1c respectively. In supervised type-generalised
classification a single classifier is trained to classify any token of all MWE types for which training data exists. Unlike type-specialised classification in which a separate model is trained for each MWE type, one model is trained and used for tokens of any type in the training corpus. This allows for any commonalities between literal or figurative language to be shared between types. However, since supervised resources are costly to acquire, a greater aim is to be able to classify tokens of MWE types for which no training data has been collected or labelled. With this aim in mind we motivate cross-type classification in which classifiers are trained on tokens of one set of MWE types, and tested on token candidates for types not found in the training set.

5.1.1 Qualities

Our discussion of candidate disambiguation subtasks will proceed with reference to the following qualities of solutions:

**Performance** refers to the accuracy of the resulting classifier, nominally on unseen data but measured relative to gold-standard annotated data as a proxy for performance “in the field”.

**Coverage** refers to the proportion of token candidates a classifier is able to accept as input. Pragmatically this is assessed in terms of the number of MWE types it is designed to accept.

**Annotation cost** is the quality of requiring a large amount of gold standard hand annotated data representing a manual solution to instances of the task. If a task solution has a high annotation cost we call it annotation intensive; when the cost fails to be met we say we have a data sparsity problem.

**Representation cost** refers to the size of model that results from training for statistical machine learning. In practice this means size on disk or in memory for the trained classifier, which may impact how well it integrates with other applications.

**Training cost** is the resources including time, disk and memory to produce a classifier from the available data.
5.1.2 Type specialised disambiguation

The first subcategory of candidate disambiguation we will look at is type-specialised disambiguation. The defining characteristic of this task is that the classifier is trained on a substantial amount of gold standard data for a specific MWE type and is only expected to be able to classify tokens of that type. Type-specialised classification is epitomised by the work of Hashimoto and Kawahara (2009). Let us examine the properties of the task.

A typical solution to this task will achieve high levels of accuracy on a targeted MWE, but will have a high annotation cost. Given that the MWEs with literal homomorphs number in their thousands, achieving high coverage over the range of MWE types is problematic. Not only is the cost of manual annotation prohibitively expensive, but additionally the solution must inevitably encode specific knowledge of every MWE type and therefore its representation may be large as well.

One advantage of this approach is in training cost. The size of a partition for a single MWE type is quite small compared to the size of a multi-type dataset. Since the time cost of training for our chosen machine learning algorithm is usually worse than linear (Chang and Lin 2011), training many small models is usually far more tractable than solving the constrained optimisation problem on a full corpus.

This task most closely resembles word sense disambiguation in the analogy explored by Section 2.3.3. In particular, type-specialised disambiguation corresponds to lexical sample classification, where systems are trained with specialist knowledge of a (usually small) selection of ambiguous word types, and model an independent set of semantic classes for each. The challenges of data acquisition that apply to supervised WSD apply to type-specialised disambiguation. As such, coverage is the greatest limitation of type-specialised classification.

5.1.3 Type generalised disambiguation

Type-specialised disambiguation has two major drawbacks:
1. it requires sufficient training data for each MWE type to allow for statistical machine learning, which makes it costly to resource.

2. it does not generalise behaviours common to different MWE types.

In situations of data sparsity it may even prove impossible to train type-specialised classifiers. In constructing the OpenMWE corpus, Hashimoto and Kawahara (2009) aimed to perform manual annotation on 1000 samples for each MWE type, but did not manage to find the full 1000 token candidate for many types in their web crawl resource. To solve this problem, Diab and Bhutada (2009) simply bundle all their training data — regardless of MWE type— into learning one classification model, which is intended to identify tokens of any of the training MWE types. This serves as a prototypical example of the task we have termed type-generalised classification. The distinguishing characteristic of type-generalised classification is that it shares information between different MWE types. A typical solution to this task will train a single machine learning model on all of the input data and distinguish the types of individual instances only by means encoded in training features. This is more amenable to efficient representation of trained models and it alleviates the problems of a lack of training data by allowing knowledge gained from common MWE types to transfer to sparse ones.

A significant drawback of this approach is the cost of training. A small amount of data for each type still adds up when trying to achieve coverage over a large number of types. Since training time usually increases superlinearly with the training set size this can soon become problematic. This can result in a loss of accuracy depending on how the terminating conditions of the machine learning algorithm trade time for precision. Therefore, on specific types we expect type-generalised classification to perform worse than type-specialised classification which has the advantage of being able to make a good fit for each MWE type separately. Additionally, type-generalised classification significantly reduces the potential to use WSD methods effectively because the semantics of the literal and idiomatic senses of each MWE are presumed to be disjoint from the other MWE types, making the semantic disambiguation aspect a high cardinality multiclass labelling problem. There-
fore, we expect type-generalised classifiers to rely more on the idiom characteristic features for their performance.

Kawahara and Palmer (2014) investigate supersense WSD for verbs using a type-generalised classification strategy. Supersense tagging parallels multiword expression token candidate disambiguation even more closely than fine grained WSD since it uses the same set of classes for all word types. Their results showed that overall performance was competitive with word specific models and that performance on word types with as few as 5 annotated token samples in the corpus was very strong.

Neither the problem of over-generalisation nor the problem of training cost are undefeatable. More advanced methodologies might start by sub-partitioning the training data into subsets of a reasonable size for training. If partitioning is done using an appropriate clustering algorithm this will even allow for specialisation to families of MWE types with similar distributional properties. However, such solutions inevitably eat away at the advantages of type-generalised classification, particularly that of cost of representation, and introduce much complexity to the machine learning problem.

5.1.4 Cross-type disambiguation

To increase coverage of type-generalised classification we might, perhaps, spread our annotation cost thinly by annotating just a few example tokens for every MWE type found to suffer ambiguity. However, even this endeavour has a large annotation cost and is limited to the classification of MWE types that have been seen at the time of data collection. If we are able to learn generalisable features of MWE types, why not go all the way and learn to classify tokens of MWE types for which we have no annotated data at all? This is, ultimately, the only pragmatic course of action for using supervised methods in an application that demands high coverage (such as the glossing software described in Chapter 4). We call the task in which training data is supplied for some

\[1\] albeit typically a different set of tags for each part of speech
MWE types but the classifier is expected to work for previously unseen\footnote{by “unseen” we mean types with no labelled examples; we still assume a human judgement has classified the type itself as ambiguous} MWE types cross-type classification.

From our starting point in type-specialised classification prioritising performance we have now swung all the way around to prioritise coverage. In the process we have almost entirely given up on treating this as a WSD task, since we have no training data for the “senses” of unseen MWE types. We expect a substantial drop in the performance on unseen types in this task when compared to a seen-types evaluation. This mirrors the typical drop in performance one tends to see for all-words WSD compared to supervised lexical sample models.

Interestingly, the other properties from Section 5.1.1 are the same as for type-generalised classification. In fact there will be a large overlap in the solution space with the main difference being that greater expectations are placed on the classifier in terms of being able to produce a reasonable output on a greater range of inputs. To put it another way, overfitting to the particular MWE types and their semantics will be penalised more in this task than in the type-generalised task. Thus we should expect the idiom characteristic features to be the most useful for this kind of classification and the WSD inspired features to be of limited use.

A good example of a cross-type methodology is Li and Sporleder (2010)\cite{Li2010}. However, they pursued it mainly as a means for overcoming annotation cost — fundamentally the same as the motivation we give for type-generalised classification. Indeed, it turned out that their results were inconclusive for many of the categories of features they experimented with, due to low frequencies of those features within their corpus. Cross-type classification is the most challenging of the tasks we have discussed, but also potentially the most useful, if solved.

**Distinction between task and solution**

Fundamentally type-specialised and type-generalised models represent different strategies for performing the same learning task: given labelled training instances for a set of MWE types build
a model for classification of unseen instances of those types. However the consequences of the choice of strategy here has a substantial impact on many of the properties described in Section 5.1.1 which we consider to justify a separate classification.

Cross-type classification on the other hand has distinctly different constraints on its inputs and outputs. One starts with a supervised corpus of one or more MWE types and uses it to inform an algorithm which classifies instances of unseen types.

5.1.5 Additional resources

We have defined type-specialised, type-generalised and cross-type classification in terms of how the task relates to a corpus of annotated examples — that is, examples for which the task has been solved by hand. This is the most critical resource to the task, however other resources may be used as well. This might mean knowledge bases — compiled expert knowledge on words or MWE types — or it might mean unannotated corpus resources not explicitly dealing with MWEs. The use of such resources is inevitable, even if by indirect means via use of linguistic tools such as dependency parsers or morphological analysers which have been trained on corpora annotated for their own task. It should be understood that the task definitions above are not meant to exclude the use of such resources.

5.1.6 Related work

In this section we briefly discuss the MWE token identification literature reviewed in Section 2.3.3 in terms of the terminology described in the preceding sections.

Hashimoto and Kawahara (2009) performed a type-specialised classification study taking advantage of the size of their OpenMWE corpus to train per-type classifiers on a substantial number of examples. In this chapter, we extend that work using the OpenMWE corpus to compare results using type-specialised, type-generalised and cross-type classification results. Additionally we extend the features used by Hashimoto and Kawahara (2009) in particular introducing features capturing information about the MWE types for type-generalised and cross-type classifiers to take advantage
of. The new features include type-level information found encoded in the Japanese dictionary of multiword expressions.

Fazly et al. (2009) use an unsupervised method to discover expression-specific canonical forms and test a heuristic whereby token candidates are classified as literal if they vary from the canonical form. They also build type-specialised models of context for idiomatic and literal usages using a supervised corpus and a corpus built using the canonical form heuristic.

Diab and Bhutada (2009) trained a single sequence-labelling classification model on a collection of MWE types and performed a type-generalised evaluation. Additionally, although Diab and Bhutada (2009) did not perform cross-type evaluation their sequence labelling model included a label for non-constituent words and it is interesting to note that the approach of labelling every word in the sentence opens up the possibility of labelling unseen types. In fact, they did observe some cases of their models identifying usages of expressions not seen in the training data.

Nasr et al. (2015), since they attempt to perform token candidate disambiguation within the framework of dependency parsing, are effectively performing a type-generalised classification as well. It is noteworthy that, like us, they integrate features from a syntactic lexicon to assist in disambiguation. The lexicon records whether certain verbs take complements introduced by the specific conjunctions that appear in the MWEs being studied. Our work is quite similar since the JDMWE specifies constraints on syntactic arguments allowed by an idiomatic usage. Another case of type-generalised disambiguation being performed as part of a more general task is the study of Schneider and Smith (2015), wherein token candidate disambiguation is performed concurrently with supersense labelling of all words in the text.

Li and Sporleder (2010) were motivated to forgo type-specialised models by a lack of sufficient labelled examples per-type, but interestingly this lead them to attempt cross-type classification. However the small size of their data set meant that most of the linguistically motivated features they were investigating were based on phenomena that did not occur sufficiently often in the corpus to have a statistically significant effect on results.

Finally, there is Gharbieh et al. (2016) who compare type-specialised and cross-type
classification using vector embeddings composed for the MWE and the context of its usage. They found that the cross-type classifier performed better than a majority class classifier but not as well as the unsupervised method of Fazly et al. (2009).

5.2 Feature extraction

To lay the groundwork for the experimental work in this chapter, we organise existing features for classification from the literature (particularly Hashimoto and Kawahara (2009)), building a taxonomy which reveals some omitted features which can be justified on the same grounds as the originals. We define the additional features and refer to them as our extensions to the feature set. We then introduce a new top-level category of features which are targeted at solving the cross-type disambiguation task and for use in type-generalised classification.

Our features are drawn from morphological and syntactic analyses of each token, from a dependency parse of the idiom context and from a specialised dictionary capturing the morpho-syntactic idiosyncrasies of Japanese MWE types.

5.2.1 Preprocessing

The features used for classification in this chapter are generally expressed in terms of the Japanese morphological and syntactic preprocessing steps applied. This section describes those steps, highlighting data to be extracted from the output of the preprocessing tools. It builds on the introduction to Japanese NLP given in Section 3.2. We include a brief discussion of how best to replicate the preprocessing steps for other languages.

Our feature extraction makes extensive use of the Japanese dependency parser KNP (Kurohashi and Nagao 1994), however features should be reproducible in other languages with a suitable dependency parser, morphological analyser, and electronic thesaurus or ontology. Each instance in http://nlp.ist.i.kyoto-u.ac.jp/EN/?KNP
To help elucidate both the details of our feature extraction and how it might be replicated for another language, we will look at the information extracted for the sentence in Example (23):

(23) 桂子さんは、サッカーの腕を上げた。
keiko-san-wa, sakkã-no ude-o ageta.
Keiko-TOPIC soccer-GEN arm-OBJ raised.

# “Keiko raised her soccer arm.” (literal)

“Keiko improved her skills at soccer.” (idiomatic)

Japanese is a non-segmenting language in that it has no clearly marked word boundaries.

In Figure 5.2 Example (23) has been segmented by KNP into tokens which are then grouped into chunks (or, more properly, bunsetsu). Each chunk has a parent link to a higher chunk describing the structure.
dependency parse of the sentence. In the conventional word order for Japanese dependency links are always forwards, so the head of a phrase is also its rightmost (final) chunk.

From Figure 5.2 we see that *keiko* “Keiko” and *ude* “arm” are the dependants of *ageta* “raised”, with *sakkä* “soccer” modifying *ude* “arm”. This gives rise to the incorrect literal interpretation #Keiko raised her soccer arm. In fact, *ude-o ageru* “to improve one’s skills” is an idiomatic homomorph of *ude-o ageru* “to raise one’s arm”.

In this chapter when we refer to words we are in fact referring to the chunks returned by *KNP*. This is the most appropriate level of segmentation for our purposes because it is the level at which the dependency relations exist and because the morphological tokenisation level includes affixes and particles which would need filtering. *KNP* labels each chunk with a normalised form, which we use as the lemma of the word for our feature extraction. We also extract the part of speech (POS), category and domain of tokens corresponding most closely to the lemma, as additional features of each word (see the lexical features below).

**Inflections**

Returning to Figure 5.2 take note of the *TOPIC* flag on the first chunk and the *ADNOMINAL* flag on the second. *KNP* produces many chunk annotations; we made use of five main kinds:

- **ADNOMINAL** appearing on adnominal modifiers;
- **TOPIC** appearing on sentential topics and emphasised chunks;
- **VOICE** of inflected verbs;
- **NEGATED** denoting negation; and
- **VOLITIONAL** denoting a volitional modality.

For all but the volitional modality group, *KNP* outputs only one or two annotation variants (e.g. there are two voices in Japanese: passive and causative). For the volitional modality,
Hashimoto and Kawahara (2009) used five classes: request, invitation, order, volition and prohibition. We made use of the same subset of modalities output by KNP.

The KNP chunk annotations we used chiefly capture inflections on words in the text, something which Diab and Bhutada (2009) approximate with a character $n$-gram feature. For implementation in other languages, the $n$-gram heuristic or any available morphological analyser might be used.

**Lexical features**

Our preprocessing step allows us to collect a number of data points on individual words. We collect each of the following features of a word’s type under the banner of “lexical features”:

1. Lemma
2. Part of Speech
3. Hypernym (“category”)
4. Domain

The category and domain information performs a similar function to the named entity information used by Diab and Bhutada (2009); it collapses classes of words into a single feature while retaining relevant semantic information. An information source such as an ontology or thesaurus could be substituted for use in other languages. Hashimoto and Kawahara (2009) translate the kategori “category” field of KNP output as hypernym, and we will adopt the same terminology hereafter.

### 5.2.2 Idiom characteristic features

A widely depended-upon characteristic of MWEs is lexico-syntactic fixedness. This characteristic features in a range of MWE research from MWE extraction to candidate disambiguation (Bannard 2007; Fazly et al. 2009; Li and Sporleder 2010; Lossio-Ventura et al. 2014).
Morimoto et al. 2016) (see Section 2.3). This motivates the use of a number of features of the token candidate itself, such as inflections on the constituent word tokens and the presence of non-constituent words interrupting the expression. In the context of our work with idiomatic expressions, token features which have been motivated by lexico-syntactic inflexibility are generally referred to as idiom features, but may interchangeably be referred to as MWE features.

The idiom features of Hashimoto and Kawahara (2009) included a single binary feature for each of the KNP chunk annotation groups listed at the end of Section 5.2.1. For each of the groups, the feature fires if one of the annotations appears on a relevant constituent of the token candidate. Details of each are given in the outline of our extensions below.

The final idiom feature of Hashimoto and Kawahara (2009) was the adjacency feature. This feature fires if the constituents of the token candidate are contiguous, i.e., there are no intervening chunks. They found that this feature had a greater impact on classification performance than the other idiom features combined.

**Extended idiom features**

Each of the Boolean idiom features of Hashimoto and Kawahara (2009) captures some variation of the form of the MWE. We introduce MWE token features capturing details on the kind of variation:

- The ADNOMINAL modification feature fires if a non-constituent adnominal modifies a noun in the MWE.
  
  Where such a modifier exists, our extensions include features capturing the four kinds of lexical information described in Section 5.2.1: lemma, POS, hypernym and domain.

- The TOPIC feature fires if a constituent noun is marked as a sentential topic.
  
  In this case our extensions include the lexical features from Section 5.2.1 as extracted for the topic-marked constituent.
The VOICE, NEGATION and VOLITIONAL features fire if the MWE has a head verb and it has voice marking or is negated.

Our extensions include features specifying what form of voice, negation or volitional marking is used.

Finally we come to the adjacency feature, which we have tweaked as well as extended. We reverse the sense of the feature to fire when the constituents are not adjacent, and rename the feature as the internal modification flag.

Our extensions include an internal modification Boolean flag, and additionally extract the lexical features from Section 5.2.1 for a single intervening chunk where any existed. When more than one intervening chunk is found, we take the rightmost.

We take a few words here to explain our transformation of the adjacency feature into an internal modification feature. This change is almost entirely superficial; it are not expected to alter the results, but instead are intended to enable a more intuitive understanding of the features. First we observe that the other idiom features fire (or rather, take a true value) where they deviate from the so-called canonical form for the MWE. This means that they fire less often than not; it also means that the feature is true for a phenomenon that may be proscribed by the MWE’s lexico-syntactic fixedness. Since constituent adjacency is the common, canonical form for a MWE it is more consistent to treat it as a false valued or non-firing feature. Second, we observe that syntactic constraints ensure that an intervening word will be a dependent modifier of a constituent. For these reasons we name the modified feature internal modification.

Examining the information in Figure 5.2 extracted by KNP for the simple sentence in Example (23), idiom features would only fire for adnominal modification. The adnominal modification Boolean feature fires because sakkâ “soccer” modifies the constituent ude “arm”; Consequently our extensions generate lexical features (lemma, POS, hypernym and domain) for the modifier: sakkâ, noun, abstract thing and sports respectively. As was the case when sakkâ “soccer” was considered in its role as a modifier of ude “arm”, we note that it is in fact informative that the adnominal
modifier is a sport and not, for example, a person.

Note that although Figure 5.2 shows the TOPIC flag for the word keiko “Keiko”, it is not a constituent of the MWE so the TOPIC flag and features do not fire.

**JDMWE conformance features**

The *Japanese dictionary of multiword expressions* (JDMWE) contains lexicographer-defined constraints on the syntactic flexibility of MWEs. Here, we introduce refined features which fire only when a MWE token has a modifier which violates the constituent modification constraints encoded in JDMWE. To explain the constituent modification constraints, Tanabe et al. (2014) use the example JDMWE rule \[[\ast N\text{wo}]\ast V\text{30}\] for the expression kao-o suru “make a face”. In the rule the N is the POS of the noun kao “face”, the wo represents the particle o that follows kao, and V30 represents the POS and conjugation class of the verb suru “to do”. The square brackets represent the nested dependency relationships between the constituents. The asterisks are used to represent modifiability at different positions in the dependency tree. In this case both constituents are modifiable: the first asterisk means that the noun kao can take adnominal modifiers and the second means that the verb suru can take internal adverbial modifiers. Thus the following example is legal according to the JDMWE:

(24) 悲しい顔をいつもする
    kanashii kao-o itsumo suru
    sad face-OBJ always do
    “always make a sad face”

However, if the first asterisk had not been there then the adjective kanashii “sad” of kao would be a proscribed adnominal modifier and if the second asterisk were not there then the adverb itsumo “always” on the verb suru would be a proscribed internal modification.

In Section 5.2.2 we defined an internal modifier as a dependent of a constituent which is not a constituent itself but divides an MWE token into two parts. For example, in the English *kick seven buckets*, seven is an internal modifier intervening the constituents of a token candidate of the
MWE *kick the bucket*. Our internal modification features and the adjacency feature of related work (Hashimoto and Kawahara 2009) flag the presence of any internal modifier unconditionally. For the *JDMWE* features we include an internal modification flag and extensions comprising lexical information about the modifier that only fire if the modification is proscribed by the type’s constraints.

Since the *JDMWE* constraints can also apply to *external* modifiers it also presents an opportunity to extend the adnominal modification feature which indiscriminately flags external modifications on a leaf noun. Sentential subjects and other external arguments of the head verb are too common to be sensibly proscribed but we did include a feature flagging proscribed external modification of leaf constituents such as *water* in *kick the bucket of water*, and included our four kinds of lexical information about the modifier.

### 5.2.3 Context features

As seen in Section 2 [Hashimoto and Kawahara (2009)] put together a set of features based on WSD best practice as espoused by [Lee and Ng (2002)]. We observe that each is a representation of the context in which the MWE token appears. Each of the WSD features they used can be sorted into one of three kinds of context, as follows:

1. local context, in the form of part of speech and lemma features for indexed word offsets to each side of the token candidate.

2. syntactic context, in the form of lemmas and POS for context words in a syntactic relationship with either the first or last constituent word of the MWE.

3. full context, in the form of bag features for lexical features of single words in the full surrounding context.

Due to the way the *OpenMWE* corpus was constructed, the full context is a single sentence, but often a relatively long one. In general, it might be more appropriate to use a paragraph, several sentences, or a wide word window. Note that we use the same definitions of local and syn-
tactic context as Lee and Ng (2002) with the specialisations to MWEs outlined by Hashimoto and Kawahara (2009).

Local context

The local context is drawn from a window three words to the left of the leftmost constituent in the token candidate and three words to the right of the rightmost constituent. They are labelled with indices $-3$ through $-1$ and $1$ through $3$ respectively. Lexical features are drawn for each of these indices separately. Additionally, paired lexical features are drawn for a selection of index parings: $(-3, -2), (-3, -1), (-2, -1)$, the symmetrical positive indexed equivalents and finally $(-1, 1)$. Hashimoto and Kawahara (2009) drew two versions of these features: one with the lemmas at the given offsets and one with parts of speech. We draw each of the four lexical features listed in section Section 5.2.1. The hypernym and domain features we have thus added for context words perform a similar function to the named-entity features used by Diab and Bhutada (2009). In particular, they are features which fire for a select subset of words and collapse collections of words with similar semantics into a single value.

Syntactic context

As with the local context, we also extend the features extracted from the syntactic context with hypernym and domain information. There is a very good motivation for doing this, apart from completeness. The intent of the syntactic features is to capture selectional restrictions involving constituents of the MWE. Violation of selectional restrictions for or by constituents of the MWE leads us to strongly suspect an idiomatic usage. In the case of Example (25) we see that having the sport sakkā “soccer” modifying ude “arm” is strongly indicative of the idiomatic ude-o ageru “to raise skills”. For this MWE, any sport has the same implication. We can see from Figure 5.2 that KNP has extracted the domain sports for sakkā “soccer” so a classification algorithm can use this feature to make a valid generalisation.
Chapter 5: Multiword Expressions

Full context

The full context represents the full text in which the token candidate is found as a bag of words. Generally speaking this might be limited to a full document, a paragraph, or a large word window. Since the OpenMWE corpus was built on long single sentences taken from the web we take the sentence as a whole for our full context. Like Hashimoto and Kawahara (2009) before us, we use a bag of lemmas found in the sentence to represent context, encoded as a vector. In our extended features we extract a bag of items for each of the lexical features listed in Section 5.2.1 including the hypernym and domain but excluding part of speech, which we do not expect to undergo meaningful variation.

Example

Figure 5.3 contains a sample of all original and new WSD features, extracted for the MWE token in Example (25), which expands Example (23):

(25) 桂子さんは 真面目に 練習して、サッカーの腕を 上げた。お母さんは喜んだ。
keiko-san-wa majime-ni renshû-shite, sakkâ-no ude-o ageta. okāsan-wa yorokonda.
Keiko-TOPIC diligent-ly practice, soccer-GEN arm-OBJ raised. Mother-TOPIC pleased.

“Keiko practised diligently and improved her skills at soccer. Her mother was pleased.”

Note that for the sake of the example we have included a second sentence to show that its content should appear in the full context and local context features.

5.2.4 Type features

The features we have discussed so far have, for the most part, ignored the constituent words of the MWE type itself. For type-specialised classifiers this is inconsequential since the constituents are constant. However features of the MWE type may be important for cross-type classification where similarities between different MWEs could be leveraged. Therefore, for each
Chapter 5: Multiword Expressions

full context features

lemma keiko, majime, renshū, okāsan, yorokobu.
hypernym abstract thing, body part, person.
domain sports, familial associations.

local context features

lemma majime_{−3}, renshū_{−2}, sakkā_{−1}, okāsan_{1}, yorokobu_{2}
   (majime, renshū)_{−3,−2}, (majime, sakkā)_{−3,−1}, (renshū, sakkā)_{−2,−1}, ...
POS adjective_{−3}, noun_{−2}, noun_{−1}, ...
   (adjective, noun)_{−3,−1}, ...
hypernym abstract thing_{−2}, abstract thing_{−1}, ...
   (NULL, abstract thing)_{−3,−1}, ...
domain sports_{−1}, ...
   (NULL, sports)_{−3,−1}, ...

syntactic context features

lemma sakkā_{child}
POS noun_{child}
hypernym abstract thing_{child}
domain sports_{child}

Figure 5.3: A sample of the WSD inspired features extracted for Example (25).

MWE type, we use the lexical features of Section 5.2.1 of the headword in particular and a bag feature for each across all constituents of the MWE.

One motivation for these features is to allow a type-generalised classifier to make more informed decisions when confronted with tokens of types it was trained on. For example, if a collection of constituents has been encountered in training, a supervised statistical classifier may capture the majority class or prior probability for idiomaticity of the training MWE type. For cross-type classification, some constituents — in particular the headword — may be indicative of the relative idiomaticity of an idiom. For example, common verbs such as take and make are common in idioms such as take a shot and make a stand.
Chapter 5: Multiword Expressions

JDMWE type-level features

We extracted the modifiability flags from the syntactic field of entries in JDMWE and generated a feature for each modifiable constituent, identified by its position in the type’s canonical parse tree. The motivation for this is to allow machine learning algorithms to capture any similarities in idiomaticity between MWE types with similar modifiability constraints. This is not expected to be as effective as the JDMWE based MWE token features. However, if MWE types come in families with similar behaviour this may be a way to detect it.

5.3 OpenMWE corpus experiments

In this section we explore the supervised classification of ambiguous MWE tokens using the OpenMWE corpus of Japanese idioms [Hashimoto and Kawahara 2009]. We put our new features tailored to the cross-type classification task to the test. We also test our refined features for type-specialised classification. We explore more deeply the interaction between several aspects of the task, including differences between cross-type and type-specialised classification; combinations of major classes of features; and finally, the size of the training corpus. We find that:

1. Our extensions for existing WSD inspired features offer consistent improvements in performance, and our extensions to idiom features usually offer marginal improvements. The extended WSD and idiom features used together for type-specialised classification outperform all comparable previous systems in terms of raw performance.

2. Our type features for cross-type classification interact with the task and with other features in interesting ways, and in some cases give substantial improvements to performance.

3. Our best results in cross-type classification use only our extended WSD features, with no idiom inspired or type features. Despite using no tagged data for the target MWE types, they achieved a performance in excess of the (supervised) type-specialised baseline.
This last result is significant because it demonstrates the readiness of our cross-type classifiers to work on previously unseen MWE types. It is also interesting because it uses only semantic features and none of the lexico-syntactic fixedness features widely expected to be effective for MWE classification (Fazly et al. 2009; Hashimoto et al. 2006; Li and Sporleder 2010).

5.3.1 Results

We evaluated classifiers using combinations of the three main classes of features: idiom (or MWE token features), WSD (or context features) and type (or MWE type features). For all tasks, a tenfold cross-validation partitioning was used, and a feature count cutoff of one was used to filter out extremely rare features. Given the binary classification nature of the task we used accuracy as our performance metric, micro-averaging across all instances in the corpus.

For statistical significance we used the sign test because our cross-type classification testing partitions were unevenly weighted, making a t-test inappropriate. Except where otherwise stated, all differences we report here were significant with \( p < 0.05 \) (at least).

We constructed two kinds of majority class baseline using class counts from the corpus: the corpus baseline, which achieved an accuracy of 0.612, and the type-specialised baseline, with an accuracy of 0.741. All other systems were linear kernel Support Vector Machine models trained using the \textit{libSVM} package\(^6\).

**Crosstype classification**

For the purposes of testing cross-type classification we partitioned the set of 90 MWE types for cross-validation. Thus classifiers were trained on the instances of 81 types and tested on the instances of the 9 unseen types. Note that since the corpus contains a different number of instances for each type, the partition size was not strictly constant. Results across all feature combinations appear in Table \(5.1\).

\(^6\)We initially used the \textit{TinySVM} package and quadratic kernels of [Hashimoto and Kawahara (2009)] for comparability reasons, but eventually changed system and kernel for consistency and speed of convergence.
Chapter 5: Multiword Expressions

Table 5.1: Results of combining different features for cross-type classification. “Basic” results restrict the idiom and WSD features to those of Hashimoto and Kawahara (2009); “extended” results include our extensions.

<table>
<thead>
<tr>
<th>Feature Types</th>
<th>Accuracy</th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>basic</td>
<td>extended</td>
</tr>
<tr>
<td>idiom type</td>
<td>0.623</td>
<td>–</td>
<td></td>
</tr>
<tr>
<td></td>
<td>0.630</td>
<td>0.627</td>
<td></td>
</tr>
<tr>
<td></td>
<td>0.626</td>
<td>0.651</td>
<td></td>
</tr>
<tr>
<td></td>
<td>0.737</td>
<td>0.745</td>
<td></td>
</tr>
<tr>
<td></td>
<td>0.736</td>
<td>0.743</td>
<td></td>
</tr>
<tr>
<td></td>
<td>0.738</td>
<td>0.746</td>
<td></td>
</tr>
<tr>
<td></td>
<td>0.739</td>
<td>0.745</td>
<td></td>
</tr>
<tr>
<td>corpus baseline</td>
<td>0.612</td>
<td></td>
<td></td>
</tr>
<tr>
<td>type-specialised baseline</td>
<td>0.741</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

The idiom type and token features did manage to improve on the corpus baseline by a little over one percentage point each. However, this is over ten percentage points behind the results when using WSD features alone. In fact, the WSD features with our extensions achieved effectively our best results. The addition of idiom features with our extensions achieved fractionally better performance (without statistical significance) and all feature combinations which did not include our full complement of WSD features had lower performance. The WSD features’ performance exceeds even the type-specialised baseline, for which the oracle majority class was computed from gold-standard data which the cross-type classifier has no access to.

It is a surprising result, for two reasons: first, that idiom features are widely assumed to be a key information source, particularly for unsupervised disambiguation (Fazly et al. 2009), and second, that WSD features — paragraph context in particular — are typically used as a model of the meaning of a token. It is counterintuitive that models of semantics are more informative than models of lexico-syntactic variations when the testing and training sets that are explicitly disjoint with respect to MWE type.

Neither the idiom token nor type features alone stand up well in comparison. However, we note that combining our type features with the extended idiom token features provided a sub-

\(^7\)In this chapter, all stated differences were statistically significant unless explicitly declared otherwise.
Chapter 5: Multiword Expressions

Table 5.2: Results for classification of MWE tokens of MWE types seen in the training corpus. As in Table 5.1, “Basic” results restrict the idiom and WSD features to those of Hashimoto and Kawahara (2009).

<table>
<thead>
<tr>
<th>Feature Types</th>
<th>Accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>basic</td>
</tr>
<tr>
<td>idiom</td>
<td>0.741</td>
</tr>
<tr>
<td>type</td>
<td>0.748</td>
</tr>
<tr>
<td>wsd</td>
<td>0.630</td>
</tr>
<tr>
<td></td>
<td>0.844</td>
</tr>
<tr>
<td></td>
<td>0.851</td>
</tr>
<tr>
<td></td>
<td>0.847</td>
</tr>
<tr>
<td></td>
<td>0.854</td>
</tr>
<tr>
<td>corpus baseline</td>
<td>0.612</td>
</tr>
<tr>
<td>type-specialised baseline</td>
<td>0.741</td>
</tr>
</tbody>
</table>

stantial boost compared to similar configurations, attaining a classification accuracy of almost four percentage points above the baseline.

What happens when a nominally cross-type classifier encounters an instance of a MWE type which it has seen in its training set? We can test this by performing a type-generalised classification task (as defined in Section 5.1). To construct the task, we partitioned the corpus stratified across types. That is, classifiers were trained on a combined 90% of instances from each of the MWE types in the corpus and tested on a set comprising the remaining 10% from each type. The results are shown in Table 5.2.

The idiom features once again improved a couple of percentage points on the baseline. In this instance, the type features produced much better results, achieving the same results as the type-specialised majority class baseline. It is not possible for any deterministic classifier to do better on the same input because the idiom type features are constant across all instances of a MWE type.

Once again the WSD features did far better than any of the others at 23 percentage points over the corpus baseline and over ten points above even the type-specialised baseline. This is more to be expected than the equivalent result for cross-type classification since, by their origin, WSD features are designed to capture differences in semantics for known types.

The indisputable dominance of WSD features observed in these experiments, particularly
Table 5.3: Results of combining different features for type-specialised classifiers. “Basic” results restrict the idiom and WSD features to those of Hashimoto and Kawahara (2009); “extended” results include our extensions.

