Knowledge Discovery and Extraction of Domain-specific Web Data

A thesis presented by

Li Wang

to
The Department of Computing and Information Systems
in partial fulfillment of the requirements for the degree of Doctor of Philosophy (with coursework component)

The University of Melbourne
Melbourne, Australia
June 2014

Produced on Archival Quality Paper
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: ________________
Knowledge Discovery and Extraction
of Domain-specific Web Data

Abstract
Web user forums (or simply “forums”) are a valuable means for users to resolve specific information needs, both interactively for the participants and statically for users who search and browse over historical thread data. However, the complex structure of forum threads can make it difficult for users to extract relevant information. Addressing this problem, we propose to parse thread discourse structure of forum threads for the purpose of enhancing information access and solution sharing over web user forums.

The discourse structure of a forum thread is modelled as a rooted directed acyclic graph (DAG), and each post in the thread is represented as a node in this DAG. The reply-to relations between posts are then denoted as directed edges (LINKs) between nodes in the DAG, and the type of a reply-to relation is defined as a dialogue act (DA). To parse the discourse structure of threads, both LINKs and DAs need to be identified. The first method we propose uses conditional random fields to either classify the LINK and DA separately and compose them afterwards, or classify the combined LINK and DA directly. Another technique we adopt is to treat
this discourse structure parsing as a dependency parsing problem, because the joint
classification of LINK and DA is a natural fit for dependency parsing. Our parsing
methods not only perform significantly better than a strong heuristic baseline, but
also can robustly handle growing threads, and achieve similar results over partial
threads compared to complete threads. Additionally, we also explore unsupervised
approaches for LINK classification by using lexical chaining.

Then, we explore ways of using thread discourse structure information to im-
prove information access and solution sharing over web user forums. Specifically,
we first demonstrate that the proposed discourse structure can help thread solved-
ness identification (i.e. automatically identify whether the question asked in a fo-
rum thread is resolved or not). The basic idea is using features derived from thread
discourse structure to help solvedness classification. For example, the last reply-to
LINK and its DA type can be indicative of whether the asked question is resolved or
not. Experimental results show that simple features derived from thread discourse
structure can greatly boost the accuracy of solvedness classification, which has been
shown to be very difficult in previous research.

We also investigate the utility of discourse structure in forum thread IR. The
proposed method first parses the discourse structure of targeted threads, then uses
information from the parsed discourse structure to augment existing IR systems.
For instance, if a post is linked to a question post with a DA type of an answer, more
weight should be given to this post during retrieval. Experimental results demon-
strate that exploiting the characteristics of discourse structure of forum threads can
benefit IR, when compared to previously-published state-of-the-art IR methods.
## 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>iii</td>
</tr>
<tr>
<td>Table of Contents</td>
<td>v</td>
</tr>
<tr>
<td>List of Figures</td>
<td>vi</td>
</tr>
<tr>
<td>List of Tables</td>
<td>viii</td>
</tr>
<tr>
<td>Citations to Previously Published Work</td>
<td>xiv</td>
</tr>
<tr>
<td>Acknowledgments</td>
<td>xv</td>
</tr>
</tbody>
</table>

1 Introduction
1.1 Contributions                                                   10
1.2 Thesis Outline                                                  11

2 Literature Review
2.1 Introduction                                                    14
   2.1.1 Glossary                                                    14
2.2 Metadata Recovery from Forum-related Data                        16
   2.2.1 Thread Attribute Analysis                                   16
   2.2.2 Dialogue Act Tagging                                         25
   2.2.3 Thread Structure Analysis                                   31
   2.2.4 Post Attribute Analysis                                     58
2.3 Forum-related Tasks                                              70
   2.3.1 Information Retrieval over Forum Data                      71
   2.3.2 Forum Thread Summarisation                                  83
   2.3.3 Knowledge Base Augmentation                                 84
2.4 Forum Data Crawling                                              86
2.5 Lexical Chaining                                                87
2.6 Summary                                                         89

3 Resources
3.1 Introduction                                                    91
3.2 CNET Dataset                                                   92
### Contents

<table>
<thead>
<tr>
<th>Section</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>3.3</td>
<td>ILIAD Dataset</td>
<td>98</td>
</tr>
<tr>
<td>3.4</td>
<td>Ancestry Dataset</td>
<td>100</td>
</tr>
<tr>
<td>3.5</td>
<td>Dependency Parsing</td>
<td>108</td>
</tr>
<tr>
<td>3.5.1</td>
<td>Introduction</td>
<td>108</td>
</tr>
<tr>
<td>3.5.2</td>
<td>MaltParser</td>
<td>114</td>
</tr>
<tr>
<td>3.6</td>
<td>Graphical Models</td>
<td>117</td>
</tr>
<tr>
<td>3.6.1</td>
<td>Introduction</td>
<td>117</td>
</tr>
<tr>
<td>3.6.2</td>
<td>CRFSGD</td>
<td>120</td>
</tr>
<tr>
<td>3.7</td>
<td>Experimental Methodologies</td>
<td>122</td>
</tr>
<tr>
<td>3.7.1</td>
<td>Feature Weighting</td>
<td>122</td>
</tr>
<tr>
<td>3.7.2</td>
<td>Evaluation</td>
<td>124</td>
</tr>
<tr>
<td>3.7.3</td>
<td>Hydrat</td>
<td>130</td>
</tr>
<tr>
<td>3.8</td>
<td>Summary</td>
<td>132</td>
</tr>
<tr>
<td>4</td>
<td>Forum Thread Structural Analysis</td>
<td>133</td>
</tr>
<tr>
<td>4.1</td>
<td>Introduction</td>
<td>133</td>
</tr>
<tr>
<td>4.2</td>
<td>Thread Discourse Structure Analysis</td>
<td>134</td>
</tr>
<tr>
<td>4.2.1</td>
<td>Introduction</td>
<td>134</td>
</tr>
<tr>
<td>4.2.2</td>
<td>Task Description</td>
<td>136</td>
</tr>
<tr>
<td>4.2.3</td>
<td>Learners and Features</td>
<td>139</td>
</tr>
<tr>
<td>4.2.4</td>
<td>Classification Methodology</td>
<td>145</td>
</tr>
<tr>
<td>4.2.5</td>
<td>Results and Evaluation</td>
<td>148</td>
</tr>
<tr>
<td>4.2.6</td>
<td>Summary of Thread Discourse Structure Analysis</td>
<td>157</td>
</tr>
<tr>
<td>4.3</td>
<td>Thread Linking Structure Analysis by Lexical Chaining</td>
<td>158</td>
</tr>
<tr>
<td>4.3.1</td>
<td>Introduction of Linking Analysis by Lexical Chaining</td>
<td>159</td>
</tr>
<tr>
<td>4.3.2</td>
<td>Lexical Chaining Algorithms</td>
<td>160</td>
</tr>
<tr>
<td>4.3.3</td>
<td>Methodology</td>
<td>165</td>
</tr>
<tr>
<td>4.3.4</td>
<td>Assumptions, Experiments and Analysis</td>
<td>166</td>
</tr>
<tr>
<td>4.3.5</td>
<td>Summary of Linking Analysis by Lexical Chaining</td>
<td>173</td>
</tr>
<tr>
<td>4.4</td>
<td>Summary</td>
<td>174</td>
</tr>
<tr>
<td>5</td>
<td>The Utility of Discourse Structure in Identifying Resolved Threads in Technical User Forums</td>
<td>175</td>
</tr>
<tr>
<td>5.1</td>
<td>Introduction</td>
<td>175</td>
</tr>
<tr>
<td>5.2</td>
<td>Data Description</td>
<td>178</td>
</tr>
<tr>
<td>5.3</td>
<td>Discourse Parsing for Thread Solvedness Classification</td>
<td>181</td>
</tr>
<tr>
<td>5.4</td>
<td>Solvedness Classification</td>
<td>184</td>
</tr>
<tr>
<td>5.4.1</td>
<td>Experiments over ILIAD 222</td>
<td>185</td>
</tr>
<tr>
<td>5.4.2</td>
<td>Experiments over ILIAD</td>
<td>187</td>
</tr>
<tr>
<td>5.5</td>
<td>Results Analysis and Simulation</td>
<td>189</td>
</tr>
<tr>
<td>5.6</td>
<td>Discussion</td>
<td>195</td>
</tr>
<tr>
<td>5.7</td>
<td>Summary</td>
<td>196</td>
</tr>
</tbody>
</table>
List of Figures

1.1 A question on the system freeze problem of 13-inch MacBook Pro Retina .................................................. 2
1.2 Posts from an Apple Discussion forum thread marked as “This solved my question” or “This helped me” ...................... 3
1.3 An example question reproduction post from an Apple Discussion forum thread ...................................................... 4
1.4 An answer post, a following answer confirmation post, and a post provides more information about the answer, from an Apple Discussion forum thread ......................................... 5
1.5 A snippeted and annotated CNET thread ................................................................. 7
1.6 A snippeted and annotated ILIAD thread ................................................................. 8

2.1 Different contexts in a thread structure ................................................................ 77

3.1 Screenshot of the example CNET thread ............................................................. 96
3.2 An example CNET thread with thread discourse structure ............................... 97
3.3 Screenshot of the example ILIAD thread ............................................................. 100
3.4 Screenshot of the example Ancestry thread in flat view .................................... 104
3.5 Screenshot of the example Ancestry thread in thread view ............................... 105
3.6 Screenshot of the Ancestry data annotation interface (Elsas 2011) ................. 106
3.7 Dependence structure of an example CNET thread ........................................... 111

4.1 An example CNET thread with thread discourse structure ............................... 135
4.2 Breakdown of post-level LinkDA results for CRFSGD and MaltParser based on post position ............................................. 152
4.3 Breakdown of post-level Link and DA F-score based on the decomposition of CRFSGD and MaltParser classifications 153
4.4 Post-level joint classification results for users binned by uscore, based on MaltParser with and without UserProf features 157
4.5 A snippeted CNET thread annotated with linking structure ........................... 160
List of Figures

<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>5.1</td>
<td>A snippeted ILIAD thread with annotated discourse structure</td>
<td>177</td>
</tr>
<tr>
<td>5.2</td>
<td>Examples of a pair and a subthread in a thread</td>
<td>183</td>
</tr>
<tr>
<td>5.3</td>
<td>Simulation over all DAs (AllDA)</td>
<td>191</td>
</tr>
<tr>
<td>5.4</td>
<td>Simulation over automatically-generated DA groups (PositiveDA\textsubscript{auto} and OtherDA\textsubscript{auto})</td>
<td>192</td>
</tr>
<tr>
<td>5.5</td>
<td>Simulation over manually-created DA groups (PositiveDA\textsubscript{manual} and OtherDA\textsubscript{manual})</td>
<td>193</td>
</tr>
<tr>
<td>6.1</td>
<td>A snippeted and annotated Ancestry thread</td>
<td>199</td>
</tr>
<tr>
<td>6.2</td>
<td>Different contexts in a thread structure</td>
<td>200</td>
</tr>
<tr>
<td>6.3</td>
<td>An example ranking comparison from the Indri-DM-PCS and Indri-DM-PCS systems</td>
<td>216</td>
</tr>
<tr>
<td>A.1</td>
<td>Calculating the counts distribution for Post 2</td>
<td>266</td>
</tr>
<tr>
<td>A.2</td>
<td>Calculating the counts distribution for Post 3</td>
<td>266</td>
</tr>
</tbody>
</table>
List of Tables

2.1 An example subjective thread (Biyani et al. 2012) .......................................................... 20
2.2 An example non-subjective thread (Biyani et al. 2012) ..................................................... 21
2.3 The components of the thread class set by Wang et al. (2010b) ........................................... 23
2.4 The dialogue act (DA) tagset proposed by Kim et al. (2010b) ........................................... 39
2.5 Dialogue acts proposed by Kim et al. (2006) ........................................................................... 42
2.6 The 12 dialogue act labels defined by (Jeong et al. 2009) ..................................................... 65

3.1 An example thread from the CNET dataset ......................................................................... 93
3.2 Data source categories and sub-forums of the CNET dataset ............................................. 97
3.3 An example thread from the ILIAD dataset .......................................................................... 99
3.4 An example thread from the Ancestry dataset .................................................................... 103
3.5 A transition sequence for the thread in Figure 3.7 (LA_ represents Left-Arc, RA_ represents Right-Arc, and SH represents Shift.) ...................................................... 110
3.6 A simple feature model with seven features. Regarding the address functions, STK[i] locates the i-th post in the stack, BUF[i] locates the i-th post in the buffer, LDEP(w) locates the farthest child of post w to the left, and RDEP(w) locates the farthest child of post w to the right. With respect to attribute functions, COUNT is the word count of a post and DEPREL represents the dependency label (dialogue act label in our case) .............................................................. 113
3.7 Feature (Feat) vectors for the configurations (Config) in Table 3.5, based on the feature model in Table 3.6 (N_P_i represents the number of words in Post i.) ............................................................ 114
3.8 Confusion matrix for class c ................................................................................................. 126
3.9 TP/TN/FP/FN of the prediction for two example multi-label multi-class instances d_i and d_j ........................................................................................................................................................................ 126

4.1 The dialogue act (DA) tagset proposed by Kim et al. (2010b) ............................................ 137
4.2 Thread length distribution of the CNET dataset ................................................................. 139
4.3 Link distribution of the CNET dataset .................................................................................. 140
4.4 DA distribution of the CNET dataset .................................................................................... 140

x
List of Tables

4.5 Parameter settings used for CRFSGD in this research ........................................... 141
4.6 Feature model used with MaltParser in this research. As explained in section 3.5, \( \text{STK}[i] \) locates the \( i \)th post in the stack, \( \text{BUF}[i] \) locates the \( i \)th post in the buffer, \( \text{RDEP}(w) \) locates the farthest child of post \( w \) to the right, and \( \text{DEPREL} \) represents the dependency label (dialogue act label in our case). Additionally, \( \text{HEAD}(w) \) locates the head of the \( w \) post in the stack, \( \text{STK}[i]-\text{STK}[j] \) merges the feature from \( \text{STK}[i] \) and the feature from \( \text{STK}[j] \) into one feature, and \( \text{RAW} \) represent all the raw structural and semantic features defined latter in the same section. Note that each entry from feature no. 1-12 creates one feature for each raw feature. .......................... 142
4.7 The feature presentation of the third post in a thread of length 8 .................................. 144
4.8 Post/thread-level component-wise classification F-scores for Link and DA classes .......................................................... 148
4.9 Post/thread-level LinkDA joint classification F-scores (“*” signifies a significantly worse result than that for the same learner with ALL features) ......................................................... 149
4.10 Post/thread-level Link and DA F-scores from component-wise classification and from LinkDA classification decomposition .......................................................... 151
4.11 Post-level LinkDA F-score for CRFSGD/MaltParser, based on in situ classification over sub-threads of different lengths (indicated in the rows), broken down over different post extents (indicated in the columns) .......................................................... 154
4.12 Statistics for the 4 groups of users .................................................................................. 156
4.13 Results from the Assumption 1 based unsupervised approach, by using three lexical chaining algorithms with four different weighting schemes ................................................................................. 167
4.14 Results from the Assumption 2 based unsupervised approach, by using three lexical chaining algorithms with four different weighting schemes ................................................................................. 169
4.15 Results from the Assumption 3 based unsupervised approach, by using ChainerSV with different weighting schemes .......................................................... 171
4.16 Supervised linking classification by applying CRFSGD over features from Section 4.2 without (NoLC) and with (WithLC) features extracted from lexical chains, created by ChainerSV with different weighting schemes ................................................................................. 172

5.1 The Dialogue Act (DA) set used for annotating ILIAD dataset (“*” signifies a difference over the original method of Kim et al. (2010b), while the definitions of other labels are consistent with those used previously) .......................................................... 180
5.2 Thread discourse structure features used for Solvedness classification .......................... 182
<table>
<thead>
<tr>
<th>Table</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>5.3</td>
<td>Results over ILIAD, using discourse structure features from the gold-standard and also the discourse parsing model (&quot;*&quot; signifies a significantly better result than both baselines; the best result in each column is indicated in <strong>boldface</strong>).</td>
<td>185</td>
</tr>
<tr>
<td>5.4</td>
<td>Results over ILIAD, using discourse structure features from the gold-standard and also the discourse parsing model (&quot;*&quot; signifies a significantly better result than both baselines; the best result in each column is indicated in <strong>boldface</strong>).</td>
<td>188</td>
</tr>
<tr>
<td>5.5</td>
<td>Entropy of each DA against the Solvedness class distribution for every DA-only feature and AllDAFeat features.</td>
<td>190</td>
</tr>
<tr>
<td>5.6</td>
<td>Micro-averaged DA classification F-scores per DA over ILIAD.</td>
<td>194</td>
</tr>
<tr>
<td>6.1</td>
<td>Summary of Elsas (2011)’s experimental setup.</td>
<td>205</td>
</tr>
<tr>
<td>6.2</td>
<td>Elsas (2011)’s IR results (Original) and our reproduced results (Reproduced) over the Ancestry dataset using Indri. Retrieval is performed at the post-level, and evaluation is conducted at the thread-level. Three aggregation methods are used for each system to transform post-level scores to thread-level scores. The best results for each column are <strong>bold-faced</strong>.</td>
<td>207</td>
</tr>
<tr>
<td>6.3</td>
<td>The effect of ( k ) in Indri-DM-PCS.</td>
<td>208</td>
</tr>
<tr>
<td>6.4</td>
<td>Elsas (2011)’s IR results (Original) and our reproduced results (Reproduced) over the Ancestry dataset using Terrier. Retrieval is performed at the post-level, and evaluation is conducted at the thread-level. Three aggregation methods are used for each system to transform post-level scores to thread-level scores. The best results for each column are <strong>bold-faced</strong>.</td>
<td>209</td>
</tr>
<tr>
<td>6.5</td>
<td>Discourse structure parsing F-scores by applying CRFSGD/MaltParser with Initiator feature using the Combined approach over different training dataset setups (The best result for each column is <strong>bold-faced</strong>).</td>
<td>210</td>
</tr>
<tr>
<td>6.6</td>
<td>The ( mAP\text{pref}/ppref@10 ) scores from Indri-DM-LD when training the discourse parser over different training data sets (CNET, ILIAD or CNET+ILIAD), and with different promotion factors for the selected DAs; <strong>boldface</strong> signifies a better result than the Indri-DM-PCS baseline at a level of statistical significance ( (p &lt; 0.05) ). The baseline results are ( mAP\text{pref} = .657 ) and ( p\text{pref}@10 = .664 ).</td>
<td>212</td>
</tr>
</tbody>
</table>
6.7 The mAPpref/ppref@10 scores from Indri-DM-LD when training the discourse parser over different training data sets (CNET, ILIAD or CNET+ILIAD), based on 10-fold cross-validation experiments. The table also shows the promotion factors learned and their corresponding standard deviations (δ). **boldface** signifies a better result than the Indri-DM-PCS baseline at a level of statistical significance (p < 0.05). The baseline results are mAPpref = .657 and ppref@10 = .664.

---

A.1 An hypothetical annotation example .......................... 264
A.2 Confusion matrix for the hypothetical example .................. 264
A.3 An hypothetical post annotation example ....................... 265
A.4 Confusion matrix for the hypothetical post annotation example ... 267
Citations to Previously Published Work

Large portions of Chapter 4 have appeared in the following papers:


Large portions of Chapter 5 have appeared in the following paper:


Large portions of Chapter 6 have appeared in the following paper:

Acknowledgments

My wholehearted thanks first go to my two supervisors, Tim Baldwin and Su Nam Kim. They are both experienced mentors and my dear friends, who have accompanied me through my long PhD journey with great patience and supportiveness. Tim has been a role model to me, from both a research and life perspective. The inspiration of his personality and diligence will serve as a lifelong guidance to me for many years to come. Su’s advice regarding my research and career have always been invaluable to me, and made my PhD life much easier. Without Tim and Su, my PhD experience would never have been so enjoyable and fruitful. I am also extremely grateful to James Bailey for being my thesis committee chair, providing critical feedback, and supporting my career development.

I am really lucky to have many supportive collaborators during my PhD research. Marco Lui, who gave me much help and played a substantial role in the first couple of years of my PhD, has been inspiring me since our first encounter. Joakim Nivre, who patiently provided support and guidance for using MaltParser throughout my experiments, which made my first international publication possible. Diana McCarthy, who opened the gate of lexical semantics for me, and motivated our lexical chaining paper. Last but not least, I am very appreciative of the mentoring from Chin-Yew Lin during my internship at Microsoft Research Asia. Collaborating with Chin-Yew has been a unique experience, which opened my eyes to world-grade industry research.

I feel it is a blessing to be part of Melbourne Natural Language Processing group, and to be surrounded by such talented and diverse colleagues. Their vibrant presence has made my PhD experience complete. Particular mention goes to people who
have given me indispensable help and stimulation: Marco Lui, Bo Han, Jey Han Lau, Clint Burford and Paul Cook.

The University of Melbourne and my friends also have played crucial roles in making my PhD research an especially pleasant journey. Julian Harris from Academic Skills has helped me tremendously to improve my academic writing and presentation skills. Robyn Gamble from the Career and Industry Unit has assisted me greatly in my career development. William Webber, who is both a mentor and friend, has given me priceless suggestions regarding my research and career. I also cherish my friendship with Jan Schröder, which has enriched my soul.

I am also heavily indebted to The University of Melbourne and NICTA, who not only have offered me a great environment to advance my research, but also have generously covered my costs of living, training, and conference travels.

Lastly, I express my dearest thanks to my wife and family. Their unselfishness support and sacrifice are unimaginable. I cannot thank them enough for everything that they have done for me. This is where I have to say: “No words can express my appreciation and love.”

Thank you.
Chapter 1

Introduction

Web user forums (or simply “forums”) are online platforms for people to share and obtain information via a text-based threaded discourse, generally in a pre-determined domain (e.g. IT support or genealogy discussions). With the advent of the Web 2.0, there has been an explosion of web authorship in this area, and forums are now widely used in various areas such as customer support, community development, interactive reporting and online eduction. In addition to providing the means to interactively participate in discussions, or obtain or provide answers to questions, the vast volumes of data contained in forums make them a valuable resource for “support sharing”, i.e. looking over records of past user interactions to potentially find an immediately applicable solution to a current problem. On the one hand, more and more answers to questions over a wide range of domains are becoming available on forums; on the other hand, it is becoming harder and harder to extract and access relevant information due to the sheer scale of the data, and its diversity and complexity.
Figure 1.1: A question on the system freeze problem of 13-inch MacBook Pro Retina

To illustrate the above described problem, consider a question\textsuperscript{1} posted on the Apple Discussion forum\textsuperscript{2} as shown in Figure 1.1. The poster bought a new 13-inch MacBook Pro Retina, and experienced two sudden system freezes within two days. Therefore, he/she tried to get help from the community. From Figure 1.1 we can see that this is a rather hot topic. The discussion spans 190 pages, includes 2837 replies, and has attracted 331615 views. Additionally, 216 users have indicated that they have the same problem too. If another user were to come across this same problem

\textsuperscript{1}https://discussions.apple.com/thread/5481839
\textsuperscript{2}https://discussions.apple.com
Figure 1.2: Posts from an Apple Discussion forum thread marked as “This solved my question” or “This helped me”
Chapter 1: Introduction

Figure 1.3: An example question reproduction post from an Apple Discussion forum thread

and try to find possible solutions from this thread, the Apple Discussion forum has the facility for the thread initiator to flag posts as “This solved my question” or “This helped me”, and it is possible to filter posts in a thread by these flags. Unfortunately, in this example, the posts marked as “This solved my question” or “This helped me”, which are shown in Figure 1.2, only reveal that people are experiencing similar problems and contain no useful information. Therefore, the only option for a user who is seeking solutions is to read the posts in chronological order.

However, reading the posts in chronological order turns out to be very tedious. Most posts on the first page are “reproduction” style posts where the posters simply complain that they have the similar problems, as shown in Figure 1.3. The only answer post on the first page is a suggestion from an answerer who has not tried the solution himself/herself, as shown in Figure 1.4. Figure 1.4 shows the interactions between the answerer and another user, where the answerer first provides a solution, followed by a post from another user to confirm more details about the answer, and finally the answerer adds more information to his/her original answer. From these
interactions, it is unlikely that the proposed solution will work. If a user were to choose to keep on reading the posts on the next couple of pages, he/she would find out that there are no answer posts on the second page of 15 posts and there are only two answer posts on the third page of 15 posts. Additionally, among the small number of answers provided in the first three pages, it is not clear whether there is a valid solution or not. The user has to try them out one by one, which is even more
time consuming. In summary, whilst there is relevant information in this thread, it is very hard to access. Now, imagine that the interactions, which are represented as reply-to links, between posts and the types of each interaction are clear to users. For example, in Figure 1.4, a user knows that the first post is an answer post linking back to the initial question, the second post is an answer confirmation post linking back to its preceding answer post, and the third post provides more information to the answer post. If this linking and tagging information is available to users who search for answers, it has great potential to improve their information browsing experience.

This thesis targets the above task. Specifically, we first propose the task of automatically parsing the discourse structure of forum threads, for the purpose of enhancing information access and solution sharing over forums. To illustrate the task, we use a second shorter example thread, made up of 5 posts from 4 distinct participants, as shown in Figure 1.5. This thread is from the CNET forum dataset, which will be described in Section 3.2. The discourse structure of the thread is modelled as a rooted directed acyclic graph (DAG) with a dialogue act (DA) label associated with each edge (Link) of the graph. The Links represent the reply-to relations between posts, and the DAs represent the type of each reply-to relation. To parse the discourse structure of threads, both Links and DAs need to be identified. In this example, User A initiates the thread with a question in the first post, by asking how to create an interactive input box on a webpage. In response, User B and User C provide independent answers. User A responds to User C to confirm the details of the solution, and at the same time, adds extra information to his/her
original question; i.e., this one post has two distinct dependency links associated
with it. Finally, User D proposes a different solution again to the original question.
It should be noted that while some forums already have the reply-to Link informa-
tion, many forums do not. For example, forums which are built using phpBB\(^3\) and
vBulletin\(^4\) either do not support the reply-to linking, or do not provide it by default.
In Section 4.2, we will detail our work on parsing the thread discourse structure.

\(^3\)https://www.phpbb.com/
\(^4\)https://www.vbulletin.com/
A natural question following the above research is how thread discourse structure can be used to help users better access information and solutions in forums. Intuitively, the discourse structure could be used directly to better visualize the thread representation, by highlighting potentially more important posts to users. We can also evaluate discourse structure extrinsically by exploiting its utility for other tasks. To this end, we explore two different tasks using thread discourse structure. The first task, which will be discussed in Chapter 5, utilises the thread discourse structure to facilitate Solvedness identification (i.e. automatically identify whether the question asked in a forum thread is resolved or not). Figure 1.6 shows another example
thread, made up of 5 posts from 3 distinct participants, with the discourse structure of the thread. This thread is from the ILIAD dataset, which will be explained in Section 3.3. In this thread, UserA initiates the thread with a question (DA = Question-question) in the first post, by asking a question. In response, UserB provides an answer (DA = Answer-answer). Then, UserA confirms more details about the answer provided (Dialogue Act = Answer-confirmation). UserB responds to UserA to add more information about his/her previous answer (DA = Answer-add). Finally, UserC proposes an independent answer again to the original question (DA = Answer-answer). It is obvious that although two independent answers are provided in Post2 and Post5, it is almost impossible to identify whether there is a correct solution unless the whole thread is understood at a technical level. In fact, in their previous research over the ILIAD dataset, Baldwin et al. (2007) showed that the Solvedness identification task is empirically very difficult. However, we will demonstrate in Chapter 5 that simple features derived from thread discourse structure can greatly boost the accuracy of Solvedness classification.

As the second task, which will be discussed in Chapter 6, we explore ways to use information from thread discourse structure to augment existing information retrieval (IR) systems, for thread-level IR over forums. The intuition of using discourse structure for IR is that certain types of posts are more important than other types of posts, in terms of satisfying users’ information needs. For instance, if a post is linked to a question post with a DA type of an answer, more weight should be given to this post during retrieval. Experimental results demonstrate that exploiting the characteristics of discourse structure of forum threads can benefit IR,
Chapter 1: Introduction

when compared to previously-published state-of-the-art IR methods. Chapter 6 will give a detailed description of this work.

Additionally, we have also worked on methods which only target Link recovery. As will be described in Section 4.3, we investigate ways of using lexical chaining (i.e. a technique for identifying lists of related words within a given discourse) for Link identification in forum threads.

1.1 Contributions

The contributions of this thesis are as follows:

- We propose the use of conditional random fields and transition-based dependency parsing to deal with the joined Link and DA classification of thread discourse structure, in the form of a rooted directed acyclic graph over posts, with edges labelled with dialogue acts. The proposed methods achieve significantly better F-scores when compared to an informed baseline. We also examine the task of in situ classification of discourse structure, in the form of predicting the discourse structure of partial threads, as contrasted with classifying only complete threads. We find that there is no drop in F-score over different sub-extents of the thread in classifying partial threads, despite the relative lack of thread context. Additionally, we propose an unsupervised approach to predict thread linking structure using lexical chaining, a technique which identifies lists of related word tokens within a given discourse.

- We investigate the utility of discourse structure in identifying resolved threads
in technical user forums. We exploit the implicit indicators of solvedness from thread discourse structure. For example, the last reply-to Link and its DA type can be indicative of whether the asked question is resolved or not. We show that simple features derived from thread discourse structure can greatly boost the accuracy of solvedness classification, which has been shown to be very difficult in previous research.

- We explore the utility of discourse structure in forum thread retrieval. We hypothesise that different types of posts in a thread should be weighted differently for forum thread retrieval. This is because, while some types of posts are more useful to a query (e.g. answer posts), some types of posts are less useful (e.g. question posts). We demonstrate that weighting posts dynamically according to thread discourse structure can benefit thread retrieval, when compared to previously-published state-of-the-art IR methods.

- We carry out an extensive and detailed literature review over forum-related research, where there is a dearth of review articles and the literature has tended to be disjointed. The review includes most of the forum research in the field of the natural language processing, excluding user profiling research in the forum domain.

1.2 Thesis Outline

Chapter 2: This chapter provides a detailed literature review over forum-related research. We first look into work which attempts to recover metadata from
forum data, such as reply-to links between posts, post-level dialogue acts, post quality, and thread-level characteristics. We also review research on forum-related tasks, including forum information retrieval, thread summarisation and knowledge base construction.

Chapter 3: This chapter describes the resources used in this research, including three forum datasets from different domains, two off-the-shelf natural language processing packages, and a declarative classification framework. Additionally, we explain the core empirical methodologies that underpin the software packages used in this research, namely graphical machine learning models, dependency parsing, relevant evaluation metrics, and feature weighting.

Chapter 4: This chapter mainly focuses on our work on parsing thread discourse structure. We first introduce the proposed generalised methods for thread discourse parsing, using conditional random fields and dependency parsing. Then, we demonstrate that the proposed approaches can be robustly applied to dynamically evolving threads, where threads grow as new posts appear. Additionally, we also investigate the inter-post reply-to linking recovery task by exploiting lexical chaining, a technique to identify lists of related word tokens within a given discourse.

Chapter 5: In this chapter, we exploit ways to use discourse structure for thread solvedness prediction, that is to automatically identify whether the question asked in a forum thread is resolved or not. The proposed approach uses features extracted from thread discourse structure for solvedness classification,
Chapter 1: Introduction

which has been demonstrated by previous research to be a very difficult task. Experimental results show that simple features derived from thread discourse structure can greatly boost the accuracy of solvedness classification.

Chapter 6: In this chapter, we present our work on the analysis of the utility of thread discourse structure in the context of forum retrieval. In this work, we demonstrate that thread discourse structure can be used to boost forum thread retrieval effectiveness. The basic idea we adopt is that certain types of posts are more important than others, especially in the context of information retrieval. For example, an answer post is arguably better than a question post because it may fulfill a user’s information need. Based on this idea, we first parse the discourse structure of the target forum threads, and then use the predicted post types to adjust thread rankings produced by information retrieval systems. Experimental results show that the proposed methods can improve thread retrieval significantly, when compared to previously published results.

Chapter 7: This chapter summarises the content of this PhD thesis, as well as contributions of our work. Additionally, we present possible extensions of this research in terms of future work. Specifically, we highlight some of the unresolved challenges of thread discourse parsing and potential solutions. Moreover, we describe interesting research directions to carry out thread discourse structure analysis across different forums and domains.
Chapter 2

Literature Review

2.1 Introduction

In this section, we first present a detailed survey of forum-related research, with a focus on metadata recovery from forum-related data, as reviewed in Section 2.2, and forum-related tasks, as examined in Section 2.3. Then, we discuss research on forum data crawling in Section 2.4. Moreover, in Section 2.5, we briefly summarise the literature of lexical chaining which will be used in Section 4.3.

2.1.1 Glossary

The same or similar concepts sometimes appear in the literature under different names. We will try to use the same terminology for each concept throughout this chapter, and this section will summarise important terminology we will use.

**Thread**: we use the term “thread” to refer to forum discussion threads, newsgroup discussion threads, mailing list threads, and email interaction threads.
Post/message: the terms post and message are often used interchangeably in the research community to refer to each posting in a forum thread. In this chapter, we use “post” to denote forum thread post, and “message” to indicate email thread message. However, sometimes, we will use “post” and “message” interchangeably for brevity (e.g. when describing experiments over a forum dataset and a email dataset at the same time).

Word/term: in this chapter, “word” and “term” are used interchangeably to indicate a word unit in a post or message.

Thread initiator: the user who starts a new discussion thread. In the context of troubleshooting-oriented discussions, thread initiators are users who present problems and seek solutions.

Quoted text: in forums and email interactions, a user may sometimes quote content from previous posts or email messages in his/her post. This quoted content is called “quoted text” in this chapter.

Initial post/message: the first post/message of a thread. In the literature, it is sometimes also called the “root post/message” or “first post/message”.

Dialogue act: a full definition of dialogue act is provided in Section 2.2.2. While, strictly speaking, a “dialogue act” is a specialised “speech act”, these two terms are often used interchangeably in the literature. In this chapter, we will use “dialogue act” to denote both.
2.2 Metadata Recovery from Forum-related Data

Much research has been done on analysing the characteristics of forum data to recover useful information or metadata. Some research has focused on thread-level analysis, by investigating the attributes of threads or the overall thread structure. Other research has explored specific characteristics of posts such as post quality. In this section, relevant literature will be reviewed in four subsections. First, research on overall thread attribute analysis will be summarised. Then, we briefly review relevant dialogue act tagging research, which is closely related to the thread structure analysis literature in Section 2.2.3 and post-level research in Section 2.2.4.