<table>
<thead>
<tr>
<th>Feature Types</th>
<th>Accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>basic</td>
</tr>
<tr>
<td>idiom wsd</td>
<td>0.768</td>
</tr>
<tr>
<td>*</td>
<td>0.882</td>
</tr>
<tr>
<td>type-specialised baseline</td>
<td>0.741</td>
</tr>
</tbody>
</table>

in cross-type disambiguation, warrants further investigation which we leave for future work.

**Type-specialised classification**

When encountering a token candidate of a known MWE type with a large amount of labelled training data, a type-generalised or cross-type classifier need not be used: we can fall back on type-specialised classifiers like those of Hashimoto and Kawahara (2009). Although a type-specialised classifier has no capacity to transfer generalised knowledge from other types, it has the advantage of being able to fit the specific type better with the same model complexity. To see how much better we can do by selecting an appropriate type-specialised classifier, we trained and tested classifiers on the same partitioning as the previous task but restricted to instances of one MWE type at a time. The results appear in Table 5.3.

In this case the idiom features performed better, exceeding even the type-specialised baseline. This indicates that the idiom features do contain information about MWE token idiomaticity even if it does not generalise well across types.

The WSD features achieved close to the best results seen across all our experiments. An improvement of four percentage points is seen compared to the previous task, which is indicative of the challenge to the machine learning process of simultaneously training on data of 90 idioms. Our best results were achieved using all of the extended idiom and WSD features together, hitting an even 0.890.

Finally, we used the type-specialised task to investigate the contribution of the size of
the *OpenMWE* corpus. To do this we measured cross-validation accuracy while limiting the total number of training instances. Training set size caps ranging between 100 and 1000 instances were used, but in practice most of the MWE types had between 500 and 1000 available training instances, so the average actual training instances used was less than the cap. We performed these experiments using our complete WSD and idiom features and, for comparison, with the original features of Hashimoto and Kawahara (2009). The results appear in Figure 5.4.

Even with 100 instances per MWE type, we achieved an accuracy of 0.834, which is an appreciable improvement on the type-specialised baseline. However the data show a definite positive trend with the number of instances, reaching 0.884 under a cap of 650 instances (and 589 average actual instances) per MWE type.

Setting the maximum number of instances per MWE type to 1000 achieved an accuracy of 0.888. Additionally, when restricted to the original features used by Hashimoto and Kawahara
a performance of 0.884 is observed. Since Hashimoto and Kawahara (2009) also capped instance counts at 1000 this is our closest reproduction of their best configuration which scored 0.893. The differences may be accounted for by a difference in SVM library used, use of linear rather than quadratic models and our relative lack of training parameter tweaking.

Error analysis

One of the more interesting results of this study was the finding that semantic context models were effective for classification of token candidates of unseen MWE types. In particular, models trained on bag of words features were far more successful on this task than models trained on features designed to capture idiom morphosyntactic inflexibility. To gain some insight into this we sampled some MWE tokens from the corpus which were incorrectly classified by models trained on our full complement of idiom features with extensions but were correctly classified by a model trained solely on paragraph level bag of words features. We focussed on the shortest tokens of the MWE types that showed the greatest improvements under bag of words based classification. Specifically, we took the tokens that were incorrectly labelled by the idiom features model and divided them by MWE type and by gold standard label, choosing the 10 largest such groups of error instances. We then chose five of those groups which contained instances with short contexts so that we could focus on cases where differences occurred with a small number of context words. Four of those expressions had only one to three non-constituent words per MWE token. The sample of the remaining expression comprised longer sentences up to fifteen words long, and was included because it represented the rarer case of errors corrected by bag of words features where the true label was idiomatic.

One of the expressions with many short examples containing only one or two non-constituent tokens was mizu to abura “incompatible” (idiom) vs. “oil and water” (literal). In the extracted sample the model trained on idiom features had incorrectly applied an idiomatic label whereas the bag of words model had correctly applied a literal label. Here are two examples:
Chapter 5: Multiword Expressions

(26) 水と油が乳化する
mizu-to abura-ga nyūka-suru
water-AND water-SUBJ emulsify

“Oil and water emulsify” (literal)

# “incompatible emulsify” (idiomatic)

(27) 水と油を混ぜる研究
mizu-to abura-o mazeru kenkyū
water-AND water-OBJ to mix research

“research on mixing oil and water” (literal)

# “research on mixing incompatible” (idiomatic)

In the full set of extracted error samples the context words almost all had a physical or scientific aspect to them:

- mazaru “to mix (intransitive)”
- mazeru “to mix (transitive)”
- kongou “mixture”
- jikken “experiment”
- kyōkai “boundary”
- nyūka “emulsification”
- atsusa “heat”
- daietto “diet”
- kasuri “medicine”
- kenkyū “research”

It makes sense that words related to physical properties and chemical process indicate that a sentence is talking about the real physical challenge of mixing oil and water. However, the MWE type
was not in the training set so the real question is how are these words an indication of the literal interpretation for an unseen MWE type? The OpenMWE corpus was constructed from a web crawl and general purpose idiom dictionaries, so perhaps the expressions it contains would generally be of too informal a register for use in scientific discussions. Although specialised domains like science have their own inventories of MWEs — *multiword expression* itself being one such example — for expressions in general use it is plausible that scientific context words are a good indicator of literal speech.

Another expression with many instances incorrectly classified idiomatic by the idiom feature model was *te o nobasu* “expand into an area” (idiom) vs. “extend a hand” (literal). The context words found in instances where the bag of words model corrects the idiom model’s idiom label to literal were less consistent in this case. We divided the context words into a few thematic groups:

- **physical action**: *todoku* “to reach”, *nigiru* “to clasp”, *mageru* “to bend”, *akeru* “to open”, *naderu* “to brush gently” and *kata* “direction”;
- **the body**: *ashi* “foot”, *koshi* “lower back”, *atama* “head” and *kao* “face”;
- **everyday objects**: *yubiwa* “ring”, *mado* “window”, *ken* “sword”, *kabe* “wall” and *hitsugi* “coffin”;
- **people**: *boku* “I”, *jijūchō* “grand chamberlain”, *sachiko* “Sachiko”, and *shyōjo* “little girl”.

These are not words from a specialised domain as we saw with *mizo to abura* and there is no particular reason to think they would co-occur with a more formal register. However, the literal interpretation of the idiom’s constituents, *extend a hand*, is a very mundane physical action using a common body part, so it does make sense that words for mundane things like physical actions, other body parts everyday objects and people might signify a literal interpretation. Any other idiom based on a metaphor for an everyday action or important feature of the body might follow the same pattern.

The expression *hana ga takai* “proud” (idiom) vs. “prominent nosed” (literal) also involves a body part and exhibits a somewhat similar selection of context words in short literal sen-
sentences that are mistaken for idiomatic by the idiom features classifier but not by the bag of words classifier:

**physical actions** : *mieru* “to see”;

**the body** : *irojiro* “fair-skinned” and *kao* “face”;

**people** : *mama* “mum”, *seiyojin* “Western people”, *maruko* “Marco” and *daredemo* “anyone”;

**miscellaneous** : *sujikan* “a few hours”, *nochini* “later on” and *yume* “dream”.

In this case the metaphor is for feature of a person’s appearance so it makes sense that words for people would appear more often. However, the idiomatic interpretation is also a description of a person’s personal qualities so it is difficult to say that people appearing in context should be a primary indicator of the literal sense.

The errors analysed so far have been for cases where the idiom feature model incorrectly applied an idiomatic label to a literal instance. However, the reverse case also happened, where the idiom features model applied a literal label and the bag of words model correctly applied an idiomatic label. One such expression is *chi ga kayou* “to show humanity” (idiom) vs. “blood flows” (literal) which, like the other expressions seen so far, had many short instances incorrectly labelled by the idiom features model:

**communication and media** : *kanso* “thoughts”, *serifu* “speech”, *hatsugen* “statement”, *toujojinbutsu* “character (in play/novel)”, *kotoba* “words” and *sakuhin* “artistic works”;

**people or persons** : *jitai* “oneself”, *bokura* “we”, *ningen* “human being” and *robotto* “robot”;

**miscellaneous** : *kokoro* “spirit/heart”, *kansei* “completion”, *ikizuku* “to breathe (heavily)” and *tokoro* “wild yam”.

We see some similarities and some differences here in the kinds of words appearing. The references to body parts and everyday objects are all but gone. Instead we see words related to language and the arts. Words related to people (or at least intelligent entities) are still present so perhaps they
should not be taken as a strong indicator of idiomaticity either way. However, we do note that *jitai* “oneself” and *ningen* “human being” in particular lend themselves to less casual speech.

Another expression with many idiomatic instances that were mis-classified by the idiom feature classifier was *mune o utsu* “to be touching” (idiom) vs. “to strike the chest” (literal). The shortest sentences available for this analysis were relatively long, comprising up to fifteen words each. Here we show a uniform sample of the context words:

**physical objects and actions** : *kata* “direction”, *sugata* “appearance”, *hirogeru* “to spread”, *sesshoku* “touch” and *tsutsumu* “to wrap up”;

**people** : *watashino* “mine”, *hitobito* “people”, *jibun* “myself”, *kodomo* “child”, *rōdōsha* “worker”,
*nakama* “colleague”, *gakusei* “student”, *otto* “husband”, *waka* “youth” and *tsuma* “wife”;

**communication and thought** : *uttaeru* “to bring to someone’s attention”, *kataru* “to talk about”,
*kyōmi* “interest in”, *omou* “to think or believe”, *gyōshuku* “condensation (of ideas, emotions, etc.)”, *mugonde* “wordlessly” and *iu* “to say”;

**abstract concepts** : *aware* “pity”, *konshun* “this spring”, *sensō* “war”, *setsusetsu* “passionate”,
*shitashii* “intimate”, *shinsotsu* “honesty”, *kokorone* “feelings” and *kandō* “being deeply moved emotionally”.

We see that with the increase in sentence length, there is a mixture of mundane physical words with people and abstract concepts. However, the mundane physical words are still small in number compared to the words related to communication and to abstract concepts, so this sample does seem consistent with the sample previously seen.

These examples provide some indication that there are in fact identifiable classes of context words which are useful for cross-type classification. However, some caution is warranted given that categories of words like body parts or scientific terms might be useful for identifying literal instances of particular kinds of expressions with constituents that are semantically related to those categories. Ultimately the patterns discovered in this analysis speak most strongly for composit-
tional semantic approaches such as those of Gharbieh et al. (2016) where a model of the individual constituent semantics is compared to the context semantics to detect literal usages.

5.3.2 Discussion

We have shown that cross-type classification of ambiguous token candidates can surpass the type-specialised supervised baseline while alleviating the requirement for labelled token instances, thus enabling classification of tokens of previously unseen MWE types.

Our type features and new idiom features, working in concert with the idiom features of Hashimoto and Kawahara (2009), substantially increase cross-type classification performance over the corpus baseline. However their effect is wholly subsumed by the inclusion of WSD features which are by a clear margin the most effective representation for disambiguation of token candidates of unseen types. This is a surprising result since one would expect a supervised context model to only capture the semantics of seen types. On type-specialised classification, our new idiom and WSD features achieve more consistent gains above the use of WSD features alone. However, the WSD features still have by far the greatest impact on performance.

Finally, we conclude that the size of the OpenMWE corpus raises potential performance at supervised classification by leaps and bounds, but additional improvements are still to be had by collection of more data.

For future work we would like to investigate the dominance of WSD features at cross-type classification. The success of semantic features where the training and test sets have different semantics by design, making this an intriguing counterintuitive result, as does the relatively poor performance of features targeted at linguistic properties of MWEs.

However, to close out this chapter, we will first take a closer look at classifying ambiguous token candidates using idiom characteristic inflexibility by applying a promising knowledge-base for knowledge-rich cross-type classification, the Japanese dictionary of multiword expressions.
5.4 *JDMWE* lexicon experiments

This section builds on the work in the previous section by exploring whether type-level MWE properties sourced from an idiom dictionary can boost the accuracy of cross-type MWE token classification. That is, we attempt to determine whether multiword expression token candidates are idiomatic or literal, based on: (a) annotated instances of other MWE types, and (b) dictionary-based information on the syntactic properties of the idiom in question.

The idiom dictionary we use, the *Japanese dictionary of multiword expressions*, is introduced in Section 3.4.2. It contains a lexicographer’s judgements about the forms of lexico-syntactic modification that are allowed by idiomatic usages of specific Japanese MWEs. However, only a subset of the dictionary had been released at the time of our study, so in this section of the chapter we work with the intersection of the *JDMWE* headwords and the types in the *OpenMWE* corpus.

We find that a type signature representing constituent modifiability judgements extracted from the idiom dictionary is more predictive of the idiomaticity of tokens than our previously introduced MWE type features such as POS or lexeme of the idiom’s constituents. However, individual token candidate violations of the dictionary’s modifiability rules have variable utility for machine learning classification, being suggestive of the literal class but not definitively entailing it. Finally, we present evidence that WSD features do model the individual semantics of literal instances of token candidates when used in a type-specialised classification setting, which re-affirms the novelty of our previous result showing that the same feature set is a powerful resource for cross-type classification.

5.4.1 Results

We worked with a subset of the *OpenMWE* corpus comprising those types having: (a) an entry in the released subset of the *JDMWE*, and (b) both literal and idiomatic classes represented by at least 50 MWE tokens each in the corpus. This leaves 27 MWE types and 23,392 MWE tokens and means that our results are not directly comparable to either those of [Hashimoto and Kawahara](#).
Figure 5.5: Cross-type classification accuracy using JDMWE type-level features and lexical type-level features in combination with various token-level features.

We constructed a cross-type classification task by ten-fold cross validation of the MWE types in the corpus subset, with micro-averaged results. Training sets were the union of all MWE tokens of MWE types in a partition. The majority class was the idiomatic sense and provided a (subcorpus) baseline accuracy of 0.594. Support Vector Machine models with linear kernels were trained on various feature combinations using the libSVM package.

Figure 5.5 shows the results on cross-type classification of adding type level features to token level feature sets. Our JDMWE type-level features represented the modifiability constraints of the type itself. Somewhat unexpectedly they performed comparatively well at the cross-type task, with an accuracy of 0.647, at 5.3 percentage points above the baseline. This is a marked improve-
ment on the lexical type-level features from the previous section, which achieved an accuracy of 4.0 points above baseline. As was observed previously the accuracy gained by using type-level features is much smaller than the WSD features. However, the relative performance of the $JDMWE$ type features to the lexical type features is sustained in combination with a variety of token level feature families.

**Token level $JDMWE$ features**

Our $JDMWE$ token-level features represented how well a token candidate conformed to the type level constraints. The token-level features perform quite badly at cross-type classification. When measured against the baseline or used to augment other token features, they degraded or only marginally improved performance. The fact that using these features resulted in worse-than-baseline performance suggests that the constituent modifiability features extracted from $JDMWE$ may not be strict constraints as they are construed.

To better examine the quality of the $JDMWE$ constituent modifiability constraint features, we constructed a heuristic classifier. The classifier applies the idiomatic class by default, but the literal class to any MWE token which violates the $JDMWE$ constituent modifiability constraints. This classifier’s precision on the literal class was 0.624, meaning that fully 0.376 of modifiability constraint violations in the corpus occurred for idiomatic tokens.

However, the classifier was correct in its literal class labels more than half the time so it achieved a better overall accuracy at 0.612 than the majority class classifier. It follows that the heuristic classifier comfortably outperformed the Support Vector Machine classifier based on the same features. This shows that our poor results with regards to the $JDMWE$ constraint violation features are due in part to failures of the machine learning model to take advantage of the information they provide.

As to the strength of the constraints encoded in $JDMWE$, we found that 4.4% of all idiomatic tokens in the corpus violated constituent modification constraints, and 10.8% of literal tokens. That is, the idiomatic tokens do not obey the constraints universally, but do violate the type
Chapter 5: Multiword Expressions

constraints half as often as the literal tokens. Thus the constraints seem sound but are not a hard rule as could be inferred from their inclusion in the JDMWE.

Error analysis

To understand better how idioms are violating their JDMWE constituent modifiability constraints, we examined the modifications on a sample of such violating tokens from the OpenMWE corpus. We extracted 98 tokens from the 10 MWE types with the most violations, selecting at most 10 tokens for each kind of violation from each of those types. Five main categories accounted for the majority of tokens with constraint violations:

1. blending literal context with the figurative space (27 instances);
2. inserting an adverb between expression constituents (14 instances);
3. embellishing the expression with a pre-modifying adjective on a noun (8 instances);
4. systematic MWE-specific parsing errors (14 instances); and
5. general parsing errors (22 instances).

The remaining 13 instances did not fit into any of the above categories, or used complex language that was difficult to interpret concretely into a cause of failure.

What we mean by blending literal context with the figurative space is that some literal subject matter of the sentence is given a specific relationship to a non-literal constituent. Some expressions showed more of a tendency for this than others. For instance, it was the predominant violation for idiom me o samasu “become aware” (idiom) vs. “wake ones eyes” (literal). The JDMWE constraint for this phrase is listed as [N wo]*V30. Although the JDMWE lists the eyes constituent as unmodifiable, in some idiomatic OpenMWE instances the eyes are expressed as being those of the person or entity who becomes aware. For example consider the following fragments of sentences containing me o samasu:

---

Section 3.4.2 has an explanation of the JDMWE constraint syntax.
Chapter 5: Multiword Expressions

(28) 眠っている 遺伝子の 目を 覚まさせる
nemutte-iru idenshi-no me-o samasaseru
The sleeping gene-GEN eyes-OBJ made to open
# “The sleeping gene’s eyes are opened” (literal)

“The dormant gene is activated” (idiomatic)

(29) 文月先生の 目を 覚ませる
Fumizuki-sensei-no me-o samasaseru
Mr Fumizuki-GEN eyes-OBJ made to open

“Make Mr Fumizuki aware”

In Example (28) the eyes in the expression are used as though they belong to the gene that is being activated and in Example (29) it is the eyes of a teacher that are being opened.

For each constituent of an idiom the JDMWE encodes whether or not it allows dependent modifiers including for any final verbal constituents. Since Japanese sentences are headed by the final verb it would be unreasonable to expect a verbal idiom constituent to have no non-constituent children at all, but in keeping with the general fixedness of MWEs a non-constituent which separates the constituents from each other is a clear violation of this constraint. Nonetheless, there are many cases in the OpenMWE of adverbs appearing between a constituent noun and the head verb. Consider the following examples of the expression nami ni noru “go with the times” (idiom) vs. “ride the wave” (literal):

(30) デジタル化の 波に うまく 乗り
dejitaru-ka-no nami-ni umaku nori
digitisation-GEN wave-DAT skillfully ride

“Skillfully ride the wave of digitisation” (literal)

“Skillfully adapt to the trend of digitisation” (idiomatic)

(31) 世界化の 波に 積極的に 乗って
“seikaiai”-no nami-ni sekkyokuteki-ni notte
globalisation-GEN wave-DAT proactively ride

“Proactively embrace globalisation”

In both cases the manner in which the subject adapts to a changing situation — umaku “skillfully” for Example (30) and sekkyokuteki-ni “proactively” for Example (31) — is given as an adverb
as if describing the manner of *riding*. The adverb separates the verb constituent from the noun constituent so that the expression does not appear contiguous in the text. Note that in both examples the constituent *wave* has a non-constituent dependee (“digitisation” in the first and “globalisation” in the second) but this modification is allowable according to the constraints of the *JDMWE*.

In a sense the adverbial modifiers we have just looked at in *nami-ni noru* “ride the wave” extend a metaphor because they appear as the manner of “riding” of the wave. Such modifications also sometimes occur as adjectives on nouns that are proscribed for modification by the *JDMWE* constraints. Here are a couple of examples for the expression *aosuji wo tateru* “becomes highly agitated” (idiom) vs. “vein stands out” (literal) where the veins have been elaborated in a somewhat literary manner:

(32) 頭に 数分の 青筋を 立てて
hitai-ni ikustumo-no aosuji-wo tatete
forehead-DAT many veins-OBJ raise
“Many veins on his head stood out” (literal)
“He was apoplectic” (idiomatic)

(33) 先生は 脈に 蚯蚓のやうな 青筋を 立てて
sensei-wa kao-ni mimizu-no-yō-na aosuji-wo tatete
teacher face-DAT worm-like vein raise
“A worm-like vein stood out on the teacher’s head” (literal)
“It looked like the teacher’s head would explode” (idiomatic)

In Example (32) the sense of agitation is heightened by describing many veins standing out on the forehead. Note that the expression already carries an implication that the vein is on the face, but this example makes explicit reference to the forehead. Example (33) uses the same pattern, this time describing a single highly pronounced vein standing out from the face like a worm. It similarly serves to emphasise the strength of the teacher’s anger. This example also extends the metaphor by locating the vein, this time on the face rather than the forehead specifically. These extensions of the metaphor to cover the size or location of the vein don’t follow a fixed form and seem to occur as a form of creative expression.
One of the MWE types had a common pattern of usage which was prone to triggering attachment errors in the parser, causing a noun constituent to appear to have a constraint-violating dependent when in fact it does not. The expression is *te o nigiru* “cooperate” (idiom) vs. “grasp hands” (literal) and is commonly used to express co-operation with someone. However, the particle *to* used to indicate “with” in Japanese can also be used like the conjunction “and”, co-ordinating a list of nouns. Thus an ambiguity arises: did we clasp hands with the person, or did we clasp the person and the hand. The latter interpretation is nonsensical but the parser has chosen it in several instances as illustrated by the following examples:

(34) 金大中氏と 手を 握り
Kimudejun-shi-to te-o nigiri
# “Grasp Kim Dae-jung and the hand”

“Grasp Kim Dae-jung’s hand” (literal)

“Co-operate with Kim Dae-jung” (idiomatic)

(35)セルビア派と 手を 握る
serubia-ha-to te-o nigiru
# “Grasp the Serb faction and the hand”

“Co-operate with the Serbs”

In Example (34) the phrase was parsed by *KNP* as *((Kimudejun-shi)-to (te)-o) nigiri* “grasp ((Kim Dae-jung) and (the hand))” which makes the constituent *te* “hand” appear to have the dependent (more strictly speaking, the co-ordinate) *Kimudejun-shi-to* “and Kim Dae-jung”. Since for this expression modifications are not allowed on *te* this is detected as a violation of the *JDMWE* constraints. In a correct parse *Kimudejun-shi-to* “with Kim Dae-jung” would be a dependent of the head verb which would not be a constraint violation. Example (35) illustrates the same error, this time regarding cooperation with a collection of people. Errors of this kind occurred occasionally with other expressions as well to some extent but it was the primary cause of error for *te o nigiru* in particular.
Finally, there was an inevitable class of non-MWE-specific parsing failures which erroneously attached non-constituents to constituents proscribed for modification. One common kind seemed to be the attachment of connecting words and adverbs as in the following examples for the expression *ashi ga deru* “overran the budget” (idiom) vs. “legs came out” (literal):

(36) 儲けに ならない 仕事 ところか、足だ出た
mouke-ni naranai shigoto dokoroka, ashi-ga deta
profit-DAT not become job let alone legs came out

# “Not only the unprofitable job, but also the legs, came out”

“Let alone being an unprofitable job, it also overran its budget”

(37) また 足だ出て しまいました
mata ashi-ga dete shimaimashita
again legs came out completely

“The budget blew out completely again”

In Example (36) the co-ordination *dokoroka*, which can mean “not only” or “let alone” depending on context, has been mis-attached to the noun constituent *ashi* “legs” rather than the head of the whole clause *deta* “came out”. The result is that the *JDMWE* constituent modifiability constraint on *ashi* is unnecessarily violated. Similarly, in Example (37) the adverb *mata* “again”, which can also act as a conjugation meaning “and also”, is erroneously attached to the constituent noun *ashi* instead of the verb *deru*.

In this error analysis we have seen a few ways in which an idiom may escape from its usual inflexibility and a few ways in which an idiom’s modifications may be detected improperly due to parsing errors. There is a trap in assuming that the constituents of an idiom will never have non-constituent modifiers due to not having a direct semantic relationship with the context. Firstly, we have seen a number of ways in which a metaphor itself can be extended coherently with adverbs and adjectives that embellish the original expression. Of the 98 errors studied in our analysis, 22 had some form of embellishment. Secondly, we saw 27 cases of mixing the literal context into the metaphor as a way of emphasising the mapping of the literal word into the metaphorical space. In fact, this kind of mixing was so common in the violations of *me o samasu* “become aware” (idiom)
vs. “wake ones eyes” (literal) where the eyes belonged to a literal subject that it is tempting to suggest that it is a standard usage pattern for the idiom. In this sense it is acting much like nami ni noru “go with the times” (idiom) vs. “ride the wave” (literal), where the JDMWE explicitly allows usages of the form the wave of X where X is a part of the literal context. However, there still exist a significant number of tokens with modifications proscribed by the JDMWE which do not follow such a fixed rule or pattern and instead represent the user of the expression playing with the boundaries of its expected form.

Overall 49 instances — exactly 50% — fit one of our patterns of legitimate violation of a constraint. 13 further instances appeared valid but did not fit one of the described patterns. The remaining 36 instances were influenced by parse errors. Violations of JDMWE constraints appear to be a genuine phenomenon with nontrivial prevalence in natural language.

5.4.2 Another look at WSD features

A second look at Figure 5.5 reveals that even with our improvements to type-level features, our finding of the previous section that WSD context features perform best at cross-type classification of unseen types still holds after the introduction of the JDMWE features. Our results do not shed light on the cause of this phenomena directly, but a breakdown of the type-specialised results does at least reveal the extent to which the WSD features model individual semantics of seen types.

For our type-specialised classification task we once again performed cross-validation for each JDMWE MWE type in isolation, aggregating final results with a micro-average. Some types had a literal majority class, so the baseline accuracy was higher than the subcorpus accuracy at 0.741. Figure 5.6a shows accuracy of various feature subsets on the idiomatic tokens of the corpus (that is, it shows recall for idiomatic tokens). The results show that type-specialised classification performance is basically constant when restricting analysis to only the idiomatic test instances. The huge performance boost produced through the use of WSD features occurs only on literal instances as seen in Figure 5.6b. That is, in our type-specialised classifiers the WSD features are improving
(a) Recall for idiomatic instances for various feature combinations with and without WSD context features, in a type-specialised classification setting.

(b) Recall for literal instances for various feature combinations with and without WSD context features, in a type-specialised classification setting.

Figure 5.6: Looking separately at recall of idiomatic and literal instances in the corpus under various combinations of feature families. Classifiers with no features at all are the type specialised majority class baseline which is literal for some types and idiomatic for others.

Recall for the literal instances of a MWE type but not for the idiomatic instances. It should be noted that idiomatic recall is already very high and that the literal sense is the minority sense in the corpus so this may be a case of improving the recall of the minority class. However, it seems clear that the WSD features are effective at highlighting contexts distinctive of particular literal senses of token candidates. If the WSD features chiefly manage to improve recall of specific expressions of non-idiomatic language it is all the more difficult to see why they are so effective at improving performance at cross-type idiom token identification.

5.4.3 Discussion

Using a MWE dictionary as input to a supervised cross-type MWE token classification task we have shown that the constituents’ modifiability characteristics tell more about idiomaticity than their lexical characteristics. We found that the constituent modification constraints in \textit{JDMWE}
are not hard-and-fast rules but do represent statistical trends for idiom flexibility in the OpenMWE corpus. Finally, we found that the MWE features improve the recall of literal instances of token candidates, which may serve as a guide to future investigations into the cross-type classification power of the same features.

5.5 Conclusion

In this chapter we have investigated various feature representations for discerning true MWE usages from literal expressions that match the canonical syntactic form for MWEs in the OpenMWE corpus. We used a supervised machine learning classification approach to determine the contribution of a variety of families of features including linguistically motivated features and features borrowed from word sense disambiguation techniques. We focused in particular on the extent to which morpho-syntactic variation — which has proven useful for identification of MWE types (Cook et al. 2007) — can provide sufficient information to identify MWE tokens.

This study is motivated by the needs of the web glossing application described in Chapter 4 which identifies dictionary headwords in submitted text: for this application it would be desirable to identify MWE tokens only where they are actually relevant. If it can be done on morpho-syntactic variation alone it could be performed as part of a preprocessing and tagging stage rather than a more compute intensive semantic analysis stage. Ideally it would be applicable to MWE types for which no labelled instances exist so in this chapter we pay particular attention to generalisable models of token candidate disambiguation.

We found that when classifying instances of types unseen in the training data, lexical features of the type including high level semantic categories of its constituents could improve performance over a corpus majority class baseline. An even greater improvement could be found by use of a type flexibility signature derived from morpho-syntactic variation rules encoded in the Japanese dictionary of multiword expressions. However, our type level features alone were not enough to approach the performance of a type-specialised majority class baseline. The inclusion of token-
level features describing specific morpho-syntactic variations improved performance somewhat, but again it was not close to achieving the type-specialised baseline. Only with features traditionally used for WSD such as bag of words and locally offset or syntactically related words were our models able to distinguish true MWE usages of types not seen in the training data at a rate equivalent to a supervised type-specialised majority class baseline. This is an interesting result because the features are traditionally used to model the semantics of a word or document but there is not an implicit common semantics between the literal interpretations of different MWE types. In future work we would like to investigate further why the WSD inspired features perform so well at cross-type classification.

5.5.1 Implications for Wakaran

Our best cross-type classifiers achieved an accuracy of under 75%, which leaves a lot to be desired where user applications are concerned. When testing classifiers on instances of types that were seen in the training set, accuracies approaching 89% were seen, especially when combining the use of WSD inspired features and idiom features from [Hashimoto and Kawahara (2009)] with our own extensions. This is getting much closer to a desirable level of performance, and our investigation of training set size indicates that some degree of greater performance could be achieved by collecting more labelled samples. However the coverage over which this performance can be achieved is very low because it is limited to MWE types with labelled training instances numbering in the order of 1000 or more.

Wakaran, the crowdsourcing application developed for this thesis, includes features allowing users to select competing glosses from a list of glosses for a single lexical unit. Interactions with these features are its primary mechanism for learning about how students interpret the senses of words in context. In Wakaran gloss promotion to inline text does enforce mutual exclusivity between a MWE and its constituents but the enforcement is indirect and the choice is not explicitly presented in the user interface the way a sense selection choice is presented. Although it may be possible to extend the interactive features to glosses that compete on different lexical units the im-
implementation and user interface would be more complicated than it is now. Future work on the user interface should investigate a way to pursue the goal of making the choice more explicit.

As Wakaran stands, the most straightforward way of integrating candidate disambiguation would be to process tokens down to a single gloss by either eliminating a MWE gloss or eliminating the glosses for all of its constituents. Removal of this information is risky because the consequences of getting the classification wrong are that relevant glosses are eliminated entirely from consideration. In this chapter we have concluded that morpho-syntactic flexibility feature alone will not achieve the levels of accuracy needed. The cost of training type-specialised classifiers with a high enough coverage of MWE types for general purpose glossing is prohibitively high, which leaves the cross-type classifiers with an accuracy of 75%. We consider this too low to justify eliminating glosses completely. Therefore we conclude that it is not appropriate to use the methods of this chapter in Wakaran to filter out MWE dictionary entries, nor to filter out literal entries for the constituents. Further investigation into lexical semantic approaches to token candidate disambiguation should be pursued in future work before such an integration occurs.

We chose for this thesis to prioritise investigating integration of WSD with Wakaran over integrating candidate disambiguation. A WSD integration has a more transparent user interaction mechanism and since it will not remove information irrevocably from the output accuracy considerations are less stringent. The next chapter includes evaluation of high coverage WSD algorithms for the JWordNet lexical inventory and Chapter 7 describes implementing WSD for the JMDict gloss inventory, integrating WSD with Wakaran and studying user interactions with senses from a WSD perspective.
Chapter 6

Connecting lexical resources to free text

Acquisition of manual semantic annotations for text from which to learn about language can be costly in terms of human effort, in response to which a number projects have sought to bootstrap new resources for one language from a translation of an existing resource in English \cite{Bentivogli2004,Lupu2005}. In this chapter we describe the transfer of manual sense annotations from the English \textit{SemCor} corpus to a Japanese translation and use the resulting gold-standard annotations to evaluate resources for knowledge-based word sense disambiguation of Japanese. Along the way we pay particular attention to an oft-neglected prerequisite for WSD: the initial identification of lexical knowledge base entries in free text. In Chapter \ref{chap:wsd}, we defined the task of WSD in terms of selecting concepts described in a lexical knowledge base to best describe the meaning of a word token in free text. However, the definition makes an assumption that is not necessarily met by naturally occurring texts: that the text comes with an unambiguous tokenisation into word types found in the available lexical knowledge base. In Chapter \ref{chap:wsdmultiword}, we investigated violations of that assumption introduced by certain ambiguous multiword expressions with idiomatic meanings. However, idioms are not the only source of multi-token ambiguity in Japanese: we will see in this chapter that there can be quite a discrepancy between the coarseness of a morpho-syntactic lexicon suitable for token segmentation and a semantic lexicon for which one might perform WSD. The main causes for this discrepancy are the lack of whitespace segmentation and that Japanese is
Chapter 6: Connecting lexical resources to free text

highly agglutinative/synthetic: this means that complex compounds are both common and undelimited from surrounding text.

A major focus of this chapter will be on translating text tokenisations from a fine grained lexicon with morpho-syntactic resources to a coarser grained lexicon with semantic resources. This is a particular challenge for synthetic languages or those without explicit word boundaries, but even in the English language we may face many choices when enumerating words into a lexicon. We saw in Section 3.1.2 that compounding of words may result in new lexicalisations such as *walking stick* that may warrant their own entry in a dictionary – depending on the purpose of the dictionary of course. Chapter 5 explores the potential inclusion of even longer phrasal constructions in the lexicon and how to recognise when the appearance of an idiom’s constituent words represent the lexicalised phrase and when they do not. In Section 3.2 we saw that the Japanese language, lacking explicit word boundaries, presents an even greater difficulty in deciding precisely what constitutes an entry in the lexicon. The *unidic* and *IPAdic* morpheme lexicons disagree in many places on the basic lexical granularity already, but compounding is common in Japanese so dictionaries like *JWordNet* may choose to recognise lexicalisations compounding several *IPAdic* morphemes. For example, the sequence of three *IPAdic* morpheme tokens *kyōsan*, *shugi* and *sha* may combine in many ways to form *JWordNet* entries: *shugi* “doctrine”, *shugisha* “ideologue”, *kyōsanshugi* “communism” and *kyōsanshugisha* “communist”. Finally, new words are constantly entering general usage, meaning that a lexicon can become outdated (O’Donovan and O’Neill 2008). General purpose tokenisers and morphological analysers (such as *MeCab*, trained on *unidic* or *IPAdic* for Japanese) are a powerful tool for normalisation of inflection but they do not directly discover compound and MWE entries from a special purpose lexicon of interest. In this chapter we will study the problem of mapping between different lexicons, particularly monolingual mappings to different granularities.