2.2.1 Thread Attribute Analysis

Research on thread attribute analysis often aims at automatically identifying one or more characteristics of a discussion thread, thus helping users better access information in forums. For example, by automatically identifying whether a thread is problem-solving oriented, whether the initial post of the thread is detailed enough to elicit valid solutions, and/or whether the problem presented in a thread is solved, a system can more reliably assist users in asking questions and searching for solutions (Baldwin et al. 2007). Another example is subjectivity classification. By automatically identifying whether a thread is seeking opinions or looking for factual information, it can help improve forum search and help forum administrators monitor abusive conversations (Biyani et al. 2012). Moreover, by automatically classifying the problem sources and solution types of troubleshooting-oriented discussions, it can help users to spell out the general nature of their support need in
their queries (Wang et al. 2010a). Additionally, there has also been work on automatically predicting what topic a thread focuses on (Feng et al. 2006a). A variety of forums have been targeted in the research, such as technical computing discussions (Baldwin et al. 2007; Biyani et al. 2012; Wang et al. 2010b), travel forums (Biyani et al. 2012), and graduate course online discussions (Feng et al. 2006a). As for classification methodologies, the problem is often treated as a document categorisation task, and supervised methods are often used. It is interesting to note that while most research papers train their chosen supervised learners over manually annotated training data, Feng et al. (2006a) automatically induced training data from textbooks. Moreover, most research has made use of the thread structure for classification, except for Wang et al. (2010a), who treated each thread as a flat document. It should also be noted that, among this research, while the attribute of interest is often determined by the first post of each thread, the whole thread is often used to help automatically identify the attribute. However, some of the tasks proposed by Baldwin et al. (2007) capture interactions of posts in a thread, as we discuss first.

Baldwin et al. (2007) investigated three specific characteristics detected from discussion threads in Linux troubleshooting:

**Task orientation:** whether a thread focuses on problem-solving or general discussions.

**Completeness:** whether the initial post of the thread includes enough information about the targeted problem for thread participants to provide valid solutions.

**Solvedness:** whether there is a valid solution provided in the thread for the problem
raised by the thread initiator. If URLs leading to other web pages, which contain valid solutions, are provided, the problem will also be considered as solved.

As mentioned at the beginning of this section, “Task orientation” and “Solved-ness” are inherent attributes of the whole threads, and they capture the interactions among posts in each thread. This is unique among the research reviewed in this section. Three binary classification tasks were identified based on these three characteristics and experiments were carried out using a range of classification and regression methods, including SVMs (Joachims 1998) (with a linear kernel and an RBF kernel), $k$-nearest neighbour classifiers, propositional rule learners, decision trees, naive Bayes (NB), linear regressors, perceptron classifiers and meta-classifiers. Moreover, Baldwin et al. (2007) explored not only bag-of-words features, but also another 18 lexical and contextual features from four subparts of the thread:

Initial post: the initial post of the thread, and immediately following posts from the thread initiator.

First response: first post from a non-initiator participant.

All responses: all posts excluding the initial post and the first post from a non-initiator.

Final post from the initiator: the last post from the thread initiator. In the case that the last post is immediately preceded by posts from the initiator, these

\footnote{In this chapter, when introducing the machine learning algorithms used in a paper for the first time and SVMs are used, we use the term “SVMs” if SVMs with a linear kernel are assumed, otherwise we explicitly describe the kernel settings.}
are concatenated with the final post.

The intuition of dividing the thread into subparts is that different subparts are more relevant to different classification tasks. For example, “Initial post” is probably more relevant to “Task orientation” and “Completeness” classification, while “Final post from the initiator” might be more relevant to “Solvedness” classification. For experiments, Baldwin et al. (2007) crawled data from the Software subforum of Linuxquestions forum, and the debian-amd64 and debian-apache lists of Debian mailing lists. 250 threads from the data were annotated and experimented with. Both classification and rank correlation results of Baldwin et al. (2007) show that the tasks are very challenging, and it is very hard to outperform a majority class baseline. In Chapter 5, we will use their dataset to demonstrate that, by using thread discourse structure information, “Solvedness” classification can be improved.

Biyani et al. (2012) explored the task of automatic forum thread subjectivity identification using non-lexical and thread specific features. They conducted this research by treating the subjectivity orientation of threads as a binary classification task. Threads which ask for subjective options, viewpoints and evaluations are considered as subjective, whereas threads which seek for factual information are considered as non-subjective. Table 2.1 and Table 2.2 show examples of a subjective thread and a non-subjective thread respectively.

2http://www.linuxquestions.org: a Linux forum where Linux newbies can ask questions and Linux experts can offer advice. The Software subforum is for software related issues.
3http://lists.debian.org/completeindex.html: the official mailing lists service provided by the Debian operation system project. The debian-amd64 list discusses issues about porting Debian to AMD x86-64 architecture, and the debian-apache list discusses the maintenance of the Apache HTTP server and related packages in Debian.
4Because what constitutes “quality bands/artists” can be subjective, this is probably not a good non-subjective example. However, because it is used in the original paper, we present it as it is.
Chapter 2: Literature Review

Do you still tip after bad service?

<table>
<thead>
<tr>
<th>Do you still tip after bad service?</th>
</tr>
</thead>
<tbody>
<tr>
<td>After looking for restaurants options for my trip to NY in September (Trip Advisor, Menu Pages, etc) I can see that most of the complains are on bad service received in the restaurant, but not the food quality. So, as I am not used much to tip in restaurants as you do in the States (since I am not American and not living there), what do you do when you suffer bad service in a restaurant, even if the food is good? Do you still tip 15%? Thanks in advance for your comments on this.</td>
</tr>
<tr>
<td>I would tip 10%. Actually, these days tipping 20% is more the norm for good service. If you get bad service, depending on how bad it is either 1) leave a smaller tip; or 2) don’t leave a tip at all. However, in all my years of dining out, there have been only two occasions where we had such bad service that we didn’t leave a tip. Needless to say, we didn’t return to those places either!</td>
</tr>
<tr>
<td>I lower the tip if the service is not good. ( and once lowered it to under a $$) HOWEVER, if you are not tipping because of bad service it is important to let someone in the restaurant know WHY you are not tipping!</td>
</tr>
</tbody>
</table>

Table 2.1: An example subjective thread (Biyani et al. 2012)

Biyani et al. (2012) experimented with four kinds of features, namely structural features, dialogue act features, subjectivity-lexicon-based features, and sentiment features (features which capture the sentiment and emotion of a thread, as proposed by Somasundaran et al. (2007)), over two datasets from TripAdvisor New York forum5 and Ubuntu Forums6. The datasets, which contain 609 threads from TripAdvisor and 621 from Ubuntu, are sampled from a dataset created by Bhatia and Mitra (2010) which will be introduced at the end of Section 2.3.1. The machine

5http://www.tripadvisor.com/ShowForum-g60763-i5-New_York_City_New_York.html: TripAdvisor is a web service to provide holiday reviews, photos and travel advice for hotels and accommodation. Its New York forum is a discussion board for people to discuss New York city travels.

6http://ubuntuforums.org: the official forum for the Ubuntu distribution of the Linux operating system; its primary function is for Ubuntu support.
Chapter 2: Literature Review

Rock, folk, pop, blues music in December

| Hi guys,                        |
|                               |
| We’re coming over to catch Oasis at Madison Square Gardens in December. What other quality bands/artists are playing from 6 December onwards? |
| Cheers                        |
| Have a look at www.pollstar.com and, in the weeks leading up to your trip, at www.timeout.com/newyork/ |

Table 2.2: An example non-subjective thread \(\text{[Biyani et al. 2012]}\)

Learning algorithms tested include multinomial naïve Bayes (NB), SVMs, logistic regression, bagging, boosting and decision trees. \(\text{[Biyani et al. (2012)]}\)’s experimental results show that structural features are the best indicator of thread subjectivity. The structural features used are:

- **InitPostLength**: total number of words in the initial post.
- **ThreadLength**: total number of words in the thread.
- **NumPost**: total number of posts in the thread.
- **NumUser**: total number of users in the thread.
- **AvgPostAuthor**: average number of posts by a user in the thread.
- **AvgLengthPost**: average number of words in a post in the thread.

\(\text{Wang et al. (2010b)}\) devised an ontology of “problems sources” and “solution types” to capture the targets and sources of the problems described in threads, as well as the types of the solution presented in threads, respectively. The class set proposed is made up of two orthogonal basic class sets (Problem and Solution).
and a miscellaneous class set (Misc), as shown in Table 2.3, based on the analysis over a dataset (contains 327 threads) from Operating Systems, Software, Hardware and Web Development subforums of the CNET forums. This dataset, which will be described in detail in Section 3.2, was also used by Kim et al. (2010b) for thread discourse structure parsing, as we will discuss later in this chapter under “Thread Linking Structure and Semantics Recovery” in Section 2.2.3.

A given thread is labelled either with one class label from each of the two basic class sets (e.g. OS-Install), or alternatively one class label from the Misc class set (e.g. Spam). Wang et al. (2010b) conducted a series of preliminary experiments over the CNET dataset with SVMs and naive Bayes learners. The experimental results show that while SVMs perform better over basic Solution classes, naive Bayes is superior for predicting basic Problem classes. Wang et al. (2010b) did not conduct in-depth error analysis over these results, and adopted a classification composition approach, which uses these two learners over the two basic class sets separately and composes the predictions into an overall thread classification. This was found to lead to the best results.

Feng et al. (2006a) proposed to predict the topic focus of graduate course online discussions using a pseudo-relevance feedback based classification method and domain ontologies induced from the course textbooks. The classification method used is a Rocchio-style algorithm, which constructs a “profile” vector for each class, as a weighted average of positive and negative training instances. When given a test instance, its term vector is compared to the profile vectors based on cosine similarity to find the best match. If the classification cosine similarity exceeds a predefined
Chapter 2: Literature Review

<table>
<thead>
<tr>
<th>Class Category</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Problem</strong></td>
<td></td>
</tr>
<tr>
<td><em>OS</em></td>
<td>Operating system</td>
</tr>
<tr>
<td><em>Hardware</em></td>
<td>Core computer components, including core external components (e.g. a keyboard)</td>
</tr>
<tr>
<td><em>Software</em></td>
<td>Software-related issues, including applications and programming tools</td>
</tr>
<tr>
<td><em>Media</em></td>
<td>Hardware which is either a non-standard external component or peripheral device</td>
</tr>
<tr>
<td><em>Network</em></td>
<td>Network issues (e.g. connection speed, and installing a physical network)</td>
</tr>
<tr>
<td><em>Programming</em></td>
<td>Coding and design issues relating to programming</td>
</tr>
<tr>
<td><strong>Solution</strong></td>
<td></td>
</tr>
<tr>
<td><em>Documentation</em></td>
<td>How to use a certain function, select a computer/component, or perform a task</td>
</tr>
<tr>
<td><em>Install</em></td>
<td>How to install a component</td>
</tr>
<tr>
<td><em>Search</em></td>
<td>Search for a particular component (e.g. a software package)</td>
</tr>
<tr>
<td><em>Support</em></td>
<td>How to fix a problem with a computer or component</td>
</tr>
<tr>
<td><strong>Misc</strong></td>
<td></td>
</tr>
<tr>
<td><em>Other</em></td>
<td>Troubleshooting-related, but the problem source is not included in the Problem set</td>
</tr>
<tr>
<td><em>Spam</em></td>
<td>The thread is not troubleshooting-related</td>
</tr>
</tbody>
</table>

Table 2.3: The components of the thread class set by [Wang et al. (2010b)]

threshold, this test instance is added into the relevant positive instances and the corresponding profile vector is updated:

\[
C^{i+1} = \alpha C^i + \beta \frac{\sum_{d \in R} d}{||R||} - \gamma \frac{\sum_{d \in NR} d}{||NR||}
\]

where \( C^{i+1} \) and \( C^i \) are profile vectors of a class at time \( i + 1 \) and \( i \) respectively, \( d \) represents the term vector of a new test instance, \( R \) includes all positive instances at time \( i \), and \( NR \) includes all negative instances at time \( i \). The weighting parameters of \( \alpha, \beta \) and \( \gamma \) were set to 1 in [Feng et al. (2006a)]'s experiments.

To construct an initial profile vectors for each class, [Feng et al. (2006a)] created a domain ontology by processing the course textbook’s table of contents and in-
index. Specifically, different sections in the table of contents represent potential class candidates, the terms in the index are used to construct vectors for respective class candidates according to which sections the terms refer to, and a weight for each term in each vector is also assigned based on the occurrence of the term references in respective sections. Finally, only classes which are relevant to the task (i.e. classes in the gold standard data) are selected.

To predict the topic class of discussion threads, Feng et al. (2006a) proposed several strategies: treat a whole thread as a document and classify its topic class directly, or classify the topic class of each component post first and then use the following ways to aggregate post-level predictions to a thread-level one:

**Count:** The class topic which appears the most.

**Sum:** The class topic which has the largest accumulated classification score.

**Nearest:** The class topic which has the largest classification score.

**First:** The class topic of the initial post in the thread.

**Last:** The class topic of the last post in the thread.

**LCS:** The class topic which results in the longest continual post sequence in the thread.

Feng et al. (2006a) conducted a series of experiments over a set of graduate advanced operating systems course discussion, which consists of 206 posts spanning 50 threads. They found that when running their classifier over all threads, all prediction strategies derived similar results with a slightly better result from the “sum”
aggregation of post-level predictions. However, when they only consider threads which are not coherent (i.e. the proportion of predicted majority topic class in the thread is small), the post-level prediction aggregation approach significantly outperforms the strategy which treats the thread as a document, and the “sum” aggregation of post-level predictions derives the best result.

### 2.2.2 Dialogue Act Tagging

Dialogue Acts (DAs), which were proposed based on the original work on speech acts (Austin 1962; Searle 1969), represent the meaning of discourse units at the level of illocutionary force, “the particular dimension of meaning along which statement, directive and question are distinguished” (Huddleston 1988:p. 129). The identification of DAs in human interactions, either in audio format (transcribed) or text format, is often regarded as an important step to recover the discourse structure in the interaction. Depending on the research focus, a discourse unit could be an utterance in a conversation (e.g. Stolcke et al. (2000)), a sentence (e.g. Lampert et al. (2008)), a paragraph (e.g. Cong et al. (2008)), or a whole message consisting of several paragraphs (e.g. Cohen et al. (2004)). In the context of dialogue act analysis over forum data, a basic discourse unit can be a sentence, a paragraph or a post. Some forum research treats a dialogue act as a relation between two discourse units (e.g. Kim et al. (2010b)), while other research uses dialogue acts to annotate each individual discourse unit (e.g. Bhatia et al. (2012)). The dialogue act sets used by research in the field of forums are often devised based on tasks and use cases, and there is no commonly adopted dialogue act set to the best our knowledge. In this
section, literature on dialogue act research in general will be reviewed briefly, with a focus on papers which are loosely related to our research. Because this thesis targets forums, we do not go into the details of each research paper presented in this section. However, we will come back to give a detailed review of publications on forum-related dialogue act analysis in Section 2.2.3 under “Joint Thread Linking Structure and Semantics Recovery”, and in Section 2.2.4 under “Post Dialogue Act Tagging”.

Dialogue acts have been applied to the analysis of mediums of communication including conversational speech (Stolcke et al. 2000; Shriberg et al. 2004; Murray et al. 2006), email (Cohen et al. 2004; Carvalho and Cohen 2005; Lampert et al. 2008), instant messaging (Ivanovic 2008; Kim et al. 2010a), edited documents (Soricut and Marcu 2003; Sagae 2009) and online forums (Xi et al. 2004; Weinberger and Fischer 2006; Wang et al. 2007; Fortuna et al. 2007; Cong et al. 2008; Kim et al. 2010b; Bhatia et al. 2012). It has been argued that automatic DA identification can help a range of applications such as meeting summarisation (Murray et al. 2006), email summarisation (Rambow et al. 2004), speech recognition (Stolcke et al. 2000), or human social intention detection (Jurafsky et al. 2009; Ranganath et al. 2009). They can also be useful for support sharing—i.e. “the ability for users to look over the logs of past support interactions to determine whether there is a documented, immediately-applicable solution to their current problem” (Kim et al. 2010b:p. 192).

Some corpora have been widely used for DA research, including the Switchboard corpus (Godfrey et al. 1992), International Computer Science Institute (ICSI)
Meeting Recorder Dialog Act (MRDA) corpus (Shriberg et al. 2004), Verbmobil corpus (Wahlster 1993), and Enron email corpus (Klimt and Yang 2004). While the dialogue act set design often varies depending on the corpus and the research focus, there are a number of key DA tagsets with strong currency. For example, the Dialogue Act Markup in Several Layers (DAMSL) annotation scheme was proposed by Core and Allen (1997) to provide a domain-independent annotation framework to annotate communicative acts in task-oriented dialogues. DAMSL allows multiple labels in three layers (Forward Communicative Functions, Backward Communicative Functions, and Utterance Features) to be assigned to an utterance. Each layer contains several independent categories consisting of more detailed but high-level labels, which could be applicable in different domains. DAMSL was then adapted and used over different corpora, such as the Switchboard corpus (Stolcke et al. 2000), and instant messaging data (Ivanovic 2008; Kim et al. 2010a). Another example is the SWBD-DAMSL tag set (Jurafsky et al. 1997), which was originally designed for the Switchboard corpus, based on the DAMSL annotation scheme. It was then adapted and used on other corpora, such as the ICSI MRDA corpus (Shriberg et al. 2004), as well as forum and email data (Jeong et al. 2009). Recently, the ISO 24617-2 standard was devised for dialogue act annotation (Bunt et al. 2010; Bunt et al. 2012). This standard focuses on the communicative function and semantic content of a dialogue act, and supports annotating utterances with more than one functional tag.

As mentioned earlier, the DA tags can be identified at different levels, such as sentence, utterance, paragraph and message levels. While heuristic methods can
reliably segment a discourse into sentences and paragraphs, automatic utterance segmentation is an open research question, especially in research on conversational speech. This is because the utterance segments used in conversational speech can comprise more than one sentence, and also span more than one turn. To address this issue, automatic segmentation has been explored. For example, Mast et al. (1996) combined a polygram language model (i.e. a mixture of \( n \)-grams with varying size of \( n \)) with multi-layer perceptrons to recognise the segment boundaries of a transcript of the Verbmobil corpus, and achieved an accuracy of 92.5% with prosodic features. Stolcke and Shriberg (1996) deployed an \( n \)-gram language model to discover the hidden segment boundaries of the Switchboard corpus transcript, and explored a range of lexical features such as function ‘cue’ words, POS labels, and turn markers. They achieved 85% recall and 70% precision on the task. Shriberg et al. (2000) used decision tree and hidden Markov modelling to segment the Broadcast News and Switchboard corpora, and experimented with a range of lexical and prosodic features. They found that prosodic features are especially useful, and achieved an accuracy of 96.8% on the Broadcast News transcript and 96% on the Switchboard corpus by using a probabilistic combination of prosodic and lexical features. Finke et al. (1998) tried both neural network and Markov model approaches for automatic segmentation, and found that both methods could produce competitive results. Ang et al. (2005) used a decision tree model (prosodic features), a language model (lexical features), and an HMM-based combination of the two to automatically segment the ICSI meeting corpus. They found that while prosodic features (in the form of a decision tree) produced superior results, the combined approach
introduced further improvement. It is also possible to integrate DA segmentation and classification (Warnke et al. 1997; Finke et al. 1998), e.g. using the A* algorithm to search over the space of possible segmentations and speech act assignments. It should be noted that, first, the quantitative results indicated should only be interpreted as broadly indicative, and are not directly comparable. Second, a general finding is that prosodic features can substantially improve segmentation accuracy. Third, Markov-like models (e.g. polygrams/n-gram language models) with lower orders (e.g. unigram and bigram) often lead to very good results.

Regarding DA classification, a range of methods have been used, including n-gram language models (Mast et al. 1996; Finke et al. 1998; Stolcke et al. 2000), semantic classification trees (Mast et al. 1996), integrated DA segmentation and classification (Warnke et al. 1997; Finke et al. 1998), decision trees (Stolcke et al. 2000; Cohen et al. 2004), maximum entropy models (Stolcke et al. 2000; Ang et al. 2005; Carvalho and Cohen 2005; Kim et al. 2010b), neural networks (Stolcke et al. 2000; Cohen et al. 2004), SVMs (Cohen et al. 2004; Lampert et al. 2010; Fortuna et al. 2007; Kim et al. 2010b; Kim et al. 2010a), rule induction methods (Shrestha and McKeown 2004; Cong et al. 2008), CRFs (Ding et al. 2008; Kim et al. 2010b; Kim et al. 2010a), and Naive Bayes (Kim et al. 2010a). While some research has reported similar results over different methods (Stolcke et al. 2000), most research has found that Markov-like models (e.g. polygrams/n-gram language models and CRFs) with lower orders (e.g. unigram and bigram) often lead to very good results. It is interesting to note that although most research has focused on supervised methods involving only DAs, some research has approached the task via unsupervised
Chapter 2: Literature Review

(Cong et al. 2008), semi-supervised (Jeong et al. 2009) and integrated methods (Warnke et al. 1997; Finke et al. 1998). For example, Warnke et al. (1997) presented an integrated approach for the segmentation and classification of DAs. In their approach, the predictions of a multi-layer perceptron classifier on dialogue act boundaries were fed into a polygram language model, which was used for the joint segmentation and classification of dialogue acts. Compared with a two-step approach similar to Mast et al. (1996), this joint approach improved the DA classification accuracy slightly. Finke et al. (1998) also proposed an integrated approach by using Markov-based models for both segmentation and classification of DAs, without empirically quantifying the improvement over the two-step approach.

Different features have also been explored in DA classification, including lexical features (Stolcke et al. 2000; Ang et al. 2005; Cohen et al. 2004; Carvalho and Cohen 2005; Ding et al. 2008; Lampert et al. 2010; Kim et al. 2010b) such as bag of words, structural features (Shrestha and McKeown 2004; Ding et al. 2008; Kim et al. 2010b) such as relative post position, context features (Ang et al. 2005; Carvalho and Cohen 2005; Kim et al. 2010b) such as DA predictions of preceding posts, semantic features (Ding et al. 2008; Lampert et al. 2010; Kim et al. 2010b) such as similarity scores, prosodic/acoustic features (Mast et al. 1996; Warnke et al. 1997; Finke et al. 1998; Stolcke et al. 2000; Ang et al. 2005) such as acoustic scores from speech recognizers, and graph-based features (Fortuna et al. 2007; Jeong et al. 2009) such as reply-to networks in forum threads. In general, lexical features are less effective than other features. Over conversational speech, prosodic features often lead to better results. It should also be noted that although context features
were considered explicitly in some research, Markov-based methods are often able to capture these features inherently.

2.2.3 Thread Structure Analysis

This section will review published research on thread structure analysis, which targets information about the interactions between posts in a thread. Specifically, we will review literature on the recovery of thread linking structure (i.e. recover the reply-to links between posts), the joint recovery of thread linking structure and semantics (i.e. recover the reply-to links between posts, and the semantics of the discussions), question-answer pair extraction (i.e. extract question and answer posts, paragraphs or sentences) and thread partitioning (i.e. divide a discussion thread into coherent dialogue segments).

Thread Linking Structure Recovery

A more general term for linking structure recovery in the literature is discourse disentanglement, which is the task of dividing a conversation thread or document thread into a set of distinct sub-discourses or discourse units. Disentanglement can be used loosely to refer to the task of recovering the linking relations among the discourse units of a targeted discourse. A discourse unit could be a clause in a document (Wolf and Gibson 2005), a post in a forum thread (Wang et al. 2008a; Seo et al. 2009; Lin et al. 2009; Kim et al. 2010b), a message in an email thread (Seo et al. 2009), an utterance in a conversation dialogue (Grosz and Sidner 1986; Rosé et al. 1995; Lemon et al. 2002; Elsner and Charniak 2008), or a comment in
Chapter 2: Literature Review

a news-article discussion thread (Schuth et al. 2007). The disentangled discourse is sometimes assumed to take the form of a tree structure (Grosz and Sidner 1986; Lemon et al. 2002; Seo et al. 2009; Lin et al. 2009), an acyclic graph structure (Rosé et al. 1995; Schuth et al. 2007; Elsner and Charniak 2008; Wang et al. 2008a; Kim et al. 2010b), or a more general cyclic chain graph (Wolf and Gibson 2005).

The task of recovering thread linking structure, which is represented as inter-post reply-to links, has been widely explored in the research community. The motivation behind this research is that the linking structure may help tasks such as discussion summarisation (Wang and Rosé 2010). It has also been demonstrated that thread linking structure can improve forum thread and post retrieval (Xi et al. 2004; Duan and Zhai 2011; Seo et al. 2009; Bhatia and Mitra 2010; Wang et al. 2011a).

However, this linking structure is not available in many forums. For example, forums which are based on phpBB7 or vBulletin8 either do not support threaded view of the linking structure, or do not provide it by default.

Methods used to recover thread linking structure vary greatly. For example, Wang et al. (2008a) and Wang and Rosé (2010) adopted unsupervised ranking methods to recover the inter-posts links. Seo et al. (2009) treated the task as a supervised ranking problem, by considering each child post as a query and its parent post as the most relevant document for the query. Wang et al. (2011a) used a supervised structured classification approach. The datasets used for experiments also differ among the reviewed research. For example, Wang et al. (2008a) used data from educational discussion forums, and Wang and Rosé (2010) adopted political

---

7 https://www.phpbb.com/
8 https://www.vbulletin.com/
discussion threads from Usenet. Seo et al. (2009) experimented with a variety of datasets from online game discussions, travel forums, and research mailing lists. A detailed review of these research papers is presented below. It should be noted that there is also research that goes beyond the thread linking structure, to recover both thread links and discussion semantics, which will be reviewed in the next section. This section focuses exclusively on thread linking structure recovery.

Wang et al. (2008a) used semantic similarity between posts within a thread to recover the thread linking structure, assigning a link when the similarity score is larger than a pre-defined threshold. The semantic similarity between two posts is calculated using cosine similarity of the two posts’ term vectors, weighted by TF-IDF (Okapi BM25: Robertson et al. 1994). Additionally, three penalisation criteria were used to adjust the similarity scores:

**Fixed window size:** two posts will never be linked if their relative distance in the chronological time frame is larger than a predefined value.

**Dynamic window size:** two posts will never be linked if their relative distance in the chronological time frame is larger than a certain value. This value is dynamically changed based on the length of the corresponding thread where these two posts reside.

**Time distance:** the similarity score between two posts is adjusted using the reciprocal of their relative distance in the chronological time frame.

Based on the proposed methods, Wang et al. (2008a) conducted experiments over a dataset from the LegSim corpus⁹ which consists of 28 threads. The LegSim

---

⁹http://www.legsim.org/
corpus was provided by John D. Wilkerson at University of Washington in Seattle, and consists of discussion threads from an educational multi-player web-based teaching tool. The experimental results show that the “fixed window size” penalisation method has the best overall performance. In addition, they found that while the performance of different algorithms often decreases when the thread length increases, the performance usually increases as the average post length increases. Moreover, simply linking each post to its immediately preceding post was also found to be very effective.

Wang and Rosé (2010) treated the thread linking structure recovery task (“initiation-response pair identification” in their paper) as a pairwise ranking problem and constructed a dataset based on threads extracted from the alt.politics.usa Usenet discussion forum. The dataset consists of 100,028 instances, where each instance is made up of three posts: a reply post, an actual parent post and a foil parent post. An actual parent post is identified by analysing the metadata and quotation texts, and a foil parent post is a random post from the same thread as the reply post, excluding the parent, the grandparent, siblings or children of the reply post. The task is to rank the actual parent post higher than the foil parent post in each instance. Wang and Rosé (2010) experimented with 3 similarity functions:

**Cosine similarity**: cosine similarity of the two posts’ word vectors.

**Latent semantic analysis (LSA) average similarity**: first use latent semantic analysis (Forsythe et al. 1977:p. 201–206) to reduce the feature space to a $k$ dimensional concept space, then calculate the cosine similarity of the two posts’ LSA vectors in that space. A post’s LSA vector is constructed by averaging
across the LSA representations for each of its words.

**LSA Cartesian similarity**: first calculate the cosine similarity for all word pairs, as represented as LSA vectors, in the Cartesian product of the two posts, and then take the mean of all the cosine values.

They found that the LSA Cartesian similarity model, which does not deemphasise unusual words as the LSA average similarity does, achieves the best performance overall. Further analysis also shows that when lexical cohesion is high between parent and reply posts, cosine similarity performs better, while when the lexical cohesion is low between initiation and response posts, LSA Cartesian similarity is superior.

Seo *et al.* (2009) treated the thread linking structure recovery task as a ranking task, by considering each child post as a query and its parent post as the most relevant document for the query. They used a ranking SVM algorithm (Joachims 2002) and experimented with various intrinsic and extrinsic features, which include:

**Intrinsic features**: IDF-weighted cosine similarity between the term vectors of a parent candidate post and a child post. The term vectors can be constructed using different strategies:

- only consider the original content of a parent and the original content of a child post, excluding quoted texts.
- only consider the original content of a parent and a quoted text in a child post.
- consider the full texts.
• construct the term vectors with unigrams or $n$-grams.\textsuperscript{10}

**Extrinsic features:** features which describe the context:

**Location prior:** the estimated probability of a parent post’s location index, given the child post’s location index. This information is derived from the training data.

**Time gap:** normalised posting time gap between the child and parent posts, normalising over the time gap between the child post and the initial post of the thread.

**Same author:** whether the child and parent posts are from the same author.

**Author reference:** whether the name or ID of the author for a parent post is mentioned in the child post.

**Inferred turn-taking:** when posts $A$, $B$ and $C$ are posted in order in a thread, if $B$ mentions $A$’s author, and $A$ and $C$ are from the same author, there is inferred turn-taking between $B$ and $C$.

Seo et al. (2009) experimented with different feature combinations over three sets of datasets:

• WOW dataset from the general discussion forum of the World of Warcraft,\textsuperscript{11} which contains 60 threads.

\textsuperscript{10} The specific choice of $n$ is not detailed in the original paper.

\textsuperscript{11} http://forums.worldofwarcraft.com/board.html?forumId=10001: the official forum of the World of Warcraft, a massively multiplayer online role-playing game.
• Cancun dataset from the Cancun forum of TripAdvisor\textsuperscript{12} which contains 60 threads.

• W3C dataset from the mailing list of the World Wide Web Consortium\textsuperscript{13} which is a subset of the dataset used for the email discussion search task of the TREC (Text REtrieval Conference) 2006 enterprise track (Soboroff et al. 2006). The W3C dataset contains 1635 threads.

For the WOW and W3C datasets, all the proposed features were experimented with, while for Cancun dataset the features relating to quoted texts were excluded because the TripAdvisor forum does not systemically support quoted texts. Seo et al. (2009)'s experimental results show that the most effective intrinsic feature is the second one in the above “intrinsic features” list with unigrams (i.e. cosine similarity between the unigram term vectors of the original content of a parent candidate and a quoted text in a child post) and the most effective extrinsic features are “location prior” and “time gap”. In addition, they also found that the authorship-based features, which include “same author”, “author reference” and “inferred turn-taking”, are only effective in formal discussions where authors’ names are generally known and cited, such as in the W3C dataset. Following Seo et al. (2009)’s work, Wang et al. (2011a) explored the thread linking structure recovery task by using a thread CRF\textsuperscript{14} model and show that the proposed thread CRF is superior to the

\textsuperscript{12}http://www.tripadvisor.in/ShowForum-g150807-i18.html: a discussion board on TripAdvisor for people to discuss Cancun travels.

\textsuperscript{13}http://lists.w3c.org: official mailing lists of the World Wide Web Consortium (W3C).

\textsuperscript{14}In this chapter, when introducing the machine learning algorithms used in a paper for the first time and CRFs are used, we use the term “CRFs” if linear-chain CRFs are assumed, otherwise we explicitly describe the type of the CRFs used. The basics of linear-chain CRFs will be explained in Section 3.6.
Joint Thread Linking Structure and Semantics Recovery

There is also research that goes beyond the thread linking structure, to include semantics, such as the type of the linking relation (Kim et al. 2010b; Kim et al. 2006) and post-level topic modelling (Lin et al. 2009). It should be noted that Kim et al. (2006) only proposed a set of dialogue acts, annotated a dataset, and did some preliminary analysis over the dialogue act distribution over the dataset. They did not do any further experiments.

Kim et al. (2010b) proposed to model the thread discourse structure of forum threads, by capturing both the reply-to links between posts and the type (i.e. dialogue act) of each link. Their work is directly related to our research. Specifically, they proposed a dialogue act set to capture different types of links between posts, which is made up of 5 super-categories: Question, Answer, Resolution, Reproduction and Other. The Question category contains 4 sub-classes: question, add, confirmation and correction. Similarly, the Answer category contains 5 sub-classes: answer, add, confirmation, correction and objection. For example, the label Question-add signifies the Question superclass and add subclass, i.e. addition of extra information to a question. The full details of the dialogue act tagset are summarised in Table 2.4. Based on this tagset, Kim et al. (2010b) created a dataset by crawling sample threads from the CNET forums, and annotated both links and dialogue acts. This dataset is used in our research and described in detail in Section 3.2.

\[^{15}\text{The reasons for combing these two cases into one dialogue act are: (1) they serve the same purpose in the typology; and (2) they are minority classes, and splitting them will create even smaller class categories.}\]
Table 2.4: The dialogue act (DA) tagset proposed by Kim et al. (2010b)

Kim et al. (2010b) mainly evaluated post dialogue act and post link classification as separate tasks. Additionally, they carried out preliminary experiments to investigate the interaction between dialogue act classification and link classification, by classifying links using dialogue act information. Four types of features are considered:

Lexical features: both unigram and bigram tokens.

Structural features:

- whether the post is from the thread initiator.
- the post’s relative position in the thread.
Chapter 2: Literature Review

Post context features:

- the predicted dialogue act of the immediately preceding post, and whether this preceding post’s author authored the current post.

- the predicted dialogue act and the relative location of the most recent post from the same author of the current post.

- the predicted dialogue acts of all the preceding posts.

Semantic features:

- the relative location of the post which has the most similar title to the current post.

- the relative location of the post which has the most similar content to the current post.

- the number of question marks, exclamation marks and URLs in the current post.

- the distribution of dialogue acts of the author of the current post, based on this author’s posts in the training data.