The ultimate goal of the chapter is to bridge the gap between free text and the traditional conception of WSD on a tokenised, word-type annotated text. We applied the methods developed in this chapter to build a *Japanese WordNet* tokenisation for a free text translation of *SemCor*, which formed the basis of the official *Japanese Semcor* release. Transferring senses from *SemCor* to our
tokenised *Japanese Semcor* gave us a sense annotated corpus of Japanese text. We then used this annotated corpus to evaluate select resources for knowledge-based WSD methods. This chapter also lays the groundwork for identifying tokens of word types in the Japanese-English translation dictionary *Japanese Multilingual Dictionary (JMDict)* which is less well resourced for knowledge-based WSD. *The Wakaran Glosser* which appeared in Chapter 4 uses the algorithms developed in this chapter for identification of words to gloss with *JMDict* entries. Chapter 7 develops methods and resources for performing WSD relative to the *JMDict* lexicon and sense inventory.

### 6.1 Related work

*Landes et al. (1998)* isolated *WordNet* lexemes in the *Brown Corpus* corpus using the *Brill Tagger* (Brill 1994). They did so by means of some preprocessing for MWEs coupled with customisation of the tagger to account for the preprocessing. The overall process was relatively straightforward, leveraging the English language’s advantage of having analytic morphology and whitespace to guide tokenisation:

1. Additional whitespace was inserted between words and punctuation.

2. MWEs that could be found in *WordNet* were collapsed into single tokens by replacing whitespace with the underscore character. A suitable POS tag based on the *WordNet* entry was appended to the token.

3. The text was then tokenised by splitting on whitespace.

4. Finally a *Brill Tagger*, modified to make use of the POS information appended to MWEs, was run over the text to get indicative POS tags for each token.

Our method for dealing with tokenisation and MWEs must be far more robust in the absence of whitespace delimitation and presence of complex compounds. Instead of splitting on whitespace during preprocessing we tokenise using an existing morphological model to identify word boundaries. Whitespace delimitation also encourages the English tokenisation to agree in granularity with
Chapter 6: Connecting lexical resources to free text

the WordNet lexicon; for Japanese we get a much higher count of MWEs for the JWordNet lexicon relative to the tokenisation. This results in a high level of ambiguity introduced by inflected tokens and overlapping matches for MWEs. To deal with this we developed algorithms to solve the problem of resolving the assignment of lexicon headwords to textual tokens in a one-to-many fashion.

MultiSemCor is a translation of SemCor into Italian with sense annotations transferred from the original English via a translation dictionary [Bentivogli et al. 2004]. The translators of MultiSemCor were asked to explicitly join Italian MWEs in their translation according to WordNet conventions to alleviate the need for MWE identification. However, the tool used for word alignment between the Italian and English SemCor resources, called KNOWA, did also make use of an automatic MWE identification preprocessor. KNOWA is in fact a general purpose Italian-English word alignment system that makes use of information from cross-lingual lexical resources [Pianta and Bentivogli 2004]. It uses morphological and MWE identification preprocessors to generate a ranked list of candidate lemmas and then seeks translations of those candidates in a general purpose English-Italian/Italian-English dictionary. The candidate lemmas are ranked using a heuristic based on their part of speech and the word alignment is designed to prefer translations of high ranked lemmas. The alignment part of the KNOWA algorithm takes SemCor tokens and performs an expanding search starting at the corresponding position in the Italian translation of the sentence for a translational candidate lemma match. It has a fallback mode that searches for matches based on graphemic similarity between English and Italian words. In the work of this chapter, we also transfer senses from SemCor to its Japanese translation by aligning ranked candidate lemmas. Our work has the following differences to the construction of MultiSemCor:

1. We solve MWE identification jointly with word alignment and seek to identify MWEs at the precise granularity of our target lexicon JWordNet. In the process, we must resolve high levels of ambiguity in overlapping matches of lexical knowledge base lemmas to text.

2. We have access to an alignment of SemCor tokens to JWordNet lemmas specified by trans-
lators during translation of the text. Thus we can make alignments that are more closely supervised than those used by MultiSemCor.

3. Unlike Italian, Japanese word order and orthography are very different to English, so we can not make use of similar sentence positions or graphemic similarity to guide alignment. As such, we only align and transfer senses from SemCor tokens selected by the translators.

Poznanski et al. (1998) also uses a ranked list of candidate lemmas for a cross-lingual word alignment but for the purpose of glossing source language text with target language dictionary entries. In this case, there is no target language sentence to align to but rather a lexicon of target language words and MWEs. They define the glossing task as a simplified version of lexicalist machine translation: tokens in the source text are matched to translational equivalencies in a bilingual lexicon; however, unlike sentence translation, target language terms not assembled into a grammatical sentence. Nevertheless, some POS and MWE ambiguity occurs in the source language and Poznanski et al. (1998) proposes a method for dealing with it called prioritised tiling. Prioritised tiling is performed in two stages:

1. Identification of (potentially MWE) candidate lemmas.

2. Selection of a non-overlapping subset of candidate lemmas.

The identification step uses a morphological tagger with sufficient granularity to identify dictionary forms of MWEs which have been inflected or conjugated in the text. Candidate lemmas are generated by taking potentially non-contiguous sequences of morphemes from the text. In this chapter we outline a general form for this step of the algorithm which also includes provision for varying the form of individual morphemes. We also specify detailed implementations for specific applications. The second step, selection of non-overlapping candidate lemmas, is the tiling step. Poznanski et al. (1998) rank candidate lemmas using a priority function and then build a non-overlapping tiling of the source sentence by greedy selection of candidates from the ranking. The priority function is a lexicographic sort key based on five levels of comparison:
Chapter 6: Connecting lexical resources to free text

1. number of source language morphemes;
2. number of intervening morphemes;
3. tagger probabilities on source language morphemes;
4. position of rightmost morpheme; and
5. frequency of target language match.

As with the first step, we explore in this chapter a general form of this step of the algorithm and develop specific variants for different applications.

After tokenising the Japanese translation of SemCor and performing a sense annotation transfer, we use the resulting JSemcor for an evaluation of lexical knowledge base resources for Personalised PageRank based WSD. Moro et al. (2014) use the same lexical knowledge base, BabelNet, together with a WSD algorithm they call Babelfy, which jointly solves MWE identification with WSD and expands the task to include linking encyclopaedic entities to text. Identification of multiword expression token candidates is achieved by extracting 1-5-grams from pre-tokenised text and filtering for exact matches to lemmas in the lexicon (substring matches are allowed for named entities). We use a similar approach but expand the possible matches by using a model of inflection for the tokenisation lexicon. This allows us to extract normalised lemmas such as the compound inshōzukeru “to impress” from its inflected usage inshōzukerareru “to be impressed” in sentence 31 of the Japanese translation of SemCor document br-f03. Babelfy only manages to extract the exactly matching constituent inshō “mental image” when given the same sentence; it does not match zuke[ru]rareru “to [be] attach[ed]” at all. Our method also allows for solving MWE identification jointly with WSD and in fact we do so when transferring gold standard senses from SemCor to its translation in Japanese. However, when we evaluate LKB based WSD on JSemcor we fix the tokenisation to that produced by the sense annotation stage so as to make use of the gold standard

http://babelfy.org accessed 2016-06-09
labels. Moro et al. (2014) include comparison to a variety of other WSD methods including Personalised PageRank for BabelNet. The comparison is performed on a variety of WSD tasks for a variety of inventories including WordNet, Wikipedia and BabelNet but for both Babelfy and Personalised PageRank uses the BabelNet concept graph regardless of the inventory specified by the task.

In this chapter we use a single evaluation task by varying the lexical knowledge base concept graph used to measure the change in WSD performance due to augmentations of the lexical knowledge base. Note that we use Personalised PageRank to perform graded WSD annotation and evaluation whereas Babelfy is inherently limited to single best sense WSD.

The Personalised PageRank WSD stage of this chapter is most closely related to Agirre and Soroa (2009) and Agirre et al. (2014) who released the UKB toolkit\(^2\) for Personalised PageRank that we use for our experiments. They compared the performance results of using Personalised PageRank with a number of extensions of WordNet to include links based on supervised or unsupervised sense annotations on WordNet glosses. Their findings showed that the extension of WordNet with unsupervised but higher quantity concept relations gave better results. We conclude this chapter by comparing a WordNet-with-gloss lexical knowledge base network to the much larger extensions in WordNet++ and BabelNet.

### 6.2 Re-tokenising text

In this chapter we seek to connect concepts described in lexical resources JWordNet and JMDict to expressions in arbitrary text. However, these lexical resources lack an encoding of morpho-syntactic rules for their entries that would allow us to identify conjugated forms such as playing for the entry of to play. Even for exact textual matches, morpho-syntactic information would be needed to disambiguate between homonyms with POS differences such as play and saw which may have entries for both Noun and Verb to distinguish

\[(38) \quad \text{I saw}_V \text{ a play}_N \text{ last night.}\]

\(^2\)See http://ixa2.si.ehu.es/ukb/
Lexical resources with Japanese morpho-syntactic information do exist to solve this problem in the form of IPAdic and unidic. However, their lexicons differ in granularity: slightly from each other but significantly from the lexicons of the semantic resources JWordNet and JMDict. For example, the word 高等学校 (kōtōgakkō) “Senior High School” in JMDict comprises two tokens from the IPAdic lexicon: 高等 (kōto) “advanced” and 学校 (gakkō) “school”. In this section we outline the general form of an algorithm for mapping from a tokenisation using one fine grained source lexicon with morpho-syntactic information into a tokenisation of a coarse grained target lexicon. We develop the algorithm for a specific set of Japanese electronic dictionaries but the same method could be applied, for example, to a dictionary of idiomatic multiword expressions in other languages.

6.2.1 Token candidate generation

In this section we describe an algorithm RETOKENISE which takes a source lexicon tokenisation of some text as input and returns a set of potential target lexicon tokens for the text. We solve the problem of mapping source lexicon tokens to target lexicon token candidates via a token compounding and inflectional variation function of the source lexicon. Formally, we assume that we have access to a function LEXICONMAP which takes a sequence of source lexicon tokens to be compounded and uses morphological knowledge encoded in the source lexicon to predict dictionary forms for a compound of the tokens in the target lexicon. LEXICONMAP must return at least one predicted candidate lemma (l-candidate), but may return multiple candidate lemmas in the case of ambiguity. In the next section we will explore specific implementations of LEXICONMAP. After that, Section 6.2.3 presents algorithms for dealing with ambiguity introduced by over-generation of target lexicon candidate tokens (t-candidates).

The full implementation of the RETOKENISE function appears in Algorithm 1. The
Algorithm 1  Algorithm for generating candidate tokens in a target lexicon from a morphologically rich source lexicon tokenisation.

**Require:** Function LEXICONMAP(tokens) compounds a sequence of source lexicon tokens into one or more potential target lexicon lemmas.

**Require:** Tokens is a sequence of tokens tagged with entries in the source lexicon.

**Require:** $L$ is the target lexicon: a set of lemmas.

```
function RETOKENISE(Tokens, L)
    limit ← max\{\mid lemma \mid \mid lemma \in L\}
    token-candidates ← ∅
    for i ← 1..\mid Tokens\mid do
        j ← i
        repeat
            ts ← ⟨Tokens$_i$...Tokens$_j$⟩
            lemmas ← \{l ∈ LEXICONMAP(ts) \mid \mid l\mid ≤ limit\}
            token-candidates ← token-candidates ∪ \{(i, j, lemma) \mid lemma ∈ lemmas\}
            j ← j + 1
        until j > \mid Tokens\mid or lemmas = ∅
    end for
    return \{(i, j, lemma) ∈ token-candidates \mid lemma ∈ L\}
end function
```

The basic idea of the RETOKENISE algorithm is to iterate over subsequences of the source lexicon tokenisation, applying LEXICONMAP to produce candidate target lexicon tokens aligned to the source tokens. In this version of the algorithm we apply LEXICONMAP only to contiguous subsequences of the source lexicon tokenisation. However, the algorithm could be modified to generate skip-grams – subsequences with items removed or replaced by a wildcard – as was used by Poznanski et al. (1998) in their preprocessing for prioritised tiling. Alternatively, implementations of LEXICONMAP could use knowledge of the target lexicon to selectively ignore tokens in their input n-gram.

Figure 6.1 illustrates with an example how RETOKENISE generates candidate tokens and filters them down by matching lexicon headwords. Figure 6.1a shows the initial input to RETOKENISE for this example: it is the first part of a sentence from JSemcor that has been segmented with the IPAdic lexicon. Figure 6.1b collects data from two stages in the algorithm’s execution: subsequences collects subsequences of the tokenisation that are passed to LEXICONMAP in the inner loop; token-candidates represents the candidate tokens that are generated using the candidate lemmas returned by LEXICONMAP. In this example, LEXICONMAP has returned only
“Sceptics [may deny] the more surprising phenomena of dreams...”

(a) The input has been tokenised and POS-tagged by a morphological analyser.

\[
\text{subsequences} = \langle \text{kaigi} \rangle, \langle \text{kaigi, shugi} \rangle, \langle \text{kaigi, shugi, sha} \rangle, \ldots, \langle \text{shugi} \rangle, \langle \text{shugi, sha} \rangle, \ldots
\]

\[
\text{token-candidates} = \left\{ \langle 1, 2, \text{kaigi} \rangle, \langle 1, 3, \text{kaigishugi} \rangle, \langle 1, 4, \text{kaigishugisha} \rangle, \ldots, \langle 2, 3, \text{shugi} \rangle, \langle 2, 4, \text{shugisha} \rangle, \langle 2, 5, \text{shugishawa} \rangle, \ldots \right\}
\]

(b) Substring inputs passed to LEXICONMAP and candidate tokens generated from the candidate lemma results.

\[
\text{token-candidates} = \left\{ \langle 1, 2, \text{kaigi} \rangle, \langle 1, 3, \text{kaigishugi} \rangle, \langle 1, 4, \text{kaigishugisha} \rangle, \langle 2, 3, \text{shugi} \rangle, \langle 2, 4, \text{shugisha} \rangle, \langle 4, 5, \text{wa} \rangle, \langle 5, 6, \text{kono} \rangle, \ldots \right\}
\]

(c) Candidate tokens returned by RETOKENISE must have a lemma that exists in the target lexicon.

Figure 6.1: Illustration of stages in RETOKENISE execution on an example from the start of sentence 22 in JSemcor document br-f03.
one lemma per token sequence but the next section will discuss cases where it may return more. *token-candidates* is initially built up by indiscriminately including candidate lemmas returned by LEXICONMAP but before the algorithm terminates it filters the list of candidate lemmas to contain only lemmas found in the dictionary, as seen in Figure 6.1c. Note that a great many overlapping candidate tokens are in fact legitimate Japanese words but that *shugishawa*, at least, has been filtered out of the return value. The implementation details of the filtering step are omitted here and may be application dependent. For example, in the corresponding step of the prioritised tiling algorithm, Poznanski *et al.* (1998) use a hash index on MWE constituents to optimise the dictionary lookup.

Filtering has been deferred to the final step of RETOKENISE as a consideration for implementation optimisation: it allows for amortised lookups of candidate lemma in batches and also allows repeated candidates to be looked up only once. These optimisations were used in early versions of the application described in Chapter 4.

Looking again at Figure 6.1b note that the number of input subsequences that LEXICONMAP is called on might grow very large. The total number of contiguous subsequences of a tokenisation is proportional to the square of the number tokens. Specifically, counting every token as an end position and every earlier token as a start position gives the number of subsequences as

\[
\sum_{e=1}^{N} \sum_{s=1}^{e} 1 = \sum_{e=1}^{N} e = \frac{1}{2}N \times (N + 1)
\]

by the basic arithmetic series. However, we assume the target lexicon is a set with finite cardinality so there is an upper limit to the character length of lemmas in the target lexicon. Although we treat LEXICONMAP as a black box for compounding and inflection, we can prune the search space of source token subsequences if we make the assumption that appending tokens to the input generates longer candidate lemmas. That is, we assume that

\[i < j \Rightarrow \text{mincandidate}(\langle \text{tokens}_1...\text{tokens}_i \rangle) < \text{mincandidate}(\langle \text{tokens}_1...\text{tokens}_j \rangle)\]
where

\[
\text{mincandidate}(\langle tokens_1 \ldots tokens_i \rangle) := \min\{|\text{lemma}| \mid \text{lemma} \in \text{LEXICONMAP}(\langle tokens_1 \ldots tokens_i \rangle)\}
\]

This means we can stop extending subsequences of tokens once the generated candidate lemmas exceed the longest in the target lexicon. For example, if the target lexicon for the example in Figure 6.1 contained no words longer than \text{kaigishugi} then after seeing that \text{LEXICONMAP}(\langle kaigi, shugi, sha \rangle) = \{\text{kaigishugisha} \} \text{ RETOKENISE could have detected that the limit was exceeded and skipped all longer token subsequences, moving directly on to the next starting token shugi.}

Even after limiting the size of candidate tokens, the \text{RETOKENISE} algorithm is potentially very memory intensive because it generates all candidate lemmas before checking for their existence in the target lexicon. In practice it may be preferable to perform target lexicon filtering incrementally inside the loop. This will be a matter of trading memory for time: depending on how the target lexicon is stored and queried, time may be saved by amortising the query costs in the final step; otherwise, space may be saved by querying one lemma at a time as candidate lemmas are returned by \text{LEXICONMAP}. This trade-off could be compromised by heuristic filtering — such as with a Bloom filter (Bloom 1970) — within the candidate generation loop itself. We address some of these issues in a specialised variant of \text{RETOKENISE} in Section 6.4.1. An approach that eliminates the generate-and-filter model completely is introduced in Section 6.4.2.

The \text{RETOKENISE} algorithm we have just introduced performs two simple high level tasks and delegates application and lexicon specific details to the \text{LEXICONMAP} function. The high level tasks performed by \text{RETOKENISE} are:

1. choosing subsequences of the source lexicon tokenisation to attempt to match to the target lexicon.

2. aggregation and bulk lookup of target lexicon headword candidates formatted by \text{LEXICONMAP}.

Implementation details specific to the source and target lexicons can be encapsulated within spe-
Algorithm 2  Map a sequence of tokens (from a source lexicon) into a set of predicted compound lemmas in a target lexicon.

Require: Function HEADINDEX(tokens) returns the (index of the) best choice of syntactic head for a token sequence.

Require: Function BASES(token) returns a sequence of possible source lexicon lemmas for token.

Require: Function SURFACE(token) returns the original text of the document.

function HEADHEURISTICMAP(Tokens)

    result ← ∅
    i ← HEADINDEX(Tokens)
    bases ← BASES(Tokens_i)
    surface ← (SURFACE(Tokens_j) | 1 ≤ j ≤ |Tokens|)
    prefix ← CONCAT((surface_j | 1 ≤ j ≤ i − 1))
    suffix ← CONCAT((surface_j | i + 1 ≤ j ≤ |Tokens|))
    for base ∈ bases do
        result ← result ∪ {prefix · base · suffix}
    end for
    return result
end function

specialised implementations of LEXICONMAP. Thus, RETOKENISE forms the common core to a number of applications in this chapter.

6.2.2 Lexicon mapping

The LEXICONMAP function that appears as an unspecified implementation detail in RETOKENISE in Section 6.2.1 is required to make predictions of target lexicon compounds for source lexicon token sequences. In this section, we illustrate the algorithm HEADHEURISTICMAP: a simple high-level template for LEXICONMAP. The template algorithm selects a candidate syntactic head token from a sequence of tokens and maps it to its source lexicon lemma; the remaining tokens retain their source text surface form. For example (traffic, lights) will map to traffic light and (ran, a, mile) will map to run a mile. In particular, note that only one of the tokens has its inflections removed, so baker’s dozens would map to baker’s dozen rather than baker dozen. The purpose of preserving all but one surface form in this way is to preserve grammatical correctness of MWEs.

Pseudo-code for the general form of HEADHEURISTICMAP appears in Algorithm 2. It relies on the availability of three functions specialised to tokens of the source lexicon:
Chapter 6: Connecting lexical resources to free text

SURFACE which retrieves the token as it appeared in the source text.

BASES which extracts a sequence of source lexicon lemmas for the token in order of decreasing likelihood.

HEADINDEX which takes a sequence of tokens and selects the one most likely to inflected if conjugating the sequence of tokens as a phrasal unit.

In Section 6.3.1 we give an implementation of HEADINDEX for Japanese that simply selects the rightmost token in the sequence, which is a good heuristic for a mostly head-final language and works sufficiently well for that application. However, for different languages and applications a more sophisticated implementation of HEADINDEX may be called for. For BASES we included the dictionary form of the token in the source lexicon but also its surface form as a potential inflection in the target lexicon. We include the surface form so as not to completely exclude exact matches to the source text. Another reason to keep the surface form is that idiomatic MWEs are often known by their characteristic inflections (see Chapter 5) and thus may appear inflected in a dictionary headword. A detailed description of a specific implementation of HEADHEURISTICMAP for the IPAdic and JWordNet lexicons appears in Section 6.3.1. For the inflected phrase inshōzukerareru “to be impressed” our implementation of HEADHEURISTICMAP returns \{inshōzukerareru, inshōzukeru\} which contains the uninflected inshōzukeru “to impress” that can be found in JWordNet. On the other hand, for the inflected form oyasuminasai “good night” HEADHEURISTICMAP returns \{oyasuminasai, oyasuminasaru\} and in this case the surface form is found in JWordNet and the inflection-normalised form is not because the inflected form is an idiomatic expression.

In this section we have outlined the RETOKENISE algorithm in the previous section, which requires application and lexicon specific implementation details to be provided by a function fitting the LEXICONMAP interface. The HEADHEURISTICMAP algorithm described in this section proposes a simple implementation template that may be tailored to a variety of source and target lexicon pairs satisfying certain criteria. In particular, the source lexicon is required to encode suffi-
cient information to propose dictionary forms for inflected tokens and to propose a single token in a compound most likely to appear uninflected in a dictionary entry. This basic template serves as a guide for implementation of LEXICONMAP for specific applications and lexicons in this chapter.

### 6.2.3 Candidate token disambiguation

The RETOKENISE algorithm in Section 6.2 may return more than one match in the target lexicon for any given source lexicon token. Firstly, it may generate compounded candidate tokens for overlapping subsequences of the source lexicon tokens. The example in Figure 6.1b illustrates a complex case of this. As a special case, overlapping sequences may be properly nested: one subsequence fully contained within the other, which represents a case of MWE token ambiguity as explored in Chapter 5. Secondly, RETOKENISE may return more than one lemma for a single sequence of tokens due to the behaviour of LEXICONMAP. This situation will commonly arise if LEXICONMAP chooses to supply a predicted dictionary form for a compound but also supply an exact match to the surface text as discussed at the end of the preceding section. If we wish only to identify a set of target lexicon lemmas present in the text then this may be sufficient. Chapter 4 describes the implementation of an application for which it is better to retain (most) candidate tokens than provide an unambiguous tokenisation in the target lexicon. However Section 6.3 which describes the tokenisation of a Japanese translation of the SemCor corpus into JWordNet tokens, requires us to fully retokenise the text into the target lexicon so we need some way to select a non-overlapping subset of token spans from the result of RETOKENISE. Additionally we may also need to resolve ambiguity resulting from multiple matches to a single token subsequence. The results of the Japanese Semcor retokenisation, given in Section 6.3.4 will show that there were indeed many cases of ambiguity between candidate tokens. In this section we outline general algorithms for candidate token disambiguation which we will later specialise to particular applications such as the task in Section 6.3.
Chapter 6: Connecting lexical resources to free text

Candidate disambiguation for RETOKENISE

The problem of selecting an unambiguous tokenisation from the candidate tokens returned by RETOKENISE bears a thematic resemblance to the task we investigated in Chapter 5 which we called the multiword expression token candidate disambiguation task. In Chapter 5 we investigated the problem of selecting between an idiomatic multiword expression candidate token and its constituent candidate tokens. In this section we generalise the task definition slightly to include candidate tokens which overlap without either fully containing the other. As seen in Section 2.3.2, one way to approach it is to classify MWEs in advance is to identify MWE types which always take precedence over their constituent-only interpretation, that is, to identify MWEs which will always be relevant whenever their constituent words appear together. However, it can also be the case that a MWE type is not consistently relevant to the context in which a candidate token appears for it. In this case we need a way to determine for a specific text whether to select the MWE candidate token or to select its constituents’ candidate tokens. Chapter 5 gives a sophisticated treatment of this task for Japanese MWEs using a supervised classification method and shows that it is difficult to succeed at the task without using a representation of contextual semantics such as a bag of words for the full text. However in this section we give a simple algorithm for MWE candidate disambiguation for situations where the supervised method used in Chapter 5 is impractical.

The general form of our algorithm, GREEDYMWEIDENT, is shown in Algorithm (3). It uses the same strategy as the prioritised tiling algorithm of Poznanski et al. (1998) which chooses a non-intersecting subset of candidate tokens by producing a global ranking of all candidate tokens returned by RETOKENISE and then selecting higher ranked matches in preference to lower ranked intersecting matches. GREEDYMWEIDENT requires a ranking function RANKTOKENS which sorts the output of RETOKENISE placing “better” matches before “worse” ones according to a heuristic that may be specialised to the source and target lexicons. As candidate lemmas are selected from the ranking, lower ranked candidates that intersect with the new selection are eliminated from consideration. This version of the algorithm only considers contiguous MWEs, but MWEs with
Algorithm 3 Heuristic algorithm for resolving conflicts in output of the RETOKENISE algorithm.

**Require:** `token_candidates` has the structure of the output of RETOKENISE.

**Require:** A function `RANKTOKENS(token_candidates)` which orders all candidate tokens (as returned by RETOKENISE) with the most preferable first.

```plaintext
function GREEDYMWEIDENT(token_candidates)
    ignore ← ∅
    result ← ∅
    for ⟨i, j, lemma⟩ ← token_candidate ∈ RANKTOKENS(token_candidates) do
        if token_candidate ∉ ignore then
            result ← result ∪ {token_candidate}
            ignore ← ignore ∪ {{it, jt, lemma} ∈ token_candidates | jt ≥ i and it ≤ j}
        end if
    end for
    return result
end function
```

Algorithm 4 Implementation of `RANKTOKENS` which maps each token candidate to a feature tuple (or “key”) and then ranks based on a lexicographic ordering of the keys.

**Require:** `token_candidates` has the structure of the output of RETOKENISE.

**Require:** Function `RANKKEY(token_candidate)` that returns a heuristic comparison key to decide the rank of a token candidate.

```plaintext
function RANKTOKENSBYKEY(token_candidates)
    n ← |token_candidates|
    sort_keys ← \{RANKKEY(token_candidate) | token_candidate ∈ token_candidates\}
    sorted_indices ← INDEXSORT(sort_keys)
    return ⟨token_candidates_{sorted_indices1}…token_candidates_{sorted_indices_n}⟩
end function
```

Intervening tokens could also be handled using the resource consumption model used by Poznanski et al. (1998).

A simple, general, implementation of `RANKTOKENS` is given in Algorithm 4. It extracts features for each candidate token into a feature tuple and then ranks them based on a lexicographic ordering of the tuples. Feature extraction is delegated to function `RANKKEY` which should be specialised to make use of application specific information. For example, one feature extracted by the version of `RANKKEY` described in Section 6.3.2 represents whether or not the candidate token lemma is in a set of hints provided by human translators. Where no special information is available, a simple implementation of `RANKKEY` can just promote sorting based on a longest-first earliest-first
heuristic:

\[ \langle i, j, \text{lemma} \rangle \xrightarrow{\text{RANKKEY}} \langle i - j, i \rangle \]

This basic implementation prefers MWE tokens over their constituents which is a good heuristic when considering that it is unambiguously the right choice for so many classes of multiwords such as noun compounds and unambiguous idiomatic expressions. Returning once again to the example in Figure 6.1 we see a complex MWE disambiguation task for source lexicon tokens (kaigi, shugi, sha) including target lexicon candidate tokens for kaigi, kaigushugi, kaigushugisha, shugi, and shugisha. The full ranking for these alternatives using the simple RANKKEY above is:

1. kaigushugisha “sceptic”
2. kaigushugi “scepticism”
3. shugisha “ideologue”
4. kaigi “doubt”
5. shugi “doctrine”

The first item, kaigushugisha “sceptic” is selected, eliminating all the others. If it had not existed, then kaigushugi “scepticism” would have been chosen over shugisha “ideologue”: they have the same number of tokens so the one that appears earlier in the text ranks higher.

The GREEDYMWEIDENT algorithm described in this section provides a lexicon-independent template for selecting a target lexicon tokenisation from a ranked list of candidate tokens. All lexicon specific implementation details are encapsulated in a ranking function. In this section we additionally described a simplification of the ranking function which reduces the lexicon specific details to a function RANKKEY which maps a candidate token to a tuple of values for use in a lexicographic sort. A GREEDYMWEIDENT implementation can be used as a disambiguation step for the output of the RETOKENISE algorithm described in Section 6.2.1 to identify tokens of target lexicon headwords in the source text. A highly specialised implementation of this collaboration will be examined in Section 6.3.
Word sense disambiguation

Now let us turn to the problem of selecting between target lexicon token candidates which match the same (potentially compounded) source lexicon tokens. We do not give a solution to this problem here but we will discuss some of the implications of the task. Some potential sources of ambiguity LexiconMap may produce include:

- **derivational morphology** may be handled differently in the source and target lexicons. Thus, for example, the noun *running* may need to be mapped to both the noun *running* and the verb *run*.

- **POS ambiguity** may arise when the target lexicon encodes distinct entries for different part of speech. For example, *distance* may have separate entries as a noun and a verb. The knowledge encoded in the source lexicon may help us resolve this ambiguity, particularly if the target lexicon uses the same POS tagset.

- **sense ambiguity** may arise when the target lexicon encodes distinct entries for different meanings of a word even within the same POS.

  Sense ambiguity is a particularly challenging problem. We explore methods for automated resolution of sense ambiguity — that is word sense disambiguation— later in this chapter in Section 6.5 and further still in Chapter 7. For now, having arrived at the task of word sense disambiguation, we have completed the bridge promised in the opening remarks of this chapter. This section has provided a high level tour of the methods we use in the following sections and chapters for bridging the gap between free text and the conventional formalism for WSD. In the next section, we take a deep dive into the topic, by applying the methods developed so far to bootstrapping a sense annotated corpus of Japanese from a translation of the English *SemCor*. 
6.3 Tokenisation of Japanese SemCor

In Section 3.3.1 we wrote about the SemCor project which took news texts from the Brown Corpus and, using a heuristic, segmented them into tokens from the lexicon of WordNet. We additionally wrote, in Section 3.3.3 of existing work translating SemCor into Japanese for the Japanese Semcor project. The goal of the project was to transfer WordNet sense annotations from SemCor onto the translated Japanese Semcor via synset links to JWordNet. The translators for Japanese Semcor encoded novel information intended to aid in sense transfer, in the form of a partial mapping from English tokens to JWordNet lexicon entries. However the raw translation data did not include a tokenisation of the Japanese text into the JWordNet lexicon, let alone a word alignment to finish the sense transfer. In this section we describe the completion of that sense transfer and word alignment.

Figure 6.2 shows the interface used by translators when translating English SemCor text into Japanese. Its main feature is the text box for entry of the Japanese translation as free text. Above the text box appears the line of English SemCor to be translated. English tokens with WordNet sense annotations are highlighted: below the text box these terms appear alongside the JWordNet synonyms that are linked to the same synset. If the translator clicks of one of the JWordNet synonyms its text is inserted into the translation and a record is made that the single word translation was used. However, the full text translation is not marked up in any way to indicate the location in the text of the translated term, so if the text is later deleted, reused or edited an ambiguous token alignment may arise. Nevertheless, this translator click through record provides a rich source for supervised sense annotation transfer. The tokenisation of Japanese SemCor described in this section was designed around enabling this supervised transfer and resolving any ambiguities that arose.

To complete the sense annotation transfer for Japanese SemCor we needed to tokenise the Japanese Semcor text into JWordNet tokens. To achieve this we first tokenised the Japanese texts into IPAdic tokens using the MeCab tokeniser and then applied an implementation of the RETOKENISE algorithm outlined in Section 6.2 to produce the JWordNet tokenisation. For this task
Last 10 translations:

<table>
<thead>
<tr>
<th>128) 悪いことをした</th>
<th>Go</th>
<th>Translator history</th>
<th>Document view</th>
<th>All data</th>
</tr>
</thead>
</table>

Hold  Unsure

Prev
Remove temptations.

English: Remove the child from the scene of his misbehavior.

Japanese: 悪いことをした現場から子供を離す。

Clear  Save

Figure 6.2: Interface used by translators to develop Japanese translation of SemCor text (Bond et al. (2012), Figure 2, page 3).
we designed an implementation of LexiconMap based on the HeadHeuristicMap template from Section 6.2.2. This application calls for a fully disambiguated tokenisation of the Japanese text into JWordNet tokens, so a MWE candidate disambiguation step is required. Our implementation of this step resembles the implementation of GreedyMWEIdent given in Section 6.2.3 in that it ranks candidate tokens returned by LexiconMap across a whole sentence, selecting the best first and eliminating lower ranked collisions. The ranking is produced using a richer feature tuple than the one given in Section 6.2.3, the most notable addition being a feature indicating whether the target lexicon (JWordNet) candidate lemma appears in the translator’s click-through data as a word translation. Such words are greatly preferred over any candidate tokens they may conflict with because:

1. there is a transferable sense annotation available for them; and
2. the translator click-through is a human judgement saying that the JWordNet lemma appears somewhere in the translated sentence.

In the remainder of this section, we go into the details of how we designed the Retokenise implementation for building a JWordNet tokenisation of Japanese Semcor, and the steps involved in transferring sense annotations from SemCor to Japanese Semcor.

### 6.3.1 Lexicon mapping for Japanese

Before we can segment the Japanese Semcor texts into JWordNet tokens we need some way of normalising inflected words to the form they appear in the lexicon. We can do this with one of many freely available tools, each of which will tokenise according to its own dictionary. Juman and MeCab are popular candidates for Japanese, but their dictionaries are more fine grained than JWordNet which contains many compounds of shorter lemmas in, for example, the lexicon of IPAdic which we chose for use with MeCab. On the other hand, chunking parsers like KNP and CaboCha produce a courser grained lexicon but do not explicitly map to JWordNet’s precise lexicon so using them to tokenise Japanese Semcor would still risk a granularity mismatch with the target JWordNet.
dictionary. By using MeCab with the a finely grained lexicon to pretokenise Japanese Semcor and building compounds with our RETOKENISE algorithm higher coverage for JWordNet tokenisation and sense transfer than by using a general purpose tokeniser or parser alone. However, we do need to specialise the general template of LEXICONMAP given in Section 6.2.2 to build candidate lemmas for JWordNet out of IPA dic token compounds.

Since the purpose of this task was to create a freely available resource we chose MeCab for our tokeniser using the IPA dic lexicon since both are freely available and popular. The account of the Japanese language and text processing tools to deal with it given in Section 3.2 then guides the implementation of LEXICONMAP. The head-final property of Japanese is particularly useful: it makes the HEADHEURISTICMAP from Section 6.2.2 a good heuristic for generating compound lemmas. The implementation of the HEADINDEX function follows immediately from this decision: it simply returns the index of the last token:

\[ \text{tokens} \xrightarrow{\text{HEADINDEX}} |\text{tokens}| \]

In Section 6.2.3 we noted that inconsistent treatment of derivational morphology between the source and target lexicon may lead to multiple target lexicon candidates for a token sequence. It turns out that Japanese is such a case because many compound nouns have constituents that are inflected verbs. For example, MeCab sees the word tsumekiri “nail clippers” as two tokens in IPA dic: the noun tsume “nail” and a nominalisation kiri “cutting” from kiru “to cut”. Thus for our present implementation of BASES we return not just the lemma of the token — which would be kiru from our example — but the surface form as well (kiri). Finally we note that for Japanese there is no need to consider whitespace when concatenating tokens to form compounds: the operation can use simple string concatenation. Thus for the example of tsumekiri the tokenisation is \langle tsume, kiri/kiru \rangle and the HEADHEURISTICMAP implementation may be used to generate the following token candidates for the target lexicon:

1. \langle 1, 1, tsume \rangle,
2. \(1, 2, \text{tsumekiri}\),

3. \(1, 2, \text{tsumekiru}\),

4. \(2, 2, \text{kiri}\), and

5. \(2, 2, \text{kiru}\)

This allows the RETOKENISE algorithm to identify each of \textit{tsume} “nail”, \textit{kiru} “cut” and the \textit{tsumekiri} “nail clippers” as candidate tokens in the target \textit{JWordNet} lexicon.

Finally, we wish to tokenise the full text even if it contains content words that have no entry in \textit{JWordNet}. To achieve this we expand the target lexicon for RETOKENISE to include both \textit{JWordNet} and the lexicon of the tokeniser:

\[ L := L_{\text{JWordNet}} \cup L_{\text{MeCab}} \]

where \(L_{\text{JWordNet}}\) is the \textit{Japanese WordNet} lexicon and \(L_{\text{MeCab}}\) is the set of \textit{IPAdic} dictionary forms.

### 6.3.2 Candidate token disambiguation for Japanese SemCor

In the previous section we described an implementation of LEXICONMAP that normalises inflections and generates compounds of tokens of the \textit{IPAdic} lexicon as candidates for tokens from the \textit{JWordNet} lexicon. However, LEXICONMAP over-generates candidates so there may be multiple candidate tokens intersecting on many spans of text. To produce our final \textit{JWordNet} tokenisation of \textit{Japanese Semcor} we therefore needed to select a subset of non-intersecting candidate tokens. We used a variant of the GREEDYMWEIDENT algorithm (Algorithm (3) in Section 6.2.3).

The basic idea of GREEDYMWEIDENT is to rank candidate tokens and then greedily select a non-intersecting subset. We used the candidate token ranking function RANKTOKENSBYKEY shown in Algorithm (4) with an implementation of RANKKEY specialised to the sense transfer task for \textit{Japanese Semcor}. Each candidate token ranking key is a feature tuple and they are compared lexicographically, meaning that candidate tokens are sorted by the first extracted feature and only if
two candidate tokens have the same value for the first feature is the next feature consulted (and so on). The implementation of RANKKEY we used for Japanese Semcor extracts the features described in Section 6.2.3 and extends it with additional features to make use of information provided by the translators while translating SemCor into Japanese. Thus, for the JWordNet tokenisation of JSemcor, RANKKEY returns a tuple of features answering the following questions:

1. Is the target lexicon lemma in the set of Japanese lemmas selected by translators during translation? (Yes is preferred.)

2. How many source lexicon tokens is the match made of? (Longer is preferred.)

3. What position is the first token in the text? (Earlier is preferred.)

4. Is the target lexicon lemma actually in JWordNet, or is it part of the IPAdic lexicon augmentation? (JWordNet is preferred.)

5. Does the target lexicon lemma match the base form or surface form of the head token in the match? (Surface is preferred since this exactly matches the text.)

Feature 1 gives the primary preference to matches to a lemma in the translator click-through data. The next two features, 2 and 3, mean that when potential matches overlap precedence is given first to longer matches (e.g., 米国政府 (beikoku-seifu) “US Federal Government” is chosen over 政府 (seifu) “government”) and then to earlier matches (e.g., 現代化 (kindaika) “modernisation” is chosen over 化する (ka-suru) “to change” where they intersect in the out-of-vocabulary 現代化する (kindaika-suru) “to modernise”). Feature 4 will only ever be false on single token matches to the LMeCab ensuring that the source lexicon fallback candidate tokens always have lower rank than any identical lemmas in JWordNet. The final feature, 5, is designed to prefer compound words formed via derivational morphology. It only comes into play when comparing matches to LJWordNet since the token surface forms not included in LMeCab and a JWordNet lemma will always be preferred to the source lexicon augmentation. The full RANKKEY implementation for the JSemcor sense transfer task is given in Algorithm (5). Note that its input domain has been augmented to
allow for the additional features. The \texttt{RANKTOKENS} and \texttt{GREEDYMWEIDENT} functions must be similarly augmented to make allowance for this.

### 6.3.3 WSD for Japanese SemCor

With the retokenisation of \textit{Japanese Semcor} complete, the remaining step is to transfer sense annotations from \textit{SemCor}. The data we have to do this is the translation click-through data which maps tokens in \textit{SemCor} to \textit{JWordNet} lemmas. Each click-through datum applies only to the translation of a single \textit{SemCor} sentence, which may be zero, one or more sentences in the (manually sentence-aligned) \textit{Japanese Semcor}, but is usually one.

For each click-through datum we nominally perform the following actions:

1. Identify the source token $t_e$ of the datum and the sentence $S_e$ in which it occurs.
2. Identify the Japanese translation $Ss_j$ of $S_e$.
3. Identify the Japanese lemma, $l$, of the datum.
4. Identify all tokens $ts_j$ in the retokenisation of $Ss_j$ which have been assigned $l$.
5. Extract the synset $s$ from the sense annotation of $t_e$.
6. Apply a sense annotation of $s$ to each each $ts_j$.

Unfortunately there are a number of conditions under which this process may fail or at least fail to be deterministic.
• More than one source token is mapped to the same Japanese lemma:
  – If each has the same sense annotation this problem is mitigated.
  – If they have different sense annotations then either or both may apply to the target token.
  – If the Japanese lemma matches more than one target token — that is, \(|ts_j| > 1\) — it is more likely that only one of the synsets applies to each.

• The target (Japanese) lemma is not in JWordNet. This situation arises because translators were initially given an interface to add their own text to the list of translation candidates available for click-through.

• The target (Japanese) lemma is not in the transferred synset \(s\). This situation also arises because translators were able to add their own translation candidates, however in this case it is suggestive that a synset membership is missing from JWordNet.

  Additionally, due to rearrangement of senses between WordNet versions 1.6 and 3.0, some SemCor tokens are annotated with deleted senses and others with more than one sense. The end result of all this is that each click-through datum may lead to zero or more senses being transferred to zero or more \(JSemcor\) tokens, and each \(JSemcor\) token may receive sense annotations from zero or more click-through data. We nevertheless proceeded with the sense transfer as described, counting the incidence of any of the above irregularities for later investigation.

  Once the sense transfer was complete we additionally tagged any \(JSemcor\) tokens which were monosemous in JWordNet with their single synset.

### 6.3.4 Results

The preceding sections revealed that our retokenisation and sense annotation transfer of \(JSemcor\) started with the tokenisation of each sentence with the morphological analyser MeCab using the IPAdic lexicon and tagset. This resulted in 382,762 tokens overall and 148,249 open class tokens, giving averages of 27 and 10.5 per sentence respectively.
The *MeCab* parse of each sentence was then retokenised into lemmas from our combined *JWordNet* and *IPAdic* lexicon by the procedure outlined in Section 6.3 up to the candidate token disambiguation in Section 6.3.2. The result is a tokenisation of the *JSemcor* corpus identifying a large number of usages of *JWordNet* with any match failures filled in by *IPAdic*. We count any open class *IPAdic* tokens and all of the *JWordNet* lemmas as open class words and count both in *SemCor* and *JSemcor*. The results are summarised in Table 6.1.

To see the importance of a robust retokenisation we consider the incidence of MWEs in the click-through translation lemmas provided by translators. Of 61,827 Japanese lemmas suggested for some *SemCor* sentence by the click-through data, 7,551 are found in the *MeCab* parse of the translated text as compounds of *IPAdic* tokens. Of the rest, 44,813 are single token. 9,463 are not found in the translated text at all: the translation interface allowed free editing of the translation text but did not allow click-through word alignments to be undone.

In Section 6.3.3 we detailed a number of conditions which may cause the word alignment (and subsequent sense transfer) from *SemCor* tokens to *JSemcor* tokens to not be one-to-one. In fact, we see 1,734 Japanese lemmas coming from more than one token in the source *SemCor* sentence, though only 190 come from more than one source lemma (and none from more than two). Conversely, 3,252 lemmas suggested by the click-through data match more than once in the Japanese translation of the sentence. Also, the alignment coverage is not complete: 51,450 sense tagged tokens in *SemCor* have not been translated, and 90,525 open class tokens in the Japanese sentence translations have no translation lemma mapped to them. Part of speech distributions for unaligned tokens in both languages are shown in Table 6.2. When compared to *SemCor*, there is a very heavy skew in *Japanese Semcor* towards nouns and, to a lesser extent, verbs. This imbalance

<table>
<thead>
<tr>
<th>Corpus</th>
<th><em>SemCor</em></th>
<th><em>JSemcor</em></th>
</tr>
</thead>
<tbody>
<tr>
<td>Sentences</td>
<td>12,842</td>
<td>14,169</td>
</tr>
<tr>
<td>Words</td>
<td>261,283</td>
<td>382,762</td>
</tr>
<tr>
<td>Content Words</td>
<td>119,802</td>
<td>150,555</td>
</tr>
</tbody>
</table>

Table 6.1: Corpus Size
in POS counts goes a long way to explaining why an alignment could not be found. In fact, the
mapping between WordNet and JWordNet only groups words of a single POS into a single synset,
so no cross POS annotation transfer is possible.

After completion of the word alignment, we perform the annotation transfer. Due to
the word alignment not being one-to-one, and for the additional reasons given in Section 6.3.3,
annotation transfer can result in zero or multiple senses being assigned to a word-aligned translation.
Of the 61,827 Japanese lemmas suggested for a SemCor sentence by the click-through data, 131 are
assigned more than one sense from their SemCor sentence and 13,771 have none. The remaining
47,925 click-through lemma suggestions are assigned a single sense. After taking into account
suggested lemmas which appear more than once — or not at all — in the target sentence, 46,121
JSemcor tokens receive tags from the annotation transfer.

In some cases the word alignment is successful and senses are available for transfer but the
transfer cannot be completed. This happens when the suggested Japanese lemma is user-contributed
and is not found in the synset to be transferred. In fact, this occurred 13,857 times during our sense
transfer, suggesting a large number of potential new synset memberships for Japanese WordNet.

After sense transfer we were left with 61,495 JWordNet tokens in JSemcor without sense
annotations, either because sense transfer failed or because the word was not aligned. Of these
12,144 were monosemous in JWordNet and so were tagged with their single sense. Applying these
single sense annotations brings the total number of sense annotated tokens to 58,265. In the final
corpus we also included open class MeCab tokens which have still not been assigned a Japanese
WordNet lemma as an additional 34,329 words. These may be useful if, for example, one wanted to
generate a bag of words model of the JSemcor text.

<table>
<thead>
<tr>
<th>Part of speech</th>
<th>English Tokens</th>
<th>Japanese Tokens</th>
</tr>
</thead>
<tbody>
<tr>
<td>Verb</td>
<td>13,457</td>
<td>24,698</td>
</tr>
<tr>
<td>Noun</td>
<td>9,979</td>
<td>41,394</td>
</tr>
<tr>
<td>Adjective</td>
<td>10,337</td>
<td>2,794</td>
</tr>
<tr>
<td>Adverb</td>
<td>12,321</td>
<td>5,635</td>
</tr>
</tbody>
</table>

Table 6.2: Part of speech distribution for tokens without word alignment
All in all, approximately one in three tokens of JWordNet lemmas identified in Japanese Semcor received a sense annotation transferred to it by translator click-through data. This is by no means a complete coverage, but it is nevertheless a substantial corpus of supervised sense annotations. Since the sense annotation transfer itself was also supervised to some extent by the translators of the text it is also a potentially more reliable source of sense annotations than a corpus bootstrapped with a fully automatic alignment. Of the sense annotations transferred, approximately one in seven were transferred to compounds of multiple IPAdic tokens illustrating the importance of having a systematic method for identifying MWEs relevant to the lexicon under study.

6.3.5 Distribution

We use the Kyoto Annotation Format (KAF) to share the corpus (Bosma et al. 2009). This is an emerging standard for WordNet annotation. We only use the two lowest layers (text and term), not including any higher levels that exist such as dependencies or geodata. In order to make the data accessible, it has been released under the same license as the English SemCor. 

A sample KAF record is presented in Figure 6.3 containing two words with JWordNet senses (学校 (gakkou) “school” and 戻る (modoru) “return”), IPAdic POS tags for all open class tokens in the MeCab parse, and file and sentence IDs which align with English SemCor.

6.4 Retokenisation in the Wakaran Glosser

In this section, we give details of the algorithm used to identify dictionary entries in text submitted by users of Wakaran, the crowdsourcing glossing system introduced in Chapter 4. In this chapter we have studied the task of identifying entries from a semantic lexicon relevant to arbitrary Japanese texts. The main challenges of the task are applicable to finding glosses for the documents submitted to Wakaran by students: a lack of whitespace segmentation, the occurrence of MWEs and compounds, and inflection of compounded verbs. We have outlined a solution strategy in Section 6.2.
with a detailed implementation for the tokenisation of JSemcor into JWordNet tokens in Section 6.3. In Wakaran we use another adaptation of the RETOKENISE algorithm, with differences arising due to the requirements of Wakaran as an interactive user application being different to those of the task of sense transfer for JSemcor.

6.4.1 Retokenisation for JMDict

JMDict is a Japanese lexicon with multilingual glosses — English being the most common — which has been constructed to serve as an aid to second language speakers of Japanese (Breen 2004). It has a permissive license and is used in a variety of software applications, including many targeting students of Japanese as a second language (see Section 4.2 for a few examples). Wakaran shares in this goal and requires an advanced method for tokenising Japanese text into entries in the JMDict lexicon. JMDict entries include part of speech information but to our knowledge no tagging resources exist for its POS labels; nor any formalised method for JMDict tokenisation. Therefore, we applied the retokenisation methods developed in this chapter to the task of identifying JMDict entries in Japanese text. Due to the difference in application, this task requires different qualities to the JSemcor tokenisation task (Section 6.3). In this section we outline the desired qualities and our implementation.

The main qualitative difference the task of tokenising into JMDict lemmas for Wakaran has to the JSemcor task is that, being a dictionary application for end users, the runtime performance requirements of dictionary software will focus more on latency and co-existence with other applications and less on batch throughput. In Section 6.2 we briefly touched on the cost of having all candidate target lexicon entries in memory at once. For this reason we developed a new algorithm structure which avoids generating a full list of compounded and inflected lemmas simultaneously into memory only to have many pruned away at a later stage of processing.

Another consideration for Wakaran is that each background worker process will need to access a copy of the dictionary for retrieving matching lemmas. Storing a full copy of the dictionary in each may lead to memory availability constraining the number of concurrent workers that can run.
However, storing the dictionary in a separate data store process (such as the SQL server) leads to a per-access overhead for dictionary lookup. Thus querying lemmas candidates one at a time rather than batched together may introduce unacceptable time costs. Thus, the Wakaran RETOKENISE algorithm performs dictionary lookup in batches of bounded size rather than waiting to accumulate all candidates or performing lookup one lemma at a time.

Even where an entry exists in the target lexicon we may not eventually want to include it in our output. Firstly, algorithms such as those described in Section 6.2.3 could be used to filter out unnecessary multiword expressions from the result (or, conversely, to remove unnecessary single constituents of a MWE). Secondly, the POS identified by the morphological tagger for a token may not be an appropriate match for the JMDict entry found. Finally, as discussed in Section 6.2.3, we might use semantic algorithms such as automatic WSD to further filter out uninteresting entries or transform the associated semantic metadata. We do not cover these considerations in this chapter but in Chapter 7 we introduce automatic WSD for Wakaran.

Algorithm (6) gives a parallel version of the RETOKENISE algorithm originally described in Section 6.2. It uses parallel processing to save on accumulating hypothesis target lexicon compounds by filtering for existence in the lexicon in parallel to generation of candidate lemmas from the source lexicon tokens. Parallelisation could be achieved by any pipelining method such as UNIX pipes, multi-threading or -processing or co-routining. Wakaran uses chained python generator functions. An additional advantage of this form of the RETOKENISE algorithm over the first form (Algorithm (1)) is that the output is written to a pipeline and so can undergo further filtering and transformation without accumulating all candidate target lexicon entries in memory.

6.4.2 Eliminating over-generation and filtering

In the previous section, we described a strategy for improving memory usage of the RETOKENISE algorithm to allow for a large pool of Wakaran NLP worker processes to co-inhabit the same server machine. Although this optimisation increases the capacity of Wakaran as a system, under a partial load it does little to increase the speed of response from the server to a user. Al-
Algorithm 6 Parallel algorithm for retokenising text into a target lexicon.

**Require:** Function `LEXICONMAP(tokens)` compounds a sequence of source lexicon `tokens` into one or more potential target lexicon lemmas.

**Require:** Function `LOOKUP(L, lemmas)` returns the subset of `lemmas` found in lexicon `L`.

**Require:** `Tokens` is a sequence of tokens mapped to entries in the source lexicon.

**Require:** `N` is the batch size for lexicon queries.

**Require:** `L` is the target lexicon: a datastore containing a set of lemmas.

```plaintext
function RETOKENISE
parallel(Tokens, L, N)
    RETOKENISE
producer(Tokens, L) | RETOKENISE
consumer(L, N)
end function

function RETOKENISE
producer(Tokens, L)
    limit ← max{\|lemma\| | lemma ∈ L}
    for i ← 1..|Tokens| do
        j ← i
        repeat
            ts ← ⟨Tokens_...Tokens_j⟩
            lemmas ← \{l ∈ L | \|l\| ≤ limit\}
            for lemma ∈ lemmas do
                WRITE(⟨\(i, j, lemma\)⟩)
            end for
            j ← j + 1
        until j > |Tokens| or lemmas = ∅
    end for
end function

function RETOKENISE
consumer(Tokens, L, N)
    lemmas ← candidates ← ∅
    while ⟨i, j, lemma⟩ ← READ do
        lemmas ← lemmas ∪ \{lemma\}
        candidates ← candidates ∪ \{(i, j, lemma)\}
        if |lemmas| ≥ N then
            WRITEMATCHES(L, lemmas, candidates)
            lemmas ← candidates ← ∅
        end if
        end while
    WRITEMATCHES(L, lemmas, candidates)
end function

function WRITEMATCHES(L, lemmas, tokencandidates)
    lemmas ← LOOKUP(L, lemmas)
    for (i, j, lemma) ∈ tokencandidates do
        if lemma ∈ lemmas then
            WRITE(⟨i, j, lemma⟩)
        end if
    end for
end function
```
though the algorithm described in Section 6.4.1 was initially deployed, it was eventually replaced by a direct JMDict tokenisation method which we describe in this section. The primary reason for using a RETOKENISE implementation is the lack of morphological resources for JMDict to allow for identification of inflected forms of dictionary entries. The approach we take in this section is to bootstrap a lexicon of inflectional variations of JMDict headwords by using the IPAdic lexicon used previously for the morphological analysis given as input to the RETOKENISE algorithm.

Bootstrapping the lexicon is a three step process for each JMDict headword:

1. Run the headword through the morphological tagger to get its best tokenisation.
2. Generate inflections of the tokenisation using information in the IPAdic lexicon.
3. For each generated inflection, generate bigram transition probabilities for the new surface form using information from the IPAdic lexicon.

To generate inflections for a tokenised JMDict headword, we reverse the HEADHEURISTICMAP algorithm described in Section 6.2.2. In HEADHEURISTICMAP, a sequence of tokens from free text has its final token converted to the source lexicon lemmatised form as a heuristic for converting the compound of the full sequence of tokens to a dictionary form. In this case we reverse the process: we take the lemmatised form of the final token of a JMDict headword and look up all inflections for that lemma in the source morphological lexicon IPAdic. Note that in English changing the inflection of the final word of MWE would not be sufficient — instead you would look for the head verb or noun of the phrase — but in Japanese syntax the final token is a very good heuristic for the point of inflection. The end result is a set of inflections for the JMDict headword as a whole (whether it is a compound of IPAdic tokens or not).

The final step is to associate bigram transition probabilities to the generated JMDict headword inflections so that they can replace the IPAdic dictionary for use with the conditional random field (CRF) engine of the MeCab tokeniser. One challenge is that the first and last tokens of a compound JMDict headword may have different IPAdic POS to each other and therefore have different transition probabilities with respect to tags to the left and right of the candidate token. However,
MeCab makes this challenge easy to overcome by allowing its dictionary entries to encode separate POS identities for their left and right roles in a bigram. Thus we simply assign the POS of the left-most token of the compound as its POS in the right hand position of a bigram. Likewise, we assign the POS of the rightmost (potentially inflected) token as the POS of the compound in the left hand position of a bigram. The one remaining parameter that needs to be filled is the unigram weight for the *JMDict* headword. For this we take the weight of the rightmost token in the compound that has an open-class POS. The reason we take an open class POS is because affixes, suffixes and particles will give a poor indication of the frequency of the compounded or inflected word.

Having generated inflections with bigram left and right transition classes and a unigram weight we have enough information to compile a *MeCab* dictionary for *JMDict*. For precision in tokenisation we merge the *JMDict* dictionary with the *IPAdic* dictionary so as to fill any gaps in the *JMDict* lexicon, particularly with particles, affixes, and chained inflection morphemes. The main entry-point for *MeCab* operates in a full disambiguation mode, outputting a single tokenisation of non-overlapping morphemes or a list of such tokenisations in order of their estimated probability. However, *Wakaran* is interested in displaying all information about potentially ambiguous tokenisations. We use the lattice labelling mode of *MeCab* to include all candidate tokens which appear in high scoring tokenisations of a sentence to detect cases of MWE and compound ambiguity.

We have thus described the two main re-tokenisation algorithms deployed in *Wakaran* to choose glosses for arbitrary Japanese text. The other large NLP component developed for *Wakaran* is the *JMDict* WSD algorithm described in Chapter 7. To close out this chapter, we discuss experiments in knowledge-based WSD over *JWordNet* which lay the groundwork for it.

### 6.5 WSD with *JWordNet*

In this section we use the sense annotations transferred to *Japanese Semcor* by the work described in Section 6.3 to evaluate resources for knowledge-based Japanese WSD using *Personalised PageRank* algorithms. *Japanese Semcor* does represent a gold standard sense annotation of
substantial size and as such it would be possible to investigate supervised word sense disambiguation training with it as we did using OpenMWE for multiword expressions in Chapter 5. However, since the annotations on Japanese Semcor were transferred from the original SemCor its coverage of word types and the number of annotations per type is inherently limited to that of SemCor and extending coverage would come at the same high cost that extending the coverage of SemCor would require. For this reason, we choose to use Japanese Semcor as an evaluation set to estimate the performance of knowledge based methods which have the potential to achieve coverage over a larger lexicon than a supervised word expert WSD method. The resources for knowledge based WSD we seek to evaluate are JWordNet itself, WordNet++ and BabelNet. The JWordNet ontology is a rich semantic network for Japanese and is based on the English Princeton WordNet. In addition, the transfer of sense annotations from SemCor to JSemcor we developed in Section 6.3.3 gives us a suitable corpus for evaluation of an all words WSD algorithm using JWordNet senses. WordNet++ (Ponzetto andNavigli 2010) and BabelNet (Navigli and Ponzetto 2012) are extensions of the Princeton WordNet lexical knowledge base to use the network of links between pages of English Wikipedia and multilingual Wikipedia. Since they extend Princeton WordNet they are also compatible with the JWordNet tokenisation of Japanese Semcor. Each of the three lexical knowledge bases, WordNet, WordNet++ and BabelNet, is a superset of the one before, so our study serves as an investigation of the increased utility of the ongoing enrichment of a lexical knowledge base. For our WSD method we focus in particular on the network method Personalised PageRank, which has provided good results for WSD over WordNet and for semantic networks in other languages (Agirre and Soroa 2009). In the next section we outline our implementation of Personalised PageRank over JWordNet. Then, in Section 6.5.2 we evaluate the resulting WSD model over JSemcor.

6.5.1 WSD for Japanese WordNet

In our experimental setup we study changes in performance of a leading knowledge-based method for WSD, Personalised PageRank, as the size of the LKB is increased. Our LKBs are all
built on a core of Princeton WordNet synsets, with Princeton WordNet and WordNet++ in particular including only English lexicalisations. In this section we describe how we implemented Personalised PageRank WSD for the Japanese WordNet tokenisation of Japanese Semcor developed in Section 6.3.

A key feature of Japanese WordNet is the mapping between its synsets and the synsets of English WordNet. This provides us with a supervised mapping from Japanese words to the English WordNet synsets at the heart of each of our lexical knowledge bases under examination, and could indeed be used in conjunction with any existing lexical knowledge base containing WordNet synsets to perform network methods for Japanese WSD. We leverage this to reproduce the Personalised PageRank method of Agirre and Soroa (2009) for the JWordNet Japanese lexicon.

The basic form for Personalised PageRank based WSD over JWordNet has the following steps:

1. identification of JWordNet lexicon words in Japanese text;
2. computation of Personalised PageRank to assign weights to WordNet senses of JWordNet headwords; and
3. mapping Personalised PageRank results to WSD judgements.

The first step, identification of JWordNet lemmas in running text, was the main result of Section 6.3 of this chapter.

For the computation of the Personalised PageRank itself we use UKB which was developed by Agirre and Soroa (2009). To this end, we constructed and compiled UKB Personalised PageRank networks for the concept graphs of each of our lexical knowledge bases:

- **WN+gloss** WordNet with sense-annotated gloss relations as used by Agirre and Soroa (2009).
- **WN++** WordNet++, which is to say the network of WN+gloss with additional relations sourced from Wikipedia links between pages mapped to the WordNet synsets (Ponzetto andNavigli 2010).
The full BabelNet 1.1.1 graph (Navigli and Ponzetto 2012), including additional concepts and relations from multilingual Wikipedia (but no Japanese lexicalisations for non-WordNet synsets).

We then constructed a UKB lexicon file linking JWordNet lemmas to their WordNet synsets by using the inherent links between JWordNet and WordNet.

UKB provides a simple mechanism for mapping Personalised PageRank results to both exact match and graded sense judgements, which serves as the third and final step for disambiguation. We configured UKB to provide graded sense judgements, despite the single best sense nature of the annotations on JSemcor, so as to detect shifts in Personalised PageRank towards the gold standard sense even in cases where it fails to pick the best sense. However, it is simple to transform the graded sense annotations into single best sense annotations after the fact. For this reason too we used both exact match and graded evaluation measures when scoring performance of Personalised PageRank. In the next section we detail evaluation of this setup using the JSemcor resource.

### 6.5.2 Evaluation with JSemcor

In Section 6.3 we detailed the transfer of SemCor sense annotations onto the sentence aligned translation, Japanese Semcor. The result is a corpus of Japanese text with links into Japanese WordNet and (transferred) supervised sense annotations. Complete coverage of all open class terms with sense annotations was not achieved: of 116,226 tokens of JWordNet lemmas, only 58,265 were successfully sense tagged (including monosemous words). We do not try to use JSemcor for supervised classification but consider it a good resource for evaluation of JWordNet word sense disambiguation.

We evaluated Personalised PageRank using 3 criteria on 9 distinct configurations. Each of the 27 evaluation settings can be distinguished by a combination of the following factors:

### Resource

$\in$ WN$\text{+gloss}$, WN++ and BBN

as defined in the previous section.
Context size $\in$ SENT, PARA and DOC

meaning sentence, paragraph and document sized contexts respectively.

Criteria $\in$ Exact, RRank, Spearman

being evaluation with the exact match criteria, the reciprocal rank criteria defined below, or evaluation by Spearman rank correlation.

One challenge for evaluation is that the annotations on our gold corpus are for a single best sense whereas our WSD algorithm Personalised PageRank naturally outputs grades for all senses of a word. For the exact match evaluation we resolve the discrepancy by choosing the highest graded sense as the single best output of WSD. For Spearman rank correlation we convert the single best gold annotations on JSemcor into ranks by assigning rank 1 to the gold sense and, applying the standard convention for tied ranks, assign the arithmetic mean of the remaining ranks to the other senses. Finally, we introduce the reciprocal rank (r-rank) method here as a measure that natively resolves the discrepancy between the single best sense gold standard corpus and the naturally graded output of the Personalised PageRank algorithm for WSD. For single token gold-annotated with a single sense, the r-rank score for a graded sense annotator is given by

$$r\text{-rank} = \frac{1}{r_g} - \frac{r_g - 1}{n(n - 1)}$$

where $r_g$ is the rank of the gold sense in the graded annotator’s output and $n$ is the number of senses for the word. The $\frac{1}{r_g}$ represents the most important component of the measure: it is the reciprocal of the ranked position of the gold sense in the WSD algorithm’s output. Thus r-rank is closely related to the mean average precision (MAP) measure commonly used for information retrieval (for example, in the TREC conferences (Voorhees and Harman 2001)): if WSD is viewed as a retrieval task for the correct sense then the $\frac{1}{r_g}$ term represents the precision-at-$r_g$. Since there is only one correct sense to retrieve this is identical to the average precision for the term. However WSD differs from tasks typically evaluated with mean average precision in that the set of query results is a very short bounded list rather than a large collection. This leads to some perverse results if $\frac{1}{r_g}$ is used
Chapter 6: Connecting lexical resources to free text

252

directly. For example, ranking the gold sense at position 2 should be a good result for a word of 10 senses but a terrible result for a word with only 2 senses but in both cases the annotator would get the same score of 0.5. For this reason we add an adjustment term \( \frac{r_n - 1}{n(n-1)} \) which rescales \( r\text{-}rank \) so that ranking the gold sense first scores 1.0 and ranking it last scores 0.0. Overall, the two parts of the \( r\text{-}rank \) can be seen as a \( MAP \) with an adjustment for polysemy of the term. Of our three metrics, \( r\text{-}rank \) is most closely targeted at measuring performance of a graded annotator against a single best sense gold standard. However, the exact match metric provides a good test of how well the annotator can exactly reproduce the gold standard and the Spearman rank correlation measure is a robust and popular measure of graded output. The combination of these three metrics of different qualities reduces the risk of unusual results arising from peculiarities of the evaluation. Final aggregated scores for each measure were calculated as the micro-average score over all annotated gold tokens. Note that when using this aggregation method the exact match criteria represents traditional WSD recall rather than precision or accuracy.

We use the following notation for a configuration of the WSD algorithm:

\[
WSD_{lkbc}^{context} := \text{Personalised PageRank over } lkb \text{ graph per context}
\]

where \( lkb \in \{\text{WN}^{+}\text{gloss}, \text{WN}^{++}, \text{BBN}\} \)

\( context \in \{\text{SENT, PARA, DOC}\} \)

For evaluation we use the notation

\[
M_{measure}(WSD, tokens)
\]

for the result of applying an evaluation measure to the annotations given by \( WSD \) to a specific set of gold standard annotated tokens from the corpus. For convenience, we define evaluation over the
whole corpus:

\[ M_{\text{measure}}(\text{WSD}) := M_{\text{measure}}(\text{WSD}, \text{corpus}) \]

Note that the scope of the evaluation does not restrict the scope of disambiguation. For example, when calculating

\[ M_{\text{Exact}}(\text{WSD}_\text{WNN+}, \text{tokens}_\text{house}) \]

where \( \text{tokens}_\text{house} = \{ t \in \text{corpus} \mid \text{lemma}(t) = \text{house} \} \)

the \text{Exact} score is aggregated only over tokens of the word \text{house} but using annotations output by \text{Personalised PageRank} run over each and every token of each paragraph of the entire corpus.