Kim et al. (2010b) used three machine learners: a maximum entropy model, an SVM-HMM (Joachims et al. 2009), and a CRF model. For dialogue act classification, they found that the structural features are the most effective features, and that CRFs are the most effective learner. The best result was achieved by applying the CRF over the combination of structural features and the title similarity feature from
semantic features. For link classification, the post context features are represented in two different ways: link-based post context or dialogue act-based post context (i.e. classifying links by using dialogue act information). Kim et al. (2010b)’s experimental results show that when using dialogue act-based post contexts, the best result is achieved by applying the CRF over the combination of structural and post context features. In Section 4.2, we carry out in-depth analysis of the task of joint classification of post link and dialogue act tags over the same dataset, to generate full thread discourse structures.

Kim et al. (2006) proposed a set of dialogue acts, which is summarised in Table 2.5, based on the theory of speech acts of Austin (1962) and Searle (1969) for annotating pairs of posts in threads. They annotated a dataset, whose size is not specified, from the student online discussions in an undergraduate Operating System course, and did some preliminary analysis over the dialogue act distribution over the dataset. No experimental results are presented in the paper.

Lin et al. (2009) proposed a topic modelling based approach, which is called SMSS, to simultaneously capture both the semantics and structure of threaded discussions. SMSS tries to minimise a loss function which is made up of four parts:

- minimise the cost of reconstructing posts from topics.
- a L1 sparse regulariser over the post topic distribution, based on the assumption that an individual post usually focuses on a limited number of topics.
- minimise the cost of approximating the current post’s topic distribution from the topic distributions of previous posts within the same thread.
Table 2.5: Dialogue acts proposed by [Kim et al. (2006)]

<table>
<thead>
<tr>
<th>Dialogue Act</th>
<th>Description</th>
<th>Positivity</th>
</tr>
</thead>
<tbody>
<tr>
<td>Question</td>
<td>Question on specific problems</td>
<td>neutral</td>
</tr>
<tr>
<td>Announcement</td>
<td>Command or announcement</td>
<td>neutral</td>
</tr>
<tr>
<td>Complex Answer</td>
<td>An answer requiring a full description of procedures, reasons, etc.</td>
<td>neutral</td>
</tr>
<tr>
<td>Simple Answer</td>
<td>An answer with short phrase or words, e.g. factoid, Yes/No</td>
<td>neutral</td>
</tr>
<tr>
<td>Suggest</td>
<td>Give advices/suggestions for some problems/solutions</td>
<td>neutral</td>
</tr>
<tr>
<td>Elaborate</td>
<td>Elaborate on a previous arguments or questions</td>
<td>neutral</td>
</tr>
<tr>
<td>Correct</td>
<td>Correct a wrong answer/solution with a new one</td>
<td>negative</td>
</tr>
<tr>
<td>Object</td>
<td>Object to some argument/suggestions/solutions</td>
<td>negative</td>
</tr>
<tr>
<td>Criticize</td>
<td>Criticize an argument</td>
<td>negative</td>
</tr>
<tr>
<td>Support</td>
<td>Support others’ arguments/solutions</td>
<td>positive</td>
</tr>
<tr>
<td>Acknowledge</td>
<td>Confirm or acknowledgement</td>
<td>positive</td>
</tr>
<tr>
<td>Compliment</td>
<td>Praise an argument or suggestion</td>
<td>positive</td>
</tr>
</tbody>
</table>

- a L1 sparse regulariser over the topic distribution associated with the previous posts, based on the assumption that a post often only comments on one or two previous posts.

The first two components capture the semantics of the discussion, while the last two components reflects the structural characteristics of the thread.

To evaluate the proposed SMSS model, [Lin et al. (2009)] conducted three tasks: thread linking structure recovery, junk post identification, and expert finding, as detailed below:

**Thread linking structure recovery**: a post is linked to the most similar preceding post. The similarity measure is the linear interpolation of term similarity, topic similarity, and structural similarity. The term similarity is the cosine similarity between two posts’ term vectors. The topic similarity is the cosine similarity
between two posts’ topic vectors, whose weights are optimised in the first two components of SMSS. The structural similarity is the cosine similarity between two posts’ structure vectors, represented by their respective preceding posts, with weights optimised in the last two components of SMSS.

**Junk post identification:** the basic idea is based on the assumption that junk posts usually have different topics compared to non-junk posts, and that the content is similar across threads. To capture these assumptions, an additional background topic, which is represented by common words in terms of post-level document frequency, is added into the SMSS model, and is fixed during the optimisation process. Posts which are close to this background topic are classified as junk posts.

**Expert finding:** a user-level reply-to graph is first constructed, where each node corresponds to a user and the weight of a direct edge is the number of posts replied to the destination user. Then the HITS (Hyperlink Induced Topic Search, \*Kleinberg (1999)\*) algorithm is used to rank the users.

For experiments, \*Lin *et al.* (2009)\* used a dataset from Apple Discussions\[16\] which contains 4486 threads spanning 80,008 posts, and a dataset from Slashdot\[17\] which consists of 1154 threads spanning 203,210 posts. Their experiments over thread linking structure recovery show that structure similarities are more effective in the task.

---

\[16\] https://discussions.apple.com: the official forum hosted by Apple Inc. for discussion of matters related to Apple products.

\[17\] http://www.slashdot.org/: a technology-related news website, where each news story submitted is followed by a threaded discussion among users.
Question-answer Pair Extraction

Rather than trying to parse the general structure of threads, another line of research has focused on identifying and extracting specific relations between posts. The most explored task in this direction is to extract question-answer pairs, where each question-answer pair consists of a question and a corresponding (best) answer from the same discussion thread. This task may help enrich the knowledge base of (community-based) question answering services (Cong et al. 2008; Ding et al. 2008; Yang et al. 2009b; Hong and Davison 2009), improve information/answer access over forum threads (Cong et al. 2008), improve thread summarisation (Ding et al. 2008), enhance search (Hong and Davison 2009), or find forum experts (Bouguessa et al. 2008; Jurczyk and Agichtein 2007; Zhang et al. 2007). It has also been demonstrated that question-answer pair extraction can help augment the knowledge base of chatbots (Feng et al. 2006b; Huang et al. 2007). Researchers have approached this task from different angles. For example, while some research has tried to address question extraction and answer identification at the same time (Cong et al. 2008; Hong and Davison 2009), other research has focused on answer identification only (Feng et al. 2006c; Catherine et al. 2012; Catherine et al. 2013; Huang et al. 2007) or extracting both question contexts (i.e. sentences which provide background information and impose constraints regarding a question) and answers (Ding et al. 2008; Yang et al. 2009b) by assuming that questions are pre-identified. The focus of granularity of the questions and answers also differs in the literature. For question detection, some research has tried to identify question sen-

---

18 A chatbot is an agent which can conduct natural language based conversations with users in a certain domain or on a certain topic.
tences \cite{cong2008} while other research has tried to identify question posts \cite{hong2009}. Regarding answer extraction, some research has aimed to extract answer sentences \cite{ding2008, yang2009}, or paragraphs \cite{cong2008}, while other research has tried to identify answer posts \cite{feng2006, hong2009, catherine2012, catherine2013, huang2007}.

Regarding the methodologies used for question classification, supervised document categorisation approaches are often adopted \cite{cong2008, hong2009}. With respect to answer classification, supervised methods are also widely used \cite{cong2008, yang2009, hong2009, catherine2012, huang2007}, and the thread structure is usually exploited to facilitate the classification. Additional explored approaches include structured supervised classification \cite{ding2008}, unsupervised ranking-based algorithms \cite{cong2008, feng2006}, and semi-supervised co-training algorithms \cite{catherine2013}. The datasets used in this research have covered domains such as travel forums \cite{cong2008, ding2008, yang2009}, photography discussions \cite{hong2009}, technical discussions \cite{hong2009, catherine2012, catherine2013}, undergraduate computer science course discussions \cite{feng2006}, and movie discussions \cite{huang2007}. It should be noted that among the research reviewed in this section, \cite{catherine2012}’s work concentrates on the evaluation of different features proposed by previous research in the context of answer post identification.

\cite{cong2008} addressed the task of question-answer pair extraction from
forums. The task is performed in two steps: (1) question sentence detection, and (2) answer paragraph identification. Specifically, to detect question sentences, they first used labelled sequential patterns (LSPs) to extract patterns from both question and non-question sentences. Only the part-of-speech (POS) tags of non-keywords, and keywords including 5W1H (i.e. who, what, when, where, why and how), modal words and a few other selected keywords (e.g. wonder and any) are considered. Then the discovered LSP patterns are used as binary features to train RIPPER (Cohen 1995) to detect question sentences.

For answer paragraph detection, the task is treated as a ranking problem where all the candidate answer paragraphs for a given question are ranked. An unsupervised graph-based propagation model was devised to either propagate ranking scores from existing methods or generate features for supervised classification approaches. In this model, for each question, an answer graph is built based on all the candidate answers where each answer is a node in the graph. KL-divergence \((KL)\) is used to decide whether there is a directed edge from answer node \(a_1\) to \(a_2\): if \(1/(1 + KL(a_1 \parallel a_2))\) (the definition of \(KL(a_1 \parallel a_2)\) is given below) is larger than a threshold \(\theta\), an edge will be formed from \(a_1\) to \(a_2\). The weight of each edge is a linear interpolation of the KL-divergence score, the distance of the destination answer node from the question, and the authority score of the author of the destination answer node (estimated using the number of the author’s replying posts and the number of threads initiated by the author in the corresponding forum). After all the weights are calculated, they are normalised using a PageRank based approach (Brin and Page 1998). Based on this graph, three unsupervised answer ranking
approaches and two supervised methods are proposed. For the unsupervised approaches, before using the graph to propagate the scores, initial ranking scores are generated by using the following three methods:

**Cosine similarity:** the ranking score of each candidate answer \( a \) for a question \( q \) is defined as the cosine similarity between the term vector of \( a \) and the term vector of \( q \), weighted by TFIDF.

**Query likelihood:** ("query likelihood language model" in their paper) the ranking score of each candidate answer \( a \) for a question \( q \) is the probability of generating \( q \) from the term distribution of \( a \).

**KL-divergence:** ("KL-divergence language model" in their paper) the ranking score of each candidate answer \( a \) for a question \( q \) is the KL-divergence between the term distribution of \( a \) \((M_a)\) and the term distribution of \( q \) \((M_q)\): 

\[
KL(a \parallel q) = KL(M_a \parallel M_q).
\]

The supervised methods use SVMs and treat each question and candidate answer pair as an instance. Then the graph is either used to propagate the initial ranking scores from the SVMs or to generate features for the SVMs.

For question identification, Cong et al. (2008) used two datasets generated from 650 TripAdvisor Forum\(^{19}\) threads. For answer detection, five datasets were used, where three of them are generated from 750 TripAdvisor forum threads, and two of them are from LonelyPlanet forum\(^{20}\) and BootsnAll Network\(^{21}\) which contain

\(^{19}\)http://www.tripadvisor.com/ForumHome

\(^{20}\)http://www.lonelyplanet.com/thorntree/index.jspa: the official discussion forum of Lonely Planet, which allows independent travellers to exchange travel information, advice, hints and tips.

\(^{21}\)http://boards.bootsnall.com/eve/ubb.x: a forum for indie travellers to ask questions and
100 threads each. Experimental results show that, within the three unsupervised approaches, KL-divergence with graph propagation achieves the best performance, even better than a simple supervised method using only SVMs. At the same time, the supervised method which uses the graph to generate features for SVMs, also outperforms the method which uses SVMs directly.

Ding et al. (2008) extended Cong et al. (2008)’s work by focusing on question context and answer detection, both at the sentence-level. They experimented with linear-chain CRFs (Lafferty et al. 2001), skip-chain CRFs (Sutton and McCallum 2007; Galley 2006), and 2D CRFs (Zhu et al. 2005). The experimental results show that CRFs are superior to SVMs with a polynomial kernel and decision trees for context and answer detection, while skip-chain CRFs, which capture the dependency between contexts and answers, outperform linear-chain CRFs for answer detection. Additionally, 2D CRFs outperform linear-chain CRFs for both context and answer detection. Yang et al. (2009b) improved on Ding et al. (2008)’s models by proposing a more comprehensive and unified graphical representation, designing special inference algorithms by exploiting the structure of thread discussions, and deploying customised structural SVMs (Joachims et al. 2009) for a more flexible framework.

Hong and Davison (2009) treated the question-answer post pair extraction task as two classification tasks, namely question post classification and answer post classification, with emphasis on feature analysis. For question post classification, they focused on the initial post of each thread, and tried to identify whether it is a question post. For answer post classification, they tried to identify the best answer post share information.
in each thread, in which the initial post is a question. The features they used for question post detection include the number of question marks, the count of each 5W1H word, thread length in terms of the total number of posts, \( n \)-grams (1-5 grams), and authorship (the number of posts the user starts and the number of posts the user replies to). The features they used for answer post detection include post position, authorship, \( n \)-grams (1-5 grams), the count of each stop word, and the score from a query likelihood model. For experiments, they adopted SVMs for both tasks and tested over two datasets: 1000 threads (500 threads for question detection and 1000 threads for answer detection) from Photography On The Net\(^{22}\) (“DC dataset”) and 1072 threads (572 threads for question detection and 1072 threads for answer detection) from UbuntuForums\(^{23}\) (“Ubuntu dataset”). Experimental results from question detection show that \( n \)-gram features are the most effective single feature type, while combinations of simpler features can achieve comparable or better performance. For example, the combination of authorship, question mark counts, 5W1H word counts and thread length achieves the best result over the DC dataset and the second best over the Ubuntu dataset. For answer detection experiments, the results show that the combination of authorship and post position achieves the best results for both datasets. Additionally, Hong and Davison (2009) conducted answer ranking experiments by linearly combining scores from classifiers trained over each of: (1) the combination of post position and authorship features; (2) only the post position feature; and (3) only the authorship feature. Their method significantly outperformed the best method proposed by Cong et al. (2008).

\(^{22}\)http://photography-on-the.net/forum/: a discussion and photo sharing forum for all Canon digital camera enthusiasts.
\(^{23}\)http://ubuntuforums.org/
Feng et al. (2006c) proposed to use the weighted graph-based algorithm HITS (Kleinberg 1999) to detect one or more best answer posts in a discussion thread. To construct a graph for a thread, each post in the thread is treated as a node, and the weighted directed links are generated based on manually annotated dialogue act relationships (e.g. post $d_i$ is a “complex answer” to post $d_j$) between posts. The dialogue act tagset used, proposed by Kim et al. (2006), was described earlier in this section, and is shown in Table 2.5. After the HITS algorithm converges based on a predefined error rate, the best answer posts are the posts with the highest “authority” scores or “hub” scores. Three methods were used to generate the weighted links for the graph:

**Lexical similarity:** given a dialogue act $SA_{ij}(d_i \rightarrow d_j)$, a link is generated from post $d_i$ to post $d_j$. The weight of this link is the cosine similarity between the term vectors of $d_i$ and $d_j$.

**User authority:** each user’s authority is first measured as the proportion of his/her positive posts (e.g. “acknowledge”, “compliment” and “support”), where the positivity is defined by certain types of dialogue acts which are also manually annotated. Then, for a given dialogue act $SA_{ij}(d_i \rightarrow d_j)$, a self-pointing link is generated for post $d_i$ with a weight of its user’s authority score.

**Dialogue act strength:** each dialogue act’s strength is first calculated based on the weighted average of user authority scores of the corresponding dialogue act over all users, and the positivity (manually annotated as “positive”, “neutral” or “negative”) of the dialogue act. Then, for a given dialogue act $SA_{ij}(d_i \rightarrow d_j)$, a link with a weight equal to the strength of this dialogue act is generated. This
Chapter 2: Literature Review

link is self-referential for post $d_i$ if the dialogue act is “neutral”, otherwise it is linked from $d_i$ to $d_j$.

Feng et al. (2006c) experimented with different combinations of the above three link generation methods, over a set of undergraduate computer science course discussion threads which consists of 1307 posts from 314 threads. The experimental results show that the combination of user authority and dialogue act strength leads to the best result, which is slightly better than using dialogue act strength alone.

Catherine et al. (2013) approached the task of answer post detection in a semi-supervised manner. At its core, their method adopts the co-training methodology of Blum and Mitchell (1998). Specifically, the algorithm starts with a small amount of training instances and goes for $n$ iterations. At each iteration $i$, two classifiers are formed by training a SVM learner with an RBF kernel over two independent features sets. The two classifiers are then used to classify unlabelled instances, and the predictions with highest confidence are moved to the current set of labelled instances for training at iteration $i + 1$. The two feature sets used are:

**Structural features:** the ratings of the author and the post, as well as the relative post position.

**Pattern features:** sequential patterns which are mined using a modified PrefixSpan (Pei et al. 2001) algorithm.

Additionally, Catherine et al. (2013) presented a parallel co-training model for answer post identification, incorporating the joint identification of acknowledgement posts. A positive acknowledgement post from the author of the question suggests
that the problem is solved, while a negative one indicates that the proposed solutions do not work. In this model, two co-training tasks run in parallel: one for answer classification and one for acknowledgement identification, using the same learner, features and framework. At each iteration, predictions from acknowledgement classification at iteration \( i - 1 \) are used as features for answer classification at iteration \( i \). The acknowledgement features used are:

- whether the post has an acknowledgement reply.
- the number of posts between the post and its acknowledgement reply, in chronological order.
- the number of posts between the post and its last acknowledgement reply, in chronological order.

In their experiments, [Catherine et al. (2013)](https://discussions.apple.com) annotated answer and acknowledgement posts, by treating the initial post in a thread as a question, of 303 threads crawled from Apple Discussions, where 3 threads are used for training and 300 for testing. Their method significantly outperforms a supervised system which uses SVMs and is trained only over the 3 training threads, as well as the unsupervised model proposed by [Cong et al. (2008)](https://discussions.apple.com).

[Catherine et al. (2013)](https://discussions.apple.com)’s work is based on their earlier work ([Catherine et al. 2012](https://discussions.apple.com)), which conducted a detailed study on the contribution of different features in the task of answer post identification in technical discussion forum threads. They annotated answer posts, by treating the initial post in a thread as a question, of
about 600 threads (the exact number of threads used is not clear in the original paper) from Apple Discussions, and conducted various statistical analyses as well as classification experiments using SVMs. Their analyses and experimental results show that while features which measure the content similarity between a post and the question post have often been used in previous research, they are among the least effective features. They found the most effective features to be:

- whether the author of the current post is the author of the question post.
- author ratings.
- whether a post contains a hyperlink or not.
- the relative post position in the thread.
- whether the current post is replied to by the author of the question post.
- whether the current post belongs to the first $n$ posts of the thread.

They also found that the feature which indicates whether a post contains a hyperlink or not is the most important one.

Huang et al. (2007) proposed to extract high-quality threadtitle-reply pairs from online discussion forums, to enrich the knowledge base for chatbots. A high-quality threadtitle-reply pair consists of the title of a thread, and a reply post from the same thread which has the following characteristics:

- should be independent of other reply posts.

\[^{25}\text{https://discussions.apple.com/community/iphone}\]
• provides descriptive, informative and trustworthy content relative to the initial post.

• has high readability, neat short and concise expressive style, and clear structure.

• should have no “intemperate sentiment,” no obscene words and not contain words which are indicative of exclusive personal information (terms beginning with my, such as my wife and my child.).

• should be of appropriate length.

Huang et al. (2007) adopted a cascaded framework by firstly classifying reply posts which are relevant to the thread title, then filtering out posts which contain words from a keyword list which contains 33 obscenities, 62 personal information terms and 17 forum specific terms, as well as posts containing more than 50 words, and finally ranking the extracted threadtitle-reply pairs based on their content quality. For relevant post classification, SVMs are used with a range of structural and content features:

**Structural features:**

• whether the current post quotes the initial post.

• whether the current post quotes other non-initial posts.

• whether the current post is posted by the thread initiator.

• number of posts between the current post and the previous post by the same user.

26This is the term used in the paper; the meaning is unclear.
Content features:

- number of words.
- number of content words (i.e. excluding stop words).
- number and ratio of overlapping words between the current post and the thread title.
- number and ratio of overlapping content words between the current post and the thread title.
- number of domain words (i.e. words that are not in a commonly used lexicon) in the current post.
- whether the current post contains other user's forum nicknames.

For threadtitle-reply ranking, ranking SVMs are used with the following set of features:

- number of times the current post is quoted within the current thread.
- number of threads initiated by the current post’s user in the forum.
- number of threads initiated by the current post’s user that get no replies.
- number of replies in the threads which are initiated by the current post’s user.
- number of threads in the forum the current post’s user participates in.
- number of replies the current post’s user posts in threads that are initiated by other users.
average length of all the posts from the current post’s user.

- number of days the user has posted to Usenet.

- user’s total influence, quantified using the method of Matsumura et al. (2002).

- number of times the user’s posts are quoted in the current threads/the whole forum.

Huang et al. (2007) conducted various experiments over a dataset crawled from the Rotten Tomatoes Forum, which consists of 2995 posts from 53 threads. They found that for relevant post classification, the structural features are the most effective and adding content features only leads to a small improvement in precision (1.01% in their experiments) without affecting the recall. For threadtitle-reply ranking, they observe that their user-level features are effective, contradicting the findings of Xi et al. (2004)’s post-level retrieval experiments over newsgroup discussion threads, as described in Section 2.3.1. Huang et al. (2007) attribute this to the removal of irrelevant posts in the first step of their framework, which makes their user-level features more salient in ranking the remaining relevant posts. Additionally, they found that using features such as cosine similarity between a relevant post and the initial post does not improve ranking performance.

Thread Partition

Kim et al. (2005) proposed a method to divide a discussion thread into coherent dialogue segments, according to posts which partition the discussion into “new”,

http://www.rottentomatoes.com/vine/ a forum for movie and video game discussion.
“general” and “specific” topics. To identify posts leading to topic divergence, the proposed method processes the posts in a thread in chronological order and adopts a two-step identification approach:

- identify whether a post is on a new topic, relative to its immediately proceeding posts, which are identified in the same dialogue segment. This is done by calculating the cosine similarity between the term vector of the current post and the term vector of the qualified proceeding posts. If the similarity score is less than a threshold, this post is regarded to be on a new topic. When constructing the term vectors, terms in quoted texts are given different weights compared to terms in unquoted texts.

- if a post is identified to be on a similar topic in the first step, the system tries to determine whether this post is more specific or more general, relative to its immediately proceeding post. Given a post $D_2$ and its parent post $D_1$, and their overlap text $(C_{D_1, D_2})$ which is extracted based on quotations, the “specialisation” ($spec$) and “generalisation” ($gen$) scores for $D_2$ can estimated by:

$$spec(D_1, D_2) = 1 - similarity(D_1, C_{D_1, D_2})$$

$$gen(D_1, D_2) = 1 - similarity(D_2, C_{D_1, D_2})$$

After the $spec$ and $gen$ scores are calculated, the system first checks whether the $gen$ score is larger than a pre-determined threshold. If yes, post $D_2$ will be marked as a generalisation boundary. Otherwise, the system checks whether the $spec$ score is larger than a pre-determined threshold. If yes, post $D_2$ will be
marked as a specialisation boundary. Otherwise, $D_2$ is regarded as the same topic as $D_1$.

For experiments, Kim et al. (2005) constructed a dataset which consists of 20 discussion threads spanning 368 posts, from a movie message board. From the results, they observed that it is important to weight the terms in quoted text separately when constructing term vectors.

### 2.2.4 Post Attribute Analysis

This section will review research that has focused on post-level attribution analysis, such as post quality assessment (i.e. identify whether a post’s content is of high quality or not), post dialogue act tagging (i.e. classify post-level or sentence-level dialogue acts), and post viewpoint identification (i.e. identify personal view points shown in posts).

#### Post Quality Assessment

One of the most important directions in post-level analysis is automatically assessing post quality, to help users better access high-quality information and high-reliability users. The research has mainly focused on two task settings: binary classification into “good” and “bad” posts (Weimer et al. 2007; Weimer and Gurevych 2007; Lui and Baldwin 2009); and ternary classification into “high”, “medium”

---

28 [http://www.hundland.com/movieboard.mv](http://www.hundland.com/movieboard.mv); note that this message board is no longer available.

29 Based on a five star rating scale used in the source forum of the dataset, where posts with more than three stars are “good” and posts with less than three stars are “bad”.
Chapter 2: Literature Review

and “low” quality posts (Wanas et al. 2008). The task has largely been treated as a supervised document categorisation task, with the main focus on feature engineering. For example, Weimer et al. (2007) and Weimer and Gurevych (2007) mainly extracted features from individual posts, with the exception of similarity features where cosine similarity between the post unigram vector and the forum unigram vector was calculated. Wanas et al. (2008) mainly focused on features that involve the thread structure. Lui and Baldwin (2009) went further, by aggregating and using a combination of features at the user-level. In general, forum-specific features and user-level features have been found to be more effective. With respect to datasets used for experiments, Weimer et al. (2007), Weimer and Gurevych (2007) and Lui and Baldwin (2009) used similar datasets from Nabble.com, while Wanas et al. (2008) used a dataset from a technology discussion forum. The details of each research paper are reviewed below.

Weimer et al. (2007) used an SVM with an RBF kernel to classify posts into either “good” or “bad”, and experimented with five groups of features:

**Surface features:**

- the number of words in a post.
- the proportion of sentences ending with ? and !.
- the proportion of all-caps words.

**Lexical features:**

---

30 Based on a −1 to 5 rating scale used in the source forum of the dataset, where posts with the ratings −1, 0, 1 or 2 are “low”, posts with the rating 3 are “medium”, and posts with the ratings 4 or 5 are “high”.

the proportion of words which are spelled incorrectly.

- the proportion of swear words, where the swear words are compiled from WordNet and Wikipedia.

**Syntactic features:** the distribution of part-of-speech tags, as defined in the Penn Treebank tag set (Marcus et al. 1993).

**Forum specific features:**

- whether the post contains HTML.
- whether the post is from a mailing list.
- the proportion of characters which are in quoted texts.
- the number of URLs.
- the number of filesystem paths.

**Similarity features:** the cosine similarity between the post unigram vector and the forum unigram vector.

Weimer et al. (2007) experimented with 1532 posts from forums which are in the Software category of Nabble.com. The experimental results show that the most effective feature group is “forum specific features”, and the most effective individual features are “whether the post is from a mailing list” and “the proportion of characters which are in quotes”.

http://www.nabble.com/Software-f94.html: Nabble provides a service for users to build free, simple and embeddable forums, photo galleries, news, blogs and mailing lists. The Software category is no longer available.
Weimer and Gurevych (2007) extended Weimer et al. (2007)’s work by experimenting with more diverse datasets from Nabble.com. Specifically, they used three datasets: a dataset from the Software category which includes 1532 posts, a dataset from the Non-Software category which contains 1886 posts, and the combination of the two. Their experimental results show that the system works best over the dataset from the Software category.

Compared to Weimer and Gurevych (2007) and Weimer et al. (2007), Wanas et al. (2008) explored the problem using a finer assessment division, by classifying posts into three categories: “high”, “medium” and “low” quality. They also used a different set of more sophisticated features, many of which take into account the thread structure. Specifically, they extracted five groups of features:

**Relevance features:** features that capture the relevance of the post compared to its corresponding subforum and thread.

**Originality features:** features that capture the maximum word overlap between the current post and its preceding posts.

**Forum-specific features:** features that capture the number and size of quoted texts, as well as the number of replies.

**Surface features:** features that capture how fast the post is posted compared to its immediately preceding post, length of the post, and the usage of punctuation, emoticons and capitals.

**Posting component features:** features that capture the number and quality of URLs, and the number and uniqueness of the questions in the post.
Wanas et al. (2008) deployed an SVM learner with an RBF kernel, and experimented with a dataset which contains 20,008 posts from 14 subforums on the Slashdot online discussion forum. Their experimental results show that the “forum-specific features” are the most effective ones, while the “relevance features” and “posting component features” are the least effective ones.

Lui and Baldwin (2009) approached the post quality classification task by exploiting features proposed by relevant previous research: the ILIAD features of Baldwin et al. (2007), the WANAS features of Wanas et al. (2008), and the network features of Fortuna et al. (2007). The novelty of Lui and Baldwin (2009)’s work lies in aggregating and using these features at the user-level. Specifically, they experimented with the following features:

**BoW:** bag of words features at the post-level.

**ILIAD:** the features proposed by Baldwin et al. (2007) as described in Section 2.2.1, used at the post-level.

**WANAS:** the features used by Wanas et al. (2008), used at the post-level.

**ILIAD-User:** aggregated ILIAD features, which takes the mean of each feature value over all posts from a user.

**WANAS-User:** aggregated WANAS features, which takes the mean of each feature value over all posts from a user.

**PostAfter:** similar to the “reply-to author network” of Fortuna et al. (2007), as described later in this section.
Thread Participation: similar to the “thread participation author network” of Fortuna et al. (2007), as described later in this section.

Common Authors: identical to the “common authors thread network” of Fortuna et al. (2007), as described later in this section.

For experiments, Lui and Baldwin (2009) used three learners, namely an SVM with an RBF kernel, a nearest-prototype learner using skew divergence as the distance metric, and a maximum entropy learner, to classify posts into either “good” or “bad”. The dataset they use was based on the data used in Weimer and Gurevych (2007)’s research, which contains 4094 posts from Nabble.com. The experimental results show significant improvements when using the user-level features.

Post Dialogue Act Tagging

Various research has focused on post dialogue act classification to capture the role and purpose of each individual post or sentence in threaded discussions. This research can potentially help forum thread-level and post-level retrieval (Bhatia et al. 2012), discussion summarisation (Zhou and Hovy 2006), and better thread visualisation (Bhatia et al. 2012). It also has been demonstrated that post-level dialogue acts can help forum thread-level retrieval (Wang et al. 2013) and user profiling (Kim et al. 2006). Some research on post dialogue act tagging in forums tries to parse both the dialogue acts and the links among them, and is reviewed in Section 2.2.3, under “Joint Thread Linking Structure and Semantics Recovery”. This section covers research on post dialogue act tagging which does not consider the link structure, and includes one paper on sentence-level dialogue act tagging.
(Jeong et al. 2009) and one paper on post-level dialogue act tagging (Bhatia et al. 2012).

Jeong et al. (2009) proposed a semi-supervised sentence-level dialogue act tagging system, which tags email and forum data by leveraging out-of-domain labelled data. The proposed method uses a word-level subtree pattern mining method (Kudo and Matsumoto 2004) to generate subtree features as weak learners, and adopt bootstrapping and semi-supervised boosting (Bennett et al. 2002) to conduct semi-supervised learning. Specifically, for each instance (sentence) $d_i$, a forest of different word-level trees is first created, such as a sequence of words and a dependency parse tree. The forest of instance $d_i$ is denoted as $d_i$. The subtree features are then extracted from the forest of each instance. Each subtree feature $t$ is defined as a weak learner:

$$f(y_i, t, d_i) \triangleq \begin{cases} +y_i & t \text{ is a subtree of } d_i \\ -y_i & \text{otherwise} \end{cases}$$

where $y_i \in \{+1, -1\}$, which is a binary class label for instance $d_i$. A one-vs.-all strategy is used to tackle the multi-class problem. For the bootstrapping and semi-supervised boosting, tree edit distance (Shasha and Zhang 1990) is used to select instances from unlabelled data which are similar to labelled data. The selected instances, along with their automatically predicted labels, are added to the training data in each iteration.

For experiments, Jeong et al. (2009) defined 12 dialogue acts, which are shown in Table 2.6, based on the Meeting Recorder Dialog Acts (MRDAs) of Dhillon et al. (2004). As labelled data, they used the Switchboard-DAMSL (SWBD) (Jurafsky et al. 1997) and MRDA (Shriberg et al. 2004) datasets, where the former contains
## Dialogue Act Description

<table>
<thead>
<tr>
<th>Dialogue Act</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>accept response</td>
</tr>
<tr>
<td>AA</td>
<td>acknowledge and appreciate</td>
</tr>
<tr>
<td>AC</td>
<td>action motivator</td>
</tr>
<tr>
<td>P</td>
<td>polite mechanism</td>
</tr>
<tr>
<td>QH</td>
<td>rhetorical question</td>
</tr>
<tr>
<td>QO</td>
<td>open-ended question</td>
</tr>
<tr>
<td>QR</td>
<td>or/or-clause question</td>
</tr>
<tr>
<td>QW</td>
<td>wh-question</td>
</tr>
<tr>
<td>QY</td>
<td>yes-no question</td>
</tr>
<tr>
<td>R</td>
<td>reject response</td>
</tr>
<tr>
<td>S</td>
<td>statement</td>
</tr>
<tr>
<td>U</td>
<td>uncertain response</td>
</tr>
</tbody>
</table>

Table 2.6: The 12 dialogue act labels defined by (Jeong et al. 2009)

1,155 telephone conversations from around US and the latter consists of 75 meeting (2 are excluded from their experiments) from the International Computer Science Institute (ICSI) at UC Berkeley. The unlabelled data used for training includes 22,391 emails from the Enron dataset[^34] (discussion_threads, all documents and calendar folder), and 11,602 threads (spanning 55,743 posts) from the TripAdvisor travel forums[^35] (Beijing, Shanghai and Hongkong forums). For testing, Jeong et al. (2009) manually annotated 40 email threads from the BC3 corpus[^36] and 100 threads from the TripAdvisor travel forums. The experimental results show strong performance of the semi-supervised systems, especially over the forum data, where the best result from the semi-supervised systems is close to the result from the supervised method.

[^34]: [http://www.cs.cmu.edu/~enron/](http://www.cs.cmu.edu/~enron/): this dataset contains email conversations between employees (mostly senior management) of the Enron Corporation.
Bhatia et al. (2012) investigated the task of post dialogue act classification with a focus on feature exploration. The dialogue act set they use is adopted from the Mailing List and Forums (MLAF) Track of Forum for Information Retrieval Evaluation (FIRE) 2010, and contains 8 dialogue acts:

**Question**: a question which initiates discussions in the thread.

**Repeat question** a question which is already asked in a previous post of the same thread.

**Clarification**: clarifying questions which aim to gather more details about a previously asked question.

**Further details**: more details about a previously asked question.

**Solution**: a solution proposed to address a previously asked question.

**Positive feedback**: feedback which indicates that a previously proposed solution works.

**Negative feedback**: feedback which indicates that a previously proposed solution does not work.

**Junk**: the post does not contain any useful information.

Four sets of features are proposed for experiments:

**Content based features**:

37 An additional dialogue act “junk” is added.
38 http://www.isical.ac.in/~fire/2010/task-guideline.html
39 http://www.isical.ac.in/~fire/2010/index.html
• whether the post quotes a previous post.

• cosine similarity between the post and the thread title/first post/entire thread, as three features.