Since each of the evaluation criteria used is a mean of token scores, we test for statistical significance of differences using the Student’s \( t \)-test or, for annotators with the same coverage, a paired Student’s \( t \)-test for the vector of differences between token scores. \text{BBN} had the same lexicon as \text{WNN++} and therefore the same coverage for \text{Personalised PageRank}. As such, the paired \( t \)-test was used for all comparisons except where \text{WNN+gloss} is compared to \text{WNN++} or \text{BBN}. If we did not find a statistically significant difference between two scores we write

\[ M_{\text{measure}}(\text{WSD}, \text{tokens}) \approx M_{\text{measure}}(\text{WSD}', \text{tokens}) \]

On the other hand, where two scores differ with statistical significance we write

\[ M_{\text{measure}}(\text{WSD}, \text{tokens}) \preceq M_{\text{measure}}(\text{WSD}', \text{tokens}) \]

Results of the evaluation appear in Table 6.3. For all evaluation measures, leaving the lexical knowledge base fixed and varying the context size usually shows a statistically significant improvement when using larger contexts. That is, \text{DOC} almost always does better than \text{PARA}, which does better than \text{SENT}. There are a handful of exceptions for the \text{Exact} and \text{RRank} criteria
Table 6.3: Performance of Personalised PageRank over JWordNet under several configurations and evaluation metrics. The best performing configuration under each metric is indicated in bold. The difference between any scores joined by snaking lines was not found to be statistically significant (by a t-test or paired t-test, LKB coverage permitting). All other differences in scores are statistically significant ($p \ll 0.05$).

<table>
<thead>
<tr>
<th>LKB</th>
<th>Exact</th>
<th>RRank</th>
<th>Spearman</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>SENT</td>
<td>PARA</td>
<td>DOC</td>
</tr>
<tr>
<td>WN+gloss</td>
<td>0.623</td>
<td>0.628</td>
<td>0.637</td>
</tr>
<tr>
<td>WN++</td>
<td>0.667</td>
<td>0.674</td>
<td>0.675</td>
</tr>
<tr>
<td>BBN</td>
<td>0.661</td>
<td>0.668</td>
<td>0.672</td>
</tr>
</tbody>
</table>

In short, on this dataset, expanding the size of the context supplied to Personalised PageRank almost always results in an increase in performance.

The results also show a significant improvement in recall (around 5 percentage points) when moving up from JWordNet (with gloss annotation relations) to WordNet++. However, this improvement is not repeated when expanding from WordNet++ to the full BabelNet1.1.1 graph. In fact, for any given context size the results using BBN are worse than WN++ with statistical significance. That is to say:

$$M_{\text{measure}}(\text{WSD}_{\text{context}}) \subseteq M_{\text{measure}}(\text{WSD}_{\text{context}}) \subseteq M_{\text{measure}}(\text{WSD}_{\text{context}})$$

$$(\text{context}, \text{measure}) \in \{\text{SENT, PARA, DOC}\} \times \{\text{Exact, RRank, Spearman}\}$$
This is a surprising result considering that, intuitively, having more information about concepts and semantic relations should improve the ability of a random walk algorithm to estimate related senses for words in the same context. That intuition is born out by the results for the augmentation of JWordNet with Wikipedia relations to form WordNet++: on every single context size expanding JWordNet to WordNet++ improved every single measure with statistical significance.

For the final step of the experiment, we did a detailed error analysis to examine how shifting from WordNet++ to BabelNet results in worse performance in our experiments.

**BabelNet vs WordNet++**

For our error analysis we examined in detail the lemmas of words affected by the shift from WordNet++ to BabelNet when performing Personalised PageRank. The difference between BabelNet and WordNet++ is of a somewhat different quality than the difference between WordNet++ and JWordNet. WordNet++ introduces only semantic relations between the existing synsets of JWordNet. By contrast BabelNet introduces new synsets to WordNet++ and adds new edges involving at least one of the new synsets. The original, static, PageRank algorithm was a measure of graph centrality and we expect that adding new links and nodes to the graph would change the relative static importance of certain nodes. Specifically, adding additional links may improve the connectivity of some nodes whilst neglecting others. Adding new nodes with connections into the graph will alter connectivity as well, but will also siphon away some of the static PageRank weight to the new nodes. We hypothesised that a change in the static PageRank centrality may have introduced a bias in the PageRank mass received by the senses of certain words. Conversely, it may interfere with the PageRank mass distributed by certain words in context. Either of these changes might lead to a change in performance: in the former case we expect to see BabelNet perform worse than WordNet++ at Personalised PageRank on a consistent set of lemmas; in the latter case we would expect it to perform broadly worse on a consistent set of documents. In this section we employ set similarity methods to look for lemmas and documents on which BabelNet has consistently poorer performance.
To test our hypotheses, we extracted the tokens, lemmas and documents of JWordNet on which the BabelNet and WordNet++ Personalised PageRank disambiguation disagree. More precisely, for each of the entity classes (single tokens, lemmas and documents) we split the corpus up into partitions along entity lines:

\[
singles := \{ \{ \text{token} \} \mid \text{token} \in \text{corpus} \}
\]

\[
\text{lemmas} := \{ \{ \text{token} \mid \text{token} \text{ has lemma } l \} \mid l \in \{ \text{lemma}(t) \mid t \in \text{corpus} \} \}
\]

\[
documents := \{ \{ \text{token} \mid \text{token from document } d \} \mid \text{s.t. } dh \text{ is a document in corpus} \}
\]

where \( \text{corpus} := \{ \text{token} \mid i \in 1 \ldots N \} \) (6.2)

We then calculated the evaluation measures by aggregating over each partition of an entity type individually. Two annotators were considered to disagree on an entity if their scores on the tokens associated with that entity were different with statistical significance. For instance, two annotators WSD and WSD' were said to disagree on a lemma \( l \in \text{lemmas} \) with WSD superior to WSD' on \( l \) if \( M(\text{WSD}, l) \succeq M(\text{WSD}', l) \). Note that in this example, \( l \) is the set of all tokens in the corpus with a specific lemma.

We define the **superiority set (s-set)** of one annotator WSD over another WSD' to be a subset of entities from one of singles, lemmas or documents on which WSD outperforms WSD' with statistical significance. Formally we say:

\[
S(\text{WSD}, \text{WSD}', \text{entity}) := \{ e \in \text{entity} \mid M_{\text{Exact}}(\text{WSD}, e) \succeq M_{\text{Exact}}(\text{WSD}', e) \}
\]

\( \text{entity} \in \{ \text{singles, lemmas, documents} \} \) Defined in Equation (6.2)

Since our focus is on the difference in behaviour of BabelNet against WordNet++, we define for
Chapter 6: Connecting lexical resources to free text

convenience:

\[ S(\text{context}, lkb, \text{entity}) := S(WSD_{lkb}^{\text{context}}, WSD_{lkb'}^{\text{context}}, \text{entity}) \]

where \( \text{context} \in \{\text{SENT}, \text{PARA}, \text{DOC}\} \),

\( (lkb, lkb') \in \{(\text{WN++}, \text{BBN}), (\text{WN++}, \text{BBN})\} \).

This means that we can use, for example, the expression \( S(\text{DOC}, \text{WN++}, \text{lemmas}) \) to represent the set of lemmas on which the \text{WordNet++} based annotator performed better than the \text{BabelNet} annotator (where both are provided whole documents as context). Conversely, the lemmas on which \text{BabelNet} performs better are found in \( S(\text{DOC}, \text{BBN}, \text{lemmas}) \). By examining the lemmas, documents and single tokens on which one of the lexical knowledge bases performs better than the other, we aim to identify any patterns in the introduction of errors when expanding the \text{Personalised PageRank} graph from \text{WordNet++} to \text{BabelNet}. Additionally, by varying the context used for disambiguation we can assess whether any differences we find are a consistent effect of the change of \text{Personalised PageRank} graph or are simply due to random chance.

**Examining the differences**

We constructed the superiority sets for \text{WordNet++} over \text{BabelNet} on all entities and disambiguation context sizes and found that, consistently,

\[ \forall \text{entity}, \forall \text{context}, |S(\text{context}, \text{BBN, entity})| \ll |S(\text{context}, \text{WN++, entity})| \]  \hspace{1cm} (6.3)

This means that for each configuration using \text{WN++}, shifting \text{WN++} to \text{BBN} mainly introduces new errors in WSD rather than swapping a large number of errors for a larger number of different errors. The statistical significance of the paired \( t \)-tests comparing \text{WN++} configurations to \text{BBN} arise naturally from this highly skewed distribution of changes between the annotators. Since the superiority sets for \text{BBN} are of negligible size, we study mainly the superiority sets from the \text{WN++}
Chapter 6: Connecting lexical resources to free text

258

Lemma  子供-n  鉄道-n  幼虫-n  トレーニング-n  舞台-n  単-n  競争-n  目-n
Translation  child  railway  chrysalis  practice_session  stage  wheel  competition  eye
Tokens  26  12  7  6  5  4  4  4

(a) Eight most frequent words on which $WSD_{DOC}^{DOC}$ performs better than $WSD_{DOC}^{DOC}$.

<table>
<thead>
<tr>
<th>Document</th>
<th>br-k24</th>
<th>Difference: 5.9%</th>
</tr>
</thead>
<tbody>
<tr>
<td>Lemma</td>
<td>子供-n</td>
<td>腿部-n 長い-a</td>
</tr>
<tr>
<td>Translation</td>
<td>child</td>
<td>leg  long</td>
</tr>
<tr>
<td>Tokens</td>
<td>26 12 1</td>
<td>12 2 1 1</td>
</tr>
</tbody>
</table>

(b) All words on which $WSD_{DOC}^{DOC}$ performs better than $WSD_{DOC}^{DOC}$ from the top two documents on which $WSD_{DOC}^{DOC}$ performs better overall.

<table>
<thead>
<tr>
<th>Document</th>
<th>br-k24</th>
<th>br-h13</th>
<th>br-j30</th>
<th>br-j53</th>
<th>br-h16</th>
<th>br-k08</th>
<th>br-n17</th>
</tr>
</thead>
<tbody>
<tr>
<td>Difference</td>
<td>5.9%</td>
<td>2.7%</td>
<td>2.0%</td>
<td>2.0%</td>
<td>1.9%</td>
<td>1.7%</td>
<td>1.7%</td>
</tr>
<tr>
<td>Lemma</td>
<td>子供-n</td>
<td>鉄道-n</td>
<td>トレーニング-n</td>
<td>負傷-n</td>
<td>競争-n</td>
<td>食-n</td>
<td>幼虫-n</td>
</tr>
<tr>
<td>Translation</td>
<td>child</td>
<td>railway</td>
<td>practice_session</td>
<td>injury</td>
<td>competition</td>
<td>dinner</td>
<td>wheel</td>
</tr>
<tr>
<td>Tokens</td>
<td>26 12 1</td>
<td>6 4 4 3</td>
<td>4 4 7</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

(c) Most frequent word on which $WSD_{DOC}^{DOC}$ performs better than $WSD_{DOC}^{DOC}$ from each of the top eight documents on which $WSD_{DOC}^{DOC}$ performs better overall.

Table 6.4: Summary of tokens for which $WSD_{DOC}^{DOC}$ gives a better annotation than $WSD_{DOC}^{DOC}$.

side for our error analysis. For convenience, we therefore define:

$$S(context, entity) := S(context, WN++, entity)$$

Table 6.4 illustrates the content of $S(\text{DOC, singles}), S(\text{DOC, lemmas})$ and $S(\text{DOC, documents})$.

Specifically, it shows samples of the words and documents where the best BabelNet based annotator $WSD_{DOC}^{DOC}^{BNN}$ did worse than the equivalent WordNet++ annotator $WSD_{DOC}^{DOC}^{DOC}$. Although some patterns emerge, no consistent semantic theme or lexical crossover between documents is immediately apparent. Most of the difference in performance between the two annotators appears to be due to $WSD_{DOC}^{DOC}^{BNN}$ making errors on a small number of lemmas with high token counts. This distribution of errors is not surprising: it is consistent with Zipf’s law which holds that when ranking lemmas by their frequency in natural language, the frequency tends to fall off in proportion to the reciprocal of the rank (Piantadosi 2014). Since Personalised PageRank makes one-sense-per-context annotations a uniform distribution of errors on lemmas would translate to a Zipfian distribution of tokens. Thus the skew in lemma distribution for tokens on which $WSD_{DOC}^{DOC}^{BNN}$ introduces an error is consistent with
Chapter 6: Connecting lexical resources to free text

(a) Seven most distinctive words of the concatenated set of documents on which $WSD_{DOC}^{DOC}^{WN++}$ performs better than $WSD_{DOC}^{DOC}^{BBN}$

<table>
<thead>
<tr>
<th>Lemma</th>
<th>Translation</th>
</tr>
</thead>
<tbody>
<tr>
<td>父ちゃん-n</td>
<td>dad</td>
</tr>
<tr>
<td>母ちゃん-n</td>
<td>mum</td>
</tr>
<tr>
<td>ウィスキー-n</td>
<td>whiskey</td>
</tr>
<tr>
<td>ダオル-n</td>
<td>terrycloth</td>
</tr>
<tr>
<td>子供-n</td>
<td>child</td>
</tr>
<tr>
<td>雪-n</td>
<td>snow</td>
</tr>
<tr>
<td>擦る-v</td>
<td>rub</td>
</tr>
</tbody>
</table>

(b) Seven most distinctive words of the document br-k24 on which $WSD_{DOC}^{DOC}^{WN++}$ performs better than $WSD_{DOC}^{DOC}^{BBN}$ by the largest margin (+5.9%).

<table>
<thead>
<tr>
<th>Document</th>
<th>Difference</th>
<th>Lemma</th>
<th>Translation</th>
</tr>
</thead>
<tbody>
<tr>
<td>br-k24</td>
<td>5.9%</td>
<td>父ちゃん-n</td>
<td>dad</td>
</tr>
<tr>
<td>br-h13</td>
<td>2.7%</td>
<td>母ちゃん-n</td>
<td>mum</td>
</tr>
<tr>
<td>br-j30</td>
<td>2.0%</td>
<td>ウィスキー-n</td>
<td>whiskey</td>
</tr>
<tr>
<td>br-j53</td>
<td>2.0%</td>
<td>ダオル-n</td>
<td>terrycloth</td>
</tr>
<tr>
<td>br-h16</td>
<td>1.9%</td>
<td>子供-n</td>
<td>child</td>
</tr>
<tr>
<td>br-k08</td>
<td>1.7%</td>
<td>雪-n</td>
<td>snow</td>
</tr>
<tr>
<td></td>
<td></td>
<td>擦る-v</td>
<td>rub</td>
</tr>
</tbody>
</table>

(c) Most distinctive word from each of the top six documents on which $WSD_{DOC}^{DOC}^{WN++}$ performs better than $WSD_{DOC}^{DOC}^{BBN}$

<table>
<thead>
<tr>
<th>Document</th>
<th>Difference</th>
<th>Lemma</th>
<th>Translation</th>
</tr>
</thead>
<tbody>
<tr>
<td>br-k24</td>
<td>5.9%</td>
<td>鉄道-n</td>
<td>railway</td>
</tr>
<tr>
<td>br-h13</td>
<td>2.7%</td>
<td>ロールプレイニング-n</td>
<td>roleplaying</td>
</tr>
<tr>
<td>br-j30</td>
<td>2.0%</td>
<td>触覚-n</td>
<td>tactual_sensation</td>
</tr>
<tr>
<td>br-j53</td>
<td>2.0%</td>
<td>株-n</td>
<td>stub</td>
</tr>
<tr>
<td>br-h16</td>
<td>1.9%</td>
<td>ルソー-n</td>
<td>rousseau</td>
</tr>
</tbody>
</table>

Table 6.5: Distinctive words in documents where $WSL_{DOC}^{DOC}^{BBN}$ performs worse than $WSL_{DOC}^{DOC}^{WN++}$. In all cases words are ranked by tf-idf, with term frequencies counted across a single document or a subset of the corpus and document frequencies calculated across the whole corpus.

We save a formal analysis of the distribution of introduced errors for later in this chapter; here we offer some observations of the raw data. The frequency distribution is not unusual and casual observation yields no semantic trends: there is no immediately apparent difference between the content of the superiority sets $S(DOC, singles)$, $S(DOC, lemmas)$ and $S(DOC, documents)$ and a random sampling of language. This is suggestive of a negative result for the hypothesis that the extension of WordNet++ to BabelNet introduces bias for specific target lemmas. Also indicative of a negative result is that the most frequent errors overall are attributable to a single document each, meaning that there are no significant cases of errors introduced on a specific lemma across a large number of documents. Later in this section we will formalise these observations with a statistic to formally test the hypothesis at hand. However, we first examine the data for the second hypothesis: that BabelNet introduces biases for specific lexical contexts.

Table 6.5 shows the most distinctive words from the set of documents $S(DOC, documents)$ where the best BabelNet based annotator performed worse than the WordNet++ annotator. Distinctiveness of corpus subsets was measured by calculation of tf-idf for the subset. That is, term
frequency was counted across one or more documents in a subset of the corpus and document frequency was counted across the whole corpus. *Tf-idf* was calculated with raw term frequencies and the log-document frequency; that is, with the formula:

\[
\text{term-frequency} \times \log(\frac{N}{\text{document-frequency}})
\]

where \( N \) is the total number of documents in *JSemcor*. There is, of course, a consistent semantic theme apparent for the words extracted from individual documents. For example, the distinctive words for document br-k24 contain many words for family members and household items. However, as was the case in Table 6.4, when counting words on which \( WSD_{DOC}^{BBN} \) introduced errors there appears to be little lexical overlap between documents. The most distinctive terms overall comprise mostly the most distinctive single term from each document in the superiority set. Casual observation of the data has not revealed an obvious trend in the content of documents on which \( WSD_{DOC}^{WN++} \) performs better than \( WSD_{DOC}^{BBN} \).

**Summarising the differences**

Since manual inspection of the differences between using *BabelNet* and *WordNet++* for *Personalised PageRank* has not yielded any clear patterns, we sought a systematic way to measure consistency in the differences. To test for consistent words or contexts on which *BabelNet* introduces errors to *Personalised PageRank* not made by *WordNet++* we examined overlap between superiority sets of similar annotator pairs. We define the **superiority overlap** (overlap):

\[
O(\{\text{WSD-pair}_A, \text{WSD-pair}_B\}, \text{entity}) := \frac{|S_A \cap S_B|}{\min(|S_A|, |S_B|)}
\]

where for \( \chi \in \{A, B\} \) \( S_{\chi} = S(\text{WSD}, \text{WSD}', \text{entity}) \) \( \langle \text{WSD}, \text{WSD}' \rangle = \text{WSD-pair}_\chi \)

The superiority overlap is the size of the intersection between two superiority sets relative to an upper bound on its value. To help understand the interpretation of superiority overlap, consider the
example of

\[ O\{\langle WSD_{\text{DOC}}^{\text{WN++}}, WSD_{\text{DOC}}^{\text{BBN}}\rangle, \langle WSD_{\text{PARA}}^{\text{WN++}}, WSD_{\text{PARA}}^{\text{BBN}}\rangle\}, \text{lemmas} \} = 0.081 \]

This result means that when switching from BBN to \textit{WN++}, the specific lemmas on which performance improves depends on whether paragraphs or whole documents are used for disambiguation to the extent that at most 8.1\% of either s-set of improved lemmas is shared by the other s-set.

We use the superiority overlap to measure the consistency in the pattern of improvements made by a change in one \textit{WSD} configuration parameter when varying a different parameter. After finding the superiority sets of entities on which each \textit{WSD} annotator surpasses its paired \textit{WSD}' annotator, the intersection of the superiority sets is found. This gives the set of entities on which every \textit{WSD} consistently performed better than its corresponding \textit{WSD}'. The size of the intersection will depend greatly on the magnitude of the superiority set sizes so we normalise the set size for the final score. The normalisation factor is the size of the smallest superiority set which is an upper bound for the size of the intersection.

Putting aside the notion of superiority sets, our notion of overlap can be understood by its relationship to the subset containment measure defined by [Broder (1997)]. The containment of one set \( A \) in another set \( B \) is given by

\[ C(A, B) := \frac{|A \cap B|}{|A|} \]

which is the proportion of elements in \( A \) that are also in \( B \) and attains a maximum of 1.0 when \( A \subseteq B \). Overlap can be considered the maximal containment of any one set within the other:

\[ O(A, B) := \max(C(A, B), C(B, A)) \]

\[ = \max\left(\frac{|A \cap B|}{|A|}, \frac{|A \cap B|}{|B|}\right) \]

\[ = \frac{|A \cap B|}{\min(|A|, |B|)} \]

The overlap calculation is asymmetric in the sense that the superiority sets of the right-
hand WSD\(^{\prime}\) annotators of the annotator pairs over their left-hand WSD pairs are ignored. That is, the calculation is specifically targeted at the entities on which the left hand annotators outperform the right hand ones. In this spirit we will refer to the right hand annotators as the baseline set and the left hand annotators as the target set in the following discussion. Overlap scores have a reasonably natural interpretation as the proportion of the smallest superiority set found in both superiority sets. For example, a score of 0.5 means that exactly half of the entities found in the smallest of the superiority sets are shared by the rest. Similarly, a score of 1.0 means that the superiority set of one target annotator is a strict superset of the superiority set of the other target annotator over its baseline.

In order to study the effect of changing one of the lexical knowledge base or context size parameters in isolation of other changes we choose baseline and target sets so that each pair differs in a single specific way. Thus, we define the special case overlap sets:

\[
O_L(lkb, lkb', context_A, context_B, entity) := O\left(\left\{\langle WSD_{lkb_x}^{context}, WSD_{lkb_x'}^{context}\rangle \mid x \in \{A, B\}\right\}, entity\right)
\]

\[
O_C(context, context', lkb_A, lkb_B, entity) := O\left(\left\{\langle WSD_{lkb_x}^{context}, WSD_{lkb_x'}^{context}\rangle \mid x \in \{A, B\}\right\}, entity\right)
\]

\(O_L\) represents the overlap in superiority sets for one lexical knowledge base versus another across a selection of disambiguation contexts, and \(O_C\) represents the overlap in superiority sets of one disambiguation context size over another across a selection of lexical knowledge bases.

We use superiority overlap to answer the questions:

1. does shifting from WordNet++ to BabelNet introduce errors to the same lemmas regardless of other parameters?

2. does shifting from WordNet++ to BabelNet introduce errors to the same topics regardless of other parameters?

In the case of the former, we expect to see large values of

\[O_L(WN++, BBN, context_A, context_B, lemmas)\] and \[O_L(WN++, BBN, context_A, context_B, singles)\]
For the latter we expect to see large values of

\[ O_L(WN++, BBN, context_A, context_B, documents) \]

However, to define “large” we need a point of comparison. Firstly, we note that if shifting from WN++ to BBN consistently introduces errors on the same lemmas or documents then the \( O \)-values will be at or near 1.0. However, there are a couple of drawbacks to comparing overlaps to an ideal of 1.0. Firstly, if an entity in one superiority set is annotated with high accuracy by the other baseline annotator it has no chance to enter the other superiority set. For example, a token \( t \in S_A = S(WSD_A, WSD'_A, singles) \) may be annotated correctly by the other baseline annotator \( WSD'_B \) and thus \( t \notin S_B = S(WSD_B, WSD'_B, singles) \). Thus \( t \) cannot be in the intersection of superiority sets \( S_A \cap S_B \) and may reduce the maximum possible superiority overlap. This will often not eventuate: when \( S_B \) is the smaller superiority set its size determines the denominator in the superiority overlap calculation, which cancels the effect. The exclusion of \( t \) from \( S_B \) only makes \( S_B \) smaller and Equation (6.3) showed that single factor changes in our data tend to bias these shifts in a single direction which makes \( S_B \) more likely to be the smaller superiority set. However, the fact remains that differences between the baseline annotators \( WSD'_A \) and \( WSD'_B \) may prevent the superiority overlap from attaining a full 1.0. Secondly, although our interpretation of superiority overlap as a proportion of a superiority set makes it intuitive to class an overlap of 0.05, say, as low and a superiority overlap of 0.95 as high, intuition does not provide an objective criteria for setting a threshold on overlap. Thus, instead of setting a fixed overlap threshold, we condition the outcome of our test on the data. Specifically, we use a non-parametric estimate of how likely the overlap is to be as large as it is under a simple null hypothesis. Our null hypothesis is that the superiority overlap we measure for a specific set of WSD-pairs is the incidental superiority overlap we would measure.
by forming a set of random pairs drawn from the same annotators. The corresponding $p$-value is:

$$p(WSD\text{-}pairs, e) := \Pr(O(WSD\text{-}pairs', e) \geq o \mid WSD\text{-}pairs' \in \text{permute-pairs}(WSD\text{-}pairs))$$

where $o := O(WSD\text{-}pairs, e)$

and $\text{permute-pairs}(\{(WSD_1, WSD_2), (WSD_3, WSD_4)\})$

$$:= \{\{(WSD_i, WSD_j), (WSD_k, WSD_l)\} \mid (i, j, k, l) \in \text{perms}(\{1, \ldots, 4\})\}$$

(6.4)

$p$-values were calculated by brute force evaluation of $O$ for every permutation of the four $WSD$ annotators into sets of pairs. Since there are only four items to permute, brute force calculation across all permutations is quite feasible. There are $4! = 24$ permutations of the annotators, but the ordering of the pairs does not change the superiority overlap so only 12 calculations are required. See Appendix 6.A for an example of the permutations of

$$\left\{ (WSD_{\text{\sc{Para}}}^{\text{\sc{WN++}}}, WSD_{\text{\sc{BBN}}}^{\text{\sc{Para}}}), (WSD_{\text{\sc{DOC}}}^{\text{\sc{WN++}}}, WSD_{\text{\sc{BBN}}}^{\text{\sc{DOC}}}) \right\}$$

used in the calculation of a $p$-value for $O^C(\text{\sc{WN++}}, \text{\sc{BBN}}, \text{\sc{Para}}, \text{\sc{DOC}}, \text{\sc{documents}})$.

We tested the consistency of the errors introduced by switching from WordNet++ to BabelNet by measuring the superiority overlap $O^C(\text{\sc{WN++}}, \text{\sc{BBN}}, \text{\sc{Para}}, \text{\sc{DOC}}, \ast)$ which should be high under both word and context similarity hypotheses. The disambiguation contexts \text{\sc{Para}} and \text{\sc{Doc}} make a good reference pair because they exhibited the most similar disambiguation results with all other parameters held equal. Note for example in Equation (6.1) the configurations that did not show statistically significant differences were mainly between \text{\sc{Para}} and \text{\sc{Doc}} context annotators. We also saw in Table 6.3 that the absolute difference in performance was quite small. All in all this means that differences between $WSD_{\text{\sc{WN++}}}^{\text{\sc{Para}}}$ and $WSD_{\text{\sc{WN++}}}^{\text{\sc{Doc}}}$ will have minimal impact on the content of superiority sets when switching both to BBN for overlap comparison.

Results in Table 6.6 show that for all entity types the superiority overlap for \text{\sc{WN++}} vs BBN is quite small compared to the overlap of BBN vs \text{\sc{WN+gloss}} and to the overlap of \text{\sc{Doc}} vs
Table 6.6: superiority overlap for various configuration parameters. Errors introduced by switching from WN++ to BBN are in much less consistent locations than switching from either BBN to WN+gloss or switching from DOC to PARA.

<table>
<thead>
<tr>
<th>entity overlap set</th>
<th>singles</th>
<th>lemmas</th>
<th>documents</th>
</tr>
</thead>
<tbody>
<tr>
<td>$O^C(\text{WN++}, \text{BBN, PARA, DOC, entity})$</td>
<td>0.056</td>
<td>0.081</td>
<td>0.250</td>
</tr>
<tr>
<td>$O^C(\text{BBN, WN+gloss, PARA, DOC, entity})$</td>
<td>0.591</td>
<td>0.665</td>
<td>0.827</td>
</tr>
<tr>
<td>$O^C(\text{DOC, PARA, WN++, BBN, entity})$</td>
<td>0.891</td>
<td>0.856</td>
<td>0.600</td>
</tr>
</tbody>
</table>

PARA. Relatively speaking, switching from DOC to PARA usually introduces errors on the same tokens, words and documents, regardless of other configuration parameters. Switching from BBN to WN+gloss also introduces consistent errors: less so on specific tokens but with high consistency on lemmas and documents. By contrast, when switching to BBN from each of the relatively similarly behaved WSD$_{\text{PARA}}^{\text{WN++}}$ and WSD$_{\text{DOC}}^{\text{WN++}}$, the errors introduced show very little superiority overlap in terms of tokens, lemmas and documents. So it seems that the magnitude of overlap between superiority sets when re-configuring BBN to WN++ is relatively small compared to other WSD configuration changes. Inspecting the $p$-values in Table 6.6 we see that under a random shuffle of the annotators in the superiority overlap calculation, 50% of results are as large or larger than $O^C(\{\text{PARA, DOC}\}, \text{WN++}, \text{BBN, entity})$ for every entity. This means that the superiority overlap calculated for the difference between BBN and WN++ is not significantly different to the superiority overlap of a random selection of superiority overlap sets taken between the annotators selected for the calculation. By contrast, the $p$-values for $O^C(\{\text{PARA, DOC}\}, \text{BBN, WN+gloss, entity})$ are substantially smaller (all 0.083), indicating that this level of superiority overlap is fairly unlikely to occur by random chance between the selected annotators. We conclude that we are unable to identify a statistically significant consistency in the tokens, lemmas or documents on which shifting to BabelNet from WordNet++ decreases performance of Personalised PageRank.

In fact, we have found that the specifics of the errors introduced when shifting the Personalised PageRank graph from WordNet++ to BabelNet are rather inconsistent when varying other parameters. Certainly, the differences introduced by dropping from BabelNet to WordNet were more consistent under variation of parameters, and so were the differences when switching from
paragraph disambiguation contexts to whole documents. As such, we are lead to conclude that the Personalised PageRank errors introduced by augmentation of WordNet++ to BabelNet is not skewed to certain topics or words. We note that, at some point, introduction of new paths between less related senses must inevitably lead to random walks distributing noise rather than signal in a Personalised PageRank network.

6.6 Conclusion

In this chapter we investigated the challenges of linking lexical knowledge base concepts to text that is not trivially segmented into lemmas of the lexicon. In particular we studied the Japanese language for which the lack of explicit segmentation and synthetic morphology makes identification of an LKB’s tokens a non-trivial task, noting that our approach may be useful to other languages for a lexicon including a large number of multiword expressions. As a first step we developed a general algorithm for identification of inflections of lexical knowledge base entry lemmas that may span multiple tokens in a morpho-syntactic tagging lexicon, and gave specifics for implementations in Japanese tailored for use with the Japanese translation of JSemcor and for identifying JMDict words in the Wakaran Glosser. We applied the JWordNet tokenisation of JSemcor to the evaluation of resources for an existing knowledge based WSD algorithm. In our evaluation we found that the extension of the JWordNet (with gloss links) network for Personalised PageRank to use the Wikipedia-enriched WordNet++ significantly increases performance. Surprisingly, use of additional interlingual links found in the BabelNet superset of WordNet++ led to a decrease in performance, although when changing other parameters to the PPR computation, the decrease did not have a consistent effect on specific tokens, lemmas, or documents in the JSemcor corpus. In future work, we would like to reproduce these experiments in other languages and using other network methods for WSD to discover where BabelNet’s strengths and weaknesses lie. We would also like to further investigate the performance drop of BabelNet in this experiment by delving into the Personalised PageRank network itself to identify more details of the differences between BabelNet
Chapter 6: Connecting lexical resources to free text

and WordNet++.

The methods developed in this chapter for tokenising Japanese text with respect to a specific lexical resource are an essential component of the Wakaran Glosser of Chapter 4 which uses the translation dictionary JMDict. In Chapter 7 usage data collected from that application is used to construct a sense annotated corpus for the JMDict translation inventory. The corpus is used to evaluate a method for bootstrapping WSD for the JMDict inventory from the BabelNet Personalised PageRank networks used in this chapter.
Appendix 6.A  Permutation example for overlap p-value computation

To make an objective assessment of whether a superiority overlap value such as

\[ O^L(\text{WN++}, \text{BBN}, \text{PARA}, \text{DOC}, \text{documents}) \]

is small or large, we calculated the probability of getting an superiority overlap of equal or larger size using the formula in Equation (6.4). The formula requires calculating permutations the input WSD annotators:

\[ \{ \langle \text{WSD}_{\text{PARA}} \text{WN++}, \text{WSD}_{\text{PARA}} \text{BBN} \rangle, \langle \text{WSD}_{\text{DOC}} \text{WN++}, \text{WSD}_{\text{DOC}} \text{BBN} \rangle \} \]

For this example, those permutations are:

\[ \{ \langle \text{WSD}_{\text{PARA}} \text{WN++}, \text{WSD}_{\text{PARA}} \text{BBN} \rangle, \langle \text{WSD}_{\text{PARA}} \text{BBN}, \text{WSD}_{\text{PARA}} \text{WN++} \rangle \} \]
\[ \{ \langle \text{WSD}_{\text{PARA}} \text{BBN}, \text{WSD}_{\text{PARA}} \text{WN++} \rangle, \langle \text{WSD}_{\text{DOC}} \text{WN++}, \text{WSD}_{\text{DOC}} \text{BBN} \rangle \} \]
\[ \{ \langle \text{WSD}_{\text{PARA}} \text{WN++}, \text{WSD}_{\text{DOC}} \text{BBN} \rangle, \langle \text{WSD}_{\text{DOC}} \text{WBNN}, \text{WSD}_{\text{PARA}} \text{WN++} \rangle \} \]
\[ \{ \langle \text{WSD}_{\text{DOC}} \text{BBN}, \text{WSD}_{\text{PARA}} \text{WN++} \rangle, \langle \text{WSD}_{\text{DOC}} \text{WN++}, \text{WSD}_{\text{PARA}} \text{BBN} \rangle \} \]
\[ \{ \langle \text{WSD}_{\text{PARA}} \text{BN}, \text{WSD}_{\text{DOC}} \text{WN++} \rangle, \langle \text{WSD}_{\text{DOC}} \text{WN++}, \text{WSD}_{\text{PARA}} \text{BN} \rangle \} \]
Figure 6.3: Sample KAF record for スコッティは学校に戻らなかった。(Scotty did not go back to school)
Part IV

Computational semantics and *the*

*Wakaran Glosser*
Chapter 7

Word Sense Disambiguation in Wakaran

In this chapter we describe the implementation and evaluation of word sense disambiguation for JMDict on documents submitted to the Wakaran Glosser, and the collection of human sense judgements from Wakaran users in both the presence and absence of automatic WSD. Our WSD integration for Wakaran starts with the Personalised PageRank models developed in Chapter 6 for which we have an ontology with a rich network of relations, JWordNet, and have previously performed an evaluation over the annotated corpus JSemcor. We describe the adaptation of those WSD models to work with the JMDict lexicon for which no supervised annotated corpora exist and which has only sparse cross-referencing rather than a structured ontological network like JWordNet. The resulting WSD algorithm was deployed with Wakaran near the end of its lifetime to automatically preselect (but not promote to an inline position in the text) the automatically determined best sense for words in document submissions. Thus we have a sizeable collection of submissions for which users saw no automatic WSD and a smaller collection for which users sometimes saw a word sense highlighted when they first opened a gloss. For both of these sets we have records of users choosing to promote certain word senses for display inline in the text. As we showed in Chapter 4, the primary usage scenario for Wakaran was classroom reading activities. As such the collection of submissions contains duplicated text from the same source documents, often in fragments of different lengths and from different positions. In Section 7.3.2 we describe the process by which we
detected this duplication and merged word sense choices from multiple users into single documents in an annotated corpus. Since the users are students and are only rewarded implicitly by choosing a good word sense for their own purposes of understanding the text, we do not count this corpus as having gold standard annotations. In this chapter we will treat them as a “bronze standard” instead. Finally, in Section 7.4 we evaluate a selection of WSD models from Chapter 6 with respect to data collected from Wakaran. The purpose of the investigation is to discover whether data crowdsourced from Wakaran can be of sufficient quality to improve models of computational semantics, what roadblocks to that quality exist, and whether improvements to models of WSD and improvements to the Wakaran glossing interface can be mutually reinforcing.

7.1 Related work

Wakaran has been designed to serve as a research testbed in two mutually reinforcing ways:

1. to crowdsource information about human sense preferences from interactions with word sense visibility features; and

2. to serve as an extrinsic evaluation of WSD as a first class NLP feature.

In Chapter 4 we presented data collected from the usage of Wakaran showing that users chose to use the optional sense visibility features in the course of their usage but we have not tracked the learning outcomes of Wakaran users. However, there have been a number of studies that have verified positive effects on vocabulary acquisition during reading tasks for students of a second language when supplied with optional word glosses and, in particular when those glosses are automatically simplified in some manner. We review a few recent studies in this section. As to the second aim, we are not aware of any studies crowdsourcing sense annotations that arise incidentally from the course of dictionary usage. However, many crowdsourcing studies do exist that set up a dedicated environment for non-professional annotators to contribute sense judgements. We review a selection of such studies in this section. Many more are covered in Section 2.4.
Chapter 7: Word Sense Disambiguation in Wakaran

7.1.1 WSD enabled glossing software

Kulkarni et al. (2008) describes a system which applies state of the art WSD to a vocabulary learning system. The REAP Tutoring System (REAP) which is designed to provide information on the meaning of a second language word in the context of an example usage. Kulkarni et al. (2008) chose to use supervised disambiguation methods placing an emphasis on accuracy as a primary motivator much as Nerbonne et al. (1998) did with their more general natural language processing integration (see Chapter 4). A corpus for 30 target words was manually created for coarse grained sense sets defined by manually lumping together related fine-grained senses in the dictionary used by REAP. The specific machine learning methods used were Support Vector Machines and Naive Bayes. Features used were bag of words for varying sized context windows around the target word and part of speech bigrams for local context. Offline evaluation of the trained WSD models was performed, followed by user studies with 47 human participants. In the user studies the sense selected by the machine learning algorithm was pushed to the top of the sense list in a gloss. This was justified by previous research showing that students often ignore all senses after the first few. WSD was also used for generation of vocabulary tests relevant to the reading task for the ambiguous words. The results showed a statistically significant (with \( p < 0.001 \)) improvement in a vocab test on the disambiguated words for students using the WSD augmented system. This is a highly encouraging result for research into the application of WSD to glossing software!

Rosa and Eskenazi (2011) updated the work of Kulkarni et al. (2008) with REAP by crowdsourcing sense-labelled training texts to expand the coverage of the supervised WSD algorithm to 192 new words. They performed another user study to compare different presentations of WSD results:

1. best sense only,
2. best sense first, and
3. best sense non-first (moved downwards if needed)
Students’ post-reading performance on a vocabulary test and 1-week vocabulary retention were highest for the best-sense-first presentation. Additionally, when asked for their opinion of the glosses, students showed a preference for the presentations that showed all senses over those that showed only one. In our implementation of automatic WSD for Wakaran we keep all senses rather than only the best one so as to avoid the risk of throwing away useful information completely. To give prominence to the automatically selected sense we choose to highlight automatically selected senses rather than re-order them.

Eom (2012) presents a similar study to Kulkarni et al. (2008) involving the implementation of a WSD-enabled glossing system and testing learning outcomes for students using disambiguated glosses. However, the English monolingual dictionary used in the glosser, COBUILD, has a different sense inventory to the WSD system, SenseRelate, which uses WordNet. This necessitated a way of using WordNet WSD results to select COBUILD senses for display. Eom (2012); Eom et al. (2012a) devised a sense alignment algorithm by performing WSD on the COBUILD example sentences for each sense to build a probability distribution for WordNet senses conditioned on COBUILD senses. They then considered the space of all alignments mapping each WordNet sense to exactly one COBUILD sense. A probability score was assigned to each alignment as a product of two factors:

1. the product over each WordNet sense of its conditional probability given the COBUILD sense it has been aligned to (as computed using the WSD output); and

2. a heuristic adjustment to penalise skewed alignments that assign many WordNet senses to one COBUILD sense whilst leaving other COBUILD senses under-aligned.

The heuristic adjustment factor takes the standard deviation of the number of WordNet senses aligned to each COBUILD sense:

1. inverts the ordering by subtracting all deviations from the greatest deviation amongst all alignments to penalise higher variance alignment counts; and
2. applies a normalising constant and optional smoothing constant to make a probabilistic output.

The end result is a static injective mapping from WordNet senses to a COBUILD sense. It is noted that WordNet senses do often bear a similarity to more than one COBUILD but the limitation of mapping to a single COBUILD sense is accepted for the study.

In Wakaran it is also necessary to map from the WSD inventory, JWordNet, to the glosser interface inventory, JMDict. Unlike Eom et al. (2012a) we do not map a single best sense WSD output to the target dictionary. Instead, we develop a dynamic method for selecting JMDict senses from graded WSD output. Our approach bears some high level similarities to that of Chen et al. (2015) in the sense that both studies seek to construct a vector representation of word senses by starting with the content words of sense glosses and transforming them by an unsupervised machine learning method. However, the specifics of our methods are completely different: Chen et al. (2015) initialises a neural sense embedding training procedure by composing precomputed word embeddings from the sense glosses; in Wakaran the goal of our approach is to embed senses in a subspace of the graded Personalised PageRank output.

Eom (2012) and Eom et al. (2012b) prepared two reading tasks with manual gold standard WSD annotations for South Korean learners of English as a second language. A second set of annotations were produced by the knowledge-based automatic WordNet WSD system SenseRelate described in Pedersen and Kolhatkar (2009). Unlike the WSD algorithms used by Kulkarni et al. (2008) and Rosa and Eskenazi (2011) SenseRelate is an all words disambiguation algorithm and does not require sense-specific training data. 60 participants were recruited to the study and divided them into four groups:

- **GS** to be displayed *only* the gold standard sense of polysemous words;
- **SS** to be displayed *only* the sense selected by SenseRelate;
- **AS** to be displayed *all* senses, with no disambiguation; and
NS to be displayed no glosses at all.

Participants were given the texts to read in the glosser system that allowed clicking on words to view dictionary glosses as prescribed for their group. They were administered pre- and post-tests for vocabulary knowledge of selected words from the texts, to measure vocabulary acquisition. Additionally, they answered multiple choice reading comprehension questions after reading the text. All groups made statistically significant improvements to their vocabulary except for the NS group who did not have access to glosses while reading. This shows the importance of dictionary access as part of using reading tasks as a learning activity. The overall ordering of group mean vocabulary improvement was GS > SS > AS > NS with both sense specialised groups showing more improvement than NS with statistical significance and the gold-sense specialised group improving more than the all-senses group with significance as well. This indicates that quality WSD does indeed lead to better learning outcomes. Interestingly however, the average number of glosses accessed by members of each group had the same GS > SS > AS > NS ordering, although in this case the differences were not statistically significant. It is nonetheless noteworthy that the students with the more specialised and more accurate glosses made more use of them, indicating that trust in the information provided by the system and the level of effort required to understand it may be factors in both usage of and benefit from the glossing software. That is not to say that number of glosses accessed is the only factor in learning: the results showed that the accuracy of WSD selections displayed to users in the SS group had an effect on vocabulary acquisition. For words where students failed on the pre-test and later viewed a gloss, 76% of correct senses shown translated to a success for that word on the post-test, whereas only 50% of incorrect senses shown lead to students using the word correctly on the post-test. Interestingly, students usually retained knowledge of words that they used correctly in the pretest irrespective of whether they viewed an incorrect gloss or a correct one. The results for reading comprehension were less well pronounced. The group of students who were displayed the WSD system derived senses (SS) received the highest scores on average, with the students who were displayed gold senses close behind. However, the only pairwise significant difference between the groups was between SS and the group of students with no glosses displayed.
(NS). Overall it seems that it is better to be shown a gloss, even of a nominally incorrect sense, as the quality of the WSD mainly affects the degree of improvement. The results also indicate that being shown the correct gloss alone is more beneficial than being shown all potential senses of the word, at the very least because students use all-senses glosses less often. In Wakaran we aim to gain the benefits of single sense selection without risking full removal of a misidentified correct sense by highlighting senses selected by automatic WSD in the glosser interface without removing the other senses entirely.

Dang et al. (2013) implemented a glosser called RoLo aiming to improve learning outcomes by decreasing the cognitive load of reading long dictionary entries with word senses irrelevant to the context. Like Wakaran, RoLo allows users to manually select a sense from a dictionary pop-up to be recorded for later access. Additionally, RoLo allows users to add their own hypothesised senses to the list. They use two strategies to assist users with long dictionary entries, both based on previous usage the user has made of the glosser:

1. presenting example usages of the same word taken from a store of documents the user has previously requested glosses for; and

2. moving dictionary senses and manually entered entries selected recently by the user (in other documents) to the top of the list.

No semantic model of WSD is used but under the assumption that readers will tend to read documents on the same topic the historic sense selections feature does leverage a one sense per discourse heuristic. Dang et al. (2013) ran an evaluation in which 34 non first language English speakers used RoLo in two sessions using two prepared texts. Half of the users were given access to the sense and usage recall features and the other half only had the sense selection features. When tested for vocabulary acquisition of words common to the two texts, the group who had access to the recall features scored higher with statistical significance. In Wakaran we implemented automatic WSD as a way of simplifying the search for an appropriate word sense and have not implemented any way to connect usage from one session to another. However, in future work the methods used by Dang
et al. (2013) in RoLo may prove useful additions to the Wakaran functionality.

Veras et al. (2014) designed a glossing software package for touch-enabled tablet computers following similar principles to Dang et al. (2013). In this case the monolingual glossing dictionary is targeted at young learners of English as a first language as well as second language learners. The application tracks words the learner has seen before and highlights them in the text. In glosses it presents synonyms of the target word, ranking them by a two factor score:

1. familiarity to the user;
2. appropriateness to the context.

Appropriateness to the context is determined by a simple lexical substitution measure that takes an adjacent word in the text and prefers synonyms of the target that frequently co-occur with it in a large corpus. A second style of available gloss presents definitions instead of paraphrases. To reduce cognitive load this gloss presents only the first sense, but has an interactive feature to show definitions of more senses. Veras et al. (2014) have plans to introduce automatic WSD to the glosser in the future. They also have plans to study the outcomes for learners using the software. Interestingly they have a focus on using these gloss simplification tools as a way of reducing learner reading anxiety when confronted with unfamiliar language.

Kulkarni et al. (2008) and Eom (2012) have started down the path of testing WSD in a human context with a small number of machine learning strategies and contextual features and a single strategy for presenting the outcome of disambiguation to users of a glossing software system. One purpose of our research is to open up the possibility of exploring this area more deeply by opening a glossing system backed by WSD to the world wide web thus gaining access to a much larger pool of potential participants. Additionally, by introducing interactive features for gloss senses we intend to collect more detailed information about user interpretations of word meaning without the need for direct observation of the users in a controlled laboratory.
Chapter 7: Word Sense Disambiguation in Wakaran

7.1.2 Crowdsourcing sense labels

There have been a number of recent studies using methods for acquiring sense annotations from non-experts. Many have used a gamification strategy, inviting study participants to play a computer game designed to elicit word sense intuitions. Examples span from traditional sense annotation tasks that award points for agreement (Venhuizen et al. 2013) through lexical substitution games (Seemakurty et al. 2010) to reflex-based image selection games (Jurgens and Navigli 2014). Gamification techniques are reviewed in detail in Section 2.4. In our study, students select senses from a list while reading a full text, and are motivated by their own information needs to select a good fit for the context rather than by competition with others. The most similar crowdsourcing studies are probably those that most resemble professional annotation but are crowdsourced to non-experts on platforms like Amazon Mechanical Turk.

Passonneau and Carpenter (2014) took a corpus that had previously been annotated by trained participants and reproduced annotations for a subset using a Mechanical Turk crowdsourcing method. They aimed not just to get the same quality annotations at lower cost but also to be able to build a probability model of sense occurrence and individual human annotation accuracy. The point of the probability model was to be able to put a confidence value on the final annotations. To achieve this they collected up to 25 human judgements per token, which is many more than would usually be collected from trained annotators. The model has parameters for the true prevalence of each sense in a categorical distribution and a categorical distribution for each human annotator of the label they will apply conditioned on the true label. The parameters are estimated using a smoothed maximum likelihood estimate for the observed annotator choices with each categorical distribution drawn from a Dirichlet prior. On some labels that the previous professional annotators had been in low agreement, Passonneau and Carpenter (2014) found that they could place a high confidence on a crowdsourced label using the probability model. In particular, the model had the flexibility to represent the extent to which each participant was biased towards selecting the first sense, or to select certain similar distracting senses for specific true senses. Comparing the crowdsourcing...
costs to the previous trained annotation. Passonneau and Carpenter (2014) conclude that each gold sense label cost $0.70USD with training and only $0.33USD from crowdsourcing. Despite the lower cost, the crowdsourced data is arguably more useful: the professionally annotated corpus had an unusually high number of single-annotated tokens whereas the crowdsourced labels come with a large number of labels per token, to the extend of allowing estimates of gold label confidence, sense similarity (Lopez de Lacalle and Agirre 2015b) and also inherent token ambiguity (Lopez de Lacalle and Agirre 2015a).

Martínez Alonso et al. (2013) crowdsources sense annotations to investigate cases where a token takes on two recognised meanings simultaneously. They study the specific case of words that undergo regular polysemy, arising from close semantic relations between concepts. For instances the word bank can vary between a building at a location and the organisation housed there:

(41) The building next to the bank.

(42) I owe the bank a lot of money.

(43) I am going to the bank today.

Here, the word bank takes two or three different meanings: Example (41) refers to the physical building; Example (42) refers to the organisation; and Example (43) refers to both simultaneously. They obtained annotated examples of these usages for various languages with the English subcorpus being annotated using crowdsourcing through Mechanical Turk, with five annotators per token. They note that, in comparison to the trained annotators for the other languages, the crowdsourced annotators very rarely chose the underspecified sense option and hypothesise that the concept is too advanced for untrained annotators. Alonso and Romeo (2014) develop a method for post-processing crowdsourced data with experts to gain more reliable data from a hybrid collection method.

Since in our method we do not control the documents collected nor the tokens for which senses are applied we cannot guarantee multiple annotations per token. Therefore we focus on aggregated statistics and trends for the experiment at this stage, and identify sources of bias in the way students use the Wakaran glossing interface. Based on our results we pose recommendations...
for further use of Wakaran as a source of lexical semantic research data.

7.2 Integrating WSD with Wakaran

While JWordNet comes with a semantic network which enables implementation of WSD using network algorithms such as Personalised PageRank, JMDict has no equivalent network. In Chapter 6 we developed procedures for adapting a tokenisation from a morphologically rich lexicon to tokenise text with respect to an under-resourced lexicon, in that case JWordNet. Using those methods described in Section 6.4.1 we can also successfully identify JMDict headwords in text but the method is oblivious to the English glosses and cannot discriminate between them. On the other hand, the method in Section 6.3.1 can be used to tokenise JWordNet headwords in text and thus to build a context of JWordNet concepts from which a Personalised PageRank network can be seeded, but a JWordNet context model does not directly discriminate between the JMDict senses described by English glosses. In this section we build a mapping between senses in the JMDict inventory and synsets in JWordNet and use it to implement Personalised PageRank WSD for JMDict. We then describe how it was used in Wakaran to highlight the automatically judged best sense when users open a gloss.

7.2.1 PPR sense embedding for JMDict

The essence of our approach is to assign each JMDict headword a semantic real-valued feature space as subspaces of the output of Personalised PageRank over JWordNet and to map its senses to vectors in the space. The dimensions of the vector space correspond to concept nodes that exist in JWordNet and their values correspond to the activation level of those nodes in a Personalised PageRank computation. To ensure relevance of the vectors to discrimination between the word’s senses, the dimensions of the semantic vector are chosen as the union of:

- If the JMDict headwords appears in JWordNet, the synsets of all of its senses therein.
For each word in each English gloss of each sense of the JMDict entry, if it exists in WordNet, the synsets of its senses therein.

Since JWordNet synsets are directly linked to WordNet synsets, we treat them as identical for the purposes of both the Personalised PageRank network and the vector representation of JMDict headword semantics.

Our method for identifying synsets to represent a JMDict headword requires identifying synsets relevant to the English glosses. In Section 6.1 we reviewed the method used by Landes et al. (1998) to assign WordNet lexemes to SemCor tokens. That method did pre-processing to identify MWEs in WordNet and utilised a custom Brill Tagger to account for them. However, their goal was to get 100% coverage of all open class tokens in the text; ours is just to produce a set of synsets related to the Japanese headword, so coverage is not as important. Therefore we use a simplified method:

1. First we preprocess to separate punctuation from tokens using Natural Language Toolkit’s nltk.tokenize.word_tokenize function. This step also implicitly splits tokens on whitespace by identifying strings of non-whitespace text.

2. For POS tags we use Natural Language Toolkit’s recommended maximum entropy Penn Treebank tagger. Tags are mapped to the WordNet POS codes using a simple mapping on the first letter of the Penn Treebank POS tags: J to adjective, V to verb, N to noun and R to adverb.

3. Tokens are post-processed by the morphy lemmatiser that comes packaged with WordNet and is designed to identify WordNet headwords from inflected tokens. It uses a heuristic for stripping common inflective suffixes.

Having chosen the combined WordNet/JWordNet synsets to participate in the semantic vector for each headword, the final step is to assign a vector to each sense. Ultimately we will want to compare these vectors to the output of Personalised PageRank run over the JWordNet graph, so

\[\text{See: https://wordnet.princeton.edu/man/morphy.7WN.html}\]
Chapter 7: Word Sense Disambiguation in Wakaran

Each sense’s vector should be an estimate of the Personalised PageRank activations of the chosen synsets in contexts where that sense of the word is used. Thus, we estimate the vector for a sense by running Personalised PageRank—personalised to the synsets we extracted from the English glosses for that sense alone—and record the final activation for all synsets chosen for the headword.

Thus, we have created a mapping between JMDict senses and JWordNet synsets. Each JMDict headword is assigned a set of related JWordNet synsets based on the Japanese headword and the English glosses. Each sense of the JMDict headword is assigned a vector giving an estimate of the Personalised PageRank result for those synsets in contexts where the sense is used. We now have all the pieces assembled to perform all words WSD over JMDict for arbitrary Japanese texts:

- the JMDict sense to JWordNet synset mapping described here;
- the model for Personalised PageRank WSD over JWordNet described in Section 6.5;
- the JWordNet tokenisation algorithm described in Section 6.3.1; and
- the JMDict tokenisation algorithm described in Section 6.4.1.

Precisely how these parts come together is the subject of the remainder of Section 7.2.

7.2.2 Disambiguating JMDict words in context

The core idea of our disambiguation algorithm for JMDict is to take a context sensitive Personalised PageRank and compare it to the vector representation developed for JMDict senses developed in Section 7.2. However, in order to do this two pieces need to be in place:

- Seed concepts for the Personalised PageRank network.
- JMDict headwords to disambiguate from the text itself.

In this section we describe how each of those are identified and how the Personalised PageRank result is compared to the precomputed vectors for each JMDict sense.
Morphological processing and tokenisation

We saw in Chapter 6 that automatic identification of words from a Japanese lexicon in ordinary text is a non-trivial task. This is for a large part due to the lack of explicit space delimited word boundaries. In Section 6.3.1 and Section 6.4.1 we developed the means to identify words from the \textit{JWordNet} and \textit{JMDict} lexicons respectively, via a mapping from a finer grained morphological tokenisation lexicon. We use both for the purposes of WSD for \textit{JMDict}: using the same core \textit{IPAdic} tokenisation, MWEs in both lexicons are identified using the methods of Chapter 6. Thus, given an input text $T$, we compute $T_{JWN}$ and $T_{JMD}$, tokenisations in \textit{Japanese WordNet} and \textit{Japanese Multilingual Dictionary} respectively.

Contextual PageRank

We seed the personalisation vector for the \textit{Personalised PageRank} network using the standard method used for monolingual WSD. That is, we take the set of lemmas found in $T_{JWN}$ and find the union of all concepts found in the lexical knowledge base for those lemmas. Although \textit{Personalised PageRank-w2w} performs better than \textit{Personalised PageRank} for standard WSD (Agirre et al. 2014), it is not appropriate to this task since we are not disambiguating single words in the lexicon of the lexical knowledge base, and so are not able to leave out concepts linked directly to a word targeted for disambiguation. Instead, we perform a standard \textit{Personalised PageRank} calculation and extract the final weights of all concepts used for the vector representations of the \textit{JMDict} lemmas in $T_{JMD}$, as computed in Section 7.2.1.

Grading of \textit{JMDict} senses

To distinguish between \textit{JMDict} senses we extract headword specific vectors from the \textit{Personalised PageRank} full context vector and score each sense according to its vector similarity to the context. A number of conventional vector space similarity measures exist that could apply to this task, including:
Cosine similarity, which gives the cosine of the angle between two vectors. It is simple to calculate:

$$\cos(\theta) = \frac{a \cdot b}{|a||b|}$$

where “$\cdot$” is the standard inner product and $\theta$ is the angle between vectors $a$ and $b$. This is a bigger-better measure since $\cos(\theta)$ decreases for vectors that have greater angles between them.

$\ell^n$ similarity, which is a family of measures on the difference of two vectors:

$$\ell^n(a,b) = \sqrt[n]{\sum |a_i - b_i|^n}$$

which for $n = 2$ is the euclidean distance. This is a smaller-better measure since $\ell^n$ increases for vectors that have greater distance between them.

Cosine similarity is independent of the magnitude of the context vector, depending only on the relative mixture of weights assigned to concepts by the Personalised PageRank. This seems at first an attractive feature if we consider a JMDict sense to be a mixture of WordNet concepts. However, some polysemous words in JMDict map to only a single concept in WordNet. These words cannot be disambiguated by the cosine measure since all such vectors line in the positive real line with pairwise similarity of $\cos(0) = 1.0$. This special case highlights the relevance of the magnitude: each JMDict sense will have either a high or low score for the WordNet concept depending on whether a synonym appears in the sense’s gloss. In general, we do not want to factor out magnitude when comparing Personalised PageRank since they are already effectively normalised by the constraints of the algorithm. Thus we use the $\ell^2$ metric for comparing JMDict senses to a Personalised PageRank context vector.

**Presentation**

In the final output we have two options for presenting the results of disambiguation:
graded by similarity may be confusing since the similarity values are not easily interpretable and even with interpretable values different people tend to operate on different magnitudes (Erk et al. 2013).

ranking by similarity is the most appropriate to account for agreement between annotators giving grades (Erk and McCarthy 2009).

single best best achieves the aim of reducing the information presented to a user of the application, however it suffers the most when the WSD algorithm gets the wrong answer.

Ranking would have been our preferred approach. However, documentation accompanying JMDict indicates that certain conventional markings on senses such as part of speech and formality labels carry on to all subsequent senses. Although these can be automatically propagated it indicates that the glosses are intended to be presented sequentially so erring on the side of caution we chose not to re-order them. Instead we choose the single best (of greatest similarity to the context) to highlight in the user interface of Wakaran. To mitigate the impact of making the incorrect selection we continue to display all other senses, just without the highlighting received by the single best. It is the nature of the interface of Wakaran that users are then free to highlight a different definition if they are not happy with the selected sense. One upside to this situation from a research perspective is that users have a self-interest and low barrier of entry to provide the Wakaran system with corrections to annotations that it gets wrong. Note that we do not want the automatic WSD feature to remove the optional nature of gloss consultation, so we do not go so far as to promote the selected sense inline in the text. We only activate the selection feature that expands a gloss and increases its display contrast. Figure 4.1 (in Chapter 4) shows the distinction between a selected sense (sense 2) an unselected sense (senses 3+) and a sense that has been promoted inline (sense 1).
7.3 A corpus of Wakaran submissions

The ultimate purpose of the present study with Wakaran is to show the feasibility of evaluating WSD algorithms with data collected from events naturally occurring from its use: that is the subject of this section. We start by describing the construction of a corpus of documents submitted to Wakaran and the conversion of user interaction data into annotations on that corpus. To handle cases where the same text — or different parts of the same text — were submitted by multiple users we also describe the method we used for near-duplicate and subdocument detection. Finally, we examine the sense selections made by users and compare them to selections made by a number of configurations of automatic WSD systems.

7.3.1 Exporting and annotating the corpus

Our first step towards using document submissions for WSD evaluation was to export a faithful copy of each submission to an on-disk format. We chose the Kyoto Annotation Format format (Bosma et al. 2009) for its expressivity and similarity to the Wakaran data model (see Section 4.5.1). A description of the KAF format with examples of how it was used in the Japanese Semcor distribution can be found in Section 6.3.5. In this section we outline how submission content and user activity data was encoded into sense annotated KAF documents.

Information exported to the corpus

Each submission was exported to a single KAF file and assigned a directory and filename based on the stored corpus and document name attributes of the submission. Wakaran assigned a new corpus name to each user session so the exported Wakaran corpus is organised into subcorpora by user session.

The exported documents are labelled with linguistic processor metadata indicating how the submission was processed by Wakaran at the time it was submitted. These include each of the following processors in one of its versions:
MeCab for tokenisation

Wakaran sentence segmenter for sentence and paragraph labels on tokens (noting that paragraph labels were used to define WSD context boundaries).

Wakaran JMDict retokeniser for linking of JMDict entries to tokens in the form of Wakaran glosses, with versions:

1.0 The initial pipelined implementation of the retokenisation algorithm described in Section 6.4.1. This version filtered out glosses for common stopwords from the interface.

1.1 A modular re-implementation of the JMDict retokenisation. This version produces the same tokenisation as the previous but stopword filtering was left out so as to present all available entries of the JMDict dictionary.

2.0 The optimisation of JMDict retokenisation outlined in Section 6.4.2 which avoids the overgenerate-and-filter approach to JMDict headword identification. The version number was increased to 2 for this version because it changes the set of JMDict entries identified as glosses on some texts.

Wakaran JMDict Personalised PageRank for automatic selection of JMDict senses in the Wakaran interface using the method described in Section 7.2.2. This existed in two versions, or three if you count the time with no automatic disambiguation:

No version of WSD ran at all for some time after the first release of Wakaran.

1.0 The initial release used cosine similarity to compare Personalised PageRank output vectors to JMDict sense representation vectors which caused perverse results for JMDict senses that had only been mapped to a single dimensional vector. This version ran for just over one week only and is not used when aggregating scores for WSD algorithms later in this chapter.

1.1 The main release using $\ell^2$ similarity.
The token layer of the Wakaran data model records the surface text segments and original character indices of the MeCab tokenisation performed in the first step of the Wakaran retokenisation algorithm (see Section 6.4.1). This translates neatly into the text section of a KAF document. Note that neither Wakaran nor the KAF export record part of speech or lemma information for tokens — for KAF such information comes only in the terms layer. Identifiers for the token elements in the KAF export are derived from the primary key of the token in the Wakaran data model.

The terms layer of the KAF export of Wakaran records the output of the Wakaran retokenisation algorithm in the form of externalRef links to JMDict senses. Gloss objects in the Wakaran data model are mapped directly to term elements in the KAF model:

**token links** of the Wakaran gloss become KAF target elements contained in the term. Each links to the KAF element ID of the exported token. In this way the exported KAF term links to the set of tokens its source gloss was linked to.

**The gloss index** of the gloss in the submission’s list of glosses becomes an element ID for the KAF term.

**headwords** assigned to the gloss from JMDict headword entries become the lemma of the KAF term.

**paraphrases** are links from the gloss to JMDict senses and are exported to the KAF term as externalRef elements with an identifier encoding the JMDict entry id and the sense number.

The paraphrases listed in a Wakaran gloss have mutable state that can be affected by user interaction and by the output of Wakaran’s automatic word sense disambiguation. Information about the state of each paraphrase is exported as externalRef elements nested inside the reference already exported for the paraphrase itself. These nested externalRef elements are marked with the source of the annotation. In the case of the user annotations, they are marked with the random identifier assigned to the browser session of the submission. Automatic WSD annotations
are marked as coming from the *Wakaran Personalised PageRank* annotator. Exported information includes:

**user selections** of paraphrases, where a paraphrase has been highlighted by the user, are exported as a session annotation and assigned a confidence of only 0.1 because we found that users almost never deselect paraphrases once selected.

**user promotions** of paraphrases, where a paraphrase from the gloss pop-up has been chosen to be displayed above the linked tokens in the main text, are exported as a session annotation and assigned a confidence of 1.0 to clearly distinguish them from selections which have a much more subtle user interface impact.

**automatic selections** made by the *Wakaran Personalised PageRank* annotator are exported as *Personalised PageRank* annotations and represented by the score assigned by the *Personalised PageRank* annotator to the paraphrase. Note that *Wakaran* normalises annotator scores to the range \([0, 1]\) and sets the initial state of a paraphrase to selected if and only if its score was 1.0 — that is, the sense had the highest score of all senses of the word.

The *Wakaran* sense identifier encoding changed between *Wakaran* retokeniser versions 1.N and 2.0 to make it future proof to upgrades of the version of *JMDict* used by *Wakaran*. When exporting a document to *KAF* we used the sense identifiers of the retokeniser that processed the submission. However, we built a map from the version 1.N encodings to the version 2.0 encodings and for the remainder of this section they are treated as identical, having been mapped to the 2.0 format where necessary.

In its lifetime, *Wakaran* had a number of incremental changes but only one major update: the release of automatic WSD for glosses. As we will later see, user behaviour changes significantly between those two major versions, thus we take care to label each submission with the WSD version used. This effectively splits the corpus into two parts: the *USER-ONLY* and *WITH-WSD* subcorpora. In fact, for approximately the first week of its deployment, the *Wakaran* WSD implementation operated with the cosine vector similarity measure before switching to the final \(\ell^2\) measure (see
Section 7.2.2. For this reason, we take USER-ONLY to comprise documents before the WSD release date of 8 October 2013, and WITH-WSD to be the release date of the $\ell^2$ measure: 16 October 2013. Documents from the interim period we denote the COS-WSD subcorpus: these documents are excluded from most of our analysis.

### 7.3.2 Near duplicate and subdocument detection

The Wakaran corpus contains a large number of segments of duplicated text, due mainly to classroom activities being its primary use case. We used well established techniques for near-duplicate and subdocument detection to identify and submissions as being sourced from the same document; and merged clusters of submissions from the same source into a single bronze standard for evaluation. The purposes of this preprocessing step include:

- to reduce work in running WSD algorithms for evaluation repeatedly on the same text;
- to retain more context for WSD algorithms to access rather than test them on arbitrary fragments of sentences;
- to prevent highly duplicated text from being assigned greater weight in evaluation;
- to increase the density of annotations on bronze standard documents and represent potential differences in human judgement in the bronze standard.

In this section we outline the methods used for near-duplicate and subdocument detection and how annotations were merged to form a more densely annotated bronze standard.

**Detecting near-duplication**

For a number of reasons text submitted to Wakaran from the same source document does not come through exactly the same way every time. A major source of differences between submissions is that many users select only a part of the whole text for submission. In addition to the major content variation, there are small insertions or deletions of what appears to be page numbers, header
or footer text and explanatory notes in parentheticals. There are also many cases of a single heading word or final punctuation being included by some and not by others. Due to these various kinds of difference we need to be able to detect not only when the text of one submission is a smaller part of another but also cases of this happening with small differences in the content as well. To account for this we used the methods of Broder (1997) for detection of duplication in document collections.

Given that we expect to find text duplication due to students copying text from a single copy distributed by a teacher, it is tempting to use a standard method from computer science for exact substring detection such as tries or suffix arrays. However, manual inspection of the data reveals a proliferation of minor differences ranging from insertion of stray characters such as triangles, underscores and single digits to insertion of short parentheticals containing Japanese paraphrases or readings. We put this down to differences in the users’ host system software in terms of web browser, operating system text clipboard behaviour and the document viewer used to view the original document. In particular, our own experience with Portable Document Format (pdf) document viewers and web browsers has been that out-of-flow content such as page numbers, footnotes — interlinear glosses — and paragraph dividers have inconsistency between different applications and platforms in how they are converted to plain text when copied to the operating system clipboard. One other change we observed was cases of a single word being replaced or corrected, which manifests as another case of small differences between texts. Consequently, detecting duplicate text in a system designed for free submission of text without control over the user software or document content requires a method for inexact matching.