• whether the post contains question marks, 5W1H words, or keywords such as same and similar, as three binary features.

Structural features:

• the absolute and relative positions of the post in the thread, as two features.

• the total number of words/unique words in the post, as two features

• the total number of words/unique words in the post after removing stop words and stemming, as two features.

User features:

• the total number of posts from the post’s user.

• whether the post’s user is the thread initiator.

• the authority score (Bhatia and Mitra 2010) of the post’s user.

Sentiment based features:

• whether the post contains one of the following (each represented as a binary feature): thank, exclamation mark, did not, or does not.

• sentiment score of the post calculated using SentiStrength (Thelwall et al. 2010).
Bhatia et al. (2012) experimented with SVMs, naive Bayes (NB), decision trees, a multi-layer perceptron and a maximum entropy learner over a set of 100 threads from Ubuntu Forums, and found that maximum entropy achieves the best results. They observed that the content based features are the most effective and the sentiment based features are, while quite novel, the least effective.

Post View Point Identification

There has also been research on more domain-dependent tasks such as “agree vs. disagree vs. insult” post identification in political forums (Fortuna et al. 2007), “question vs. answer” post identification in technical forums (Fortuna et al. 2007), and “support vs. not support” viewpoint discovery over controversial discussions (Qiu and Jiang 2013). The two papers reviewed in this section explore quite different tasks in different domains.

Fortuna et al. (2007) explored the task of post type classification over Usenet newsgroup posts from two different domains: political discussion and debates, and technical question answering. For the political domain, they used posts from two newsgroups: alt.politics.immigration (733 posts used) and talk.politics.guns (298 posts used), and tried to predict whether a post is of type “agree” (the post agrees with the point of view of its parent post), “disagree” (the post disagrees with the point of view of its parent post) or “insult” (the post insults the author of its parent post). For the technical domain, they used posts from another two newsgroups: microsoft.public.internetexplorer.general (177 posts used) and microsoft.public.windowsxp.general (253 posts used), and tried to classify whether a post

---

40http://ubuntuforums.org
is of type “question” or “answer”. For both tasks, Fortuna et al. (2007) utilised a SVM learner with a linear kernel, and experimented with various structural and content features derived from three author networks and two thread networks. In an author network, the nodes represent authors and the edges between authors are formed based on three different criteria for three different author networks:

**Reply-to network:** if author A has replied to at least one of author B’s posts, there will be an edge from A to B.

**Thread participation network:** if author A and author B have participated in the same thread more than 4 times, there will be an edge between them.

**Text similarity network:** if the centroid keyword vector of user A’s posts is similar to the centroid keyword vector of user B’s posts (cosine similarity $\geq 0.3$), there will be an edge between them.

Similarly, in a thread network, the nodes represent threads and the edges between threads are formed based on two different criteria for two different thread networks:

**Common authors network:** if thread A and thread B have more than 2 authors in common, there will be an edge between them.

**Text similarity network:** if the centroid keyword vector of thread A’s posts is similar to the centroid keyword vector of thread B’s posts (cosine similarity $\geq 0.3$), there will be an edge between them.

Fortuna et al. (2007)'s experimental results show that while the author network features help both tasks, the thread network features are only effective for the first task (i.e. “agree”, “disagree” and “insult” classification in the political domain).
Qiu and Jiang (2013) explored the task of post viewpoint discovery (e.g. “support” vs. “not support” against the issue of “tax cuts” in one or more threads) in an unsupervised fashion. They proposed a generative latent variable model, based on three assumptions:

**Viewpoint-based topic distribution:** different viewpoints often focus on different topics. Therefore, each viewpoint can be captured by its own topic distribution.

**User identity:** the same user often has the same viewpoint on an issue across all their posts. Therefore, each user can have a unique viewpoint distribution over a variety of issues.

**User interaction:** users with the same viewpoints tend to interact with each other in a positive way, while users with different viewpoints tend to interact in a negative way. The polarity of the interactions was predicted using subjectivity lexicons together with heuristics.

To evaluate the proposed model extrinsically, Qiu and Jiang (2013) conducted two experiments: post clustering based on viewpoint, and user clustering based on viewpoint. Experimental results show that “user interaction” features, which are novel to this research, are very effective.

### 2.3 Forum-related Tasks

The main forum-related tasks that have been researched are forum information retrieval (i.e. given a query, find and rank the most relevant posts or threads) and
forum thread summarisation (i.e. summarise the content of a thread). The following two subsections will review literature from these two research fields.

2.3.1 Information Retrieval over Forum Data

The task of information retrieval (IR) over forums has been explored at two different granularities: post-level (i.e. each post is treated as a document: 

- Xi et al. (2004), Feng et al. (2006b), Seo et al. (2009), Duan and Zhai (2011), Wang et al. (2011a)) and thread-level (i.e. each thread is treated as a document: 

- Seo et al. (2009), Bhatia and Mitra (2010), Elsas and Carbonell (2009)). For post-level IR, most research (Xi et al. 2004, Duan and Zhai 2011, Seo et al. 2009, Wang et al. 2011a) has found that using the rely-to linking structure, as described under “Thread Linking Structure Recovery” in Section 2.2.3 of the threads is helpful. It has also been shown that when ranking a particular post according to a query, using information from other posts in the same thread can help (Feng et al. 2006b, Seo et al. 2009, Wang et al. 2011a, Duan and Zhai 2011). For thread-level IR, explicitly modelling the thread structure, such as individual posts and reply-to links between posts, can improve retrieval effectiveness significantly (Seo et al. 2009, Bhatia and Mitra 2010).

With respect to the methodologies used for forum IR, ranking algorithms are commonly used. Earlier research has explored ways to train supervised ranking functions for ranking (Xi et al. 2004), or adopted unsupervised semantic similarity measures for ranking (Feng et al. 2006b). Later research has mainly adopted language models, which are often smoothed based on thread structure (Duan and Zhai...
2011; Seo et al. 2009; Bhatia and Mitra 2010; Elsas and Carbonell 2009), with some research (Bhatia and Mitra 2010) also adopting thread priors. It should be noted that, among the research reviewed in this section, Elsas and Carbonell (2009)’s work mainly focuses on evaluating language model based methods proposed by previous research. As for datasets, different research papers often use their own datasets, which include technical discussions (Xi et al. 2004; Duan and Zhai 2011; Bhatia and Mitra 2010), undergraduate course discussions (Feng et al. 2006b), online game discussions (Seo et al. 2009), travel forums (Seo et al. 2009; Bhatia and Mitra 2010), research mailing lists (Seo et al. 2009), and news/rumors/reports discussions (Elsas and Carbonell 2009). The details of these research papers are presented below.

Xi et al. (2004) explored the task of post-level retrieval over newsgroup discussion threads from the microsoft.public.* sub-hierarchy of Usenet. They proposed to use a combination of features derived from the linking structure and users of the posts, and adopt linear regression and SVMs with a linear kernel to learn the weights for each feature. The features they extracted include:

**Linking structure based content features:** first extract different contexts of the target post, and then use three ranking functions, namely Okapi BM25 (Robertson et al. 1994), binary score, and term frequency score, to derive three scores for each context. The contexts considered are:

- the current post (both with and without quoted texts).
- the title of the current post.
- the initial post of the current post.
• direct parent of the current post.

• all ancestor posts of the current post.

• all posts except for the initial post.

• all the direct children posts of the current post.

• all the descendant posts of the current post.

**Linking structure based non-content features:**

• whether the post is the initial post.

• the post position.

• the number of direct children.

• the number of descendants.

• the greatest depth of all descendants.

• the number of leaf posts in the descendants.

**User-level features:** (based on Fiore et al. (2002)’s work.)

• number of posts from the user in a fixed period of time.

• number of reply posts from the user in a period of time.

• number of response posts the user gets.

• average length of the user’s posts.

• number of days the user has posted to Usenet.

• number of threads the user has participated in.
• number of threads the user has initiated.

• number of the user’s posts which do not have replies.

• number of newsgroups the user is active in.

Xi et al. (2004) conducted experiments over a dataset which consists of 343 queries and 5552 (query, post) pairs with binary relevance judgments. They found that linear regression performs better and is more robust than SVMs. They also observed that the linking structure based features boost retrieval effectiveness. Additionally, while the user-level features are quite novel, they do not play a significant role in the experiments.

Duan and Zhai (2011) explored the forum post retrieval task by using language models smoothed based on thread structure. Four different presentations of the thread structure are used to smooth their language models:

• using all posts from the same thread.

• using the initial post from the same thread.

• using all posts, which are posted earlier than the target post, in the same thread.

• using all posts on the reply path from the target post to the initial post in the same thread.

To weight the above smoothing components, where each smoothing component is made up of multiple posts, three different weighting methods were considered:

Equal weight: all posts have the same weight.
**Inverse structural distance:** the inverse of the number of posts between the target post and the smoothing post according to the respective representations of thread structure.

**Contextual similarity:** the cosine similarity between term vectors of the target post and a smoothing post.

Duan and Zhai (2011) tested two smoothing schemes with different combinations of the above representations and weighting methods, over a dataset from the Computer Help forum of CNET, which consists of 29,413 threads spanning 135,752 posts from 25,413 users. The experimental results show that for both smoothing schemes, the best results are achieved using the “reply path” presentation with both “inverse structural distance” and “contextual similarity” weighting methods, demonstrating the effectiveness of using reply-to thread linking structure for post-level retrieval. The authors also observed that when smoothing their language models with just the initial post, fairly good performance can be achieved.

Feng et al. (2006b) conducted information retrieval research in the context of building a discussion-bot for answering student questions in an undergraduate operating systems course discussion forum. They used two sources for extracting answers: a set of supplementary course documents and threaded course discussions from past semesters. The basic retrieval units for the course documents are semantically-related segments (tiles) of the documents, where the segmentation is done using TextTiling (Hearst 1994). The basic retrieval units for the thread data are posts. Given a question, cosine similarity between document term vectors

41 http://forums.cnet.com/computer-help-forum/
(weighted by TF-IDF) is first used to find the most similar unit among all the data, where a unit can be a document tile or thread post. Then, if the retrieved unit is a document tile, the tile text with a reference link to the original whole document is returned. If the retrieved unit is a thread post, a rule-based method, which uses manually annotated post-level dialogue acts, is used to find the best answer post in the corresponding thread. The dialogue acts are defined based on the work of Winograd (1987), and are very similar to the dialogue acts proposed by Kim et al. (2006), as shown in Table 2.5.

There have also been shared tasks on post-level information retrieval in the TREC 2005 enterprise track (Craswell et al. 2005), in the form of two email search tasks were explored over a mailing list dataset from W3C. The first task requires participants to search relevant email messages for a set of queries, while the second task requires participants to also identify whether a relevant email message contains “a pros and cons statement”\footnote{http://lists.w3.org}. In this shared task, most participants exploited thread structure and quoted material (Craswell et al. 2005). The email search task of the TREC 2006 enterprise track (Soboroff et al. 2006) used the same W3C dataset of Craswell et al. (2005) with a different approach to relevance judgements. For a query, a email message can be: (1) not relevant; (2) relevant but does not contain a pro/con argument; (3) relevant and contains a con argument; (4) relevant and contains a pro argument; or (5) relevant and contains both pro and con arguments.

Seo et al. (2009) investigated forum thread-level and post-level retrieval using thread linking structure, based on the idea that a thread is made up of different self-
contained contexts. Four types of contexts are considered including posts, pairs, dialogues and threads, as shown in Figure 2.1.

Specifically, Seo et al. (2009) used the language modelling approach to IR, which is based on the idea that if the document model is more likely to generate a query (i.e. $P(Q|D)$ is larger), this document is a better match with the query. The query likelihood $P(Q|D)$ is calculated by assuming term independence:

$$P(Q|D) = \prod_{q \in Q} (q|D) = \prod_{q \in Q} ((1 - \lambda)P_{ML}(q|D) + \lambda P_{ML}(q|C))$$

where $Q$ represents a query, $D$ represents a document, $q$ is a word in $Q$, $C$ is the whole document collection, $\lambda$ is a smoothing parameter, and $P_{ML}$ signifies a probability based on maximum likelihood estimation.

For thread-level retrieval, Seo et al. (2009) consider a thread to be a document (i.e. the global representation) and incorporate local contexts (i.e. posts, pairs or dialogues). A thread's ranking score is then determined by its global representation ($GR$) and its local contexts. The ranking score for the $GR$ of a thread $T_i$ regarding
query $Q$, is:

$$
\Phi_{GR}(Q, T_i) = P(Q|T_i)
$$

where $P(Q|T_i)$ is the query likelihood score of query $Q$ for thread $T_i$.

To compute the ranking score for a thread’s local contexts (i.e. posts, threads or dialogues), pseudo-cluster selection (PCS: Seo and Croft (2008)) is used. The basic retrieval element for this method is the concatenation of posts from targeted local contexts of threads, in terms of a post, a pair of posts, or a dialogue of posts.

Given a query $Q$, first, the top-$N$ targeted local contexts are retrieved. Then these $N$ local contexts are grouped into pseudo-clusters, where all the local contexts in the same pseudo-cluster belong to the same thread. That is, each pseudo-cluster represents a thread. Finally, a ranking of pseudo-clusters (i.e. threads) for the query $Q$ is calculated according to a geometric mean of scores of the top $k$ local contexts in each pseudo-cluster:

$$
\Phi_{PCS}(Q, T_i) = \left( \prod_{j=1}^{k} P(Q|L_{ij}) \right)^{1/k}
$$

where $P(Q|L_{ij})$ is the query likelihood score of the local context $L_{ij}$, which is the $j$th local context in thread $T_i$.

If a pseudo-cluster of a thread contains less than $k$ local contexts, the following formula is used:

$$
L_{min} = \arg\min_{L_n\in L} P(Q|L_n)
$$

$$
\Phi_{PCS}(Q, T_i) = \begin{cases} 
(P(Q|L_{min})^{k-m_i} \prod_{j=1}^{m_i} P(Q|L_{ij}))^{1/k} & \text{if } m_i < k \\
(\prod_{j=1}^{k} P(Q|L_{ij}))^{1/k} & \text{if } m_i \geq k
\end{cases}
$$
where $L$ represents all the targeted local contexts retrieved from all threads for query $Q$, and $m_i$ is the number of local contexts from thread $T_i$.

After the scores for the global representation and local context are calculated, a weighted-product of both can be derived:

$$
\Phi_{Product}(Q, T_i) = \Phi_{PCS}(Q, T_i)^{(1-\pi)} \cdot \Phi_{GR}(Q, T_i)^{\pi}
$$

where $\pi$ is a weighting parameter.

For post-level retrieval, Seo et al. (2009) introduced thread context smoothing and local context smoothing. The former is a two-stage smoothing by considering the thread $T$ which contains the target post $D$:

$$
P(q|D) = (1 - \lambda_1)P_{ML}(q|D) + \lambda_1((1 - \lambda_2)P_{ML}(q|T) + \lambda_2P_{ML}(q|C))
$$

Local context smoothing considers both the thread $T$ as well as local contexts, including pairs and dialogues, which contain the target post:

$$
P_z(q|D) = (1 - \lambda_1)P_{ML}(q|D) + \lambda_1((1 - \lambda_2)P_{ML}(q|X_z) + \lambda_2((1 - \lambda_3)P_{ML}(q|T) + \lambda_3P_{ML}(q|C)))
$$

$$
P(q|D) = \left(\prod_{z=1}^{Z} P_z(q|D)\right)^{1/Z}
$$

where $Z$ represents all the pair contexts or dialogue contexts which contain post $D$ (a post can be included in more than one pair or dialogue context).

For retrieval experiments, Seo et al. (2009) used the WOW, Cancun and W3C datasets, which were introduced in Section 2.2.3. The WOW and Cancun datasets were used for thread retrieval experiments, and the W3C dataset was used for post retrieval experiments. The thread retrieval experimental results show that the weighted products of thread contexts and local contexts outperform solely thread
contexts or local contexts. For both the WOW and Cancun datasets, the best results are achieved by using the combination of thread contexts and dialogue contexts. The post retrieval results also show that when using thread structure based local contexts (i.e. pairs and dialogues), retrieval effectiveness improves significantly. *Wang et al.* (2011a) confirm *Seo et al.* (2009)’s finding that dialogue contexts can improve retrieval effectiveness, by conducting similar experiments over a dataset from CNET forums, which was created by *Duan and Zhai* (2011), as described earlier in this section.

*Bhatia and Mitra* (2010) explored forum thread-level retrieval by using language model based inference networks to combine different structural units of threads, as well as query-independent priors. They divided a forum thread into three structural units: the thread title, the initial post and all the reply posts. The following formula is proposed to calculate ranking scores for a candidate thread $T$ given a query $Q$:

$$P(T|Q) = P(T) \prod_{i=1}^{n} \left\{ \sum_{j=1}^{m} \alpha_j P(q_i|S_{jT}) \right\}$$

where $P(T)$ is a query-independent prior for $T$, $\alpha_j$ is the weight for the structural unit $j$ of $T$ (i.e. $S_{jT}$), and $P(q_i|S_{jT})$ captures the probability of $S_{jT}$ generating the query term $q_i$. This probability is estimated using a language model with Dirichlet smoothing:

$$P(q|S_{jT}) = \frac{f_{q,jT} + \mu f_{q,JC}}{|jT| + \mu}$$

where $f_{q,jT}$ represents the frequency of the query term $q$ in the structural unit $j$ of thread $T$, $f_{q,JC}$ is the accumulated frequency of the query term $q$ in the structural units $j$ of all threads in the collection $C$, $|jT|$ is the term count of the structural unit $j$ of thread $T$, $|j|$ is the total term count of the structural units $j$ of all the threads in
the collection $C$, and $\mu$ is the Dirichlet smoothing parameter, which is set to 2000 following the empirical findings of Zhai and Lafferty (2001).

Bhatia and Mitra (2010) also experimented with three different thread priors (i.e. $P(T)$):

**Thread length:** the number of reply posts in the thread

**User authority:** the averaged user authorities of all posts in the thread. The user authority $A(u)$ of user $u$ is estimated as:

$$A(u) = \lambda \left\{ \frac{N_p(u) - N_{ip}(u)}{N_p} + \frac{1}{N_u} \right\}$$

where $N_p(u)$ is the number of posts by user $u$, $N_{ip}(u)$ is the number of initial posts by $u$, $N_p$ represents the number of posts in the collection, $N_u$ represents the number of users, and $\lambda$ is a normalisation factor.

**Linking information:** a forum graph is first built where each node represents a thread $T$, and a directed edge from $T'$ and $T$ indicates that a post $p$ in thread $T'$ contains a link to thread $T$. The weight of the edge $w(T', T)$ is the user authority score of post $p$'s user. Then a thread’s linking information prior is the sum of the weights of all incoming edges.

Bhatia and Mitra (2010) conducted two sets of experiments over two datasets from Ubuntu Forums which contains 113,277 threads with 4512 threads assigned relevance judgments, and the New York subforum of TripAdvisor which consists

---

44 No detailed explanation for this formula is provided in the original paper, and we present it as is.

45 [http://ubuntuforums.org](http://ubuntuforums.org)

46 [http://www.tripadvisor.com/ShowForum-g28953-i4-NewYork.html](http://www.tripadvisor.com/ShowForum-g28953-i4-NewYork.html)
of 83,072 threads with 4478 thread assigned relevance judgements. The first set of experiments did not consider the thread prior and mainly focused on testing the structural units. They found that a method which combines the three structural units with proper weights, which are learned through 5-fold cross-validation, outperforms methods which use each structural unit individually and a method which treats the whole thread as a document. The second set of experiments introduced the thread priors. The results show that when incorporating each of the three thread priors individually into the best system from the first set of experiments, the performance of the system improves, with the “linking information” prior being the most effective. However, when using the three priors at the same time, the system’s performance deteriorates.

Elsas and Carbonell (2009) investigated various language modelling based retrieval models for thread-level retrieval, by assuming term independence and a uniform thread prior. Their focus is mainly on assessing the effectiveness of using post-thread structure and post selection for forum thread retrieval. The models they explored are:

\[
P_{LD}(T|Q) = \frac{\text{rank}}{P(Q|T)}
\]

\[
P_{SD}(T|Q) = \frac{\text{rank}}{\sum_{D \in T} P(Q|D)P(D|T)}
\]

\[
P_{PCS}(T|Q) = \frac{\text{rank}}{\prod_{i=1}^{k} P(Q|D_i)^{\frac{1}{k}}}
\]

\[
P_{MAX}(T|Q) = \frac{\text{rank}}{\max_{D \in T} P(Q|D)}
\]

\[
P_{START}(T|Q) = \frac{\text{rank}}{P(Q|D_0)}
\]
where $T$ represents a thread, $Q$ represents a query, $D$ represents a post, the top-$k$ ranked posts are used for the PCS model (i.e. $P_{PCS}$), and $D_0$ is the initial post of the corresponding thread.

Elsa \& Carbonell (2009) applied the above models over a dataset from the MacRumors Forum,\footnote{\url{http://forums.macrumors.com}: a forum for people to discuss Apple Inc. related news, rumors and reports.} which includes more than 3 million posts from around 375,000 threads. The worst performance of $P_{LD}$ which considers the whole thread as a single document, and the superior performance of $P_{PCS}$ which uses posts as the base retrieval unit and selectively chooses among them, indicate the importance of post-thread structure and post selection.

### 2.3.2 Forum Thread Summarisation

Another line of research that relates to thread-level tasks is discussion summarisation. For example, Zhou \& Hovy (2005) investigated discussion summarisation over technical Internet Relay Chat (IRC) discussions and email archives. They used Kernel Traffic discussion data on GNUe development, in which a discussion thread may contain more than one subtopic, and in turn can lead to more than one mini-summary. To deal with this, Zhou \& Hovy (2005) first use TextTiling (Hearst 1997) to partition messages into multi-paragraph segments. They then cluster message segments in a thread using a hierarchical agglomerative clustering method from Ward Jr \& Hook (1963), with each cluster representing a subtopic in the target thread. The topic-initiating segment and corresponding response segments are then extracted from each cluster to form this cluster’s mini-summary.
The topic-initiating segment in a cluster is the one with the earliest timestamp, and the response segments are classified using supervised discriminative models. Zhou and Hovy (2005) experimented with a maximum entropy learner and an SVM learner, by using a number of structural and lexical features. The structural feature they used counts the number of messages between the initiating message segment and the responding message segment. The lexical features they used include the number of overlapping words/content words/tech words, and the ratio of overlapping words/content words. The experimental results show that the SVM learner is more effective. There has also been work on email summarisation (Lam et al. 2002; Newman and Blitzer 2003; Nenkova and Bagga 2003; Rambow et al. 2004; Wan and McKeown 2004), concentrating primarily on summarising and organising email archives by extracting overview sentences to help the users find the most useful email threads. Because this literature review focuses on forum research, the work on emails is only mentioned here for reference.

2.3.3 Knowledge Base Augmentation

Watanabe et al. (2004) proposed to build a knowledge base from threaded discussions from mailing lists, for a question-answering (QA) system to answer “how” type questions. To build the knowledge base, only threads starting with questions were selected and used. All the subsequent answer messages were classified into three types using reference relations and the sender’s email address:

**Direct answer:** direct answers to the original question.

**Questioner’s reply:** initial questioner’s replies to the direct answer messages.
**Other:** other types of messages.

After that, significant sentences in question messages, direct answer messages and questioner’s reply messages were extracted using surface clues. When the QA system accepts a question from a user, it compares the user question to the significant sentence in every question message in the knowledge base according to their content words and dependency parse trees, and identifies the most relevant discussion thread. The significance sentences from the question message, direct answer messages and questioner’s reply messages of the corresponding thread are finally displayed to the user. Watanabe *et al.* (2005) and Nishimura *et al.* (2005) further extended Watanabe *et al.* (2004)’s work by adding a component to give a confirmation label (“positive”, “negative” or “other”) to each threaded discussion based on whether the questions and answers are confirmed by questioner’s reply messages. This is done by exploring the surface clues in the significant sentences of questioner’s reply messages.

As reviewed at the end of Section 2.2.3, Huang *et al.* (2007) devised a method for extracting high quality threadtitle-reply pairs from online discussion forums to build the knowledge base for chatbots. Kim *et al.* (2005) proposed an algorithm to divide discussion threads into coherent dialogue segments, which was used to help blind students to access online course discussion thread records.
2.4 Forum Data Crawling

One of the most important steps of forum research is to automatically extract relevant information from one or more forum sites. However, this task is often not easy because of the large amount of duplicate and invalid\textsuperscript{48} pages, the complex layout designs which are different across forums and can change within a forum, as well as unrestricted user created posts that can contain HTML content as part of their post texts, which can mislead crawlers. Cai \textit{et al.} (2008) presented a forum crawler which is based on two observations over various forum sites: (1) similar types of pages have similar repetitive regions which appear at similar locations; and (2) links which appear repetitively at a similar location often have the same function. The proposed crawler first rebuilds the sitemap, which is a directed graph consisting of vertices representing different types of pages and edges representing linking relations between vertices, of a target forum site. The vertices are discovered by clustering a set of sample pages of the target forum site according to their repetitive regions (i.e. pages with similar repetitive region layout are likely to be of the same type), and the edges are constructed by analysing the URL pattern and the location of related links of the clustered sample pages. After the sitemap is reconstructed, it is used to find the optimal traversal path for crawling all the informative pages of the target forum. Based on Cai \textit{et al.} (2008)\textsuperscript{'s work, Wang \textit{et al.} (2008b) proposed an approach to find an appropriate traversal strategy to guide the forum crawling. The method first reconstructs a forum’s sitemap by using the strategy proposed by Cai \textit{et al.} (2008), then decides whether and how to follow every link \textsuperscript{48}Often caused by login failure issues.
in the sitemap by identifying “skeleton links” and “page-flipping links”. Skeleton links are principal links which point to valuable and informative pages, and page-flipping links are linkages between multiple pages of a long discussion thread. Yang et al. (2009a) proposed a generic template-independent approach to extract structured data, such as post title, post author, post time and post content, from web forum sites. The method uses not only page-level information, but also site-level knowledge automatically extracted from the reconstructed forum sitemap using the method proposed by Cai et al. (2008), which includes linkage information between pages and interrelationships of pages sharing similar layout design. Markov logic networks (Richardson and Domingos 2006) were then deployed to integrate these pieces of information by learning their weights. Experiments over 20 forums show promising extraction performance.

2.5 Lexical Chaining

Lexical chaining is a technique for identifying lists of related words (lexical chains) within a given discourse. The extracted lexical chains represent the discourse’s lexical cohesion, or “cohesion indicated by relations between words in the two units, such as use of an identical word, a synonym, or a hypernym” (Jurafsky and Martin 2009:pp. 685). In Section 4.3, we will describe our work on using a lexical chaining technique to recover forum thread linking structure. Because there is no previous lexical chaining research on forums or threading and most of the lexical chaining papers have been applied to unrelated tasks such as spell checking and word sense disambiguation, we will review the papers only briefly in this section,
and return to present details of the lexical chaining methodology in Section 4.3.

The first computational model for lexical chain extraction was proposed by Morris and Hirst (1991), based on the use of the hierarchical structure of Roget’s International Thesaurus, 4th Edition (1977). Because of the lack of a machine-readable copy of the thesaurus at the time, the lexical chains were built by hand. Research in lexical chaining has then been investigated by researchers from different research fields such as information retrieval, and natural language processing. It has been demonstrated that the textual knowledge provided by lexical chains can benefit many tasks, including text segmentation (Kozima 1993; Stokes et al. 2004), word sense disambiguation (Galley and McKeown 2003), text summarisation (Barzilay and Elhadad 1997), topic detection and tracking (Stokes and Carthy 2001), information retrieval (Stairmand 1997), malapropism detection (Hirst and St-Onge 1998), and question answering (Moldovan and Novischi 2002).

Many types of lexical chaining algorithms rely on examining lexicographical relationships (i.e. semantic measures) between words using domain-independent thesauri such as the Longmans Dictionary of Contemporary English (Kozima 1993), Roget’s Thesaurus (Jarmasz and Szpakowicz 2003), Macquarie Thesaurus (Marathe and Hirst 2010) or WordNet (Barzilay and Elhadad 1997; Hirst and St-Onge 1998; Moldovan and Novischi 2002; Galley and McKeown 2003). These lexical chaining algorithms are limited by the linguistic resources they depend upon, and often only apply to nouns.

Some lexical chaining algorithms also make use of statistical associations (i.e. distributional measures) between words which can be automatically generated from
domain-specific corpora. For example, Stokes et al. (2004)’s lexical chainer extracts significant noun bigrams based on the $G^2$ statistic (Pedersen 1996), and uses these statistical word associations to find related words in the preceding context, building on the work of Hirst and St-Onge (1998). Marathe and Hirst (2010) use distributional measures of conceptual distance, based on the methodology of Mohammad and Hirst (2006) to compute the relation between two words. This framework uses a very coarse-grained sense (concept or category) inventory from the Macquarie Thesaurus (Bernard 1986) to build a word-category co-occurrence matrix (WCCM), based on the British National Corpus (BNC) (Burnard 1995). Lin (1998a)’s measure of distributional similarity based on point-wise mutual information (PMI) is then used to measure the association between words.

2.6 Summary

In this chapter, we carried out a detailed review of literature on metadata recovery from forum-related data, and forum-related tasks. Our review shows that, to tackle different research problems relating to forums, researchers often adapt various existing methods from other areas or and design new methods. However, it has frequently been found that methods which capture forum-specific characteristics (e.g. inter-post links and quoted text) and features (e.g. time stamps and user information) usually lead to superior performance. We also briefly investigated research on forum data crawling and lexical chaining. In the next chapter, we present the main datasets and the most important software packages used in our research. Additionally, we also explain the core empirical methodologies behind these soft-
ware packages.
Chapter 3

Resources

3.1 Introduction

This chapter details the main datasets and the most important software packages used in our research. Additionally, we explain the core empirical methodologies that underpin these software packages. Specifically, there are three datasets involved:

- CNET dataset (Kim et al. 2010b)
  - Dataset content: 327 threads
  - Original annotation: discourse structure
  - Annotation novel to this research: none

- ILIAD dataset (Baldwin et al. 2007)
  - Dataset content: 250 threads
  - Original annotation: thread-level annotations of “Task orientation”, “Completeness” and “Solvedness”
Chapter 3: Resources

- Annotation novel to this research: discourse structure

  - Ancestry dataset [Elsas 2011]

    - Dataset content: 9,040,958 threads with 191 queries
    - Original annotation: pairwise preference relevance judgements [Carterette et al. 2008]
    - Annotation novel to this research: discourse structure of 50 sample threads

We detail the base datasets (with the “Original annotation”) in this chapter, and return to describe novel annotations to this research in Chapters 4, 5, and 6.

Regarding the adoption of these datasets in the research of others, the CNET dataset has been used mainly in our own research, the ILIAD dataset was used mainly by Baldwin et al. (2007) as well as in this research, and the Ancestry dataset was used extensively by its original author but very few others, and is no longer available for download. The major software packages used include MaltParser, CRFSGD and Hydrat. For each software package, we first introduce related core empirical methodologies, and then describe the package with a focus on its functionality as is relevant to this research.

3.2 CNET Dataset

The CNET dataset, which was created by Kim et al. (2010b), was crawled from CNET forums,1 a vibrant online community where people can ask for help and discuss various technical topics. An example thread is shown in Table 3.1. In this

1http://forums.cnet.com
**HTML Input Code - CNET Coding & scripting**

<table>
<thead>
<tr>
<th>User A</th>
<th>HTML Input Code</th>
</tr>
</thead>
<tbody>
<tr>
<td>Post 1</td>
<td>Hi, I am new to HTML Coding. Please can someone tell me how to create an input box that asks the user to enter their ID, and then allows them to press go. It will then redirect to the page <a href="http://www.sample.com/*****.htm">www.sample.com/*****.htm</a> For example, if they input an ID of 12345, they would be redirected to <a href="http://www.sample.com/12345.htm">www.sample.com/12345.htm</a> Thank You Leo</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>User B</th>
<th>Re: html input code</th>
</tr>
</thead>
<tbody>
<tr>
<td>Post 2</td>
<td>Part 1: create a form with a text field. See: <a href="http://www.w3schools.com/html/html_forms.asp">http://www.w3schools.com/html/html_forms.asp</a> Part 2: give it a Javascript action that composes the full address of the new page and jumps to it. But as you don't ask for Javascript coding, I won't elaborate on that. Kees</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>User C</th>
<th>asp.net c# video</th>
</tr>
</thead>
<tbody>
<tr>
<td>Post 3</td>
<td>I've prepared for you video link click <a href="http://www.ahsapdekorasyonbul.com/user.html">http://www.ahsapdekorasyonbul.com/user.html</a></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>User A</th>
<th>Thank You!</th>
</tr>
</thead>
<tbody>
<tr>
<td>Post 4</td>
<td>Thanks a lot for that.. Really appreciate it. I have Microsoft Visual Studio 6, what program should I do this in? Lastly, how do I actually include this in my site? Thanks again Leo</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>User D</th>
<th>A little more help</th>
</tr>
</thead>
<tbody>
<tr>
<td>Post 5</td>
<td>Leo, I noticed that the above example did not exactly answer your original question. In the video example he showed the Response.Redirect(&quot;<a href="http://www.forums.cnet.com">http://www.forums.cnet.com</a>&quot;). In your question you asked how to take the ID and direct to <a href="http://www.sample.com/id.html">www.sample.com/id.html</a> You would simply do it this way: Response.Redirect(&quot;<a href="http://www.sample.com/">http://www.sample.com/</a>&quot; + TextBox1.Text + &quot;.html&quot;); Now of course you would have needed to create the page they are going to. You could also just as easily create a .aspx page that accepts an id and does something with it in the code. An example of this is: Response.Redirect(&quot;<a href="http://www.sample.com/example.aspx?ID=">http://www.sample.com/example.aspx?ID=</a>&quot; + TextBox1.Text); Remember that I was using the video example that the other poster had posted for you so TextBox1.Text held the users ID. To be honest I would go to your favorite bookstore (online or otherwise) and grab some books on ASP.NET and C#. I can suggest a few that I have read and found very informative if you want. To answer your other questions. Visual Studio 6 is not going to be much help. You will want Visual Studio 2005 or 2008 (recommend 2008) or the cheaper route is to go with Visual Web Developer which is free and can be downloaded from <a href="http://www.microsoft.com/express/vwd/">http://www.microsoft.com/express/vwd/</a> As far as your next question you might want to start here: <a href="http://msdn.microsoft.com/en-us/beginner/default.aspx">http://msdn.microsoft.com/en-us/beginner/default.aspx</a> You can also purchase some books or if you do not feel like it is worth your time you could always hire someone to build this for you. v/r Nick</td>
</tr>
</tbody>
</table>

Table 3.1: An example thread from the CNET dataset
example, User A initiates the thread by asking how to create an interactive input box on a webpage using HTML coding. In response, User B provides a short two-step answer with an external link to an informal page. Then, User C posts an independent answer which is an external link to a tutorial video. User A responds to User C to confirm the details of the proposed solution, and at the same time adds extra information to his/her original question. Finally User D gives a comprehensive answer based on User A’s questions in both Post 1 and Post 4, as well as on previous provided answers. This example is a typical troubleshooting-oriented thread in CNET dataset, where a user is seeking ways to solve a technical problem.