An old method for measuring similarity of two pieces of text is the Ratcliff/Obershemp algorithm of Ratcliff and Metzener (1988). The basic idea of the algorithm is to find the longest common substring between the two texts and then to split each text at the location of the common substring. The algorithm then recurses on the parts of the texts before the common substring and, likewise, the parts of the text after the substring. The final similarity is the proportion of the number of characters of the text aligned as common substrings divided by the total number of characters in both texts. A side effect of the algorithm is to produce a character by character alignment of the two
texts. This method ought to be effective on our data but its time complexity is cubic in the length of the texts and could therefore be quite costly to do pairwise comparisons between all texts of a large corpus.

Broder (1997) published algorithms for duplicate text detection which do not require iterating over alignments between document positions and are therefore much more efficient. In addition to this, the method lends itself to effective dimension reduction heuristics to further decrease the cost of doing comparisons between documents. The fundamental idea is to break each document into the set of its \( n \)-grams and to compare documents on the basis of their set intersection. Two kinds of similarity are addressed: resemblance and containment. Resemblance, the degree to which two documents contain the same text, is measured as the Jaccard similarity of the \( n \)-gram sets. That is, resemblance is the proportion of \( n \)-grams in the intersection divided by the total number of \( n \)-grams present in either:

\[
R(D_1, D_2) := \frac{|S_n(D_1) \cap S_n(D_2)|}{|S_n(D_1) \cup S_n(D_2)|}
\]

where \( S_n(D) \) is the set of \( n \)-grams (Broder (1997) calls them “shingles”) of document \( D \). Containment is a measure of the extent to which one document \( D_1 \) is contained in the other, \( D_2 \), and as such is the size of the intersection divided by the number of \( n \)-grams in \( D_1 \) alone:

\[
C(D_1, D_2) := \frac{|S_n(D_1) \cap S_n(D_2)|}{|S_n(D_1)|}
\]

This latter measure of similarity is very useful for detecting cases of a submissions that contain different sized portions of the same document.

In addition to defining the comparison efficient measures of resemblance and similarity, Broder (1997) gave methods for reducing the number of \( n \)-grams that need to be compared to get a confident estimate of the true similarity. The first is an optimisation of resemblance only. It requires a deterministic mapping \( g \) of \( n \)-grams into a well-ordered space where the ordering will be statistically independent of the original document order of the \( n \)-grams. Broder (1997) recommends using a text hashing function for \( g \) and we chose to use the md5 hashing algorithm for
our implementation. The set of \( n \)-grams for each document \( D \) is then mapped itemwise under \( g \) into the ordered space (in our case, of md5 hashes) to get the image of \( S_n(D) \) under \( g \):

\[
g(S_n(D)) := \{g(s) \mid s \in S_n(D)\}
\]

When two documents are compared, resemblance is calculated using the standard formula but on only the smallest \( N \) hashes of the union of the two documents’ \( n \)-grams. Because of this, it is sufficient to store only the smallest \( N \) hashes of each document: this reduced set of hashes is called the document’s sketch. Thus the method approximates resemblance by drawing (at most) \( N \) random samples without replacement from the union of two document’s \( n \)-gram sets and counts the proportion of the samples that lie in the intersection. This method is now commonly referred to as a \textit{minhash} algorithm and advances on the concept continue to be developed today (see, for example, Shrivastava and Li (2015)). The simplicity of the \textit{minhash} — particularly its simplicity of implementation — led us to select it for the present study, however in future expansions of this work newer descendents of the algorithm may be called for.

Broder (1997) also gives an approximation method for calculating containment of one document in another. The method is similar to the \textit{minhash} method but instead of taking the smallest \( N \) hashes the size of the sketches is reduced by converting hashes to integers and keeping only items that are equal to 0 modulo \( m \). Containment of the \( n \)-gram sets is then approximated by the containment of the sketches.

For resemblance and containment we place a threshold \( t \) on the value of the similarity measure to designate when the relationship is held to exist. The sketch based approximations of resemblance and containment can both be viewed as variables with a hypergeometric distribution where samples are drawn from the union of the \( n \)-gram sets (in the case of resemblance) or the contained document only (in the case of containment) and a success is a sample drawn from the intersection. For large enough documents it can be approximated by a binomial distribution with probability of success being the true similarity. Broder (1997) uses this fact to derive a formula for
the expected number of $\epsilon$ sized errors when using sketches to estimate resemblance in a document collection. We use it as part of our detection algorithm itself: if our approximated similarity $e$ is less than threshold $t$ we calculate the cumulative probability $p$ of getting an estimate $e' < e$ if the true similarity were in fact exactly equal to $t$. In cases where the approximated similarity does not exceed the threshold but is reasonably likely ($p \geq 0.05$) under a qualifying true resemblance we give it a second chance by calculating the true resemblance or containment on the $n$-gram sets themselves. We also calculate the true similarity directly in cases where document size is too small to use the approximated distribution — but in these cases the computational cost is small.

**Deduplication of submissions**

Since the resemblance detection caps the size of the sketches to a maximal value and containment does not, we chose to do a deduplication based on resemblance first to reduce the size of the corpus before running the containment detection on the deduplicated corpus. We performed our deduplication on paragraphs rather than whole submissions as this was the size of the context used by Wakaran’s automatic WSD. As such, we refer to this unit of text as a **context** in this section. The final result of our deduplication of document submissions is a set of representative original contexts found not to be contained in any other, with user annotations from any near-duplicate or contained contexts having been merged.

In preparation, we split the corpus of document submissions into three parts, **USER-ONLY**, **COS-WSD** and **WITH-WSD**, along the lines of the major releases of automatic WSD for Wakaran as explained in Section 7.3.1. We then split each submission into paragraphs which is the disambiguation context size used for Wakaran’s automatic WSD. Contexts smaller than 11 tokens we filtered out, considering them to be too fragmentary to contain useful annotations. In all, 10160 contexts were filtered out and 8891 contexts of reasonable size were retained. Table 7.1 gives the size and part of speech distribution of the corpus after short contexts and **COS-WSD** submissions have been filtered out.

To first cluster near-duplicates, we compared each context in the corpus to each other con-
text using resemblance with threshold $t = 0.9$ using 5-gram sketches of size $N = 200$. We started each context in its own singleton cluster and for any pair of contexts with resemblance meeting the threshold we merged their clusters. This procedure is similar to single linkage clustering but instead of selecting links in order of similarity to form a hierarchy we just produce a flat clustering by applying a threshold to the similarity. Once we had clusters of near-duplicate documents, we selected the member of each cluster with the most user annotations to be its representative in the subsequent containment detection. In all, 3074 clusters of nearly equal text were formed from 8891 contexts.

To perform substring detection we took the representatives of the near-duplicate clusters and again compare documents pairwise, this time for containment. We used 5-gram sketches reduced by a factor of $m = 2$ and again set the similarity threshold to $t = 0.9$. The modulus factor is quite low, resulting in only a halving of sketch size, but since our filtering allowed contexts as small as 11 tokens any greater reduction would have been too severe. Once all pairwise containment relationships had been established or rejected we selected contexts which had no containers as super-contexts to represent containment clusters. To each containment cluster we assigned all contexts contained within the representative super-context of the cluster. This resulted in some smaller contexts being assigned to more than one cluster. In all, 35 contexts were assigned more than once, affecting a total of 55 clusters. An error analysis later in this section will show a number of steps which could be taken to reduce this redundancy further. However, it cannot be eliminated entirely.

<table>
<thead>
<tr>
<th>Subcorpus</th>
<th>Count</th>
<th>USER-ONLY</th>
<th>WITH-WSD</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>Noun</td>
<td>253819</td>
<td>80422</td>
<td>334241</td>
<td></td>
</tr>
<tr>
<td>Verb</td>
<td>59212</td>
<td>33283</td>
<td>92495</td>
<td></td>
</tr>
<tr>
<td>Adjective</td>
<td>31913</td>
<td>13840</td>
<td>45753</td>
<td></td>
</tr>
<tr>
<td>Adverb</td>
<td>13540</td>
<td>3891</td>
<td>17431</td>
<td></td>
</tr>
<tr>
<td>Conjunction</td>
<td>7626</td>
<td>2158</td>
<td>9784</td>
<td></td>
</tr>
<tr>
<td>Preposition</td>
<td>117316</td>
<td>51518</td>
<td>168834</td>
<td></td>
</tr>
<tr>
<td>Other</td>
<td>87217</td>
<td>47739</td>
<td>134956</td>
<td></td>
</tr>
<tr>
<td>Total</td>
<td>570643</td>
<td>232851</td>
<td>803494</td>
<td></td>
</tr>
</tbody>
</table>

Table 7.1: Part of speech counts of the corpus after filtering out short contexts and the COS-WSD subcorpus.
Table 7.2: Part of speech counts at each stage of the deduplication procedure: before duplicate removal, after duplicate removal and after subdocument elimination.

<table>
<thead>
<tr>
<th>Part of Speech</th>
<th>Filtered</th>
<th>Deduplicated</th>
<th>Subdocuments merged</th>
</tr>
</thead>
<tbody>
<tr>
<td>Noun</td>
<td>334241</td>
<td>91254</td>
<td>73973</td>
</tr>
<tr>
<td>Verb</td>
<td>92495</td>
<td>25631</td>
<td>20766</td>
</tr>
<tr>
<td>Adjective</td>
<td>45753</td>
<td>11805</td>
<td>9407</td>
</tr>
<tr>
<td>Adverb</td>
<td>17431</td>
<td>4479</td>
<td>3673</td>
</tr>
<tr>
<td>Conjunction</td>
<td>9784</td>
<td>2389</td>
<td>1888</td>
</tr>
<tr>
<td>Preposition</td>
<td>168834</td>
<td>48977</td>
<td>40366</td>
</tr>
<tr>
<td>Other</td>
<td>134956</td>
<td>38332</td>
<td>31903</td>
</tr>
<tr>
<td><strong>Total</strong></td>
<td>803494</td>
<td>222867</td>
<td>181976</td>
</tr>
</tbody>
</table>

and we are satisfied that compared to the total number of clusters — 2443 after containment detection — the duplication has been sufficiently reduced for our purposes. Table 7.2 shows the POS counts for the corpus at each stage of the redundancy elimination procedures: before deduplication, after elimination of near-duplicate contexts and after elimination of contained contexts.

Having identified contexts belonging to sets of near-duplicates or a containment hierarchy, we merged all user annotations from each cluster of contexts into a single representative context for each cluster. We merge annotations by aligning the terms they were placed on in accord with the position they are linked to in the original document text. We consider every context in a content duplication cluster to have come from the same source document and merge annotations from terms linked to the same token position in the original document. This presents a challenge, particularly for containment clusters: the starting position of each member of a cluster in the original document may be different. Therefore we first need an alignment for the content of each context in the cluster. To achieve this, we first ordered the terms linked to each context in a consistent way. Specifically, terms of each context were first ordered by the position of the earliest token they were linked to and then by the number of tokens linked to the term. We then aligned the terms of each member context to the terms of the representative context based on equality of the JMDict entry linked to the term and the surface text of its tokens. The algorithm we used for fuzzy text alignment was a recursive longest common substring based sequence alignment similar to that of Ratcliff and Metzener (1988).
described in Section 7.3.2. For our implementation we used the `SequenceMatcher` from the `difflib` module in the Python programming language’s standard library. It has the disadvantage of not precisely implementing the published algorithm but the advantage of being part of a common, freely available open source programming language distribution. Any terms from cluster members that do not align to a term from the representative were not merged into the master copy so as to maintain consistency with the representative copy’s text. Figure 7.1 shows a visualisation of the alignment of one of the larger containment clusters. Note that to simplify the visualisation it has been produced based on an alignment of tokens rather than the potentially overlapping MWE terms.

Since small differences between text exist between cluster members, we use the text of the representative context selected during clustering as a basis for the master copy. For resemblance clusters the representative was the text with the most user annotations. Terms from non-representative member contexts may fail to align and therefore fail to be included so this choice provides a foundation of guaranteed inclusions. In the case of containment clustering the outermost containing submission was chosen for the master text of each cluster. We merged user annotations into the representative of their resemblance clusters first, and then merged the collected annotations from those representatives into the outermost contexts of their containment clusters.
Once term alignments were complete, we merged user sense annotations into the terms of the representative context. For each sense of each term in the representative context we totalled the number of contexts in which that sense had been promoted by a user and used that total as the final score for the sense. However, for our later WSD evaluation we rescaled the final scores (after containment clustering) of each annotated term so that the maximum sense score was 1.0 and the minimum was 0.0. This was done for consistency with the automatic WSD of Wakaran which performs the same scaling. Finally, note that we did not use user sense selections (as opposed to sense promotions) in our evaluation data since users tended not to deselect senses once selected. Overall, the deduplication process did merge some user annotated terms into each other, but many user annotations were unique to the term. Thus, the total number of annotations on terms in the corpus started at 2304 (after filtering out short documents), reduced to 1983 after merging near-duplicates, and only reduced to 1926 after merging short contexts into containing contexts. This reduction is quite small compared to the nearly fourfold decrease in total number of terms (see Table 7.2). Many of the user annotations were on monosemous terms. Out of the 2304 user annotations on the filtered corpus, 440 are on polysemous words. Once all deduplication is complete the count of polysemous annotated terms has reduced to 375.

**Analysis of remaining duplication**

In the previous section we found that a number of short disambiguation contexts from submissions were contained in more than one supercontext selected by containment clustering. In this section we study the situations in which multiple cluster membership occurred and develop methods that could be used to more effectively identify duplicated content in the corpus.

We started with a manual examination of sets of supercontexts with common contained subcontexts to identify the source of the duplication and identified several common causes:

**Overlap**

In many cases the supercontexts both appear to come from a single larger text, unattested by a single submission in the corpus. Specifically a substantial prefix of one is identical — or
nearly identical — to a suffix of the other and the the smaller text that is contained in both appears in the overlapping region.

**Failed similarity identification**

In some cases we found that resemblance or containment detection had failed to identify a relationship between the supercontexts due to a significant amount of noise in the text. In particular, parentheticals appearing in one copy but not the other was a major source of lack of resemblance.

**Multiple source documents**

In one case, shared text appeared to come from multiple distinct source documents. A shared introductory sentence causes there to be some common content between contexts that are not true candidates for merging.

Cases where we have not been submitted a copy of the full document and only have overlapping portions present are not accounted for by our resemblance or containment detection. The main reason for this is that we set a high cut off of 0.9 for both, requiring 90% of the \( n \)-grams of (at least one of) the documents to overlap the other. However, we can still use these measures as a starting point for detecting overlap — two contexts that overlap must have \( n \)-grams in common after all. Containment is best suited to this since the proportion of either document in the overlap must be no smaller than the proportion of their union.

We ran a test to discover the extent of duplication remaining due to overlapping contexts. For any cases where the containment test failed but the estimated containment was greater than or equal to 0.2, we ran the `SequenceMatcher` on the two texts and recorded the number of matched tokens \( o \). The computation cost of performing this alignment is mitigated by the relatively small number of context pairs — 2593 out of 5117876 — having containment greater than 0.2 without qualifying for a containment relationship outright. We then took prefixes and suffixes of the two contexts of length \( o \) and compared the prefix of each to the suffix of the other using a resemblance test. If the prefix of either resembled the suffix of the other to a threshold of 0.9 we designated the
pair as overlapping. We used a smaller threshold than the 0.9 used for our other near-duplication tests because errors in the affixes tend to compound: an inserted token inside the overlap also pushes a potentially matching token out of the end of the $o$-length affix. With overlap defined in this way, 348 deduplicated contexts had a detected overlap with another context, with 10 unique pairwise overlaps detected between different containment clusters.

In our merge procedure for clusters of overlapping contexts terms that do not exist in the representative (super)context were discarded. For the cases of near-duplication and containment this discarding causes minimal losses. For example, during the merging of small contexts into containing contexts, 859 terms with annotations are merged upwards and only 24 terms are lost due to lack of alignment. However, if overlapping text detection were used to form clusters of contexts then no single context can be used to determine the full set of terms to keep. Our context merging procedure in Section 7.3.2 can be extended using recursion to provide an immediate solution to merging a synthetic master document from overlapping documents. As a property of the SequenceMatcher algorithm, terms from the cluster member contexts which are not aligned with the longest cluster member are instead aligned to intervals between two terms of the master or to a pre-initial or post-final interval. A complication is that the method merges more than two pieces of text simultaneously, so gaps between supercontext tokens may be filled by content from more than one subcontext. In the main merging algorithm the inter-token content is excluded from the merging process but for a more general algorithm we can apply the merging procedure recursively to sets of subordinate context terms that are aligned to the same inter-term interval of the longest context. In detail, for each space between terms of the longest context of a cluster:

1. From each other context, collect text aligned to that space

2. If more than one is non-empty, recurse on the non-empty term sequences aligned to the space

Once duplication has been eliminated from the inter-term content, the result can be spliced into the supercontext.

The other source of duplication that we found that might reduce cases of low resemblance
due to high numbers of insertions of parenthetical comments. Performing containment detection after first stripping all parentheticals of 10 or fewer tokens did result in reducing the total number of top-level contexts from 2443 to 2386.

Beyond removing parentheticals and detection of overlaps, further attempts at removing duplication show diminishing returns. For example, we had a single curious case where one copy of a document was stripped of a particular kind of Japanese consonant modifying marks called dakuten and handakuten. Such an error may be due to a flaw in an automatic process such as optical character recognition. Alternatively, in some character encodings the dakuten and handakuten are represented as a combining character so an encoding error might also be at fault. Having only one sample of this means we cannot properly assess effective measures of accounting for it — nor did we have a strong reason to do so for the present study.

### 7.3.3 Distribution of user selections vs. automatic selections

The most important property of the *Wakaran* corpus is its annotation with senses selected by users to appear inline in the text. Table 7.3 shows the distribution of senses selected by users on the *USER-ONLY* subcorpus. That is, selections made while no senses were being highlighted

<table>
<thead>
<tr>
<th>User sense</th>
<th>Automatic sense</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td>None</td>
</tr>
<tr>
<td>None</td>
<td>84382</td>
</tr>
<tr>
<td>1</td>
<td>30</td>
</tr>
<tr>
<td>2</td>
<td>10</td>
</tr>
<tr>
<td>3</td>
<td>1</td>
</tr>
<tr>
<td>4</td>
<td>0</td>
</tr>
<tr>
<td>5</td>
<td>0</td>
</tr>
<tr>
<td>6</td>
<td>0</td>
</tr>
</tbody>
</table>

Table 7.3: Confusion table of senses selected by users while automatic WSD was not operational, and the sense that would have been selected had WSD been enabled at that time.

2 These marks are used to modify consonants. For example, the syllabic | telefono becomes | telef | with the dakuten and | telep | with the handakuten.
by automatic WSD in the *Wakaran* interface. It shows that without any sense hints being displayed users had a strong bias towards the first sense in the list. In particular, users are predominantly selecting the first sense even on tokens for which the WSD algorithm would have computed one of the later senses as being most strongly related to context.

Table 7.4 shows the distribution of senses on the \textit{WITH-WSD} subcorpus, for which automatic WSD was highlighting its selections in the *Wakaran* interface. By comparison to Table 7.3, a great deal of the user preference has shifted towards the sense selected by the automatic WSD algorithm. This is apparent in the larger counts of sense selections appearing in the main diagonal of the table. However, users still exhibit a strong bias towards the first sense, splitting annotations between first and WSD-selected senses rather evenly.

It is noteworthy that on both the \textit{USER-ONLY} and \textit{WITH-WSD} subcorpora, users do not show an inclination to select any sense beyond the first four. This is in part because many words have fewer than five sense entries, but the results in Table 7.3 and Table 7.4 show that there are thousands of tokens for which the automatic WSD would have suggested senses displayed with rank of five or greater for which users made no selection, and many such tokens for which users selected senses one to four. This phenomena serves to highlight the impact of information presentation in glossing systems: information overload leads to users ignoring all but the most prominent gloss information; and ordering and highlighting of senses has a substantial effect on the information considered by the users.
7.4 Evaluation of word sense disambiguation algorithms

After deduplication, the corpus of Wakaran documents serves as a corpus of JMDict sense annotations. In this section we compare the distribution of those annotations to the output of a variety of JMDict WSD implementations.

Under test are four variations of the JMDict WSD algorithm described in Section 7.2. The key feature of this algorithm was its use of a vector representation for the meaning of each JMDict word, embedded in a word-specific subspace of the results of a Personalised PageRank calculation over the concepts in JWordNet. The details of the subspace selection are given in Section 7.2.1 but the salient feature here is that the subspace for a word comprises concepts that are related to any of its JMDict senses. A Personalised PageRank calculation needs to be seeded with weights given to LKB concepts related to the context. For the Wakaran WSD algorithm we used a separate tokenisation of the source text into JWordNet lemmas and used their senses to seed the calculation. However, the direct mapping of JMDict headwords to sets of related JWordNet senses used in our JMDict sense embedding provides an alternative method of seeding the Personalised PageRank and we test that method in this section. In Chapter 6 we saw that the WordNet++ LKB performed much better at Personalised PageRank over JSemcor than the JWordNet LKB, which was used in the deployed Wakaran WSD algorithm. As such, in this section we compare the following four configurations of Personalised PageRank to the Wakaran corpus data:

**wnpp-map**  WordNet++ LKB with Personalised PageRank seeded using JWordNet concepts mapped to JMDict senses.

**wnpp-tok**  WordNet++ LKB with Personalised PageRank seeded using a JWordNet tokenisation of the source text.

**jwn-map**  JWordNet LKB with JMDict mapped senses.

**jwn-tok**  JWordNet LKB with JWordNet tokenised senses.
Chapter 7: Word Sense Disambiguation in Wakaran

Table 7.5: Results for evaluation of all WSD configurations on the section of the Wakaran corpus for which automatic WSD was not enabled. All values are different (or not) from best \( (p < 0.05) \) by test: “m” = Montecarlo test, “t” = Independent t-test, “pt” = Paired t-test or “s” = Sign test as appropriate.

<table>
<thead>
<tr>
<th>annotator</th>
<th>spr</th>
<th>wp</th>
<th>wr</th>
</tr>
</thead>
<tbody>
<tr>
<td>random</td>
<td>0.000*pt</td>
<td>0.424*m</td>
<td>0.421*s</td>
</tr>
<tr>
<td>firstsense</td>
<td>0.678</td>
<td>0.811</td>
<td>0.805</td>
</tr>
<tr>
<td>wnpp-map</td>
<td>0.033*t</td>
<td>0.410*m</td>
<td>0.515*s</td>
</tr>
<tr>
<td>wnpp-tok</td>
<td>0.197*t</td>
<td>0.465*m</td>
<td>0.587*s</td>
</tr>
<tr>
<td>wn-map</td>
<td>-0.244*t</td>
<td>0.277*m</td>
<td>0.348*s</td>
</tr>
<tr>
<td>wn-tok</td>
<td>-0.241*t</td>
<td>0.278*m</td>
<td>0.344*s</td>
</tr>
</tbody>
</table>

All configurations use the cosine similarity to compare JMDict senses to the result of the Personalised PageRank, normalised to assign a numeric weight between 0.0 and 1.0 to each sense JMDict sense. We also use a random baseline and a first sense baseline for comparison. The first sense baseline must inevitably be highly correlated with user preferences given the strong bias towards first sense we saw from users in Section 7.3.3.

Although the users’ selections were single best in nature, the graded output of the WSD might be quite high for a user chosen sense without actually being the single best. Therefore, for comparison of the WSD results to user annotations on the Wakaran corpus we used Spearman rank correlation (spr). In addition to this, we calculated weighted precision and recall scores for the automatic WSD annotations, as though the user selections were a gold standard.

We look first at the results for the USER-ONLY subsection of the corpus. For the documents in this subcorpus, senses were displayed to users with no discriminating display method other than the dictionary ordering of the senses. Table 7.5 gives the results of evaluation of all WSD configurations on the USER-ONLY contexts. The dominance of the first sense baseline in the results show a clear preference for the first sense over the other senses. However, as we saw in Section 7.3.3 when WSD was turned on user preferences shifted dramatically towards the highlighted senses. Thus, these results do not necessarily show a true preference for the first translation but rather a bias towards it. Removing the first sense baseline from the results, Table 7.6a reveals the
Chapter 7: Word Sense Disambiguation in Wakaran

Table 7.6: Results for evaluation of all WSD configurations on the section of the Wakaran corpus for which automatic WSD was not enabled. All values are different (or not) from best \( (p < 0.05) \) by test: “m” = Montecarlo test, “t” = Independent t-test, “pt” = Paired t-test or “s” = Sign test as appropriate.

<table>
<thead>
<tr>
<th>annotator</th>
<th>measure</th>
<th>spr</th>
<th>wp</th>
<th>wr</th>
</tr>
</thead>
<tbody>
<tr>
<td>random</td>
<td></td>
<td>0.000(t)</td>
<td>0.424</td>
<td>0.421(t)</td>
</tr>
<tr>
<td>wnpp-map</td>
<td></td>
<td>0.033(pt)</td>
<td>0.410</td>
<td>0.515</td>
</tr>
<tr>
<td>wnpp-tok</td>
<td></td>
<td>0.197</td>
<td>0.465</td>
<td>0.587</td>
</tr>
<tr>
<td>wn-map</td>
<td></td>
<td>-0.244(t)</td>
<td>0.277(m)</td>
<td>0.348(s)</td>
</tr>
<tr>
<td>wn-tok</td>
<td></td>
<td>-0.241(t)</td>
<td>0.278(m)</td>
<td>0.344(s)</td>
</tr>
</tbody>
</table>

(a) Scores aggregated across all terms.

<table>
<thead>
<tr>
<th>annotator</th>
<th>measure</th>
<th>spr</th>
<th>wp</th>
<th>wr</th>
</tr>
</thead>
<tbody>
<tr>
<td>random</td>
<td></td>
<td>0.000(t)</td>
<td>0.386</td>
<td>0.386</td>
</tr>
<tr>
<td>wnpp-map</td>
<td></td>
<td>0.076</td>
<td>0.383</td>
<td>0.522</td>
</tr>
<tr>
<td>wnpp-tok</td>
<td></td>
<td>0.211</td>
<td>0.416</td>
<td>0.570</td>
</tr>
<tr>
<td>wn-map</td>
<td></td>
<td>-0.155(t)</td>
<td>0.276</td>
<td>0.388</td>
</tr>
<tr>
<td>wn-tok</td>
<td></td>
<td>-0.111(t)</td>
<td>0.291</td>
<td>0.399</td>
</tr>
</tbody>
</table>

(b) Scores aggregated across only terms for which a user selected a non-first sense.

The best WordNet++ annotator has a better Spearman rank correlation with the user data than any other Personalised PageRank annotator or the random baseline (with statistical significance). Note in particular that using a JWordNet tokeniser on the source text to build the semantic context (wnpp-tok) rather than using the full set of JWordNet senses linked to JMDict headwords (wnpp-map) results in better score on this measure. WordNet++ with the JWordNet tokenisation annotator also has better weighted recall than the other annotators though the difference between the two WordNet++ configurations is not statistically significant in this case. Finally, we note that the the annotators’ weighted precision values are generally lower than their weighted recall scores. This is a natural result of total weight assigned by the Personalised PageRank annotators being greater than one (since the grades are normalised to span the interval \([0, 1]\)). Since the bronze data primarily comprises single best sense selections it assigns a weight of 1.0 per term which acts as a cap on the denominator in the weighted recall formula but does not affect weighted precision. The random baseline also assigns grades with total weight 1.0 and has a precision not statistically worse than the best WordNet++ annotator.

Given the suspected bias in user annotations towards the first presented sense, good results may simply indicate the inclination for a WSD algorithm to choose the first sense. However, if we
Table 7.7: Results for evaluation of all WSD configurations on the section of the Wakaran corpus for which automatic WSD was enabled. All values are different (or not) from best \((p < 0.05)\) by test: “m” = Montecarlo test, “t” = Independent t-test, “pt” = Paired t-test or “s” = Sign test as appropriate.

restrict the analysis to cases where users chose a non-first sense we achieve two things:

1. more confidence that the user’s choice was made based on gloss content; and
2. a comparison between automatic WSD algorithms that does not benefit a first sense bias.

Table 7.6b shows results for all classifiers on terms in USER-ONLY where users selected a non-first sense. Numerically the results turn out to be very similar to the results on the corpus as a whole though the dataset is too small for many of the differences to have statistical significance. However, the WordNet++ based Personalised PageRank annotators still exceed the other annotators on Spearman rank correlation with statistical significance. As such, we conclude that these results show WordNet++ improves over JWordNet for Personalised PageRank based WSD of the JMDict sense inventory.

Table 7.7 shows results of evaluating our selection of JMDict WSD algorithms on the WITH-WSD submissions for which automatic WSD was actively preselecting senses for users. As with Table 7.6 two tabulations are given: Table 7.7a shows evaluation over the whole subcorpus, while Table 7.7b shows evaluation on only the tokens that exclude sources of bias. The results on the whole corpus are very similar for all annotators. Understandably the configuration that was used...
in *Wakaran* for sense highlighting, *wn-tok*, has the highest correlation with the user selections. However, it is interesting to note that all of the other annotators are competitive, even exceeding *wn-tok* on some aspects of the evaluation. Table 7.7b shows the results on tokens where the users have chosen a sense that is neither the first sense, nor a sense that was highlighted by the automatic WSD algorithm. In such cases the user has avoided both of the most obvious choices to select their own preference. Although the number of tokens is too few to get statistically significant differences out of the subtly different configurations *wnpp-map* and *wnpp-tok*, we do note that there is a large numerical difference between them. Specifically the configuration that seeds the context with senses of *JWordNet* words identified by a separate tokenisation pass appears to perform better than the configuration that seeds the *Personalised PageRank* with senses of English words in the glosses. This inequality holds for all metrics in all of the corpus subsets presented in this section with the exception of the full *WITH-WSD* evaluation. It seems likely — but not conclusive — that the *JWordNet* tokenisation is the more effective context model. Finally we note that *wn-tok* is severely disadvantaged on this subset of tokens because we have specifically eliminated tokens for which it graded the user-selected sense with the top score of 1.0. However, it still manages to achieve a non-zero score because it is a graded annotator: the presentation bias filter allows tokens on which *wn-tok* assigned the bronze sense any score strictly less than 1.0.

## 7.5 Discussion

The results of deploying automatic WSD in *Wakaran* show that it had a strong influence on user attention, which suggests that performance of the WSD algorithm is very important to user experience. Since users otherwise tend to gravitate strongly to the first sense, the deployment of automatic WSD has the potential to encourage users to look at a more appropriate translation further down the listing. Additionally, the split in user selections between WSD selected senses and first senses in the *WITH-WSD* dataset could suggest two things: that users are taking more care choosing between the two, or that users’ attention bias is split between the two. These hypotheses should be
tested in future research. If users can be encouraged to more often consider the appropriateness of at least two of the glosses they may gain some of the benefit to vocabulary acquisition of practising meaning inference as described by Fraser (1999). This too is a question for future research.

One purpose of testing WordNet++ and JWordNet LKBs for Personalised PageRank as we have done in this chapter is to see whether the crowdsourced data draws us to the same conclusions as the gold data used in Chapter 6. The strength of the influence automatic sense highlighting had on user sense selections means that JWordNet can only be fairly compared to WordNet++ on the USER-ONLY subcorpus. On this subcorpus the crowdsourced data clearly reproduces the result of Chapter 6 that WordNet++ works better as a resource for Personalised PageRank. This is true on the USER-ONLY corpus as a whole but in particular the WordNet++ annotators has a significantly better Spearman rank correlation score over tokens where the user annotations select non-first senses.

In these experiments we lacked a good measurement of the extent of the user bias due to sense ordering. This bias could be measured by randomising the order of gloss display. Caution surrounding certain elements of gloss content that carried forward from sense to sense prevented us from applying this technique in this round of data collection. However, a manual vetting of a subset of the dictionary could be used to establish the impact that a random re-ordering would have on the coherence of gloss text. Thus, future work may be able safely deploy a randomisation technique and other algorithmic sense re-ordering techniques.

### 7.5.1 Quality control implications

So far in this chapter we have looked at the degree to which the senses users choose to consult while reading Japanese as a second language agree with WSD algorithms and sources of bias such as sense order and highlighting. In this section we take what we have learned and make an assessment of what quality control measures might be used to derive gold standard sense labels from future work using the Wakaran system, in combination with providing more helpful automatic disambiguation within the system. We discuss the implications of the output-based quality control decision matrix of Kern (2014) for Wakaran, and make additional comments on other families of
Kern (2014) lays out a decision procedure for worker output validation in crowdsourcing based on four properties of the crowdsourced task: determinancy, validation effort, granularity and difficulty. Determinancy, referring to whether the task is deterministic or non-deterministic, is decided by asking whether repeated valid performances of a task will always yield the same result. For example, asking a worker to paraphrase a paragraph of text would be a non-deterministic task with many valid responses. For the purpose of gold standard label selection, if we define a correct result as the sense agreed upon by a committee of professional annotators, there will usually be a single deterministic best sense. However, it has been well noted that sometimes multiple senses are applicable for a variety of reasons (Jurgens et al. 2014) so sense label selection does not sit perfectly at one end of the determinancy scale. Validation effort refers to the difficulty of checking worker output compared to performing the original task. In the case of sense labels this effort is moderate to high: it is moderately easy to judge the relevance of a single selected sense to context, but to do so may ignore the presence of another more relevant sense. Fully accounting for all sense possibilities when evaluating a selection is very close to performing a sense selection task anew. Granularity refers to whether a task is decomposable into subtasks. An all-words annotation task for a piece of text is a granular composition of token annotations tasks, whereas a lexical selection task is more atomic, especially when labeling a single token or lexeme in a single text. In the case of Wakaran the task is more loosely defined: users will not, on the whole, label all words. However, the default mode of operation does not highlight a specific lexical selection for the users to annotate so it is best thought of as a granular task that no individual user will fully complete. Finally there is the difficulty of the task, which comes down to whether a worker who has passed any selection procedures can be reasonably expected to produce correct output in most cases. Based on the biases in sense labelling identified in this chapter the sense labelling task should be treated as difficult for the workers involved.