Figure 3.1 shows the screenshot of the example thread on the web. It is worth noting that the CNET forums have a built-in post-linking system, where every post other than the first post is linked back to a previous post. The indented post view shown in Figure 3.1 is based on the post-linking of the thread. It is also interesting to note that there is an extra post at the end of the thread in the screenshot. This is because this final post was posted after the initial crawl of the CNET data. This aspect of dynamically evolving threads will be discussed in Section 4.2.4.

The forum is organised into several categories, each of which contains many subforums. Kim et al. (2010b) only collected data from four categories: Operating Systems, Software, Hardware, and Web Development. The subforum information of these categories is shown in Table 3.2. Kim et al. (2010b) also only collected threads that contained 2 to 16 posts, as threads containing only 1 post have no answers and cannot provide solutions, and long threads tend to be more discussion-oriented and/or contain multiple sub-threads. The initial crawl contains 1000 threads.
Kim et al. (2010b) annotated a subset of the CNET dataset to capture the discourse structure of the threads, by using the dialogue act tagset described in Section 2.2.3. Two annotators annotated label 327 threads, made up of 1,332 posts. Because multilabels are allowed, a modified version of Cohen’s Kappa, proposed by Wang (2009), is used. Details of the calculation are described in Appendix A. The $\kappa$ values of post label and link annotations are 0.59 and 0.78, respectively. Any disagreements in labelling were resolved through adjudication. Of the 1332 posts, 65 posts have multiple labels (which possibly link to a common post), and 22 out of these posts link to multiple posts. The majority post label in the dataset is Answer-answer (40.30%). An example annotated thread, made up of 5 posts from 4 distinct participants, is shown in Figure 3.2.
Chapter 3: Resources

Figure 3.1: Screenshot of the example CNET thread
Table 3.2: Data source categories and sub-forums of the CNET dataset

<table>
<thead>
<tr>
<th>Category</th>
<th>Sub-forum</th>
</tr>
</thead>
<tbody>
<tr>
<td>Operating Systems</td>
<td>Windows 7</td>
</tr>
<tr>
<td></td>
<td>Windows Vista</td>
</tr>
<tr>
<td></td>
<td>Windows XP</td>
</tr>
<tr>
<td></td>
<td>Windows 2000/NT</td>
</tr>
<tr>
<td></td>
<td>Windows ME</td>
</tr>
<tr>
<td></td>
<td>Windows XP 95/98</td>
</tr>
<tr>
<td></td>
<td>Windows Mobile</td>
</tr>
<tr>
<td></td>
<td>Mac OS</td>
</tr>
<tr>
<td></td>
<td>CNET Download site</td>
</tr>
<tr>
<td></td>
<td>Audio &amp; video</td>
</tr>
<tr>
<td></td>
<td>Browsers</td>
</tr>
<tr>
<td></td>
<td>Office &amp; productivity</td>
</tr>
<tr>
<td></td>
<td>E-mail, chat, &amp; VoIP</td>
</tr>
<tr>
<td></td>
<td>Mac software</td>
</tr>
<tr>
<td></td>
<td>Spyware, viruses, &amp; security</td>
</tr>
<tr>
<td></td>
<td>PC utilities</td>
</tr>
<tr>
<td></td>
<td>Photography &amp; design</td>
</tr>
<tr>
<td></td>
<td>Windows Live</td>
</tr>
<tr>
<td></td>
<td>Mac hardware</td>
</tr>
<tr>
<td></td>
<td>Networking &amp; wireless</td>
</tr>
<tr>
<td></td>
<td>PC hardware</td>
</tr>
<tr>
<td></td>
<td>Peripherals</td>
</tr>
<tr>
<td></td>
<td>Storage</td>
</tr>
<tr>
<td>Web Development</td>
<td>Coding &amp; scripting</td>
</tr>
<tr>
<td></td>
<td>Web design &amp; hosting</td>
</tr>
</tbody>
</table>

Figure 3.2: An example CNET thread with thread discourse structure
3.3 ILIAD Dataset

The ILIAD (Improved Linux Information Access by Data Mining) dataset was created by Baldwin et al. (2007), and contains threads crawled from Linuxquestions\textsuperscript{2} and Debian mailing lists\textsuperscript{3}. The ILIAD dataset is made up of 1158 posts from 250 threads, which are annotated with three thread-level labels, namely “Task orientation”, “Completeness” and “Solvedness” by 3 Linux experts. The original annotation was done on a five point scale, and the average score across the 3 annotators was used. The value 2.5 was then used as the breakpoint to convert the average score into a binary value, as the gold-standard class. The $\kappa$ values of “Task orientation”, “Completeness” and “Solvedness” annotations are 0.64, 0.21 and 0.38, respectively. For details of this research, see Section 2.2.1. In this thesis, we only explore the classification task of Solvedness, which addresses the question of “is there a documented solution to the original problem described by the thread initiator in the thread (including the possibility of URLs pointing off to solutions elsewhere on that same forum or generally on the web)?” (Baldwin et al. 2007:pp. 74).

An example thread is shown in Table 3.3. In this example, User A initiates the thread by asking for information about the Debian distribution of Linux and its installation. User B responds with a brief answer in Post 2. Following User B’s response, User A asks for more information about the answer. Then, User B explains his/her answer in more detail. Finally, User C provides another independent answer.

Figure 3.3 shows a screenshot of this example thread on the web. As shown in
\textsuperscript{2}http://www.linuxquestions.org
\textsuperscript{3}http://lists.debian.org/completeindex.html
Debian VS. Red Hat

<table>
<thead>
<tr>
<th>User A</th>
<th>I've been using Red Hat for along time now. I'm about to buy a book on linux so I have a reference to look stuff up with and to learn about this scripting thing, etc. But I hear a lot of fuss about Debian. I've used their apt-get and synaptic too. I like apt-get a lot. So what's so great about Debian? And ummm... which of those CD's do I need? There's like a million of them and I can't find a strait answer upon which ones are for what. I don't think I need them all. Thanx</th>
</tr>
</thead>
<tbody>
<tr>
<td>User B</td>
<td>if you like apt-get, you only need disk 1, everything else you need, you can just apt-get it.</td>
</tr>
<tr>
<td>User A</td>
<td>Quote:</td>
</tr>
<tr>
<td></td>
<td>Originally posted by asb</td>
</tr>
<tr>
<td></td>
<td>if you like apt-get, you only need disk 1, everything else you need, you can just apt-get it.</td>
</tr>
<tr>
<td></td>
<td>Kewl! Is that going to be an obvious option in the installer or do I have to just select the minimal stuff and then do a dist upgrade?</td>
</tr>
<tr>
<td>User B</td>
<td>there is a spot where you choose ftp or http sites for downloading files, cant re-mem-</td>
</tr>
<tr>
<td></td>
<td>ber where exactly at the moment, but it should be fairly obvious. At the end of the installer,</td>
</tr>
<tr>
<td></td>
<td>there is a taskel and dselect part that installs programs - I didn't run either and ended the installer. After this you are left with the bare-bones system, and you will need to install any program, including x. I have woody and apt-get only looked at the stable packages list, which isn't all that up to date. I added the testing packages by adding those files to the sources.list file.</td>
</tr>
<tr>
<td>User C</td>
<td>I mostly use a minimal boot CD (based on b52.4) to install Debian since I have decent cable mod-</td>
</tr>
<tr>
<td></td>
<td>em connection. It contains the bare bones system and is only a 20MB download (or thereabouts). Use it to install the base system, then apt-get or dselect to get whatever you need. This is particularly nice if you like to use testing or unstable since you don't have to spend time installing the stable first. hw</td>
</tr>
</tbody>
</table>

Table 3.3: An example thread from the ILIAD dataset
Figure 3.3: Screenshot of the example ILIAD thread

Figure 3.3, the threads in ILIAD dataset only offer a flat view, with no post-linking information.

3.4 **Ancestry Dataset**

The Ancestry.com Forum Dataset (Ancestry) was jointly created by Jonathan Elsas and Ancestry.com, a website for historical genealogical research. The Ancestry dataset contains a full snapshot of the Ancestry.com online forum, from December 1995 to July 2010. The dataset includes 22,054,728 messages (posts) spanning

\[\text{http://boards.ancestry.com}\]
9,040,958 threads, from 165,358 sub-forums. The total number of unique participants (users) is 3,775,670. The Ancestry dataset is presented at the post-level, and the information associated with each post includes: message identifiers, subforum name, thread identifier, author name/identifier, timestamp at day level, URL of the original message, message title and message body. Inter-post linkings within each thread, as generated by users, are also provided.

An example thread is shown in Table 3.4. In this example, User A initiates the thread by seeking information about his/her great-grandfather’s grave, because it is hard for him/her to visit the grave in person. In response, User B asks for additional information about the initiator’s great-grandfather. Following User B’s response, User A adds more information about his/her great-grandfather in Post 3. Then, in Post 4, User C provides an answer which includes information about the user’s grandmother who is buried in the same cemetery, as well as a suggestion to visit the cemetery. Finally, the initiator acknowledges User C’s reply, and at the same time states that he/she has made contact with the cemetery.

Compared to the example threads the from CNET and ILIAD datasets, this example thread from the Ancestry dataset is more on the information-seeking side rather than troubleshooting. In fact, the discussion is almost like information-exchange when User B expresses interest in knowing more about User A’s great-grandfather in Post 2. Additionally, it is interesting to note that the keyword cemetery is mentioned five times throughout the discussion, and is spelled corrected only once.

A screenshot of this example thread on the original website is shown in Figure 3.4. Although the thread shown in Figure 3.4 is presented in flat view, Ances-
try.com also supports a thread view, based on the post-linking information, as in Figure 3.5.
### Henry Smith - buried Tower Hamlets

| User A Post 1 | Henry Smith - buried Tower Hamlets  
I am trying to discover details of my GGrandfather’s grave at Tower Hamlets cemetery. Henry Smith died 28th Aug 1924, and was buried in grave 2019-R. As I live a long way away, I am hoping someone can tell me how to find details of inscriptions, who else might be buried there etc. |
| User B Post 2 | Re: Henry Smith - buried Tower Hamlets  
Hi, would like to know about your ggrandfather Henry Smith. Was wondering if he is the same Henry I am looking for. Did he ever live in GA? Did he have a son he named Henry also? Thanks gari Lup |
| User A Post 3 | Re: Henry Smith - buried Tower Hamlets  
Hello Gari Lu, 
In partial answer to your enquiry, the Henry Smith who died in 1924 was actually the son of a Henry Smith (who was a Weaver apparently), but my Family records don't tell of him naming any of his own four sons as Henry. I'm afraid I don't know where GA is, so I can't begin to tell you if Henry ever lived there.  
Duncan Smith, Dundee. |
| User C Post 4 | Re: Henry Smith - buried Tower Hamlets  
Hi there. My Grandmother Susan Townsend was buried in Tower Hamlets Cemetery, Bow. Grave No. 3438R. She died 9/11/32 aged 50. Apparently it is now a "nature reserve", (my aunt lives at the back of the cemetery.) You can visit the "cemetery" one Sunday a month and they will try and locate the position of the grave for you. Hope this is of some help. I was disgusted when I found out as I am sure you will be.  
Regards Lynn |
| User A Post 5 | Re: Henry Smith - buried Tower Hamlets  
Hello Lynn,  
Thank you for your comments. I have made contact with the "Friends of Tower Hamlets Cemetery". It seems they try to do walks around the place in mid-winter to note down the position of graves and any legible engravings, because this is the only time of year that the undergrowth has died down sufficiently.  
Duncan, Dundee |

Table 3.4: An example thread from the Ancestry dataset
Figure 3.4: Screenshot of the example Ancestry thread in flat view
Figure 3.5: Screenshot of the example Ancestry thread in thread view

The Ancestry dataset also comes with a selected set of 191 queries from Ancestry.com’s query log, and pairwise preference relevance judgements for a thread retrieval task using Ancestry.com forum data.

To create the pairwise preference relevance judgements, a document pool is simulated as the first step. Firstly, Indri\(^5\) Terrier\(^6\) Zettair\(^7\) and Ancestry.com’s

---

\(^5\)Indri version 2.12 (Lemur version 4.12) from [http://lemurproject.org/](http://lemurproject.org/)

\(^6\)Terrier version 3.0 from [http://terrier.org/](http://terrier.org/)

\(^7\)Zettair Version 0.9.3 from [http://www.seg.rmit.edu.au/zettair/](http://www.seg.rmit.edu.au/zettair/)
ranked Boolean system are applied over the whole dataset to produce post rankings, with each ranking containing 1000 posts. Then, three aggregation methods, namely Mean, Max and Pseudo-Cluster Selection (PCS) (Seo et al. 2011), are used to convert each post ranking to a thread ranking. The Mean aggregation method uses the mean score of the retrieved posts as the thread score. Max derives the thread score by selecting the max score of the retrieved posts. PCS calculates the geometric mean of the top-\(k\) (\(k = 5\) is used in the original paper) retrieved posts as the thread score. Lastly, the document pool is created by combining the top 100 threads of each thread ranking. The document pool contains 374 unique threads per query on average.

Relevance assessment is conducted by Ancestry.com, by collecting document
pair preferences \cite{Carterette2008}. The annotation interface is modelled after the one proposed by \cite{CarteretteBennett2008}. A screenshot of the interface is shown in Figure \ref{fig:annotation_interface}, and presents side-by-side document pairs \((L, R)\) and collects the following judgements:\footnote{It is not clear that what the judgement should be when both documents are equally relevant, and can only assume that this did not occur when collecting the judgements, or that a suitable tie-breaking mechanism was employed.}

- \(L\) is preferred to \(R\)
- \(R\) is preferred to \(L\)
- \(L\) and \(R\) are duplicates
- \(L\) is bad
- \(R\) is bad.

During the assessment process, a document pair selection algorithm, described in detail in \cite{Elsas2011}, is used to reduce the number of assessments.

Out of the 191 queries, 50 queries were first selected for a pilot assessment, with each query annotated by two assessors. The results of the pilot assessment were analysed and used as a guide to set the parameters of the document pair selection algorithm, as well as adjust assessor training and assessment guidelines. Then, each of the remaining 141 queries was assessed by one assessor, with the adjusted parameters of the pair selection algorithm.
3.5 Dependency Parsing

In this section, we first give an introduction to dependency parsing, with a focus on concepts that are related to this thesis. Then, the dependency parsing software package used in this research will be described.

3.5.1 Introduction

Dependency parsing (Kübler et al. 2009) is the task of automatically analysing a sentence’s dependency structure, in the form of binary, asymmetrical dependency relations between word pairs, with a dependency type associated with each dependency relation. A dependency structure is defined as a labelled directed graph (i.e. dependency graph), and the task of dependency parsing is that of mapping an input sentence $S$, made up of words $w_0w_1...w_n$ (where $w_0$ is an artificial ROOT), to its dependency graph $G$. In general, the methods used for dependency parsing can be grouped into two categories: “data-driven” and “grammar-based”. Data-driven approaches use machine learning with training data to parse new sentences, while grammar-based methods make use of formal grammars. Data-driven approaches can further be divided into “transition-based” and “graph-based” methods, and grammar-based methods can further be divided into “context-free” and “constraint-based” methods. This report will only cover the basis of transition-based methods, which are the most relevant to our research. Also, we assume the dependency structure to be a more restricted dependency tree, “which is any well-formed dependency graph that is a directed spanning tree originating out of the root word $w_0$” (Kübler et al. 2009: p. 19).
The idea of transition-based approaches is to treat the dependency tree for the input sentence as a sequence of valid “transitions”, which starts from an initial “configuration/state” for the sentence and ends up in one of several terminal configurations/states. To illustrate the process, we focus on a stack-based transition system which implements a form of shift-reduce parsing. In this system, each configuration consists of a partial analysis of the input sentence, a stack of words (σ), a buffer of words (β) and a set of dependency arcs (A). σ contains partially processed words, β contains the remaining input words, and A represents a partially constructed dependency tree. In each transition, the system chooses one of three transition types:

**Left-Arc:** \((σ|w_i, w_j|β, A) \Rightarrow (σ, w_j|β, A \cup \{(w_j, r, w_i)\}), i \neq 0\): add a dependency arc \((w_j, r, w_i)\) to the arc set A and remove \(w_i\), where \(w_i\) is the top word in the stack \(σ\), \(w_j\) is the first word in the buffer \(β\), and \(r\) is the type of the dependency between \(w_j\) and \(w_i\).

**Right-Arc:** \((σ|w_i, w_j|β, A) \Rightarrow (σ, w_i|β, A \cup \{(w_i, r, w_j)\})\): add a dependency arc \((w_i, r, w_j)\) to the arc set A, replace the first buffer word \(w_j\) with the top stack word \(w_i\), and remove \(w_i\).

**Shift:** \((σ, w_i|β, A) \Rightarrow (σ|w_i, β, A)\): remove the first buffer word \(w_i\), and push \(w_i\) to the top of the stack.

In order to illustrate the idea of applying dependency parsing over discussion threads, where a thread is treated as a sentence and a post is treated as a word, we use the CNET thread example introduced in Chapter 1, as shown in Figure 1.5. Table 3.5 shows a transition sequence that derives the dependency tree of the example
No.  | Transition     | Configuration
-----|----------------|-------------------
0    | –              | ([ROOT], [P1, P2, P3, P4, P5], ∅)
1    | SH             | ([ROOT, P1], [P2, P3, P4, P5], ∅)
2    | RA_answer-answer | ([ROOT], [P1, P3, P4, P5], A₁ = [(P1, Answer-answer, P2)])
3    | SH             | ([ROOT, P1], [P3, P4, P5], A₁)
4    | SH             | ([ROOT, P1, P3], [P4, P5], A₁)
5    | RA_answer-confirmation | ([ROOT, P1], [P3, P5], A₂ = A₁ ∪ ((P3, Answer-confirmation, P4))), A₃ = A₂ ∪ ((P1, Answer-answer, P3)))
6    | RA_answer-answer | ([ROOT], [P1, P5], A₂ = A₁ ∪ ((P1, Answer-answer, P5)))
7    | RA_answer-answer | ([ROOT], [P1], A₃ = A₂ ∪ ((ROOT, Question-question, P1)))
8    | SH             | ([ROOT], [P5], A₃)
9    | RA_question-question | ([], [ROOT], A₃ = A₂ ∪ ((ROOT, Question-question, P1)))
10   | SH             | ([ROOT], [], A₃)

Table 3.5: A transition sequence for the thread in Figure 3.7 (LAᵣ represents Left-Arcᵣ, RAᵣ represents Right-Arcᵣ, and SH represents Shift.)

thread shown in Figure 3.7. Three points should be noted in this example: (1) to convert the thread discourse structure to a format which complies with dependency structure, the directions of all the direct links in the discourse structure are reversed; (2) standard dependency parsing does not support the case where a post is the dependent of more than one head post, such as Post 4 in the original example of Figure 1.5; therefore, one of the dependency links pointing to Post 4 is removed for this example; and (3) in the context of forum threads, head posts always precede dependent posts, in terms of the chronological sequencing of posts. Therefore, no Left-Arcᵣ transition is present in the transition sequence shown in Table 3.5. For a detailed discussion about special properties of dependency parsing over threads that distinguish it from standard dependency parsing, see Section 4.2.2.

Machine learning models are often used to decide what transition types to choose at each configuration. One of the most popular and successful models is support vector machines (SVMs). SVMs (Tan et al. 2005) are maximum-margin classifiers which search for a maximum margin hyperplane: a decision boundary that can
maximise the margin between training instances which belong to different classes. When combining SVMs with non-linear kernel functions, such as a polynomial kernel, the feature space is mapped to a higher dimensional space, hence achieving implicit feature combination and non-linear classification. The basic instance unit for SVMs in dependency parsing is a “configuration”. The following paragraphs will explain how to construct feature vectors for each configuration, and how to define class labels for each configuration, in the context of parsing threads.

Figure 3.7: Dependence structure of an example CNET thread
The features used in transition-based parsing are mainly “configurational word features”, which are defined by two functions: an “address function” and an “attribute function”.

An address function is used to locate a particular word in a configuration, e.g. the $k$th word from the stack/buffer or the parent/child/sibling of a word in the dependency graph defined by the configuration. It should be noted that an address function may fail to locate a word in a configuration, and thus all features defined with this address function will have NULL values.

An attribute function extracts the actual feature(s) from the words located by address functions, e.g. part-of-speech tag or dependency label for a word. The features can be “static” (e.g. a part-of-speech tag), where they stay the same throughout all the configurations if available at all, or “dynamic” (e.g. dependency label), where the feature values are only available after a certain number of configurations. Dynamic features are one of the main advantages of transition-based parsing, where the transition history and partially built dependency tree are captured.

The address functions and attribute functions can be defined in a feature model by users. To illustrate the idea of dependency parsing over threads, we use a simple feature model as shown in Table 3.6. Table 3.6 describes seven features, where each feature is defined by the combination of an address function and an attribute function. Table 3.7 shows the feature vectors created for each configuration in Table 3.5, based on the feature model defined in Table 3.6. Take configuration 5 for example. According to Table 3.5, the first post in the current stack is P1, and the first and second posts in the current buffer are P3 and P5 respectively. Therefore,
Table 3.6: A simple feature model with seven features. Regarding the address functions, \( \text{STK}[i] \) locates the \( i \)th post in the stack, \( \text{BUF}[i] \) locates the \( i \)th post in the buffer, \( \text{LDEP}(w) \) locates the farthest child of post \( w \) to the left, and \( \text{RDEP}(w) \) locates the farthest child of post \( w \) to the right. With respect to attribute functions, COUNT is the word count of a post and DEPREL represents the dependency label (dialogue act label in our case).

the first three features based on the feature model in Table 3.6 are \( N_{P1} \), \( N_{P3} \) and \( N_{P5} \) respectively. As described earlier, it is not possible for a post to have children to its left. Therefore, Features 4 and 5 will always be NULLs. Regarding the last two features, the current farthest children of P1 and P3 to the right are P2 and P4, respectively. Therefore, Features 6 and 7 are Answer-answer and Answer-confirmation, respectively.

Regarding the gold-standard label of a configuration \( c = (\sigma, \beta, A) \) for SVMs, the following scheme is used to decide \( c \)'s label:

**Left-Arc:** if \((\beta[0], r, \sigma[0])\) is in the gold-standard dependency tree.

**Right-Arc:** if all the outgoing arcs from \( \beta[0] \), according the gold-standard dependency tree, are already added into \( A \) at the current configuration, and
Table 3.7: Feature (Feat) vectors for the configurations (Config) in Table 3.5, based on the feature model in Table 3.6 ($N_{Pi}$ represents the number of words in Post $i$.)

($\sigma[0], r, \beta[0]$) is in the gold-standard dependency tree.

**Shift:** otherwise.

While dependency parsing was originally designed for parsing sentences, it has also been used in other domains, such as document discourse parsing (Sagae 2009), information extraction (Culotta and Sorensen 2004), question-answering (Wang et al. 2007a), and joint extraction of event structures (McClosky et al. 2011).

### 3.5.2 MaltParser

MaltParser (Nivre et al. 2007) implements data-driven transition-based dependency parsing. The idea is to learn a transition system, or state machine, to map a sentence onto its dependency graph. The most up-to-date version\(^9\) implements four groups of parsing algorithms, namely Nivre (Nivre 2003; Nivre 2004), Covington (Covington 2001), Stack (Nivre 2009; Nivre et al. 2009) and Planar (Gómez-Rodríguez and Nivre 2010). Nine deterministic parsing algorithms are included,

\(^9\)Available at [http://www.maltparser.org/download.html](http://www.maltparser.org/download.html), version 1.5 is used in this research.
namely Nivre arc-eager, Nivre arc-standard, Covington projective, Covington non-projective, Stack projective, Stack swap-eager, Stack swap-lazy, Planar, and 2-Planar. The Nivre algorithms (i.e. Nivre arc-eager and Nivre arc-standard) are the ones used in this research. The algorithms use two data structures: a stack of partially processed tokens and a queue (buffer) of remaining input tokens, as explained in earlier. There are two machine learning packages used in MaltParser, namely LIBSVM ([Chang and Lin 2011]) and LIBLINEAR ([Fan et al. 2008]). In our research, LIBSVM is used in all experiments. It should be noted that, as described earlier in Section 3.5.1, dependency parsing is primarily intended for parsing sentences, where each word in the sentence is considered to be a token. However, in our research, dependency parsing is applied over forum threads, with each post in the thread as a token. This was explained in detail earlier in Section 3.5.1 with a thread parsing example.

One feature of MaltParser that makes it well suited to this research is that it is possible to define feature models of arbitrary complexity for each token.\(^\text{10}\) The feature models are history-based, where the next action in the deterministic derivation of a dependency structure is predicted using features of already partially built dependency structures. Each feature in the feature models is defined using a functional notation with three types of functions, namely: (1) address functions, (2) feature (attribute) functions, and (3) feature map functions. Address functions and feature (attribute) functions are described in Section 3.5.1. Feature map functions operate over feature values returned by feature functions. For example, two or three

\(^{10}\text{To configure and use MaltParser, we also closely collaborated with Joakim Nivre, one of the authors of MaltParser.}\)
feature values can be merged into one feature value by using a feature map function. Moreover, if a feature value is a string, it can be split with an arbitrary delimiter, or a suffix/prefix of arbitrary length can be extracted. In the context of parsing forum threads, feature map functions implemented in MaltParser are less useful because these functions often operate over word-level string features. Nevertheless, it is still possible to use feature map functions to merge dependency label predictions from partially built dependency trees, e.g. merging the dependency labels of the parent and grandparent of a targeted post.

Additionally, MaltParser supports three different prediction strategies:

**Combined prediction**: Predict transition and the arc label predictions in a combined mode with one feature model.

**Sequential prediction**: First predict the transition with a transition feature model, and subsequently predict the arc label with an arc label feature model if the transition requires an arc label.

**Branching prediction**: First predict the transition with a transition feature model and if the transition does not require an arc label, then the nondeterminism is resolved; if the predicted transition requires an arc label then the parser continues to predict the arc label using separate feature models. If the transition is a left arc transition it predicts the arc label using a left arc feature model, and if it is a right arc transition it uses a right arc feature model.
3.6 Graphical Models

This section will first present a few important graphical models, which are relevant to this research. Then, a software package, which implements one of the introduced graphical models, will be described. This package will be used extensively throughout this research.

3.6.1 Introduction

In machine learning, the aim of classification is often to predict the categories of an instance. In the field of Natural Language Processing (NLP), classification sometimes involves the prediction of multiple dependent output variables/classes/labels, such as part-of-speech (POS) tagging, where the POS tag of a word in a sentence is generally dependent on the POS tags of other words (Sutton and McCallum 2012). In the context of forum thread discourse structure analysis, this dependence is even more complex: not only because the link classes and dialogue act classes are dependent on each other, but also due to dependence existing within the link classes and dialogue act classes. One way to deal with dependent output variables is using graphical models, which are probabilistic models using a “graph-based representation as the basis for compactly encoding a complex distribution over a high-dimensional space” (Koller and Friedman 2009:p. 3). While there is a wide range of graphical models used in the context of NLP, this report will only briefly go over the basis of the models which are the most relevant to our research.

In general, the classification task in graphical models is to predict an output vector $y = \{y_0, y_1, ..., y_T\}$ of random variables (i.e. classes of all the instances), given
an input vector \( \mathbf{x} = \{ \mathbf{x}_0, \mathbf{x}_1, \ldots, \mathbf{x}_T \} \) of observed features. Each \( \mathbf{x}_t = \{ x_{t,1}, x_{t,2}, \ldots, x_{t,K} \} \) contains various features of an instance at position \( t \) in the graph. One way to approach the task is using \textbf{generative models} (Sutton and McCallum 2012) which try to derive a joint probability distribution \( p(y, \mathbf{x}) \) over inputs and outputs. A simple example is naive Bayes (NB) classifiers, which model the joint probability of the input vector and a possible output class for each instance:

\[
p(y, \mathbf{x}_t) = p(y) \prod_{k=1}^{K} p(x_{t,k} | y) \tag{3.1}
\]

In Equation 3.1, we want to calculate the joint probability of an observed feature vector \( \mathbf{x}_t \) with each possible output class \( y \). By making a strong independence assumption between every feature, rather than calculating the probability of generating feature vector \( \mathbf{x}_t \) given a class \( y \), naive Bayes computes the product of the probabilities of generating each individual feature \( x_{t,k} \) given a class \( y \).

First-order hidden Markov Models (HMMs) are slightly more complex and model the joint probability of the input/observation sequence \( \mathbf{x} \) and output/prediction sequence \( \mathbf{y} \) by assuming: (1) each state \( y_t \) in the graph only depends on its preceding state \( y_{t-1} \); and (2) each observation \( \mathbf{x}_t \) only depends on the current state \( y_t \):

\[
p(\mathbf{y}, \mathbf{x}) = \prod_{t=1}^{T} p(y_t | y_{t-1}) p(\mathbf{x}_t | y_t) \tag{3.2}
\]

The main limitations of these generative models are related to their inability to handle input variables \( \mathbf{x} \) with large dimensionality and complex dependencies (Sutton and McCallum 2012). \textbf{Discriminative models}, which model the conditional distribution \( p(y | \mathbf{x}_t) \) or \( p(y | \mathbf{x}) \) directly, are usually used to overcome these problems. A simple example of a discriminative model is a maximum entropy (MaxEnt)
model (also known as “logistic regression”) which assumes that the logit function, logit$(p(y|x_t))$, of each class $y$ for an instance is a (normalised) linear function of the input vector $x_t$ of that instance, plus a normalisation constant:

$$p(y|x_t) = \frac{1}{Z(x_t)} \exp \left( w_y + \sum_{k=1}^{K} w_{y,k} x_{t,k} \right)$$  \hspace{1cm} (3.3)

where $Z(x_t) = \sum_y \exp \left( w_y + \sum_{k=1}^{K} w_{y,k} x_{t,k} \right)$ is a normalisation factor used to convert the exponential term into a probability, and $w_y$ and $w_{y,k}$ are weight parameters. To unify the notation of the formula for easier comparison, we introduce a set of feature functions where $f_{y',k}(y, x_t) = 1_{\{y'=y\}} x_{t,k}$ is used with $w_{y,k} x_{t,k}$, and $f_y(y, x_t) = 1_{\{y'=y\}}$ is used with $w_y$, where $1_{\{y'=y\}}$ returns 1 if $y' = y$, and returns 0 otherwise. Equation 3.3 can then be rewritten as:

$$p(y|x_t) = \frac{1}{Z(x_t)} \exp \left( \sum_{k=1}^{K} w_k f_k(y, x_t) \right)$$  \hspace{1cm} (3.4)

Maximum entropy Markov models (MEMMs) (McCallum et al. 2000) extend the MaxEnt model to sequence classifiers which model the conditional distribution of a whole sequence $p(y|x)$. This is done by calculating the probability of the current state $y_t$ conditioned on its preceding state $y_{t-1}$ and the current observation $x_t$ (Jurafsky and Martin 2009:p. 230):

$$p(y|x) = \prod_{t=1}^{T} p(y_t|y_{t-1}, x_t) = \prod_{t=1}^{T} \frac{1}{Z(x_t)} \exp \left( \sum_{k=1}^{K} w_k f_k(y_t, y_{t-1}, x_t) \right)$$  \hspace{1cm} (3.5)

In Equation 3.5, if we regard observations as being associated with state transitions rather than with states themselves, then $p(y_t|y_{t-1}, x_t)$ can be thought as the probability of the transition from state $y_{t-1}$ to state $y$ on input $x_t$. Also, it is worth noting that the normalisation is done per-state.

---

11. $f(x) = \frac{1}{1 + e^{-x}}$

12. Note that the $K$ in this equation is a superset of the $K$ in Equation 3.3.
MEMMs often suffer from the *label bias problem* because of per-state normalisation, which creates a bias towards states with fewer outgoing transitions (Lafferty et al. 2001). To address this problem, conditional random fields (CRFs) were proposed (Lafferty et al. 2001), where the entire output/predicted class sequence is conditioned on the entire input/observed feature sequence. A CRF is a random field (i.e. a collection of variables $Y$ over states) globally conditioned on the whole observation sequence $x$:

$$p(y|x) = \prod_{t=1}^{T} p(y_t|y'_t, x)$$

where $y_t$ and $y'_t$ are neighbours in a graph. When the graph is a linear chain, the linear-chain CRF takes a form very similar to MEMMs:

$$p(y|x) = \frac{1}{Z(x)} \prod_{t=1}^{T} \exp \left( \sum_{k=1}^{K} w_k f_k(y_t, y'_t, x) \right)$$

In Equation 3.7, the normalisation is done globally at the instance-level. Empirical experimental results over sequence segmentation and labelling tasks such as POS tagging show that CRFs outperforms both MEMMs and HMMs (Lafferty et al. 2001). However, CRFs generally converge slower during training compared to MEMMs and HMMs.

### 3.6.2 CRFSGD

CRFSGD (Bottou 2011) implements a linear-chain conditional random field (CRF), as described in Section 3.6.1, using stochastic gradient descent to efficiently solve the convex optimisation problem, and scales well to large feature sets.

CRFSGD makes use of feature templates to generate complex features. The templates utilise special macros to locate any raw features from tokens of the input, and
use them to generate (expand) features for a given token. A generated feature can take the string value of a single raw feature, or the combination of two or more raw features. Additionally it is also possible to add arbitrary characters to a generated feature.

There are two types of feature templates, namely unigram templates and bigram templates. Unigram templates create a set of binary feature functions for the combinations of each generated feature with each possible output class. Bigram templates create a set of binary feature functions for the combinations of each generated feature with each possible output class of both the current and previous tokens. Examples of a unigram feature function and a bigram feature function are given below:

- def eg_unigram_func(output, feature):
  
  if (output = classX) and (feature = featureM):
      return 1
  else
      return 0

- def eg_bigram_func(output_current, output_previous, feature):
  
  if (output_current = classX) and (output_previous = classY) and
      (feature = featureM):
      return 1
  else
      return 0
3.7 Experimental Methodologies

This section will focus on two relevant empirical methodologies, namely feature weighting and evaluation, as well as the software package used in this research for experimental management.

3.7.1 Feature Weighting

Feature weighting is a way of numerically estimating the utility of each feature as a means of improving classification accuracy. It can also be used to select the most useful features, which can reduce the feature space and improve classification efficiency without sacrificing accuracy. Feature weighting can be done in either a supervised or unsupervised manner. This section will present two representative feature weighting methods which are used in this research: term weighting \( \text{tf-idf} \) and information gain (IG).

In the context of feature weighting, the dataset often contains feature-rich instances, such as text documents, forum threads and forum posts.

\( \text{tf-idf} \) is an unsupervised feature weighting approach, and reflects how distinct a feature is of a given instance, relative to the entire dataset. The weights for each feature are different across different instances. \( \text{tf-idf} \) is the product of two statistics: term frequency (tf) and inverse document frequency (idf). The intuition behind tf is that if a feature \( t \) occurs frequently in a given instance \( d \), more weight
should be given to \( t \) for \( d \). The idea behind \( \text{idf} \) is that if a feature \( t \) appears in many instances of the dataset \( D \), less weight should be given to \( t \) for all instances in \( D \). A classic \( \text{tf-idf} \) formulation for calculating the weight of a feature \( t \) in instance \( d \) is:

\[
w_{d,t} = f_{d,t} \times \log \frac{N}{f_t}
\]

where \( f_{d,t} \) is the frequency of feature \( t \) in instance \( d \), \( N \) is the total number of instances in the dataset, and \( f_t \) is the number of instances containing feature \( t \). It is worth noting that \( \text{tf} \) and \( \text{idf} \) may also be used separately as two simple weighting methods. While the \( \text{tf} \) weights of each feature are different across different instances, the \( \text{idf} \) weight of each feature is the same across different instances.

IG is a supervised feature weighting method, and indicates how discriminative each feature is relative to labelled data. The weight of each feature is the same across instances in the dataset. The idea is based on information theory, or entropy to be more specific. Entropy captures the uncertainty of a random variable. It is typically measured in bits, and represents the minimum number of bits required on average to encode an event associated with a given probability distribution. Formally, the entropy \( H \) of a discrete random variable \( x \) is:

\[
H(x) = -\sum_{i=1}^{n} P(x_i)\log P(x_i)
\]

where \( P(x_i) \) denotes the probability of \( x = x_i \), and \( n \) is the total number of discrete values that \( x \) may take.

To calculate the IG of a feature \( x_i \), we first split the whole dataset \( D = \{d_j| j \in [1, n]\} \) into different groups \( R = \{r_k| k \in [1, m]\} \), where all the instances in a group \( r_k \) have the same value for feature \( x_i \). Then we calculate the entropy of the class dis-
distribution for the whole dataset $H(D)$, as well as the entropy of the class distribution for each group $H(r_k)$. The IG of the feature $x_i$ is:

$$IG(D, x_i) = H(D) - H(D|x_i) = H(D) - \sum_{k=1}^{m} P(r_k)H(r_k)$$

where $P(r_k) = \frac{|r_k|}{|D|}$ is the proportion of instances in group $r_k$ to the total number of instances in the dataset $D$. A higher IG value for a feature $x_i$ indicates that this feature is better able to divide the dataset into purer groups relative to the class distribution.

3.7.2 Evaluation

Experimental evaluation is a critical component of this research. This section will cover the basics of experimental evaluation methodology.

Cross-validation—also called “rotation estimation”—is a statistical validation method used for evaluating the generalisability of a statistical analysis model. In the context of machine learning, cross-validation is used to assess and compare learning algorithms.

The basic form of cross-validation is $k$-fold cross-validation. In a $k$-fold cross-validation experiment, the whole dataset is first partitioned into $k$ (roughly) equally sized folds. Then $k$ experiments are conducted, where each experiment takes a unique combination of $k - 1$ folds for training, and the remaining 1 fold for testing. The final experimental result is the aggregation of testing results from each of the $k$ experiments. In machine learning, $k = 10$ is one of the most common settings, and is used throughout all the experiments conducted in this research.
To further reduce the potential estimation bias during experiments, a stratified $k$-fold cross-validation can be used (Kohavi 1995). In a standard stratified $k$-fold cross-validation, the whole dataset is partitioned such that the class distribution of each fold is roughly the same as the class distribution of the whole dataset. In most of our experiments, our datasets are web user forum threads where the basic instance unit is a post and multiple posts are grouped into threads. In order to make use of the thread information of the posts, we use a thread-level stratified 10-fold cross-validation, where the whole dataset is partitioned such that the thread length distributions of each fold are roughly the same as the thread length distribution of the whole dataset.

The evaluation metrics used for experiments in this research include accuracy, macro-averaged precision, recall and F-score, as well as micro-averaged precision, recall and F-score. For a given instance $d$ and class $c$, the basic units for calculating these scores are:

**True positive (TP):** $d$ is correctly identified as belonging to class $c$, i.e. the prediction for instance $d$ is $c$, and the actual class for instance $d$ is also $c$.

**False positive (FP):** $d$ is incorrectly identified as belonging to class $c$, i.e. the prediction for instance $d$ is $c$, while the actual class for instance $d$ is not $c$.

**True negative (TN):** $d$ is correctly rejected as belonging to class $c$, i.e. the prediction for instance $d$ is not $c$, and the actual class for instance $d$ is also not $c$.

**False negative (FN):** $d$ is incorrectly rejected as belonging to class $c$, i.e. the prediction for instance $d$ is not $c$, while the actual class for instance $d$ is $c$. 
Chapter 3: Resources

Table 3.8: Confusion matrix for class $c$

<table>
<thead>
<tr>
<th>Actual Class</th>
<th>Class Prediction</th>
</tr>
</thead>
<tbody>
<tr>
<td>$c$</td>
<td>TP</td>
</tr>
<tr>
<td>$\overline{c}$</td>
<td>FN</td>
</tr>
<tr>
<td>FP</td>
<td>TN</td>
</tr>
</tbody>
</table>

Table 3.9: $TP/TN/FP/FN$ of the prediction for two example multi-label multi-class instances $d_i$ and $d_j$

<table>
<thead>
<tr>
<th></th>
<th>Actual</th>
<th>Prediction</th>
<th>Evaluation</th>
</tr>
</thead>
<tbody>
<tr>
<td>$d_i$</td>
<td>Y</td>
<td>N</td>
<td>N</td>
</tr>
<tr>
<td>$d_j$</td>
<td>Y</td>
<td>N</td>
<td>N</td>
</tr>
</tbody>
</table>

It is worth emphasising that the number of TPs, FPs, TNs and FNs are all computed per instance for each class. Another classic way of describing TP, FP, TN and FN is by using a confusion matrix, as shown in Table 3.8.

Many of the classification tasks in the thesis are multi-label multi-class ones. In a multi-label classification task, each instance may be labelled by a different number of classes from the same class set. In a multi-class task, the class set contains more than two class labels. An example of multi-label multi-class classification evaluation is presented in Table 3.9.

Classification accuracy ($ACC$) for a class $c$ is the proportion of correct prediction for $c$, which includes both $TP_c$ and $TN_c$:

$$
ACC_c = \frac{|\{TP_c\}| + |\{TN_c\}|}{|\{TP_c\}| + |\{FP_c\}| + |\{TN_c\}| + |\{FN_c\}|}
$$

The overall $ACC$ for a multi-label multi-class classification task is defined as the
proportion of $TP$ in all the class labels:

$$ACC = \frac{\sum_{c=1}^{C} |TP_c|}{\text{Number of all gold-standard class labels}}$$

The precision ($P$) for a class $c$ is the proportion of instances correctly predicted as $c$ among all instances predicted as $c$:

$$P_c = \frac{|TP_c|}{|TP_c| + |FP_c|}$$

The recall ($R$) for a class single $c$ is the proportion of instances correctly predicted as $c$ among all instances that are of type $c$:

$$R_c = \frac{|TP_c|}{|TP_c| + |FN_c|}$$

The overall precision and recall for a multi-label multi-class classification task can be computed by either micro-averaging or macro-averaging. The micro-averaged precision ($P_\mu$) and recall ($R_\mu$) are averaged at the level of each individual class label prediction:

$$P_\mu = \frac{\sum_{c=1}^{C} |TP_c|}{\sum_{c=1}^{C} (|TP_c| + |FP_c|)}$$

$$R_\mu = \frac{\sum_{c=1}^{C} |TP_c|}{\sum_{c=1}^{C} (|TP_c| + |FN_c|)}$$

The macro-averaged precision ($P_M$) and recall ($R_M$) are averaged at the level of each class:

$$P_M = \frac{\sum_{c=1}^{C} P_c}{C}$$

$$R_M = \frac{\sum_{c=1}^{C} R_c}{C}$$

The F-score in each case is the weighted harmonic mean of $P$ and $R$:

$$\text{F-score} = (1 + \beta^2) \frac{P \times R}{R + \beta^2 P}$$
where $\beta = 1$ is used in all experiments reported in this thesis.

It is worth noting that, only $TP$, $FP$ and $FN$ counts are considered when calculating $P$, $R$ and F-score. Therefore, if there are only $TN$ counts for a particular class, such as the class $c_m$ in Table 3.9, this class is often omitted when computing macro-averaged statistics. Additionally, in this thesis, statistical significance is tested using randomised estimation (Yeh 2000) with $p < 0.05$, unless otherwise stated.

The above evaluation metrics are the ones used in our research to evaluate machine learning models. Later, we also investigate information retrieval (IR) tasks. The basic evaluation metrics in IR are also precision ($P$) and recall ($R$):

$$P = \frac{\text{# of relevant documents retrieved}}{\text{total # of documents retrieved}}$$

$$R = \frac{\text{# of relevant documents retrieved}}{\text{total # of relevant documents}}$$

The recall is often ignored in IR evaluation, because it is usually implausible to label all the available documents (e.g. the data of the whole web). Instead, the precision for the top-$k$ ranked documents ($P@k$) is frequently used to evaluate IR systems. Another popular IR evaluation metric is average precision ($AP$), which is the average of the precision values at each relevant document $D_j$ in the document ranking:

$$AP = \frac{1}{|D|} \sum_j \text{Precision}(D_j)$$

If a relevant document $D_j$ is not retrieved (i.e. not in the ranking), a precision score of 0 is often used. The mean average precision ($MAP$) is used as an overall evaluation metric.
metric, which is the mean $AP$ across all queries:

$$\text{MAP} = \frac{\sum_{j=1}^{N} AP_j}{N}$$

The above basic IR evaluation metrics are used for the traditional absolute preferences, where a gold standard ranking of documents for each query is provided. In our IR experiments we use the Ancestry dataset ([Elsas 2011](#)), which will be described in detail in Section 3.4. The collection of preferences $P$ with the Ancestry dataset are pairwise preferences, rather than traditional absolute preferences:

$$P = \{(D_i, D_j) | \text{document } D_i \text{ is preferred to document } D_j\}$$

The set of correctly ranked preferences $P_{\text{correct}}$ is:

$$P_{\text{correct}} = \{(D_i, D_j) | (D_i, D_j) \in P \text{ and } \rho(D_i) < \rho(D_j)\}$$

where $\rho(D_i)$ represents the rank of document $D_i$ assigned by a ranking algorithm.

An analogue to the absolute evaluation measure precision at a cutoff $k$ ($P@k$) is Precision of Preferences at a cutoff ($ppref@k$), which represents the proportion of correctly ordered preferences to ordered preferences, where at least one document/thread in the pair is ranked above $k$:

$$ppref@k = \frac{|\{(D_i|D_j) | (D_i, D_j) \in P_{\text{correct}} \text{ and } \min(\rho(D_i), \rho(D_j)) \leq k\}|}{|\{(D_i|D_j) | (D_i, D_j) \in P \text{ and } \min(\rho(D_i), \rho(D_j)) \leq k\}|}$$

Similarly, an analogue to recall at a cutoff $k$ ($R@k$) is $rpref@k$, which is the proportion of correctly ordered preferences to the total number of preferences made by assessors:

$$rpref@k = \frac{|\{(D_i|D_j) | (D_i, D_j) \in P_{\text{correct}} \text{ and } \min(\rho(D_i), \rho(D_j)) \leq k\}|}{|P|}$$
An analogue to average precision (AP) is Average Precision of Preferences (AP$_{pref}$) (Carterette and Bennett 2008), which is the average of $ppref$ values over the ranks (i.e. $k$) at which the recall of preferences ($r_{pref}$) increases:

$$AP_{pref} = \frac{\sum_{D \in P} p_{pref@r(D)}}{|P|}$$

where $P_r = \{D_i | (D_i, D_j) \in P \text{ and including } D_i \text{ in the ranking increases } r_{pref}\}$.

A modified version of AP$_{pref}$ (mAP$_{pref}$) was proposed by Elsas (2011), and used to evaluate the IR results from the initial experiments over the Ancestry dataset. mAP$_{pref}$ is the average of $ppref$ values over the ranks (i.e. $k$) of all documents which have ever been preferred to any other documents:

$$mAP_{pref} = \frac{\sum_{D \in P} p_{pref@r(D)}}{|P|}$$

where $P_+ = \{D_i | (D_i, D_j) \in P\}$

Additionally, $k$-fold cross-validation is also often used in IR experiments where parameters need to be tuned. In this context, the cross-validation is carried out over a set of queries. The training queries are first used to find the best parameter assignment. Then, the learned parameters are used to carry out retrieval over the test queries.

### 3.7.3 Hydrat

Hydrat (Lui and Baldwin 2010) is a declarative classification framework, which supports advanced data management and experimental results analysis. It is extensively used in this research, for running experiments, managing data and analysing results. Natively, it supports standard text classification, sequential classification/parsing,
and meta classification tasks, with great flexibility to extend to include more complex classification processes.

Hydrat is capable of handling all stages of a typical machine learning experiment, including raw document data preprocessing (to token stream representations such as BoW), feature engineering, cross-validation, meta-classification, and interactive results analysis (in web browsers). Hydrat provides the option to store data from different stages of an experiment separately, and uses a metadata system to avoid potentially expensive re-computation. Therefore users can trade off time and space. For example, when comparing different machine learners where other experimental settings stay the same, a user only needs to specify the machine learners to be tested. Hydrat will then automatically reuse preprocessed data from finished experiments (if the user has chosen to store the data), for stages such as raw data preprocessing and feature engineering of the current experiments. By avoiding this kind of re-computation, time and resources can be saved, especially for repetitive and time/resource intensive tasks. Moreover, the reduced computation can also minimise the chances of mistakes being introduced.

As a classification framework, Hydrat has great extensibility. It is easy to write wrappers to integrate existing natural language processing tools and machine learning packages. Hydrat also has an advanced dataset management system, which manages different datasets in an integrated framework. This makes it possible to conduct different experiments for different tasks in a unified system, thus making results analysis much easier. Hydrat implements many wrappers for existing natural language processing tools and machine learning packages which are used
in this research, such as the Porter Stemmer (Porter 1980), GENIA tagger (Tsuruoka et al. 2005), NLTK (Bird et al. 2009), WEKA (Hall et al. 2009), LIBSVM (Chang and Lin 2011), LIBLINEAR (Fan et al. 2008), CRFSGD (Bottou 2011), and CRF++ (Kudo 2010). Hydrat also contains native implementations of many natural language processing and machine learning techniques, including simple tokenisation, TF-IDF weighting, naive Bayes (McCallum and Nigam 1998), k-nearest neighbour algorithm, hidden Markov models and stacking.

Additionally, Hydrat is implemented with scalability in mind. It automatically uses sparse matrices to store sparse feature representations if necessary, and tries to carry out operations directly over sparse matrices. This feature not only saves disk space and memory, but also makes Hydrat capable of handling large datasets with sparse, high-dimensional features.

3.8 Summary

In this chapter, we detailed the datasets and software packages that are actively used in this thesis. For each software package, we also explained the related core empirical methodologies. With the datasets, we focused on the novel facets of the original datasets that are relevant to this research. For the software packages, we concentrated on explaining the related core empirical methodologies and functionality in the context of our research.

In the next three chapters, we will present details of our research in the domain of forums, over the three datasets described in this chapter. We start by describing forum thread structural analysis in Chapter 4.
Chapter 4

Forum Thread Structural Analysis

4.1 Introduction

As shown in Chapter 2, thread structural information, including thread linking structure and post/sub-post semantics, can be used in a range of applications such as forum thread and post retrieval (Xi et al. 2004; Duan and Zhai 2011; Seo et al. 2009; Bhatia and Mitra 2010; Wang et al. 2011a), knowledge base augmentation of chatbots (Feng et al. 2006b; Huang et al. 2007), post quality assessment (Lui and Baldwin 2009), and user profiling (Kim et al. 2006). In the subsection “Joint Thread Linking Structure and Semantics Recovery” of Section 2.2.3, we described Kim et al. (2010b)'s work on thread discourse structure, which was introduced in Chapter 1. Kim et al. (2010b) conducted preliminary experiments over the thread discourse structure parsing task, and tried to recover the links and identify dialogue acts separately. In Section 4.2, we will explore generalised approaches for joint classification of both links and dialogue acts. Moreover, we will investigate the
behaviour of the proposed approaches over dynamically evolving threads, where threads grow as new posts appear. Additionally, in Section 4.3, we will also look into unsupervised methods for recovering inter-post links, by using lexical chaining, as introduced in Section 2.5.

4.2 Thread Discourse Structure Analysis

4.2.1 Introduction

We first introduced the intuition of thread discourse structure in Chapter 1, and then described Kim et al. (2010b)’s work on thread discourse structure parsing as well as the proposed dialogue act set in Section 2.2.3. To illustrate the task in more detail, we use the same CNET example thread as used in Chapter 1 and Section 3.2, made up of 5 posts from 4 distinct participants, as shown in Figure 4.1. The discourse structure of the thread is modelled as a rooted directed acyclic graph (DAG) with a dialogue act label associated with each edge of the graph. In this example, User A initiates the thread with a question (dialogue act = Question-question) in the first post, by asking how to create an interactive input box on a webpage. In response, User B and User C provide independent answers (dialogue act = Answer-answer). User A responds to User C to confirm the details of the solution (dialogue act = Answer-confirmation), and at the same time, adds extra information to his/her original question (dialogue act = Question-add); i.e., this one post has two distinct dependency links associated with it. Finally, User D proposes a different solution again to the original question.
HTML Input Code

...Please can someone tell me how to create an input box that asks the user to enter their ID, and then allows them to press go. It will then redirect to the page ...

Re: html input code
Part 1: create a form with a text field. See ... Part 2: give it a Javascript action

asp.net c# video
I’ve prepared for you video.link click ...

Thank You!
Thanks a lot for that ... I have Microsoft Visual Studio 6, what program should I do this in? Lastly, how do I actually include this in my site? ...

A little more help
... You would simply do it this way: ... You could also just ... An example of this is ...

Figure 4.1: An example CNET thread with thread discourse structure

To predict thread discourse structure of this type, we jointly classify the links and dialogue acts between posts, experimenting with a variety of supervised classification methods, namely dependency parsing as described in Section 3.5, and linear-chain conditional random fields (CRFs) as described in Section 3.6. In this, we build on the earlier work of Kim et al. (2010b) which was described in Section 2.2.3, and used the CNET dataset as detailed in Section 3.2.1. Kim et al. (2010b)

1We only used the CNET dataset in this chapter, because the ILIAD dataset was annotated with dialogue acts after the research in this chapter was completed.
Chapter 4: Forum Thread Structural Analysis

first proposed the task of thread discourse analysis, but only carried out experiments on post linking and post dialogue act classification as separate tasks. In addition to achieving state-of-the-art accuracy over the task, we carry out in-depth analysis of classification effectiveness at different thread depths, and establish that the accuracy of our method over partial threads is equivalent to that over full threads, indicating that the method is applicable to in-situ thread classification. Finally, we investigate the role of user-level features in discourse structure analysis.

4.2.2 Task Description

The main task performed in discourse structure analysis is joint classification of inter-post links (Link) and dialogue acts (DA) within forum threads, as is shown in previous section. In this, we assume that a post can only link to an earlier post (or a virtual root node), and that dialogue acts are labels on edges. It is possible for there to be multiple edges from a given post, e.g. if a post both confirms the validity of an answer and adds extra information to the original question (as happens in Post4 in Figure 4.1). We use the CNET dataset of Kim et al. (2010b), which is described in Section 3.2. The dialogue act set used to annotate thread discourse structure is detailed in Table 2.4, which we reproduce as Table 4.1.

We experiment with two different approaches to joint classification: (1) a linear-chain CRF over combined Link/DA post labels; and (2) a dependency parser. As shown in Section 3.5, the joint classification task is a natural fit for dependency parsing, in that the task is intrinsically one of inferring labelled dependencies between posts, but it has a number of special properties that distinguish it from stan-
Table 4.1: The dialogue act (DA) tagset proposed by Kim et al. (2010b)

<table>
<thead>
<tr>
<th>Super-category</th>
<th>Sub-class</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Question</td>
<td>question</td>
<td>the post contains a new question which is independent of the posts before it.</td>
</tr>
<tr>
<td></td>
<td>add</td>
<td>the post provides additional information or asks a follow-up question, regarding a previous question.</td>
</tr>
<tr>
<td></td>
<td>confirmation</td>
<td>the post confirms details or error(s) in a question.</td>
</tr>
<tr>
<td></td>
<td>correction</td>
<td>the post corrects error(s) in a question.</td>
</tr>
<tr>
<td>Answer</td>
<td>answer</td>
<td>the post proposes an answer to a question.</td>
</tr>
<tr>
<td></td>
<td>add</td>
<td>the post provides additional information to an answer.</td>
</tr>
<tr>
<td></td>
<td>confirmation</td>
<td>the post confirms details or error(s) in an answer.</td>
</tr>
<tr>
<td></td>
<td>correction</td>
<td>the post corrects error(s) in an answer.</td>
</tr>
<tr>
<td></td>
<td>objection</td>
<td>the post objects to an answer.</td>
</tr>
<tr>
<td>Resolution</td>
<td>—</td>
<td>the initiator confirms that an answer works.</td>
</tr>
<tr>
<td>Reproduction</td>
<td>—</td>
<td>a non-initiator asks a similar question, or confirms that an answer should work.</td>
</tr>
<tr>
<td>Other</td>
<td>—</td>
<td>the post does not belong to any of the above classes.</td>
</tr>
</tbody>
</table>

strict reverse-chronological directionality: the head always precedes the dependent, in terms of the chronological sequencing of posts.

non-projective dependencies: threads can contain non-projective dependencies, e.g. in a 4-post thread, posts 2 and 3 may be dependent on post 1, and post 4 dependent on post 2; around 2% of the threads in our dataset contain non-projective dependencies.

multi-headedness: it is possible for a given post to have multiple heads, including the possibility of multiple dependency links to the same post (e.g. adding extra information to a question [Question-add] as well as retracting information from
the original question [Question-correction]; around 6% of the threads in our dataset contain multi-headed dependencies.

**disconnected sub-graphs:** it is possible for there to be disconnected sub-graphs, e.g. in instances where a user hijacks a thread to ask their own unrelated question, or submit an unrelated spam post; around 2% of the threads in our dataset contain disconnected sub-graphs.

The first constraint potentially simplifies dependency parsing, and non-projective dependencies are relatively well understood in the dependency parsing community (Tapanainen and Jarvinen 1997; McDonald et al. 2005). Multi-headedness and disconnected sub-graphs pose greater challenges to dependency parsing, although there has been research done on both (McDonald and Pereira 2006; Henderson et al. 2008; Sagae and Tsujii 2008; Eisner and Smith 2005). The combination of non-projectivity, multi-headedness and disconnected sub-graphs in a single dataset, however, poses a challenge for dependency parsing.

In addition to performing evaluation in batch mode over complete threads, we consider the task of “in situ thread classification”, whereby we predict the discourse structure of a thread after each post. This is intended to simulate the more realistic setting of incrementally crawling/updating thread data, but needing to predict discourse structure for partial threads. We are interested in determining the relative degradation in accuracy for in situ classification vs. batch classification.

Unless otherwise noted, evaluation is over the combined link and dialogue act tag, including the combination of superclass and subclass for the Question and An-

\(^2\)It is possible to combine multiple labels into one label in these cases. However, data scarcity would be an issue.
### Table 4.2: Thread length distribution of the CNET dataset

<table>
<thead>
<tr>
<th>Thread length</th>
<th>Count</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>105</td>
</tr>
<tr>
<td>3</td>
<td>59</td>
</tr>
<tr>
<td>4</td>
<td>57</td>
</tr>
<tr>
<td>5</td>
<td>25</td>
</tr>
<tr>
<td>6</td>
<td>18</td>
</tr>
<tr>
<td>7</td>
<td>10</td>
</tr>
<tr>
<td>8</td>
<td>14</td>
</tr>
<tr>
<td>9</td>
<td>7</td>
</tr>
<tr>
<td>10</td>
<td>4</td>
</tr>
<tr>
<td>11</td>
<td>4</td>
</tr>
<tr>
<td>12</td>
<td>3</td>
</tr>
<tr>
<td>13</td>
<td>7</td>
</tr>
<tr>
<td>14</td>
<td>2</td>
</tr>
</tbody>
</table>

To predict thread discourse structure, we use a structured classification approach — based on the findings of Kim et al. (2010b) and Kim et al. (2010a) — and a dependency parser. The structured classification approach we experiment with is a linear-chain conditional random field learner (CRF: Lafferty et al. (2001)), within which we explore two simple approaches to joint classification, as is explained in Section 4.2.4. Dependency parsing (Kübler et al. 2009) is the task of automatically
Table 4.3: Link distribution of the CNET dataset

<table>
<thead>
<tr>
<th>Link Count</th>
<th>DA</th>
<th>Count</th>
</tr>
</thead>
<tbody>
<tr>
<td>0 321</td>
<td>Question-question</td>
<td>316</td>
</tr>
<tr>
<td>1 801</td>
<td>Question-add</td>
<td>157</td>
</tr>
<tr>
<td>2 151</td>
<td>Question-correction</td>
<td>3</td>
</tr>
<tr>
<td>3 49</td>
<td>Question-confirmation</td>
<td>54</td>
</tr>
<tr>
<td>4 27</td>
<td>Answer-answer</td>
<td>560</td>
</tr>
<tr>
<td>5 17</td>
<td>Answer-add</td>
<td>108</td>
</tr>
<tr>
<td>6 11</td>
<td>Answer-objection</td>
<td>29</td>
</tr>
<tr>
<td>7 11</td>
<td>Answer-confirmation</td>
<td>14</td>
</tr>
<tr>
<td>8 4</td>
<td>Resolution</td>
<td>118</td>
</tr>
<tr>
<td>9 2</td>
<td>Reproduction</td>
<td>20</td>
</tr>
<tr>
<td>10 2</td>
<td>Other</td>
<td>18</td>
</tr>
<tr>
<td>11 1</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 4.4: DA distribution of the CNET dataset

predicting the dependency structure of a token sequence, in the form of binary asymmetric dependency relations with dependency types.

Standardly, CRFs have been applied to tasks such as part-of-speech tagging, named entity recognition, semantic role labelling and supertagging, where the individual tokens are single words. Similarly, dependency parsing is conventionally applied to sentences, with single-word tokens. In our case, our tokens are thread posts, with much greater scope for feature engineering than single words, and technical challenges in scaling the underlying implementations to handle potentially much larger feature sets.

As our learners, we deploy CRFSGD (Bottou 2011) to learn the CRF, and MaltParser (Nivre et al. 2007) as our dependency parser. CRFSGD, as described in Section 3.6.2, uses stochastic gradient descent to efficiently solve the convex optimisation problem, and scales well to large feature sets. We used the default parameter
settings\textsuperscript{3} for CRFSGD, as shown in Table 4.5, with feature templates including all unigram features of the current token as well as bigram features combining the previous output token with the current token.

MaltParser, as detailed in Section 3.5.2, implements transition-based parsing, where no formal grammar is considered, and a transition system, or state machine, is learned to map a sentence onto its dependency graph. One feature of MaltParser that makes it well suited to our task is that it is possible to define feature models of arbitrary complexity for each token. In presenting the thread data to MaltParser, we represent the null-link from the initial post of each thread, as well as any disconnected posts, as the root.

To the best of our knowledge, there is no past work on using dependency parsing to learn thread discourse structure. Based on extensive experimentation, we determined that the MaltParser configuration that obtains the best results for our task is the Nivre algorithm in arc-standard mode \cite{Nivre2003, Nivre2004}, using LIBSVM \cite{Chang2011} with a linear kernel as the learner, and the feature model summarised in Table 4.6. As such, MaltParser is actually unable to predict any non-projective structures, as experiments with algorithms supporting

\begin{table}[h]
\centering
\begin{tabular}{|l|c|}
\hline
Description & Value \\
\hline
Capacity control parameter & 1.0 \\
Threshold on the occurrences of each feature & 3 \\
Total number of epochs & 50 \\
Epochs between each testing phase & 5 \\
\hline
\end{tabular}
\caption{Parameter settings used for CRFSGD in this research}
\end{table}

\textsuperscript{3}We carried out preliminary parameter tuning in our experiments. However, it did not give us significantly better results when compared to using the default settings.
<table>
<thead>
<tr>
<th>Feature No.</th>
<th>Address Function</th>
<th>Attribute Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>STK[0]</td>
<td>RAW</td>
</tr>
<tr>
<td>2</td>
<td>STK[1]</td>
<td>RAW</td>
</tr>
<tr>
<td>3</td>
<td>STK[2]</td>
<td>RAW</td>
</tr>
<tr>
<td>4</td>
<td>BUF[0]</td>
<td>RAW</td>
</tr>
<tr>
<td>5</td>
<td>BUF[1]</td>
<td>RAW</td>
</tr>
<tr>
<td>6</td>
<td>BUF[2]</td>
<td>RAW</td>
</tr>
<tr>
<td>7</td>
<td>HEAD(STK[0])</td>
<td>RAW</td>
</tr>
<tr>
<td>8</td>
<td>STK[0]-STK[1]</td>
<td>RAW</td>
</tr>
<tr>
<td>9</td>
<td>STK[0]-STK[1]-STK[2]</td>
<td>RAW</td>
</tr>
<tr>
<td>10</td>
<td>BUF[0]-BUF[1]-BUF[2]</td>
<td>RAW</td>
</tr>
<tr>
<td>11</td>
<td>STK[0]-STK[1]-BUF[2]</td>
<td>RAW</td>
</tr>
<tr>
<td>12</td>
<td>STK[0]-BUF[1]-BUF[2]</td>
<td>RAW</td>
</tr>
<tr>
<td>13</td>
<td>RDEP(STK[0])</td>
<td>DEPREL</td>
</tr>
<tr>
<td>14</td>
<td>RDEP(STK[1])</td>
<td>DEPREL</td>
</tr>
<tr>
<td>15</td>
<td>STK[0]</td>
<td>DEPREL</td>
</tr>
<tr>
<td>16</td>
<td>STK[1]</td>
<td>DEPREL</td>
</tr>
<tr>
<td>17</td>
<td>STK[2]</td>
<td>DEPREL</td>
</tr>
</tbody>
</table>

Table 4.6: Feature model used with MaltParser in this research. As explained in Section 3.5, STK[i] locates the i-th post in the stack, BUF[i] locates the i-th post in the buffer, RDEP(w) locates the farthest child of post w to the right, and DEPREL represents the dependency label (dialogue act label in our case). Additionally, HEAD(w) locates the head of the w post in the stack, STK[i]-STK[j] merges the feature from STK[i] and the feature from STK[j] into one feature, and RAW represent all the raw structural and semantic features defined latter in the same section. Note that each entry from feature no. 1-12 creates one feature for each raw feature.
non-projective structures invariably led to lower results. In our choice of parsing algorithm, we are also unable to detect posts with multiple heads, but can potentially detect disconnected sub-graphs.

The raw features used in our classifiers are as follows:

**Structural Features:**

- **Initiator** a binary feature indicating whether the current post’s author is the thread initiator. Initiator takes a value of either 0 (not the initiator) or 1 (initiator).

- **Position** the relative position of the current post, as a ratio over the total number of posts in the thread. Position $\in [0, 1]$.

**Semantic Features:**

- **TitSim** the relative location of the post which has the most similar title (based on unweighted cosine similarity) to the current post. TitSim $\in [0, 1]$.

- **PostSim** the relative location of the post which has the most similar content (based on unweighted cosine similarity) to the current post. TitSim $\in [0, 1]$.

- **Punct** the number of question marks (QuCount), exclamation marks (ExCount) and URLs (UrlCount) in the current post. QuCount, ExCount, and UrlCount are three individual features which take a non-negative integer value.

- **UserProf** the class distribution (in the training thread) of the author of the current post. UserProf $= \{x_1, x_2, \ldots, x_n\}$, where $n$ is the number of distinct classes in

---

4We also tried Swap algorithms, which can handle non-projective structures, but no improvements were shown. One of the possible reasons is that there are not many threads (i.e. only 5 out of 315), which contain non-projective structures.
<table>
<thead>
<tr>
<th>Feature</th>
<th>Value</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Initiator</td>
<td>1.0</td>
<td>post from the initiator</td>
</tr>
<tr>
<td>ExCount</td>
<td>4.0</td>
<td>4 exclamation marks</td>
</tr>
<tr>
<td>QuCount</td>
<td>0.0</td>
<td>0 question marks</td>
</tr>
<tr>
<td>UrlCount</td>
<td>0.0</td>
<td>0 URLs</td>
</tr>
<tr>
<td>Position</td>
<td>0.25</td>
<td>(\frac{i-1}{n} = \frac{3-1}{8})</td>
</tr>
<tr>
<td>PostSim</td>
<td>2.0</td>
<td>most similar to post 1</td>
</tr>
<tr>
<td>TitSim</td>
<td>2.0</td>
<td>most similar to post 1</td>
</tr>
<tr>
<td>UserProf</td>
<td>(\bar{x})</td>
<td>counts for posts of each class from the same author in the training data</td>
</tr>
</tbody>
</table>

Table 4.7: The feature presentation of the third post in a thread of length 8

the training data, and \(x_i\) is the occurrence of this user’s posts which have a class label of \(i\).