One method we can rule out right away is the control group or validation pattern. In this pattern one worker performs a task and a committee of workers validates their output. It is
good for non-deterministic tasks with a low validation effort. Since sense labelling is effectively deterministic and has a relatively high validation effort this quality control method distributes the costs of quality control in all the wrong places.

Two quality control methods which are suitable for deterministic tasks and are applicable to word sense labelling are the gold pattern and the voting pattern. In the ground truth or gold pattern, a subset of tasks that the user performs have a known-correct answer. In the case of sense labelling this may come by professionally annotating a subset of the documents that are to receive crowdsourced labels. The performance of workers on the gold subset of their tasks then serves as a measure of the quality of their novel labels. Gold standard tasks can also be used as feedback to train users in the expectations of the work (Kern 2014). In Wakaran, user submitted documents present something of a challenge to the gold pattern. One way it could be managed is with a post-hoc professional annotation of selected documents submitted to the system. Additionally, special modules may be introduced for which gold standard data could be prepared. For instance, offering practice exercises for self-test or targeted at popular standardised examinations such as the Japanese Language Proficiency Test (JLPT) would involve supplying texts to the users which could have some senses pre-annotated by professionals. In contrast to the gold pattern, the voting pattern uses crowdsourced workers alone to validate the output produced. A number of workers are given the same task and expected to produce the same output. Typically majority voting is used to decide the correct output, where the most popular answer is considered correct. This method validates task output, but like the gold pattern it can also be used to evaluate the quality of workers. In fact, Passonneau and Carpenter (2014) propose a more robust redundancy method than majority vote by building a probability model of the annotations produced by each worker that account for reliability and bias. It still relies on having each token annotated by a number of users which does not occur very frequently with ad-hoc usage of Wakaran. Multiple annotation could potentially be achieved by setting special tasks as discussed above for the gold pattern. Another interesting avenue for investigation would be to use knowledge-based disambiguation algorithms as pseudo workers.

http://www.jlpt.jp/e/
to increase the number of sense selections on each token. Additionally, annotations produced by professionals contributing to any gold annotated subsets would be easy to include in such a model. The voting pattern incurs a high cost in the sense that it requires more workers and work per task completed. However, *Wakaran* has been designed to be open to the widest possible audience of language learners and Chapter 4 showed that it attracts dedicated long term users, so workforce size is one of our method’s strengths. It is also the case that the probability modelling method of *Passonneau and Carpenter (2014)* can achieve more reliable results with lower redundancy as long as it has a large number of tokens labelled by each user. If their method is to be applied to *Wakaran* it will help to have an estimation of selection bias towards the first sense. This could be accomplished by displaying glosses in a randomised or semi-randomised order.

The gold pattern and voting pattern are well attested in crowdsourced sense disambiguation and similarity literature. *Jurgens et al. (2014)* use the gold pattern to assess the quality of workers in their gamified sense labelling projects *Puzzle Racer* and *Ka-boom!*. In *Puzzle Racer* users who select the correct label for the gold instances are shown proportionally more genuine task data as they play. In *Ka-boom!* users lose the game if they fail to destroy too many of the known-incorrect instances, which is additionally an example of how the gold pattern can be used for feedback and training. *Venhuizen et al. (2013)* use the voting pattern in a word sense labelling game called *Wordrobe*. They use gold data but rather than use it to assess individual workers they use it to measure the reliability of the voting scheme itself under different agreement cutoff parameters.

In the decision matrix put forward by *Kern (2014)* difficult tasks have a uniform preference for two methods above all others: the comparison pattern and the iteration pattern. For coarse grained (or non-decomposable) tasks, such as sense labelling for a selected token, the comparison pattern is recommended. It works by having a number of workers produce independent answers for the same task, and then having a committee of different workers vote on which is the best. For sense labelling this could mean taking senses selected by a number of *Wakaran* users for a token and displaying a reduced selection to other users based on their output. This method works on non-deterministic difficult tasks so it should be robust to the stronger cases of sense ambiguity.
However, the engineering cost of sharing information between user sessions in real time would be much higher than many of the methods recommended so far. The other pattern recommended by Kern (2014) that fits the use case of Wakaran is the iteration pattern, which would also necessitate sharing of labels between user sessions. The iteration pattern is for fine-grained decomposable tasks and involves passing a single task to multiple workers in sequence, giving each worker the output of the previous. This allows gradually building quality into the output. It is good for fine grained tasks in particular because subsequent workers can choose a small part of the task to refine the answers of other workers. In the case of Wakaran we saw in Section 7.3.2 that users largely selected glosses for non-overlapping sets of tokens. Thus the iteration pattern would mean passing on a partially annotated document from each user to the next to get a more complete annotation, but potentially gaining refinement of the previous users’ annotations as well. This method may be problematic for Wakaran’s natural classroom use case because multiple user submission of the same text is unlikely to extend over a long enough timespan to build a long chain of workers on the one task. It is less of a problem for the comparison pattern because that only requires two steps and so would be feasible within the timeframe of a single class of students reading the same document. However, as with many of the methods discussed here, special recruitment of users to perform practice tests on common documents would enable the iteration pattern to be implemented. Kittur (2010) found that seeding an iterative crowdsourcing task with a small amount of initial output greatly increased engagement of their workers. This is achieved in Wakaran with the use of automatic WSD based sense selection, even in cases where a long chain of iteration is not performed.

Overall the best output based quality management methods for Wakaran appear to be gold standard, redundancy with a probability model and the comparison method. The gold standard method in particular would require converging students on to pre-prepared texts but practice tasks for popular examinations like the JLPT would be a strong draw card for students. Annotations supplied by professional annotators for the gold method could also be folded into the estimation process for the annotator model of Passonneau and Carpenter (2014) to improve the overall quality of the model. Although the comparison method would require more engineering up front it has
the advantage of being applicable to the general use case of Wakaran for ad-hoc classroom work. Annotations selected by students could be shared amongst the class, particularly where different students made different selections, and these disagreements could be raised as points of discussion.

So far we have covered output-based methods for quality control but other methods exist as well that are applicable to getting high standard sense labels from Wakaran usage. One major aspect of crowdsourcing quality is the expertise of the workers themselves and qualification tests are one way to control for it. In the context of output based quality control we discussed reasons for administering practice tests for the purposes of converging student activity on common documents but another use for it can be for assessing the learning level of students themselves. Additional ways the qualification level of users could be tested are voluntary surveys on Japanese learning and usage experience. A sense of contribution to research may be sufficient to motivate a moderate degree of participation in such surveys. Additionally, if users could be prompted to create an account with Wakaran to gain access to the practice tests then their proficiency with certain words and overall proficiency with Japanese could be tracked over time. The administration of practice tests may also increase the motivation of students to take care in selecting their interpretations of words.

The other main area of quality control with applicability to Wakaran is execution process monitoring. In Chapter 4 we described collecting detailed interaction timeseries data from usage sessions of Wakaran and performing unsupervised clustering on a variety of fingerprint models to identify major usage patterns. This data could be used to perform an analysis like that of Rzeszotarski and Kittur (2011) where supervised machine learning is used to predict worker output quality on the basis of their activity fingerprints.

Finally, there are a number of ways in which we can leverage the open-to-all crowd selection strategy (Allahbakhsh et al. 2013). One finding of Chapter 4 was that users stated a preference for browser based extensions over Wakaran for general web browsing. Implementation of such an extension could greatly increase access to Wakaran’s userbase. We also found during our experiments that teaching staff can be quite interested in trialling new technologies in the classroom. Making contacts with Japanese teaching staff at universities around the world could be an effective
way of targeting the particular classroom environment that we are interested in. Finally, the implementation of longitudinal tracking of users via user accounts would enable smaller scale reputation or credential based crowdsourcing via the Wakaran environment.

7.6 Conclusion

The big question we sought to answer in this chapter was whether the sense preferences of Wakaran users can be used to inform the development of better models of word senses. The answer comes in two parts: firstly that evaluation using the user preferences reproduced the conclusions about resources for Personalised PageRank as the gold standard data used in Chapter 6; secondly that we identified the main sources of bias in user selections and propose methods for quantifying and dealing with that bias.

Whether we choose to automatically highlight a word sense or not, the presentation of senses in a particular order is by itself a strong influence on the focus of users attention. It is likely that sense entries further down the sense ranking are rarely being consulted, in spite of their relevance to the text and the lexicographer’s intentions in including them in the gloss. If we can find ways to increase engagement with more salient senses regardless of their positioning in the dictionary sense ordering, we can get better help about word meanings into the hands of students at the times that they have requested it. More importantly, if we can encourage students to make an informed choice between two or more distinct options this may have some of the vocabulary acquisition benefits of a meaning-inference strategy to unknown words. This goal is well aligned with the crowdsourcing goals of Wakaran since a careful consideration of two possibilities would give a high quality data point: in the data collected so far it was only possibly to rely on aggregate statistics to correlate automatic WSD with user preferences, but with careful interface design a silver standard could be built at the same time as improving learning outcomes for users. We now know that sense highlighting and sense ordering both have a strong effect on user attention. Sense ordering in particular draws focus to the top three to four senses; presenting a semantically diverse
subset of senses in those positions may maximise incentive to choose whilst also providing good
course grained WSD judgements. In Chapter 8 we outline a research plan for encouraging students
to study and choose senses more closely. In this way we have shown that we can advance computer
assisted language learning and computational semantics research goals in tandem, using *Wakaran*
as a crowdsourcing and second language learning tool.
Chapter 8

Conclusion

In this final chapter, we return to the original research questions and state our findings. We then give a summary of the contributions of each chapter to thesis and to the field. Finally, building off the findings of the thesis, we develop a plan for several directions of future work.

In this thesis we sought to discover whether human judgements of word meanings can be collected from naturally occurring interactions of students with dictionaries in a second language (L2) learning context. In particular, we were interested in finding out:

- whether students would use word sense visibility features in an interactive Japanese-English glosser to satisfy their information needs; and
- whether it is possible to draw sensible conclusions about lexical semantic models based on a record of sense choices made by students in that setting.

To that end we developed a web-enabled glossing application, the Wakaran Glosser, and advertised it to several intermediate Japanese L2 classes. It was introduced and its design was discussed in detail in Chapter 4. The experimental work of the thesis comprises two main parts:

- Evaluation of NLP models supporting the design of Wakaran and integration of WSD features.
Analysis of records of *Wakaran* usage to determine patterns of interaction confirming feature usage and patterns of sense selection for comparison to models.

In Chapter 4 we demonstrated that students of Japanese as a second language will use interactive features of the *Wakaran Glosser* for paraphrase visibility in class reading tasks (and for class preparation). Offline evaluation of NLP models was performed in Chapter 6 and Chapter 5. Finally, in Chapter 7 we showed that relative WSD evaluation on the *JSemcor* corpus does transfer to relative correlation with user word sense preferences. However, we also showed that user sense selections are highly influenced by the order in which senses are presented and any prominence given to senses by automatic WSD. We conclude that interactive glosser features can be a rich source of cheap information about user sense judgements but that care must be taken to investigate the language learning outcomes for users.

### 8.1 Contributions

In this section, we go into more detail of the contribution of each chapter to the thesis. In the next section we use the overall findings discussed here to lay out a plan for future work using *Wakaran* for crowdsourcing lexical semantic judgements.

In Chapter 4 we outlined how the *Wakaran Glosser* was designed to offer interactive word sense visibility features for second language students which have, as a side effect of their use, the result of collecting human judgements of word senses. A major design goal was to make the software an attractive choice for assistance in classroom reading tasks for students of Japanese as a second language. We established that the interactive feature design in *Wakaran* is compatible with the most effective vocabulary acquisition strategies for second language students encountering unknown words, but did not verify learning outcomes for students using *Wakaran* itself. Using a variety of aggregation and visualisation methods on the application usage logs we showed that around half of all usage sessions of *Wakaran* were for reasonably intensive reading sessions. Around 20% of sessions lasted for longer than 20 minutes and examining those revealed a variety of interesting
reading strategies, including many users who interacted with glosses in a single monotonic pass, some who read in two passes, some who read in a haphazard order and a handful who seemed to read their document forwards once and then backwards once. Based on these logs, and the bulk of usage occurring during university semester weeks on specific times during the week, we concluded that students were in fact using Wakaran for its intended purpose of in-class reading tasks and class preparation.

In Chapter 5, we investigated information sources for classifying candidate MWE tokens as idiomatic or literal text. This was motivated by the utility a computationally cheap feature base would have for construction of glossing software. The focus of the investigation was on whether the morpho-syntactic fixedness properties of idioms that are useful in identifying their types are a viable means of identifying specific usages as well. We found that SVM models built on features identifying variations of idioms from their canonical forms were barely more effective than a supervised first sense baseline, whereas models of the context semantics were substantially more effective, especially at classifying the literal instances correctly. Indeed, even on MWE types that did not appear in the training set, models of the context semantics had similar performance to the type-specialised first sense baseline. We examined candidate MWE tokens in the corpus for conformance with syntactic constraints on idiomatic usages as encoded by a lexicographer in the JDMWE, finding that true idiomatic usages violate their constraints about half as often as literal usages make use of their implicit syntactic freedom to do so. We conclude that a model of an idiom’s lexico-syntactic flexibility is unlikely to ever provide a high level of accuracy for distinguishing between literal and idiomatic instances of MWE candidate tokens. The results of evaluation on unseen MWE types suggest that identification of textual contexts that prefer idiomatic (or literal) language may be a novel alternative line of approach. However, overall it seems that MWE candidate disambiguation will not be solved to a level of accuracy that engenders trust from users of a glossing application without a solution that accounts for semantics, either via distributional models of the constituent words or through use of information from a lexical knowledge base. Either way a great challenge will lie in simultaneously satisfying the accuracy and latency requirements of an interactive glossing
application.

In Chapter 6 we investigated resources for developing and evaluating knowledge-based models of word sense disambiguation in Japanese. These are of interest in two respects:

1. As a way of implementing high coverage automatic WSD in a glossing application, knowledge-based methods are not blocked by the same knowledge acquisition bottleneck as supervised methods.

2. Traditional evaluation methods for automatic WSD provide a reference point for interpretation of users’ sense judgements crowdsourced from the glossing application.

We transferred senses from the English *SemCor* onto an existing Japanese translation using logs of lemma to lemma translations selected by the human translators as part of the translation process. We then used the resulting corpus, *JSemcor*, as a gold standard for evaluation of different LKBs in the task of Japanese WSD. We found that *WordNet++*, which is an extension of *WordNet* with additional relations automatically extracted from *Wikipedia*, performed better than *JWordNet* as an lexical knowledge base for *Personalised PageRank* on *JSemcor*. However, the further extension to *BabelNet* containing concepts and relations from multiple languages of *Wikipedia* did not result in a further improvement. In fact, it resulted in a slight decrease in performance, though the specific tokens, lemmas and documents on which performance decreased were found to be highly sensitive to other configuration parameters. We conclude that the expansion of *WordNet++* to *BabelNet* introduced a saturation of the network that *Personalised PageRank* is unable to account for, and hypothesise that an investigation of assigning weights to relations may be required to leverage the information of multilingual *Wikipedia* in this task.

Finally, in Chapter 7 we investigated interactions between *Wakaran* users and word senses, including some interaction with automatic WSD. We outlined a method for adapting the knowledge-based WSD models used in Chapter 6 to the glossing dictionary, *JMDict*, by embedding each sense in a word specific subspace of the *Personalised PageRank* output. The simplest version, using *JWordNet* as the lexical knowledge base, was running in *Wakaran* for a few weeks highlighting
word senses for users where possible. We found that user sense selections were highly biased by presentation:

1. to the first sense when WSD was turned off

2. split between the first sense and the highlighted sense while it was turned on.

However, when the automatic WSD was not highlighting senses, user sense preferences did correlate better with selections made by WSD algorithms that performed better in the offline evaluation against *JSemcor* in Chapter 6. We conclude that statistical trends in user sense preferences can be captured using *Wakaran* and that they can reproduce the results of evaluations using professionally produced corpora. Additionally, it is clear that the use of automatic WSD sense highlighting has a substantial impact on which senses users focus on and therefore quality WSD may have a substantial impact on the outcomes for a user of *Wakaran*. Properly interpreted it should be possible to use records of user interactions with automatic WSD output to evaluate the quality of WSD models and, in turn, to improve them in ways that increases their utility in the glossing application.

### 8.2 Future work

Our agenda for future work spans four major axes:

1. Establishing the effect of word sense presentation in *Wakaran* on student’s learning outcomes.

2. Identifying improvements to the current data collection mechanisms in *Wakaran* and new ways to analyse the data being collected.

3. Reconciling MWE identification and entity linking with WSD.

4. Investigating latency of WSD as an orthogonal metric for evaluation to label accuracy metrics.

Each of these directions for future work builds on the possibilities opened up by *Wakaran*, and is reliant on its ongoing usage as a glossing application.
Given the clear impact sense order and automatic sense highlighting had on user selections, studying the effect changing sense presentation has on learning experience will be a critical component of future work involving *Wakaran*. There are two main directions in which this can proceed: direct observation of users in a controlled environment and longitudinal tracking of remote users. In both cases assessment performance of reading comprehension and vocabulary acquisition will be the primary effects to be measured. Direct observation of think-aloud exercises followed by exit interviews additionally provides a rich source of information for the interpretation of user interface events allowing us, for instance, to interrogate the reasons for observed behaviours such as choices regarding use — or disuse — of interactive options include gloss showing, sense selection and deselection, and inline insertion of paraphrases. Longitudinal study of learning outcomes is possible in both a direct observation setting and in a setting where users are registered with *Wakaran* but are otherwise monitored using only the glosser’s standard logging apparatus. In the case of a remote study it will be feasible to use a much larger pool of participants. Volunteers drawn from either a pool of established users or recruited from Japanese Language courses with the assistance of teaching staff could be given a small participation incentive in exchange for a greater commitment than ad-hoc users. For instance, this might include:

- provision of simple demographic information,
- information on their extent of previous and ongoing exposure to Japanese,
- participation in a fixed number of language tests at regular time intervals,

The demographic and language exposure data provides a degree of control in interpretation of test results under different sense presentation schemes and changes in results over time. Finally, longitudinal data on learning outcomes could be collected in a less controlled manner if *Wakaran* were to provide its own assessment material, such as practice exams for standardised tests such as the Japanese Language Proficiency Test (JLPT). In such cases, participation timelines would not be controlled but a small amount of demographic information may still be collectable and participants
would be self motivated to return for repeat participation. At a minimum, any sense preferences expressed by a user while administering a self-test for learning progress are likely to be more carefully made than users in an arbitrary usage session under less intense performance pressure.

With monitoring in place to ensure learning outcomes are tracked, experiments with sense presentation in controlled and ad-hoc usage settings can be implemented, starting with refinements to the existing data collection and analysis. Further changes to sense presentation should be investigated with a split testing method — assigning different users concurrently to different presentation schemes — stratifying the split where possible to distribute experimental variables evenly across different users submitting the same document and across users from different demographics. Some adjustments may be made to the way automatic WSD is presented interactively. Specifically, users may be given an option to provide feedback on how well the automatic WSD performed in a way that does not produce a potentially unwanted change in the document such as an inline insertion. Additionally, satisfaction with automatic WSD could be measured implicitly by making it an optional feature with gloss level opt-in, however the effects of this change on participation would need to be measured and considered. The results in Chapter 7 showed a strong preference for the first sense, so the effect of sense ordering is a high priority for further investigation. Randomisation of sense order would be a good way to ascertain whether the first sense preference is semantic, a result of the presentation, or a mixture of both. Additionally, the effect of re-ordering senses using automatic WSD could be explored, which would extend research into annotation schemes other than single best sense. The introduction of learning assessment tasks to Wakaran, as discussed in the previous paragraph, may result in students applying themselves more to the task of sense selection and improve the quality of sense annotations produced. Direct observation studies will give a detailed insight into student choices in a controlled environment but the Wakaran usage logs may also provide a window into how much consideration students put into sense selections when they are not subject to the scrutiny of a laboratory environment. A simple measure of consideration which could be applied to the data already collected is the time spent viewing each gloss pop-up before making a selection. This method could be elaborated to study reading patterns on shorter timescales to analyse
patterns in behaviours such as time per paragraph, backtracking and forward tracking which can be used to measure engagement with text (Vo et al. 2010). Even with motivation structures to encourage thoughtful sense selections and metrics to detect deliberation, the quality of sense judgements provided by a single Wakaran user could not be considered the same as a professional annotator. This being the case, we would like to explore the relationship between professional annotators and users of Wakaran with a more advanced analysis than standard measures of agreement. In particular, we would like to extend the work of Passonneau and Carpenter (2014) who model different human word sense annotators as random variables with individualised distribution parameters. This method could be applied to comparing Wakaran users, professional annotators, and automatic WSD output. It even has the potential to provide confident labels for single tokens if sufficiently many annotations are collected.

A major contribution that Wakaran can make to computational semantics is to provide a setting for studying the unification of WSD, entity linking and MWE token identification. A notable development in this direction is the Babelfy algorithm which uses Personalised PageRank sourced at single concepts to build an unweighted graph of closely related concepts and then disambiguates words, multiwords and named entities in context using a heuristic densest subgraph algorithm to find cohesive interpretations (Moro et al. 2014). This method is a big success in unified modelling of lexical semantics but it does regress to the paradigm of treating word senses as a categorical variable rather than concepts with a non-exclusive relationship with word usages. We would like to use Wakaran as an environment and data source for evaluation of graded models of lexical semantics that are inspired by the Babelfy algorithm in building cohesive subgraphs of the Personalised PageRank output. For instance, we would like to investigate the use of single source Personalised PageRank vectors to solve for a weighted Personalised PageRank where the contribution of each sense to the context most closely matches its activation in the context. To do this we would constrain the linear sum of each token’s sense contributions to one. This constraint dovetails nicely with the convex addition properties of Personalised PageRank vectors. Specifically, the normalised linear sum of Personalised PageRank vectors for single concepts is the same as the Personalised PageRank vector.
for the union of the concepts. Setting an objective function to minimise the difference between
the contribution of each sense to the context and the activation of the sense in context will then
ensure cohesiveness by giving weight to the senses the most support each other. Thus, we can
solve a convex constrained optimisation problem to get a graded solution to WSD, entity linking
and MWE token identification. The results in Chapter 6 suggested another avenue for developing
knowledge-based WSD: exploration of the weights assigned to lexical knowledge base relations
to account for differences in strength or quality of the relationship. Weighting schemes could be
evaluated using Personalised PageRank and compared to results of Babelfy, particularly any graded
adaptation thereof. However, convergence properties of Personalised PageRank under a weighting
scheme would require investigation since the damping parameters of PageRank were originally
tuned by Brin and Page (1998) for a uniform exit distribution for each node. Finally, the method we
used in Chapter 7 for embedding JMDict senses in the Personalised PageRank output space requires
more investigation and evaluation. Specifically, we would like to investigate different choices for the
embedding space including a universal deep learning based word and sense embedding (Johansson
and Pina 2015).

With the elaboration of WSD schemes implemented within the context of Wakaran one
factor will increasingly affect the user experience that has not received extensive study: latency of
the WSD algorithm. It has long been known that page load times are a significant factor in web
page usability, with significant drop offs in user attention around the one second and ten second
marks (Nah 2004). As such, long running WSD algorithms may be a liability to usability in spite of
high accuracy. Although there are absolute time thresholds for usability, with computation power
continuously increasing they are a moving target, so our interest is in the progressive trade-off
between WSD accuracy and computation time. In Chapter 4 we outlined an computation time
optimisation for identification of JMDict entries in text by collapsing our POS tagging and MWE
compounding preprocessors into a single conditional random field model, but we did not investigate
any time-accuracy trade-offs. Similarly, Moro et al. (2014) introduce a novel heuristic for the NP-
hard densest subgraph problem in Babelfy but do not give an analysis of the time-accuracy trade-off
of the heuristic, touching only briefly on the accuracy benefit of including the densest subgraph algorithm in the first place. A number of opportunities exist to move the position of existing WSD algorithms on the time-accuracy graph, and perhaps even move it closer to the ideal, which could be evaluated offline using JSemcor and Wakaran evaluation data or online measuring return visitor rates for ad-hoc visitors. Firstly, due to the linear composability property of Personalised PageRank it is possible to precompile single source results and construct contextual Personalised PageRank results by convex composition at runtime (Jeh and Widom 2003). However, the storage requirements for precomputed Personalised PageRank scale quadratically with the number of concepts (Fogaras and Rácz 2004), so compression and approximation methods would be required to make this approach practical. [Tong et al. (2008)] give a good example of one such method that would be worth exploring in the context of word sense disambiguation. In particular, it is notable that with precomputed single source Personalised PageRank vectors the Personalised PageRank-w2w method of Agirre and Soroa (2009) could be performed with the same computational complexity as standard Personalised PageRank (albeit with a constant factor cost). We would like to investigate these and other trade-offs between Personalised PageRank accuracy and speed in both offline experiments and with reference to user satisfaction in Wakaran.

Further work with Wakaran requires acquiring and sustaining a base of both ad-hoc and longitudinally tracked users. One potential source of ad-hoc usage was suggested by user feedback regarding Wakaran discussed in Chapter 4: the implementation of a browser extension would make Wakaran more appealing for use in general web browsing. An added advantage of this approach would be the ability to identify URLs of publicly available content for which sense preference data had been obtained, which could lead to publishing an open corpus. However, in the work studied so far we have found that classroom reading activities are the primary use case for Wakaran, and we have identified avenues for future work that involve usage of Wakaran in language learning assessment conditions. It is clear, then, that outreach to teachers of Japanese as a second language will be the most effective way to ensure that Wakaran’s crowdsourcing potential — and its potential to have an impact on language learning as a glossing application — is realised. Our experience so
far has been that in the field of tertiary education there is a strong interest in finding ways to use technology in the classroom, which augurs well for *Wakaran*’s future.

### 8.3 Final remarks

In this thesis we have developed a novel crowdsourcing method for gathering human judgements of word sense relevance to context. We designed a platform, the *Wakaran Glosser*, in which students of Japanese as a second language may interact with the senses of words and multi-words to help them mitigate the information overload that occurs due to presentation of irrelevant glosses. In so doing, they implicitly provide feedback as to their preferred sense of the word in a given context. Using logs gathered from instrumentation of the *Wakaran*, we verified that students engage with the platform and its interactive features in a variety of common patterns. From the same logs we compiled a corpus of user sense preferences and verified that it can be used to reproduce certain results of offline evaluation of WSD algorithms on professionally annotated corpora. In the process of analysing sense selections made by users of *Wakaran*, we discovered the greatest sources of bias in user sense selections and have outlined a robust plan for future work to control for those biases and expand knowledge in several areas of both computational semantics and computer assisted language learning. Overall, we have showed that crowdsourcing of annotations, extrinsic evaluation of computational semantic models and delivery of those models as an application feature can be mutually sustaining objectives in our future research.
Bibliography


Bosma, Wauter, Piek Vossen, Aitor Soroa, German Rigau, Maurizio Tesconi, Andrea Marchetti, Monica Monachini, and Carlo Aliprandi. 2009. KAF: a generic semantic annotation format. In Proceedings of the 5th International Conference on Generative Approaches to the Lexicon, Pisa, Italy.


Chen, Tao, RuiFeng Xu, Yulan He, and Xuan Wang. 2015. Improving distributed representation of word sense via WordNet gloss composition and context clustering. In Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing, Beijing, China.


COOPER, SETH, FIRAS KHATIB, ADRIEN TREUILLE, JANOS BARBERO, JEEHYUNG LEE, MICHAEL BEENEN, ANDREW LEAVER-FAY, DAVID BAKER, ZORAN POPOVIĆ, and 57,000 FOLDIT PLAYERS. 2010. Predicting protein structures with a multiplayer online game. Nature 466(7307).


BIBLIOGRAPHY


FRANTZI, KATERINA T, and SOPHIA ANANIAZOU. 1999. The C-value/NC-value domain-independent method for multi-word term extraction. Journal of Natural Language Processing 6(3).


GALE, WILLIAM A., KENNETH W. CHURCH, and DAVID YAROWSKY. 1992b. One sense per discourse. In Proceedings of the DARPA Speech and Natural Language Workshop, Harriman, USA.


——, and Roberto Navigli. 2014. It’s All Fun and Games until Someone Annotates: Video Games with a Purpose for Linguistic Annotation. Transactions of the ACL 2.


KITTUR, ANIKET. 2010. Crowdsourcing, collaboration and creativity. XRDS: crossroads, the ACM magazine for students 17(2).


KREMER, GERHARD, KATRIN ERK, SEBASTIAN PADÓ, and STEFAN THATER. 2014. What
substitutes tell us – analysis of an “all-words” lexical substitution corpus. In 14th Conference of
the European Chapter of the Association for Computational Linguistics, Gothenburg, Sweden.

KUDO, TAKU, and YUJI MATSUMOTO. 2002. Japanese dependency analysis using cascaded
chunking. In Proceedings of the 6th Conference on Natural Language Learning, Taipei, Tai-
wan.

— —, KAORU YAMAMOTO, and YUJI MATSUMOTO. 2004. Applying conditional random fields to
Language Processing, Barcelona, Spain.

KULKARNI, ANAGHA, MICHAEL HEILMAN, MAXINE ESKENAZI, and JAMIE CALLAN. 2008.
Word sense disambiguation for vocabulary learning. In Proceedings of the 9th International
Conference on Intelligent Tutoring Systems, Montreal, Canada.

KUROHASHI, SADAO, and DAISUKE KAWAHARA, 2017. Juman (a user-extensible morphological

structures in Japanese. In Proceedings of the 14th conference on Computational linguistics,
Nantes, France.

the Workshop on Sharable Natural Language Resources, Nara, Japan.

LAFFERTY, JOHN, ANDREW MCCALLUM, and FERNANDO PEREIRA. 2001. Conditional random
fields: Probabilistic models for segmenting and labeling sequence data. In Proceedings of the
Eighteenth International Conference on Machine Learning, Williamstown, USA.

LANDAUER, THOMAS K, and SUSAN T DUMAIS. 1997. A solution to Plato’s problem: The
latent semantic analysis theory of acquisition, induction, and representation of knowledge.
Psychological review 104(2).

LANDES, SHARI, CLAUDIA LEACOCK, and RANDEE I. TENG. 1998. Building semantic concor-
Cambridge, USA: MIT Press.

similarity. Studia Linguistica 54(2).

LE, PHONG, and WILLEM ZUIDEMA. 2015. Compositional distributional semantics with long short
term memory. In Proceedings of the Fourth Joint Conference on Lexical and Computational
Semantics, Denver, USA.

LEE, YOONG KEOK, and HWEE TOU NG. 2002. An empirical evaluation of knowledge sources
and learning algorithms for word sense disambiguation. In Proceedings of the 2002 Conference
on Empirical Methods In Natural Language Processing, Philadelphia, USA.


LIEBEROTH, ANDREAS, MADS KOCK PEDERSEN, ANDREEA CATALINA MARIN, Tilo PLANKE, and JACOB FRIIS SHERSON. 2014. Getting humans to do quantum optimization - user acquisition, engagement and early results from the citizen cyberscience game quantum moves. Human Computation 1(2).


———, and ———. 2015b. A methodology for word sense disambiguation at 90% based on large-scale crowdsourcing. In Proceedings of the Fourth Joint Conference on Lexical and Computational Semantics, Denver, USA.

LOSSIO-VENTURA, JUAN ANTONIO, CLEMENT JONQUET, MATHIEU ROCHE, and MAGUELONNE TEISSEIRE. 2014. Yet another ranking function for automatic multi-word term extraction. In International Conference on Natural Language Processing, Warsaw, Poland.


MIKOLOV, TOMAS, KAI CHEN, GREG CORRADO, and JEFFREY DEAN. 2013. Efficient estimation of word representations in vector space. In International Conference on Learning Representations Workshop, Scottsdale, USA.


Nah, Fiona Fui-Hoon. 2004. A study on tolerable waiting time: how long are web users willing to wait? Behaviour & Information Technology 23(3).


PAGE, LAWRENCE, SERGEY BRIN, RAJEEV MOTWANI, and TERRY WINOGRAD. 1998. The PageRank citation ranking: bringing order to the web.

PALMER, MARTHA, HOA TRANG DANG, and CHRISTIANE FELLBAUM. 2007a. Making fine-grained and coarse-grained sense distinctions, both manually and automatically. Natural Language Engineering 13(02).


PREISS, JUDITA, JON DEHDARI, JOSH KING, and DENNIS MEHAY. 2009. Refining the most frequent sense baseline. In *Semantic Evaluations: Recent Achievements and Future Directions*, Boulder, USA.


ROSA, KEVIN DELA, and MAXINE ESKENAZI. 2011. Impact of word sense disambiguation on ordering dictionary definitions in vocabulary learning tutors. In Proceedings of the 24th International Florida Artificial Intelligence Research Society Conference, Palm Beach, USA.


SALEHI, BAHAR, and PAUL COOK. 2013. Predicting the compositionality of multiword expressions using translations in multiple languages. In Second Joint Conference on Lexical and Computational Semantics (*SEM), Volume 1: Proceedings of the Main Conference and the Shared Task: Semantic Textual Similarity, Atlanta, USA.


BIBLIOGRAPHY


TAN, PANG-NING, MICHAEL STEINBACH, and VIPIN KUMAR. 2006. Introduction to Data Mining. Boston, USA: Pearson Education.


Author/s:
Fothergill, Richard James

Title:
Crowdsourcing lexical semantic judgements from bilingual dictionary users

Date:
2017

Persistent Link:
http://hdl.handle.net/11343/194282

File Description:
Complete thesis: Crowdsourcing lexical semantic judgements from bilingual dictionary users

Terms and Conditions:
Terms and Conditions: Copyright in works deposited in Minerva Access is retained by the copyright owner. The work may not be altered without permission from the copyright owner. Readers may only download, print and save electronic copies of whole works for their own personal non-commercial use. Any use that exceeds these limits requires permission from the copyright owner. Attribution is essential when quoting or paraphrasing from these works.