These features are drawn largely from the work of Kim et al. (2010b), with two major differences: (1) we do not use post context features because our learners (i.e. CRFSGD and MaltParser) inherently capture Markov chains; and (2) our UserProf features are customised to the class set associated with the task at hand, e.g. the UserProf features for the standalone linking task take the form of the link labels (and not dialogue act labels) of the posts by the relevant author in the training data. Table 4.7 shows the feature representation of the third post in a thread of length 8. The values of each feature are scaled to the range \([0, 1]\) before being fed into the learners.
4.2.4 Classification Methodology

All our experiments were carried out based on stratified 10-fold cross-validation, stratifying at the thread level to ensure that all posts from a given thread occur in a single fold. The results are primarily evaluated using post-level micro-averaged F-score ($F_{\beta}: \beta = 1$),\(^5\) and additionally with thread-level F-score/classification accuracy (i.e. the proportion of threads where all posts have been correctly classified\(^6\)).

The detailed description of cross-validation and F-score were given in Section 3.7.2. Initial experiments indicated that it is hard for learners to discover which posts have multiple links, largely due to the sparsity of multi-headed posts (which account for less than 5% of the total posts). Therefore, only the most recent link for each multi-headed post was included in training\(^7\), but evaluation still considers all links.

Combined Classification of Link and DA

One way to achieve combined classification of Link and DA is joint classification. Joint classification has been applied in a number of different contexts, based on the intuition that it should be possible to harness interactions between different sub-tasks to the mutual benefit of both. For example, Warnke et al. (1997) jointly performed segmentation and dialogue act classification over a German spontaneous speech corpus. In their approach, the predictions of a multi-layer perceptron classifier on dialogue act boundaries were fed into an $n$-gram language model, which

---

\(^5\)It should be noted that Link $F_{\mu}$ is equivalent to unlabelled attachment score (UAS), and LinkDA $F_{\mu}$ is equivalent to labelled attachment score (LAS).

\(^6\)Classification accuracy = F-score at the thread-level, as each thread is assigned a single label of correct or incorrect.

\(^7\)If a link has more than one dialogue act, the dialogue acts are sorted alphabetically and only the first one is used.
was used for the joint segmentation and classification of dialogue acts. Sutton and McCallum (2005) performed joint parsing and semantic role labelling (SRL), using the results of a probabilistic SRL system to improve the accuracy of a probabilistic parser. Finkel and Manning (2009b) built a joint, discriminative model for parsing and named entity recognition (NER), addressing the problem of inconsistent annotations across the two tasks, and demonstrating that NER benefited considerably from the interaction with parsing. Dahlmeier et al. (2009) proposed a joint probabilistic model for word sense disambiguation (WSD) of prepositions and SRL of prepositional phrases (PPs), and achieved state-of-the-art results over both tasks.

In our experiments, we adopt two simpler compositional approaches for the CRF: (1) classifying the Link and DA separately, and composing the predictions to form the joint classification (Compose) and (2) combining the Link and DA labels into a single class, and applying the learner over the posts with the combined class (Combine). Note that Compose has the potential for mismatches in the number of Link and DA predictions it generates, causing complications in the class composition. Even if the same number of labels is predicted for both Link and DA, if multiple tags are predicted in both cases, we are left with the problem of determining which link label to combine with which dialogue act label. As such, we have our reservations about Compose, but as the CRF performs strict 1-of-n labelling, these are not issues in the experiments reported herein.

MaltParser natively handles the combination of Link and DA in its dependency parsing formulation.

---

8Another similar approach is to first classify the Link and then classify the DA of the Link. This idea has been explored by Kim et al. (2010b), and does not produce significantly better results.
In Situ Thread Classification

One of the biggest challenges in classifying the discourse structure of a forum thread is that threads evolve over time, as new posts are posted. As shown in Figure 3.1 of Section 3.2, the CNET example thread in Figure 4.1 has grown since the crawl of Kim et al. (2010b). In more popular forums such as the Apple Discussion forum, as shown in Chapter 1, threads can be actively evolving for many months or even years. A natural question following this phenomenon is whether the proposed discourse structure parsing models can cope with these dynamically growing threads. More specifically, when the “full” thread is not available yet, will the proposed method accurately parse partial threads? To examine this concern, we conduct experiments to compare the accuracy of different models when applied to partial thread data (artificially cutting off a thread at post N) vs. complete threads.\footnote{In practice, completeness is defined at a given point in time, when the crawl was done, and it is highly likely that some of the “complete” threads had extra posts after the crawl.}

This is done in the following way: classification over the first two posts only ([1, 2]), the first four posts ([1, 4]), the first six posts ([1, 6]), the first eight posts ([1, 8]), and all posts ([all]). In each case, we limit the test data only, meaning that the only variable in play is the extent of thread context used to learn the thread discourse structure for the given set of posts. We break down the results in each case into the indicated sub-threads, e.g. we take the predictions for [all], and break them down into the results for [1, 2], [1, 4], [1, 6], [1, 8] and [all], for direct comparison with the predictions over the respective sub-thread data.
Table 4.8: Post/thread-level component-wise classification F-scores for Link and DA classes

<table>
<thead>
<tr>
<th>Method</th>
<th>Link</th>
<th>DA</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Kim et al. (2010b)</strong></td>
<td>.863</td>
<td>.751</td>
</tr>
<tr>
<td>CRFSGD</td>
<td>.815</td>
<td>.750</td>
</tr>
</tbody>
</table>

4.2.5 Results and Evaluation

Joint classification

As our baseline for the task, we first use a simple majority class classifier in the form of the single joint class of 1+Answer-answer for all posts, which has a post-level F-score of 0.284. A stronger baseline is to classify all first posts as 0+Question-question and all subsequent posts as 1+Answer-answer, which achieves a post-level F-score of 0.515 (labelled as Heuristic).

As described in Section 4.2.4, one approach to joint classification with CRFSGD is to firstly conduct component-wise classification over Link and DA separately, and compose the predictions. The results for the separate Link and DA classification tasks are presented in Table 4.8, along with the best results for Link and DA classification from Kim et al. (2010b).

Next, we compose the component-wise classifications for the CRF into joint classifications (Composition). We contrast this with the combined class approach for CRFSGD and MaltParser (jointly presented as Joint in Table 4.9). With the combined class results, we additionally ablate each of the feature types from Section 4.2.3, and also present results for a dummy model, where no features are provided and
### Table 4.9: Post/thread-level LinkDA joint classification F-scores ("*" signifies a significantly worse result than that for the same learner with ALL features)

<table>
<thead>
<tr>
<th>Method</th>
<th>CRFSGD</th>
<th>MaltParser</th>
</tr>
</thead>
<tbody>
<tr>
<td>Heuristic</td>
<td>.515*</td>
<td>.311*</td>
</tr>
<tr>
<td>Dummy</td>
<td>.508*/</td>
<td>.394*</td>
</tr>
<tr>
<td>Composition</td>
<td>.627 /</td>
<td>.498</td>
</tr>
<tr>
<td>Joint +ALL</td>
<td>.619*/</td>
<td>.484*</td>
</tr>
<tr>
<td>—Initiator</td>
<td>.554*</td>
<td>.403*</td>
</tr>
<tr>
<td>—Position</td>
<td>.603 /</td>
<td>.451</td>
</tr>
<tr>
<td>—PostSim</td>
<td>.602 /</td>
<td>.463</td>
</tr>
<tr>
<td>—TitSim</td>
<td>.598 /</td>
<td>.467</td>
</tr>
<tr>
<td>—Punct</td>
<td>.605 /</td>
<td>.467</td>
</tr>
<tr>
<td>—UserProf</td>
<td>.665 /</td>
<td>.524</td>
</tr>
</tbody>
</table>

the prediction is based simply on sequential priors/labels of posts in the training data (Dummy). That is, each post has no feature representation, and classification is based solely on the structured prior in the training data. The results are presented in Table 4.9, along with the Heuristic baseline result.

Several interesting things can be observed from the post-level F-score results in Table 4.9. First, with no features (Dummy), while CRFSGD performs slightly worse than the Heuristic baseline, MaltParser significantly surpasses the baseline. This is due to the richer sequential context model of MaltParser. Second, the single feature with the greatest negative impact on results is UserProf, i.e. user profile features extracted from the training data. We return to explore this effect later in this section. Third, the single feature with the greatest positive impact on results is Initiator. Further experiments show that the Initiator feature by itself is sufficient to achieve a competitive result (i.e. F-score of 0.678).
Looking to the thread-level F-scores, we observe some interesting divergences from the post-level F-score results. First, with no features (Dummy), CRFSGD significantly outperforms both the baseline and MaltParser. This appears to be because CRFSGD performs particularly well over short threads (e.g. of length 3 and 4), but worse over longer threads. Second, the best thread-level F-scores from CRFSGD (i.e. 0.524) and MaltParser (i.e. 0.540) are not significantly different, despite the discrepancy in post-level F-score (where MaltParser is markedly superior in this case). With the extra features, the performance of MaltParser on short threads appears to pick up noticeably, and the difference in post-level predictions is over longer threads.

If we evaluate the two models over DA superclasses only (ignoring mismatches at the subclass level for Question and Answer), the post-level F-scores for joint classification, with ALL features excluding UserProf, for CRFSGD and MaltParser are 0.703 and 0.731, respectively.

Looking at the performance of CRFSGD (in Combined mode) and MaltParser on disconnected sub-graphs, while both models did predict a small number of non-initial posts with null-links (including MaltParser predicting 5 out of 6 posts in a single thread as having null-links), none were correct, and neither model was able to correctly predict any of the 6 actual non-initial instances of null-links in the dataset.

Finally, we took the joint classification results from CRFSGD and MaltParser using ALL features except for UserProf, and decomposed the predictions into Link and DA. The results are presented in Table 4.10, along with the results for component-wise classification from Table 4.8. As shown in Table 4.10, the decomposed predic-
Chapter 4: Forum Thread Structural Analysis

Table 4.10: Post/thread-level Link and DA F-scores from component-wise classification and from LinkDA classification decomposition

<table>
<thead>
<tr>
<th>Approaches</th>
<th>Link</th>
<th>DA</th>
</tr>
</thead>
<tbody>
<tr>
<td>Component-wise</td>
<td>.815 / .638</td>
<td>.750 / .581</td>
</tr>
<tr>
<td>CRFSGD decomp</td>
<td>.823 / .654</td>
<td>.734 / .571</td>
</tr>
<tr>
<td>MaltParser decomp</td>
<td>.850 / .695</td>
<td>.757 / .565</td>
</tr>
<tr>
<td>Kim et al. (2010b)</td>
<td>.863 / .676</td>
<td>.751 / .543</td>
</tr>
</tbody>
</table>

Post Position-based Result Breakdown

One question in thread discourse structure classification is how accurate the predictions are at different depths in a thread (e.g. the first two posts vs. the second two posts). A breakdown of results across posts at different positions is presented in Figure 4.2.

The overall trend for both CRFSGD and MaltParser is that it becomes increasingly hard to classify posts as we continue through a thread, due to greater variability in discourse structure and greater sparsity in the data. In general, the greater the depth, the worse the results. This is largely similar to the findings in parsing literature. However, it is interesting to note that the results for CRFSGD stay similar from posts 7 and 8 ([7,8]) to posts 9 and onwards ([9,]). To further investigate this

---

Section 4.2 is based on our paper Wang et al. (2011b). A mistake in the original experiments was found after the paper was published, and we updated all the numbers based on new experiments. This update breaks the flow of the writing in some places. Another by-product of the update is that, in Table 4.10, our best Link results are worse than that of Kim et al. (2010b).
effect, we performed class decomposition over the joint classification predictions, and performed a similar breakdown of posts for Link and DA; the results are presented in Figure 4.3. It is clear that the anomaly comes from the DA component. Interestingly, the DA results for both CRFSGD and MaltParser increase slightly from posts 7 and 8 ([7, 8]) to posts 9 and onwards ([9, ]). This is due to there being greater predictability in the dialogue for final posts in a thread (users tend to confirm a successful resolution of the problem, or report on successful external reproduction of the solution). Moreover, although MaltParser outperforms CRFSGD in almost...
all results in Figure 4.3, CRFSGD performs slightly better than MaltParser on Link classification over posts 9 and onwards ([9,]). This observation is congruous with the findings of McDonald and Nivre (2007) that errors propagate, due to MaltParser’s greedy inference strategy. Additionally, we carried out error analysis using the confusion matrix, but could not find any systematic errors.
Table 4.11: Post-level LinkDA F-score for CRFSGD/MaltParser, based on in situ classification over sub-threads of different lengths (indicated in the rows), broken down over different post extents (indicated in the columns)

In Situ Structure Prediction

As described in Section 4.2.4, we simulate in situ thread discourse structure prediction by removing differing numbers of posts from the tail of the thread, and applying the trained model over the resultant sub-threads. The results for in situ classification are presented in Table 4.11, with the rows indicating the size of the test sub-thread, and the columns being a breakdown of results over different portions of the classified thread. We do not provide numbers for all cells in the table. This is because the size of the test sub-thread determines the post extents we can breakdown the results into. For example, we cannot return results for posts 1–4 ([1,4]) when the size of the test thread was only two posts ([1,2]). From the results, we can see that both CRFSGD and MaltParser are very robust when applied to partial threads, to the extent that we actually achieve higher results over shortened versions of the thread than over the complete thread in some instances. This indicates that, even without the context of the “full” threads, both CRFSGD and MaltParser
can parse the partial threads at least as accurately as with the “full” thread context. Therefore, even applying the proposed models over forums such as the Apple Discussion forum, where there are many active incomplete threads, the proposed models can perform effectively. From this, we can conclude that it is possible to apply our method to partial threads without any reduction in effectiveness relative to classification over complete threads. As such, our method is shown to be robust when applied to real-time analysis of dynamically evolving threads. Additionally, it is not surprising to see that as we get deeper in the thread, the results become worse, as there are more possibilities for both Link and DA.

**User profile feature analysis**

In our experiments, we noticed that the user profile feature (UserProf) has a significant negative effect on both CRFSGD and MaltParser. To gain a deeper insight into the behaviour of the feature, we binned the posts according to the number of times the author had posted in the training data, evaluated based on a user score (uscore) for each user:

\[
uscore_i = \frac{\sum_{j=1}^{n_i} s_{p_{i,j}}}{n_i}
\]

where \(n_i\) is the number of posts by user \(i\), and \(s_{p_{i,j}}\) is the number of posts by user \(i\) that occur as training instances for other posts by the same author. uscore reflects the average training–test post ratio per user in cross-validation. Note that as we include all posts from a given thread in a single partition during cross-validation, it is possible for an author to have posted 4 times, but have a uscore of 0 due to those posts all occurring in the same thread.
We ranked the users in the dataset in descending order of uscore, sub-ranking on \( n_i \) in cases of a tie in uscore. The users were binned into 4 groups of roughly equal post size. The detailed statistics are shown in Table 4.12, noting that the high-frequency bin ("High") contains posts from a single user. We present the post-level micro-averaged F-score for posts in each bin based on MaltParser, with and without user profile features, in Figure 4.4.

From Figure 4.4, we can see that the UserProf feature has a more detrimental effect over the Medium group, especially for the Low group where discourse parsing is the most difficult. It seems that the UserProf cannot help in all four user groups, and only introduces noise to confuse our classifiers. All in all, UserProf is not successful. Also, because it is represented as a vector of features, which implicitly has high weight due to its length, it introduces a significant negative effect. As future work, we could use a binary alternative to indicate whether the user used a particular DA more than \( n \) times in the training data.

<table>
<thead>
<tr>
<th>Bin</th>
<th>uscore per user</th>
<th>Total users</th>
<th>Total posts</th>
</tr>
</thead>
<tbody>
<tr>
<td>High</td>
<td>224.6</td>
<td>251</td>
<td>1</td>
</tr>
<tr>
<td>Medium</td>
<td>1~41.7</td>
<td>4~48</td>
<td>45</td>
</tr>
<tr>
<td>Low</td>
<td>0</td>
<td>2~4</td>
<td>157</td>
</tr>
<tr>
<td>Very Low</td>
<td>0</td>
<td>1</td>
<td>309</td>
</tr>
</tbody>
</table>

Table 4.12: Statistics for the 4 groups of users
4.2.6 Summary of Thread Discourse Structure Analysis

In this research, we explored the joint classification of web user forum thread discourse structure, in the form of a rooted directed acyclic graph over posts, with edges labelled with dialogue acts. Three classification approaches were proposed: separately predicting Link and DA labels, and composing them into a joint class; predicting a combined LinkDA class using a structured classifier; and applying dependency parsing to the problem. We found the combined approach based on de-
pendency parsing with MaltParser performs the best over the task, closely followed by CRFSGD.

We also examined the task of in situ classification of dialogue structure, in the form of predicting the discourse structure of partial threads, as contrasted with classifying only complete threads. We found that there was no drop in F-score over different sub-extents of the thread in classifying partial threads, despite the relative lack of thread context.

In summary, the proposed generalised models can not only reasonably accurately parse the discourse structure of static threads, but also be used effectively over active forums to parsing the discourse structure of dynamically evolving threads. With thread discourse structure parsed, the next step is to investigate its utility in the context of helping users better access information in forums. However, before that, we present a more speculative line of research, which explores thread linking structure recovery using lexical chaining, as described in Section 2.5.

4.3 Thread Linking Structure Analysis by Lexical Chaining

One of the biggest drawbacks of the thread discourse structure parsing models presented in Section 4.2 is its need for manually annotated training data, which are difficult to obtain. While identifying the dialogue act component of the thread discourse structure definitely requires annotated data, it is possible to use unsupervised approaches to recover only the linking structure component, as demonstrated
in previous research summarised in “Thread Linking Structure Recovery” of Section 2.2.3. This section explores a novel unsupervised approach to forum thread linking structure recovery, by using lexical chaining to analyse the inter-post lexical cohesion. Lexical chaining, as explained in Section 2.5, is a technique which identifies lists of related word tokens within a given discourse.

4.3.1 Introduction of Linking Analysis by Lexical Chaining

To illustrate the task of thread linking recovery, we use the same thread example as shown in Section 4.2.1. However, only Link labels are considered in this task, as indicated in Figure 4.5. The linking structure of the thread is modelled as a rooted directed acyclic graph (DAG). In this example, UserA initiates the thread with a question in the first post, by asking how to create an interactive input box on a webpage. This post is linked to a virtual root with link label 0. In response, UserB and UserC provide independent answers. Therefore their posts are linked to the first post, with link labels 1 and 2 respectively. UserA responds to UserC (Link = 1) to confirm the details of the solution, and at the same time, adds extra information to his/her original question (Link = 3); i.e., this one post has two distinct links associated with it. Finally, UserD proposes a different solution again to the original question (Link = 4).

We investigate three lexical chaining algorithms, including one that only uses statistical associations between words, which can be calculated from the raw text of the targeted domain. We show our method to be empirically superior to an informed baseline.
4.3.2 Lexical Chaining Algorithms

Three lexical chaining algorithms are experimented with in this research, as detailed in the following sections.

Method 1: Chainer\textsubscript{Roget}

Chainer\textsubscript{Roget} is a Roget’s Thesaurus based lexical chaining algorithm (Jarmasz and Szpakowicz 2003) based on an off-the-shelf package, namely the Electronic
Algorithm 1 \textit{Chainer}_{Roget}

Select a set of candidate nouns

\textbf{for} each candidate noun \textbf{do}

Build all the possible chains, where each pair of nouns in each chain are either the same word or included in the same \textit{Head} of \textit{Roget's Thesaurus}, and select the strongest chain for each candidate noun.

\textbf{end for}

Merge two chains if they contain at least one noun in common

Lexical Knowledge Base (ELKB) (Jarmasz and Szpakowicz 2001).

The underlying methodology of \textit{Chainer}_{Roget} is shown in Algorithm 1. Methods used to calculate the chain strength/weight are presented in Section 4.3.3. While the original Roget’s Thesaurus-based algorithm by Morris and Hirst (1991) proposes five types of thesaural relations to add a candidate word in a chain, \textit{Chainer}_{Roget} only uses the first one, as detailed in Algorithm 1. Moreover, while Jarmasz and Szpakowicz (2003) use the 1987 Penguin’s Roget’s Thesaurus in their research, the ELKB package uses the Roget’s Thesaurus from 1911 due to copyright restrictions.

\textbf{Method 2: Chainer}_{WN}

\textit{Chainer}_{WN} is a non-greedy WordNet-based chaining algorithm proposed by Galley and McKeown (2003). We reimplemented their method based on an incomplete implementation in NLTK.}

The algorithm of \textit{Chainer}_{WN} is based on the assumption of one sense per dis-

\footnote{http://people.virginia.edu/~ma5ke/classes/files/cs65lexicalChain.pdf}
course, and can be decomposed into three steps. Firstly, a “disambiguation graph” is built by adding the candidate nouns of the discourse one by one. Each node in the graph represents a noun instance, which is divided into as many senses as the noun has, and each weighted edge represents the semantic relation (based on WordNet) between two senses of two nouns. The weight of each edge is calculated based on the distance between nouns in the discourse, as well as the kind of semantic relationship between the two senses. Secondly, word sense disambiguation (WSD) is performed. In this step, a score is calculated for every sense of each noun node, by summing the weight of all edges emanating from that sense. The sense of the noun node with the highest score is considered as the correct sense of this noun in the discourse. Lastly, all the edges of the disambiguation graph connecting (assumed) wrong senses of every noun node are removed, and the remaining edges linking noun nodes form the lexical chains of the discourse. The semantic relations exploited in this algorithm include hypernyms/hyponyms and siblings (i.e. hyponyms of hypernyms).

Method 3: Chainer$_{SV}$

Chainer$_{SV}$, as shown in Algorithm 2, is adapted from Marathe and Hirst (2010)’s lexical chaining algorithm. The main difference between Chainer$_{SV}$ and the original algorithm is the method used to calculate associations between words. Marathe and Hirst (2010) use two different measures, including Lin (1998b)’s WordNet-based measure, and Mohammad and Hirst (2006)’s distributional measures of concept distance framework. In Chainer$_{SV}$, we use word vectors from WORDSPACE
Algorithm 2 ChainersSV

\[
\text{chains} = \text{empty}
\]
Select a set of candidate tokens

\[
\text{for each candidate token } t_i \text{ do}
\]
\[
\text{max}_\text{score} = \max_{c_j \in \text{chains}}(\text{sim}_{\text{sv}}(t_i, c_j))
\]
\[
\text{max}_\text{chain} = \arg \max_{c_j \in \text{chains}}(\text{sim}_{\text{sv}}(t_i, c_j))
\]
\[
\text{if } \text{chains} = \text{empty} \text{ or } \text{max}_\text{score} < \text{threshold} \text{ then}
\]
Create a new chain \(c_k\) containing \(t_i\) and add \(c_k\) to \(\text{chains}\)

\[
\text{else if more than one } \text{max}_\text{chain} \text{ then}
\]
Merge chains if the two chains’ similarity is larger than \(\text{threshold}_{\text{sv}}\), and add \(t_i\) to the resultant chain or the first \(\text{max}_\text{chain}\)

\[
\text{else}
\]
Add \(t_i\) to the \(\text{max}_\text{chain}\)

\[
\text{end if}
\]
\[
\text{end for}
\]
\[
\text{return } \text{chains}
\]

(\text{Schütze 1998}) models and apply cosine similarity to compute the associations between words. WORDSPACE is a multi-dimensional real-valued space, where words, contexts and senses are represented as vectors. A vector for word \(w\) is derived from words that co-occur with \(w\). A dimensionality reduction technique is often used to reduce the dimensionality of the vector. We build the WORDSPACE model with SemanticVectors (\text{Widdows and Ferraro 2008}), which is based on Random Projection dimensionality reduction (\text{Bingham and Mannila 2001}).
The underlying methodology of ChainerSV is shown in Algorithm 2. This algorithm requires a method to calculate the similarity between two tokens (i.e. words): $sim_{tt}(x, y)$, which is done by computing the cosine similarity of the two tokens’ semantic vectors. The similarity between a token $t_i$ and a lexical chain $c_j$ is then calculated by:

$$sim_{tc}(t_i, c_j) = \sum_{t_k \in c_j} \frac{1}{l_j} sim_{tt}(t_i, t_k)$$

where $l_j$ represents the length of lexical chain $c_j$. The similarity between two chains $c_i$ and $c_j$ is then computed by:

$$sim_{cc}(c_i, c_j) = \sum_{t_m \in c_i, t_n \in c_j} \frac{1}{l_i \times l_j} sim_{tt}(t_m, t_n)$$

where $l_i$ and $l_j$ are the lengths of $c_i$ and $c_j$ respectively.

As is shown in Algorithm 2, ChainerSV has two parameters: the threshold for adding a token to a chain, threshold$_a$; and the threshold for merging two chains, threshold$_m$. A larger threshold$_a$ leads to conservative chains where tokens in a chain are strongly related, while a smaller threshold$_a$ results in longer chains where the relationship between tokens in a chain may not be clear. Similarly, a larger threshold$_m$ is conservative and leads to less chain merging, while a smaller threshold$_m$ may create longer but less meaningful chains. Our initial experiments show that the combination of threshold$_a = 0.1$ and threshold$_m = 0.05$ often results in lexical chains with reasonable length and interpretability. Therefore, this parameter setting will be used throughout all the experiments described in this thesis.
4.3.3 Methodology

To the best of our knowledge, no previous research has adopted lexical chaining to predict inter-post links. The basic idea of our approach is to use lexical chains to measure the inter-post lexical cohesion (i.e. lexical similarity), and use these similarity scores to reconstruct inter-post links. To measure the lexical cohesion between two posts, the texts (with stop words and punctuations removed) from the titles and bodies of the two posts are first combined. Then, lexical chainers are applied over the combined texts to extract lexical chains. Lastly, the following weighting methods are used to calculate the lexical similarity between the two posts:

**LCNum**: the number of the lexical chains which span the two posts.

**LCLen**: find the lexical chains which span the two posts, and use the sum of tokens contained in each as the similarity score.

**LCStr**: find the lexical chains which span the two posts, and use the sum of each chain’s chain strength as the similarity score. The chain strength is calculated using a formula suggested by Barzilay and Elhadad (1997):

\[
\text{Score(Chain)} = \text{Length} \times \text{Homogeneity}
\]

where \(\text{Length}\) is the number of tokens in the chain, and:

\[
\text{Homogeneity} = 1 - \frac{\text{number of distinct token occurrences}}{\text{Length}}
\]

**LCBan**: find the lexical chains which span the two posts, and use the sum of each chain’s balance score as the similarity score. The balance score is calculated
using the following formula:

\[
Score(\text{Chain}) = \begin{cases} 
\frac{n_1}{n_2} & \text{if } n_1 < n_2 \\
\frac{n_2}{n_1} & \text{else}
\end{cases}
\]

where \(n_1\) is the number of tokens from the chain belonging to the first post, and \(n_2\) is the number of tokens from the chain belonging to the second post.

### 4.3.4 Assumptions, Experiments and Analysis

We conducted experiments over the CNET dataset, as used in Section 4.2, and the same evaluation metrics (i.e. \(P_\mu\), \(R_\mu\) and \(F_\mu\) as described in Section 4.2.4) are used for evaluating results over links only. We also use the same informed heuristic (Heuristic) baseline, as mentioned in Section 4.2.5, where all first posts are labelled with link 0 (i.e. link to a virtual root) and all other posts are labelled with link 1 (i.e. link to the immediately preceding post).

Furthermore, as explained in Section 4.2.4, it is possible for there to be multiple links from a given post in the CNET dataset. Although additional thresholds can be added to address posts with multiple links, these kinds of posts, which account for less than 5% of the total posts, are sparse in the dataset. Therefore, we only consider recovering one link per post in our experiments. However, our evaluation still considers all links (meaning that it is not possible for our methods to achieve an F-score of 1.0).
Chapter 4: Forum Thread Structural Analysis

Classifier Weighting $P_\mu$ $R_\mu$ $F_\mu$

<table>
<thead>
<tr>
<th>Heuristic</th>
<th>Weighting</th>
<th>$P_\mu$</th>
<th>$R_\mu$</th>
<th>$F_\mu$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Chainer$_{Roget}$</td>
<td>LCNum</td>
<td>.755</td>
<td>.720</td>
<td>.737</td>
</tr>
<tr>
<td></td>
<td>LCLen</td>
<td>.737</td>
<td>.703</td>
<td>.720</td>
</tr>
<tr>
<td></td>
<td>LCStr</td>
<td>.802</td>
<td>.764</td>
<td>.783</td>
</tr>
<tr>
<td></td>
<td>LCBan</td>
<td>.723</td>
<td>.689</td>
<td>.706</td>
</tr>
<tr>
<td>Chainer$_{WN}$</td>
<td>LCNum</td>
<td>.685</td>
<td>.644</td>
<td>.660</td>
</tr>
<tr>
<td></td>
<td>LCLen</td>
<td>.676</td>
<td>.651</td>
<td>.667</td>
</tr>
<tr>
<td></td>
<td>LCStr</td>
<td>.718</td>
<td>.685</td>
<td>.701</td>
</tr>
<tr>
<td></td>
<td>LCBan</td>
<td>.683</td>
<td>.651</td>
<td>.667</td>
</tr>
<tr>
<td>Chainer$_{SV}$</td>
<td>LCNum</td>
<td>.648</td>
<td>.618</td>
<td>.632</td>
</tr>
<tr>
<td></td>
<td>LCLen</td>
<td>.630</td>
<td>.601</td>
<td>.615</td>
</tr>
<tr>
<td></td>
<td>LCStr</td>
<td>.627</td>
<td>.598</td>
<td>.612</td>
</tr>
<tr>
<td></td>
<td>LCBan</td>
<td>.645</td>
<td>.615</td>
<td>.630</td>
</tr>
</tbody>
</table>

Table 4.13: Results from the Assumption 1 based unsupervised approach, by using three lexical chaining algorithms with four different weighting schemes.

Initial Assumption and Experiments

We observe that in web user forum threads, if a post replies to a preceding post, the two posts are usually semantically related and lexically similar. Based on this observation, we make the following assumption:

**Assumption 1.** *A post should be similar to the preceding post it is linked to.*

This assumption leads to our first unsupervised model, which compares each post (except for the first and second) in a given thread with all its preceding posts one by one, by firstly identifying the lexical chains using the lexical chainers described in Section 4.3.2 and then calculating the inter-post lexical similarity using the methods explained in Section 4.3.3. The experimental results are shown in Table 4.13.
From Table 4.13 we can see that no results surpass the Heuristic baseline. Further investigation reveals that while Assumption 1 is reasonable, it is not always correct—i.e. similar posts are not always linked together. For example, an answer post later in a thread might be linked back to the first question post but be more similar to preceding answer posts, to which it is not linked, simply because they are all answers to the same question. The initial experiments show that more careful analysis is needed to use inter-post lexical similarity to reconstruct inter-post linking.

Post 3 Analysis

Because Post 1 and Post 2 are always labelled with link 0 and 1 respectively, our analysis starts from Post 3 of each thread. Based on the analysis, the second assumption is made:

**Assumption 2.** If the Post 3 vs. Post 1 lexical similarity is larger than the Post 2 vs. Post 1 lexical similarity, then Post 3 is more likely to be linked back to Post 1.

Assumption 2 is based on the observation described in the preceding subsection. The intuition is that because Post 2 replied to Post 1, they are usually relevant and coherent (i.e. Post 2 vs. Post 1 lexical similarity is large). If Post 3 is even more relevant to Post 1 (i.e. Post 3 vs. Post 1 lexical similarity is even larger), then it is more likely that Post 3 replied to Post 1. This assumption leads to an unsupervised approach which combines the three lexical chaining algorithms introduced in Section 4.3.2 with the four weighting schemes explained in Section 4.3.3 to measure Post 3 vs. Post 1 similarity and Post 2 vs. Post 1 similarity. If the former is larger, Post 3 is linked back to Post 1, otherwise Post 3 is linked back to Post 2. As for the
Classifier Weighting $P_\mu$, $R_\mu$, $F_\mu$

<table>
<thead>
<tr>
<th>Classifier</th>
<th>Weighting</th>
<th>$P_\mu$</th>
<th>$R_\mu$</th>
<th>$F_\mu$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Heuristic</td>
<td>—</td>
<td>.810</td>
<td>.772</td>
<td>.791</td>
</tr>
<tr>
<td>Chainer$_{Roget}$</td>
<td>LNum</td>
<td>.811</td>
<td>.773</td>
<td>.791</td>
</tr>
<tr>
<td></td>
<td>LLen</td>
<td>.811</td>
<td>.773</td>
<td>.791</td>
</tr>
<tr>
<td></td>
<td>LStr</td>
<td>.810</td>
<td>.772</td>
<td>.791</td>
</tr>
<tr>
<td></td>
<td>LCBan</td>
<td>.813</td>
<td>.775</td>
<td>.794</td>
</tr>
<tr>
<td>Chainer$_{WN}$</td>
<td>LNum</td>
<td>.806</td>
<td>.768</td>
<td>.786</td>
</tr>
<tr>
<td></td>
<td>LLen</td>
<td>.806</td>
<td>.769</td>
<td>.787</td>
</tr>
<tr>
<td></td>
<td>LStr</td>
<td>.806</td>
<td>.769</td>
<td>.787</td>
</tr>
<tr>
<td></td>
<td>LCBan</td>
<td>.809</td>
<td>.771</td>
<td>.789</td>
</tr>
<tr>
<td>Chainer$_{SV}$</td>
<td>LNum</td>
<td>.813</td>
<td>.775</td>
<td>.794</td>
</tr>
<tr>
<td></td>
<td>LLen</td>
<td>.813</td>
<td>.775</td>
<td>.794</td>
</tr>
<tr>
<td></td>
<td>LStr</td>
<td>.816</td>
<td>.778</td>
<td>.797</td>
</tr>
<tr>
<td></td>
<td>LCBan</td>
<td>.818</td>
<td>.780</td>
<td>.799</td>
</tr>
</tbody>
</table>

Table 4.14: Results from the Assumption 2 based unsupervised approach, by using three lexical chaining algorithms with four different weighting schemes.

other posts, the link labels are the same as the ones from the Heuristic baseline. The experimental results are shown in Table 4.14.

From the results in Table 4.14, we can see that Chainer$_{SV}$ is the only lexical chaining algorithm that leads to results which are better than the Heuristic baseline. Analysis over the lexical chains generated by the three lexical chainers shows that both Chainer$_{Roget}$ and Chainer$_{WN}$ extract very few chains, most of which contain only repetitions of the same word. This is probably because these two lexical chainers only consider nouns, and therefore have limited input tokens. This is particularly the case for Chainer$_{Roget}$ which uses an old dictionary (1911 edition) that does not contain modern technical terms, such as Windows, OSX and PC. While Chainer$_{WN}$ uses WordNet, which has a larger and more modern vocabulary, the chainer considers very limited semantic relations (i.e. hypernyms, hyponyms, and hyponyms.
of hypernyms). Moreover, the texts in forum posts are usually relatively short and informal, and contain typos and non-standard acronyms. These factors make it very difficult for $Chainer_{Roget}$ and $Chainer_{WN}$ to extract lexical chains. As for $Chainer_{SV}$, because all the words (except for stop words) are considered as candidate words, and relations between words are flexible according to the thresholds (i.e. $threshold_a$ and $threshold_m$), relatively abundant lexical chains are generated. While some of the chains clearly capture lexical cohesion among words, others are hard to interpret. Nevertheless, the results from $Chainer_{SV}$ are encouraging for an unsupervised approach, and therefore further investigation is conducted using only $Chainer_{SV}$.

Because the experiments based on the Assumption 2 lead to promising results, further analysis is conducted to refine this assumption. We notice that the posts from the initiator of a thread are often outliers compared to other posts — i.e. these posts are similar to the first post because they are from the same author, but at the same time an initiator rarely replies to his/her own posts. This observation leads to a stricter assumption:

**Assumption 3.** *If Post 3 vs. Post 1 lexical similarity is larger than Post 2 vs. Post 1 lexical similarity and Post 3 is not posted by the initiator of the thread, then Post 3 is more likely to be linked back to Post 1.*

Based on Assumption 3, experiments are carried out using $Chainer_{SV}$ with different weighting schemes. We also introduce a stronger baseline ($Heuristic_{user}$) based on Assumption 3, where Post 3 is linked to Post 1 if these two posts are from different users, and all the other posts are linked as Heuristic. The experimental results are shown in Table 4.15.
Table 4.15: Results from the Assumption 3 based unsupervised approach, by using Chainer SV with different weighting schemes.

From Table 4.15 we can see that while all the results from Chainer SV are significantly better than the result from the Heuristic baseline, with the LCBan weighting leading to the best $F_\mu$ of 0.816, these results are not significantly different to the Heuristic user baseline. It is clear that the improvements can be attributed to the user constraint introduced in Assumption 3.

**Lexical Chaining for Supervised Learning**

The methods used so far are mainly based on assumptions and rules. To better exploit the potential of lexical chaining, we can deploy supervised models and use different lexical similarity scores as features. It is also interesting to see whether these additional lexical similarity features can improve our existing systems described in Section 4.2. As a preliminary investigation, we add a lexical chaining based feature to the CRFSGD classifier used in Section 4.2 based on Assumption 3.
Table 4.16: Supervised linking classification by applying CRFSGD over features from Section 4.2 without (NoLC) and with (WithLC) features extracted from lexical chains, created by ChainerSV with different weighting schemes.

The feature value for each post is calculated using the following formula:

$$\text{feature} = \begin{cases} 
\frac{\text{sim}(\text{post3}, \text{post1})}{\text{sim}(\text{post2}, \text{post1})} & \text{if Post3} \\
0 & \text{otherwise}
\end{cases}$$

where sim is calculated using ChainerSV with different weighting methods.

The experimental results are shown in Table 4.16. From the results we can see that, by adding the additional feature extracted from lexical chains, the results improve slightly. The feature from the ChainerSV with LCBan weighting leads to the best $F_\mu$ of 0.897. These improvements are statistically insignificant, possibly because the information introduced by the lexical chaining feature is already captured by existing features. It is also possible that better feature representations are needed for the lexical chains (e.g. using raw sim values rather than ratios).

These results are preliminary but nonetheless suggest the potential of utilising lexical chaining in the domain of web user forums.
Experiments over All the Posts

To date, all experiments have been based on just the first three posts in a thread, where the majority of our threads contain more than just three posts. We carried out preliminary experiments over full thread data, by generalising Assumption 3 to Post $N$ for $N \geq 3$. However, no significant improvements were achieved over an informed baseline with our unsupervised approach. This is probably because the situation for later posts (after Post 3) is more complicated, as more linking options are possible. Relaxing the assumptions entirely also led to disappointing results. What appears to be needed is a more sophisticated set of constraints, to generalise the assumptions made for Post 3 to all the posts.

4.3.5 Summary of Linking Analysis by Lexical Chaining

In this research, we explored unsupervised approaches for thread linking structure recovery, by automatically analysing the lexical cohesion between posts. Lexical cohesion between posts is measured using lexical chaining, a technique to extract lists of related word tokens from a given discourse. Most lexical chaining algorithms use domain-independent thesauri and only consider nouns. In the domain of web user forums, where the texts of posts can be very short and contain various typos and special terms, these conventional lexical chaining algorithms often struggle to find proper lexical chains. To address this problem, we proposed the use of statistical associations between words, which are captured by the WORDSPACE model, to construct lexical chains. Our preliminary experiments indicated that the method is empirically superior to an informed baseline.
4.4 Summary

In this chapter, we mainly investigated thread discourse structure parsing, as a task of the joint classification of inter-post links (Link) and dialogue acts (DA). We proposed generalised models to deal with this parsing task, including a novel method which borrows techniques from dependency parsing. The presented methods achieve significantly better results when compared to a strong heuristic baseline. Additionally, we demonstrated that the proposed generalised models can effectively parse short and incomplete threads when later posts are not present, and therefore are robust over dynamically evolving threads in active forums. On top of exploring supervised thread discourse structure parsing, we also tentatively explored unsupervised approaches for parsing Links only, using lexical chaining. Our preliminary experiments show the potential of utilising the lexical chaining technique in forum thread structure analysis.

So far we have been focusing on exploring methods to accurately parse the thread discourse structure. As described in Chapter 1, our ultimate goal is to improve information access and support sharing in the domain of forums. In order to validate our hypothesis that thread discourse structure can help users better access information in forums, in the following two chapters, we investigate two forum related applications, namely forum thread solvedness identification in Chapter 5 and forum thread retrieval in Chapter 6.
Chapter 5

The Utility of Discourse Structure in Identifying Resolved Threads in Technical User Forums

5.1 Introduction

In the domain of troubleshooting-oriented forums, one potential way to enhance information access and support sharing is to automatically identify threads where the original information need has been resolved. By filtering out threads which do not contain a valid answer, we can focus the attention of users on threads which have a greater chance of containing the required solution. As described in Section 2.2.1, Baldwin et al. (2007) explore this task of Solvedness classification, and find that it is an extremely difficult problem. This is because the annotation was often based on expert knowledge of Linux, and a great deal of information not
explicitly mentioned in the thread. Figure 5.1 shows an example thread, made up of 5 posts from 3 distinct participants, from the ILIAD (Improved Linux Information Access by Data Mining) dataset of Baldwin et al. (2007), as described in Section 3.3. In this thread, Post1 and Post3 are both from the thread’s initiator UserA. Post1 asks a question, and Post3 asks for more information about an answer provided by UserB in Post2. In response to Post3, UserB adds more information to his/her original answer, and Post5 provides another independent answer. In threads like this, it is important to identify whether the problem is solved or not, and also where solution(s) are likely to be found. However, in this example, although two independent answers are provided in Post2 and Post5, it is almost impossible to identify whether there is a correct solution unless the whole thread is understood at a technical level.

This research proposes to use information derived from thread discourse structure to help predict Solvedness of threads, without validating the answers provided in the threads. In the annotated version of the example ILIAD thread, as is shown in Figure 5.1, UserA initiates the thread with a question (dialogue act = Question-question) in the first post, by asking a question. In response, UserB provides an answer (dialogue act = Answer-answer). Then, UserA confirms more details about the answer provided (dialogue act = Answer-confirmation). UserB responds to UserA to add more information about his/her previous answer (dialogue act = Answer-add). Finally, UserC proposes an independent answer again to the original question (dialogue act = Answer-answer).

Specifically, we explore features extracted from the thread discourse structure which can be used to help classify the Solvedness of threads. We experiment with
both gold-standard and automatically predicted discourse structure, and find that thread discourse structure (which in no way evaluates the correctness of each post) can, indeed, boost thread classification accuracy, achieving state-of-the-art results over the task. We also investigate the correlation between thread discourse structure prediction F-score and thread **Solvedness** classification accuracy, and demonstrate a positive correlation. Finally, we show that focusing on improving the F-score over certain dialogue acts is able to boost **Solvedness** classification.

Figure 5.1: A snippetsed LIAD thread with annotated discourse structure
5.2 Data Description

To explore the task of using discourse structure to predict the Solvedness of a thread, we annotated the ILIAD threads, as described in Section 3.3, for discourse structure, based on Kim et al. (2010b)’s dialogue act tagset detailed in Section 2.2.3.

The original dialogue act tagset was developed primarily over troubleshooting-oriented threads. However there are non-troubleshooting threads present in the ILIAD dataset (hence the Task Orientation thread classification task is addressed in Baldwin et al. (2007)). After manual analysis of the ILIAD data, we identified that the dialogue act tagset was largely transferable in its original state, but needed the addition of the information sub-class to the Question super-category (i.e. Question-information). Question-information is used to tag posts in threads which are not troubleshooting-oriented and only provide information (e.g. on developer mailing lists to report on a bug fix). We also relaxed the definition of Resolution slightly to accommodate non-troubleshooting threads. For example, in one thread, the initiator requests an update to a wiki page, and this update is confirmed later by a non-initiator. In this case, this non-initiator’s post is labelled as Resolution. In the original definition, Resolution can only be used on posts from the initiator of the thread.

The modified dialogue acts (DAs) used to annotate the ILIAD dataset for discourse structure are described in Table 5.1. The annotation was performed by two annotators. The main annotator annotated all 250 threads (containing 1158 posts), and the secondary annotator independently annotated 26 randomly-selected threads (containing 113 posts) for quality assurance purposes. During annotation, annotators
first annotate the Links between posts in a thread, and then identify the type of each link (DA). The $\kappa$ values for agreement between the two annotators are 0.64 for combined Link and DA tagging, 0.79 for just the Links and 0.68 for just the DAs. The agreement is slightly better than that of Kim et al. (2010b), where their inter-annotator agreement over the CNET data annotation achieves $\kappa$ values 0.78 for Links and 0.59 for DAs. In general, a $\kappa$ value between 0.41 and 0.60 is considered moderate, and a $\kappa$ value between 0.61 and 0.80 is considered substantial (Landis and Koch 1977).

While both the ILIAD and CNET datasets are mainly troubleshooting-oriented and technical, they come from different domains. Therefore, we expect the DA and Link distributions in them to be different. However, to our surprise, the distributions of both DAs and Links in the two datasets are remarkably similar, supporting the suggestion that the DA label set has cross-domain applicability.

Regarding the Solvedness label for ILIAD dataset, the original thread-level annotations were done by three annotators on a five-point scale, with 1 indicating high confidence in Solvedness for a given thread and 5 indicating low confidence (Baldwin et al. 2007). The $\kappa$ value for agreement among three annotators is 0.38, based on a three-point scale (positive, neuter and negative). These annotations were aggregated by taking the simple mean across the three annotators and discretising into binary classes, with 2.5 as the breakpoint. Out of the 250 threads in the ILIAD dataset, 28 threads had a score of 2.5 and were discarded in the original paper. In the interests of comparability with the original research, we experiment over this reduced dataset (denoted ILIAD$_{222}$), but question the theoretical soundness of
Chapter 5: The Utility of Discourse Structure in Identifying Resolved Threads

<table>
<thead>
<tr>
<th>Super-category</th>
<th>Sub-class</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Question</td>
<td>question</td>
<td>the post contains a new and independent question.</td>
</tr>
<tr>
<td></td>
<td>add</td>
<td>the post provides additional information or asks a follow-up question, regarding a previous question.</td>
</tr>
<tr>
<td></td>
<td>confirmation</td>
<td>the post confirms details or errors in a question.</td>
</tr>
<tr>
<td></td>
<td>correction</td>
<td>the post corrects errors in a question.</td>
</tr>
<tr>
<td></td>
<td>information*</td>
<td>the post is in a non-troubleshooting thread, and only provides information.</td>
</tr>
<tr>
<td>Answer</td>
<td>answer</td>
<td>the post proposes an answer to a question.</td>
</tr>
<tr>
<td></td>
<td>add</td>
<td>the post provides additional information to an answer.</td>
</tr>
<tr>
<td></td>
<td>confirmation</td>
<td>the post confirms details or errors in an answer.</td>
</tr>
<tr>
<td></td>
<td>correction</td>
<td>the post corrects errors in an answer.</td>
</tr>
<tr>
<td></td>
<td>objection</td>
<td>the post objects to an answer.</td>
</tr>
<tr>
<td>Resolution</td>
<td>—</td>
<td>a user confirms that an answer works.*</td>
</tr>
<tr>
<td>Reproduction</td>
<td>—</td>
<td>a non-initiator asks a similar question, or confirms that an answer should work.</td>
</tr>
<tr>
<td>Other</td>
<td>—</td>
<td>the post does not belong to any of the above classes.</td>
</tr>
</tbody>
</table>

Table 5.1: The Dialogue Act (DA) set used for annotating ILIAD dataset ("*" signifies a difference over the original method of Kim et al. (2010b), while the definitions of other labels are consistent with those used previously.)

removing these threads from the dataset, so additionally experiment with the full dataset (denoted ILIAD).
5.3 Discourse Parsing for Thread Solvedness Classification

Although predicting Solvedness is challenging, as shown by Baldwin et al. (2007), we believe that the use of thread discourse structure should assist in the task. As a first step, we need to do thread discourse parsing, which includes predicting both the linkings (Links) between posts and the type (DA) of each link. Thread discourse parsing, as discussed in Section 4.2.4, can be addressed in several ways. If a structured classification approach, such as Conditional Random Fields (CRFs: Lafferty et al. (2001)), is used, we can either classify the Link and DA separately and compose them afterwards (denoted as Composition), or we can classify the combined Link and DA (e.g. treat 0+Question-question as a single label) directly (denoted as Combined). Another approach is to treat discourse structure parsing as a dependency parsing problem.

For thread discourse parsing, we use methods explained in Chapter 4. Specifically, all experiments were carried out based on 10-fold cross-validation, stratifying at the thread level to ensure that all posts from a given thread occur in a single fold. The results are evaluated using post-level micro-averaged F-score ($\beta = 1$). All three discourse structure parsing methods introduced in Section 4.2.4 were tested in our experiments, by using CRFSGD and MaltParser. As for features, we experimented with all the features described in Section 4.2.3, including Initiator, Position, TitSim, PostSim, Punct and UserProf. We found that using CRFSGD with a simple Initiator (i.e. whether a post’s author is the initiator of the thread) feature and the Combined
Table 5.2: Thread discourse structure features used for Solvedness classification

<table>
<thead>
<tr>
<th>Feature Category</th>
<th>Feature Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DA-only</td>
<td>LastPostDA</td>
<td>The DA of the last post in the thread.</td>
</tr>
<tr>
<td></td>
<td>LastNonInitDA</td>
<td>The DA of the last post from a non-initiator in the thread.</td>
</tr>
<tr>
<td></td>
<td>HasResolution</td>
<td>Whether the thread contains a Resolution post.</td>
</tr>
<tr>
<td>LinkDA-based</td>
<td>LastPairDA</td>
<td>The DA pair for the deepest post pair in the thread tree.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>In the case of ties, the pair containing the latest post is</td>
</tr>
<tr>
<td></td>
<td></td>
<td>chosen.</td>
</tr>
<tr>
<td></td>
<td>LastSubthreadDA</td>
<td>The sequence of DAs in the longest subthread in the thread</td>
</tr>
<tr>
<td></td>
<td></td>
<td>tree. In case of ties, the sequence containing the latest</td>
</tr>
<tr>
<td></td>
<td></td>
<td>post is chosen.</td>
</tr>
</tbody>
</table>

When using the thread discourse structure (i.e. Link and DA) for Solvedness prediction, one natural question is “could we simply use Resolution to identify solved threads?” While Resolution is a clear identifier of solved threads with 100% precision, only 8% of the threads contain Resolution posts, and yet 80.4% of the threads are labelled as solved. Therefore, by only using Resolution, a classifier could not do better than a majority class baseline. Instead, we propose to use a combination of discourse structure features to address the Solvedness classification problem. Table 5.2 displays all the discourse structure features used in this research, which can be grouped into two categories: (1) those based on only the DAs (DA-only); and
(2) those based on Link and DA (LinkDA-based). When using the Link information, we rely on the notions of “pairing” and “subthreading”. A pair is defined to be the combination of a post with the parent post it links to (noting that a given post can participate as a child in multiple “pairs”, as it can link to multiple posts), and a subthread contains all posts in a given path from a leaf node to the root node following the link structure. Figure 5.2 shows an example of each, based on the sample thread from Figure 5.1. As the LastPostDA is Answer-answer from Post5 (the final post), the LastNonInitDA for the thread is Answer-answer, HasResolution is 0 (as there are no Resolution posts), LastPairDA is Answer-add/Answer-confirmation from the pairing of Post4 and Post3, and LastSubthreadDA is Answer-add/Answer-confirmation/Answer-answer/Question-question from the subthread Post4/Post3/Post2/Post1.
5.4 Solvedness Classification

Baldwin et al. (2007) experimented with various learners from three machine learning software packages, namely LIBSVM (Chang and Lin 2011), TiMBL (Daelemans et al. 2010) and Weka (Hall et al. 2009), and found that LIBSVM achieves the best performance on the Solvedness classification task. Therefore, LIBSVM is used for Solvedness classification in this research.

In our initial experiments, we experimented with different kernel functions for LIBSVM, including linear, polynomial, radial basis function (RBF) and sigmoid kernels, with various parameter settings, and found the linear kernel to outperform the other kernels. Therefore, LIBSVM with a linear kernel is used throughout our experiments. We approach the Solvedness classification task by firstly following the procedure of Baldwin et al. (2007), where ILIAD is used. Subsequently, we carry out experiments over the full 250-thread ILIAD dataset. In both cases, various combinations of the features introduced in Section 5.3 are used. To generate these features, both the gold-standard LinkDAs and the automatically predicted ones are used.

All our Solvedness classification experiments were carried out based on stratified 10-fold cross-validation. The results are evaluated using classification accuracy (ACC). As our baselines, we use a majority classifier (ZeroR), as well as the best Solvedness classifier provided by Baldwin et al. (2007) (ADCS). As mentioned earlier, randomised estimation (Yeh 2000) (at a significance level of $p < 0.05$) is used for statistical significance testing.
Table 5.3: Results over ILIAD\textsubscript{222}, using discourse structure features from the gold-standard and also the discourse parsing model (“*” signifies a significantly better result than both baselines; the best result in each column is indicated in \textbf{boldface}).

<table>
<thead>
<tr>
<th>Feature Category</th>
<th>System/feature(s)</th>
<th>$ACC_{\text{gold}}$</th>
<th>$ACC_{\text{auto}}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Baseline</td>
<td>ZeroR</td>
<td>.779</td>
<td></td>
</tr>
<tr>
<td></td>
<td>ADCS</td>
<td>.788</td>
<td></td>
</tr>
<tr>
<td>DA-only</td>
<td>LastPostDA</td>
<td>.833*</td>
<td>.775</td>
</tr>
<tr>
<td></td>
<td>LastNonInitDA</td>
<td>.766</td>
<td>.792</td>
</tr>
<tr>
<td></td>
<td>HasResolution</td>
<td>.779</td>
<td>.779</td>
</tr>
<tr>
<td></td>
<td>LastPostDA + LastNonInitDA</td>
<td>.834*</td>
<td>.779</td>
</tr>
<tr>
<td></td>
<td>LastPostDA + HasResolution</td>
<td>.883*</td>
<td>.775</td>
</tr>
<tr>
<td></td>
<td>LastNonInitDA + HasResolution</td>
<td>.874*</td>
<td>.792</td>
</tr>
<tr>
<td></td>
<td>AllDAFeat</td>
<td>.883*</td>
<td>.779</td>
</tr>
<tr>
<td>LinkDA-based</td>
<td>LastPairDA</td>
<td>.851*</td>
<td>.792</td>
</tr>
<tr>
<td></td>
<td>LastSubthreadDA</td>
<td>.833*</td>
<td>.779</td>
</tr>
<tr>
<td></td>
<td>AllLinkDAFeat</td>
<td>.833*</td>
<td>.792</td>
</tr>
<tr>
<td></td>
<td>AllDAFeat + AllLinkDAFeat</td>
<td>.865*</td>
<td>.792</td>
</tr>
</tbody>
</table>

5.4.1 Experiments over ILIAD\textsubscript{222}

Table 5.3 presents the results from experiments over ILIAD\textsubscript{222}, using the thread discourse structure features generated from both the gold-standard ($ACC_{\text{gold}}$) LinkDAs, and automatically predicted ones ($ACC_{\text{auto}}$). The automatically predicted discourse structure of the whole ILIAD\textsubscript{222} dataset is obtained by aggregating the discourse structure predictions from each fold of the 10-fold cross-validation experiments described in Section 5.3. The combination of all DA-only features (i.e. LastPostDA, LastNonInitDA and HasResolution) is denoted AllDAFeat, and the combination of all LinkDA-based features (i.e. LastPairDA and LastSubthreadDA) is denoted AllLinkDAFeat. Results which are significantly better than both baseline results are signified by “*”, and the best result(s) in each column are presented in \textbf{boldface}. 
Looking first at the $ACC_{\text{gold}}$ results in Table 5.3 we can see that, not surprisingly, HasResolution by itself does not have any effect on the prediction (see our comments in Section 5.3). Moreover, while LastPostDA leads to a significant improvement, LastNonInitDA does not have a significant effect. More interestingly, the combination of LastPostDA or LastNonInitDA with HasResolution leads to further improvements. This is because the classifiers trained on LastPostDA or LastNonInitDA are aggressive and misclassify many solved threads as unsolved, which HasResolution can correct.

The $ACC_{\text{gold}}$ column shows both the potential and shortcomings of LinkDA-based features — i.e. while both LastPairDA and LastSubthreadDA lead to significantly better results in isolation, combining them does not lead to further improvements. Moreover, combining all the features (i.e. AllDAFeat + AllLinkDAFeat) leads to a drop in results compared to just using AllDAFeat. We hypothesise that there are a number of reasons for this. Firstly, the LastPairDA, LastSubthreadDA and DA-based features have dependencies between each other, in that they all draw on the same set of DAs. While they are closely related, the classifiers do not have any access into the internals of the features to leverage them, causing the learner to overfit the training data. Secondly, while LastPairDA and LastSubthreadDA lead to low results in isolation, this is almost certainly because of their sparse nature (LastPairDA and LastSubthreadDA have 72 and 135 distinct values, respectively), much moreso than the DA-based features. When combined with the other features, however, some of these features are found to have utility.

Looking next to the $ACC_{\text{auto}}$ results in Table 5.3, we can see that we surpass the
Chapter 5: The Utility of Discourse Structure in Identifying Resolved Threads

187
two baselines, delivering on the promise of discourse parsing aiding in Solvedness classification. The results drop appreciably relative to those achieved with the gold-standard labels, and in fact the improvements over the baselines aren’t statistically significant. This is perhaps not surprising, however, given than the F-score for discourse parsing was a modest 0.626, meaning that errors will propagate through to the thread-level classification.

While these results are certainly encouraging, and were worthwhile in terms of establishing the superiority of our method when discourse parsing features are used, we always had reservations about the ILIAD dataset, due to the most contentious instances having been removed from the dataset. In introducing these instances back into the dataset and labelling them as solved, the task becomes both more realistic and more challenging, including the ZeroR baseline rising up further. In the next section, we reapply our methods to the ILIAD dataset.

5.4.2 Experiments over ILIAD

We carry out the same experiments done in Section 5.4.1 over the whole ILIAD dataset, and present the results in Table 5.4. Again, the results which are significantly better than both baseline results are signified by “∗”, and the best result in each column is presented in boldface.

From Table 5.4 we can see a similar trend to that in Section 5.4.1, with our method improving over both baselines when we use either gold-standard or automatically-predicted features. However, there are some notable differences. Looking first at the ACC gold column, firstly, none of LastPostDA, LastNonInitDA and HasResolution
Table 5.4: Results over ILIAD, using discourse structure features from the gold-standard and also the discourse parsing model (“*” signifies a significantly better result than both baselines; the best result in each column is indicated in boldface).

<table>
<thead>
<tr>
<th>Feature Category</th>
<th>System/feature(s)</th>
<th>ACC\textsubscript{gold}</th>
<th>ACC\textsubscript{auto}</th>
</tr>
</thead>
<tbody>
<tr>
<td>Baseline</td>
<td>ZeroR</td>
<td>0.804</td>
<td></td>
</tr>
<tr>
<td></td>
<td>ADCS</td>
<td>0.804</td>
<td></td>
</tr>
<tr>
<td>DA-only</td>
<td>LastPostDA</td>
<td>0.784</td>
<td>0.780</td>
</tr>
<tr>
<td></td>
<td>LastNonInitDA</td>
<td>0.792</td>
<td>0.788</td>
</tr>
<tr>
<td></td>
<td>HasResolution</td>
<td>0.804</td>
<td>0.804</td>
</tr>
<tr>
<td></td>
<td>LastPostDA + LastNonInitDA</td>
<td>0.848*</td>
<td>0.776</td>
</tr>
<tr>
<td></td>
<td>LastPostDA + HasResolution</td>
<td>0.864*</td>
<td>0.780</td>
</tr>
<tr>
<td></td>
<td>LastNonInitDA + HasResolution</td>
<td>0.872*</td>
<td>0.788</td>
</tr>
<tr>
<td></td>
<td>AllDAFeat</td>
<td>0.884*</td>
<td>0.776</td>
</tr>
<tr>
<td>LinkDA-based</td>
<td>LastPairDA</td>
<td>0.832</td>
<td>0.816</td>
</tr>
<tr>
<td></td>
<td>LastSubthreadDA</td>
<td>0.832</td>
<td>0.792</td>
</tr>
<tr>
<td></td>
<td>AllLinkDAFeat</td>
<td>0.824</td>
<td>0.792</td>
</tr>
<tr>
<td></td>
<td>AllDAFeat + AllLinkDAFeat</td>
<td>0.852*</td>
<td>0.792</td>
</tr>
</tbody>
</table>

led to any improvement in isolation. However, the combination of these three features led to results that are significantly better than the baselines, with AllDAFeat achieving the best result of 0.884. Secondly, neither LastPairDA nor LastSubthreadDA has a significant impact on results, and their combination (i.e. AllLinkDAFeat) also does not outperform the baselines significantly. Looking next to ACC\textsubscript{auto}, we achieve the best results with LastPairDA once again, surpassing the baselines but not at a level of statistical significance. Overall, while it is clear that the Solvedness classification task becomes harder when we experiment with the full ILIAD dataset, we were able to reproduce the overall results from Section 5.4.1.
5.5 Results Analysis and Simulation

Examining the differences between the results for $ACC_{gold}$ and $ACC_{auto}$ in Section 5.4.1 and Section 5.4.2 leads us to suspect that if the F-score of the thread discourse parsing could be boosted, we would be able to achieve better Solvedness classification accuracy. Furthermore, because the most effective discourse structure features, i.e. LastPostDA, LastNonInitDA and HasResolution, only make use of a subset of the DAs, we anticipate that if we can improve the F-score over certain DAs, we will be able to significantly boost our Solvedness classification accuracy.

To test these hypotheses, firstly, we examine the entropy (presented in Table 5.5) of every DA against the Solvedness class distribution for each DA-only feature (i.e. LastPostDA, LastNonInitDA and HasResolution) and the combination of all DA-only features (i.e. AllDAFeat). From Table 5.5, we can see that Answer-answer, Answer-add and Resolution have relatively low entropy compared to the rest of the DAs. Therefore, it seems that these three DAs can contribute more in Solvedness classification.\footnote{Note that this entropy analysis can only capture the association between a single DA and the Solvedness class, and we are not able to capture more subtle feature interactions.}

Secondly, we conducted simulation experiments to examine the potential relation between DA classification and Solvedness classification. The simulation starts with a seed DA classification result (SeedResults), based on CRFSGD and the Initiator feature. This seed DA classification achieves a F-score of 0.651, significantly better than a strong heuristic baseline (i.e. 0.515) which classifies all first posts as Question-question and all subsequent posts as Answer-answer. Then, an arbitrary higher goal (e.g. 0.8) is set and an artificial classification result (ArtificialResults) is created by
Table 5.5: Entropy of each DA against the Solvedness class distribution for every DA-only feature and AllDAFeat features

<table>
<thead>
<tr>
<th>DA</th>
<th>LastPostDA</th>
<th>LastNonInitDA</th>
<th>HasResolution</th>
<th>AllDAFeat</th>
</tr>
</thead>
<tbody>
<tr>
<td>Question-add</td>
<td>0.999</td>
<td>0.811</td>
<td>—</td>
<td>0.999</td>
</tr>
<tr>
<td>Question-confirmation</td>
<td>0.881</td>
<td>0.991</td>
<td>—</td>
<td>0.961</td>
</tr>
<tr>
<td>Question-information</td>
<td>0.918</td>
<td>—</td>
<td>—</td>
<td>0.918</td>
</tr>
<tr>
<td>Answer-answer</td>
<td>0.500</td>
<td>0.513</td>
<td>—</td>
<td>0.498</td>
</tr>
<tr>
<td>Answer-add</td>
<td>0.461</td>
<td>0.550</td>
<td>—</td>
<td>0.489</td>
</tr>
<tr>
<td>Answer-confirmation</td>
<td>0.918</td>
<td>—</td>
<td>—</td>
<td>0.918</td>
</tr>
<tr>
<td>Answer-objection</td>
<td>0.918</td>
<td>1.000</td>
<td>—</td>
<td>0.954</td>
</tr>
<tr>
<td>Reproduction</td>
<td>1.000</td>
<td>1.000</td>
<td>—</td>
<td>1.000</td>
</tr>
<tr>
<td>Resolution</td>
<td>0.332</td>
<td>0.918</td>
<td>0.229</td>
<td>0.237</td>
</tr>
<tr>
<td>Other</td>
<td>0.971</td>
<td>0.934</td>
<td>—</td>
<td>0.952</td>
</tr>
</tbody>
</table>

randomly correcting errors in the output of the discourse parsing model. The corrections are made evenly across all DA labels, relative to the original error rates for each DA. Next, a simulator is used to predict the labels of each instance, by randomly selecting from the labels returned by SeedResults and ArtificialResults with equal chance. In order to generate enough simulated results, we pick 20 goal F-score figures between 0.651 and 1.0, and run the simulator 100 times for each of these figures. Finally, we use these 2000 simulated discourse structure predictions to classify Solvedness using AllDAFeat features, and plot each pair of discourse structure F-score and Solvedness accuracy in a scatter plot. We also try to fit a series of simple polynomial models of the form $y = ax^n + b \ (n \in \{1, 2, 3, 4, 5\})$\(^3\) to the plot. We find that the model for $y = ax^5 + b$ provides the best fit with the data, although the differences in the range $n \in \{2, 3, 4, 5\}$ are negligible. Figure 5.3 shows the graph, along with the curve of best fit for the function $y = ax^5 + b$.

\(^3\)Choosing $n > 5$ does not result in better fit with the data.
Figure 5.3: Simulation over all DAs (AllDA)

From Figure 5.3 we can see that there is a clear correlation between the F-score of DA classification and the accuracy of Solvedness classification, and that the impact of DA classification on Solvedness classification is, in fact, accentuated for higher F-scores. Theoretically, therefore, by improving the DA classification F-score, the Solvedness classification accuracy will increase accordingly.

The entropy analysis showed that not all DAs have the same utility for the task of Solvedness classification — i.e. some DAs are more important (lower entropy) than others. We select the three DAs (i.e. Answer-answer, Answer-add and Resolution) with lowest entropy values from Table 5.5, because these DAs seem to be
the most effective across the three feature types (i.e. LastPostDA, LastNonInitDA and HasResolution). Then, we carry out an analogous simulation over this set of automatically-selected DAs (PositiveDA\textsubscript{auto}). Additionally, we conducted a simulation over the 8 non-selected DAs (OtherDA\textsubscript{auto}). Once again, a line of best fit for $y = ax^5 + b$ is generated for the resulting simulations. The curves of best fit are shown in Figure 5.4, along with the original curve of best fit for all DAs (AllDA).

\footnote{Question-correction and Answer-correction are never used in annotating the discourse structure of ILIAD dataset.}
Figure 5.5: Simulation over manually-created DA groups (PositiveDA_{manual} and OtherDA_{manual})
Chapter 5: The Utility of Discourse Structure in Identifying Resolved Threads

Table 5.6: Micro-averaged DA classification F-scores per DA over ILIAD

<table>
<thead>
<tr>
<th>DA Group</th>
<th>DA</th>
<th>F-score</th>
</tr>
</thead>
<tbody>
<tr>
<td>PositiveDA_{auto}</td>
<td>Answer-answer</td>
<td>0.782</td>
</tr>
<tr>
<td></td>
<td>Answer-add</td>
<td>0.641</td>
</tr>
<tr>
<td></td>
<td>Resolution</td>
<td>0.514</td>
</tr>
<tr>
<td>OtherDA_{auto}</td>
<td>Question-question</td>
<td>0.992</td>
</tr>
<tr>
<td></td>
<td>Question-add</td>
<td>0.678</td>
</tr>
<tr>
<td></td>
<td>Question-confirmation</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>Question-information</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>Answer-confirmation</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>Answer-objection</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>Reproduction</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>Other</td>
<td>0</td>
</tr>
</tbody>
</table>

From Figure 5.3, we can see that, as suspected, the PositiveDA_{auto} group is much more important than the OtherDA_{auto} group for Solvedness classification. Therefore, to improve Solvedness classification, we should focus our attention on improving the DA classification F-score for DAs such as Answer-answer, Answer-add and Resolution. Table 5.6 shows the micro-averaged F-scores for DA classification, calculated per DA. When we do a breakdown of the results for the discourse parsing model, we can see that there is definitely room for improvement with Answer-answer, Answer-add and Resolution. Moreover, Answer-answer and Answer-add are the most-frequent and third most-frequent DAs in the ILIAD dataset, respectively, appearing 354 and 147 times. Therefore, there appears to be considerable scope for improvement.

While the identification of the more important DAs can be done automatically as shown above, we also attempted to select them in a more ad hoc way, based on our understanding and analysis of the data set. Intuitively, if a thread’s last post or
the last post from a non-initiator is Question-confirmation (i.e. confirm details or errors in a question), Question-information (i.e. only provide non-answer information), Answer-confirmation (i.e. confirm details or errors in an answer) or Answer-objection (i.e. object to an answer), this thread is more likely to be unresolved. At the same time, we can observe that the micro-averaged F-scores for all these DAs are 0: that is the model never predicts a post to be one of these DA types correctly. To explore the utility of these additional DAs, we conducted an additional simulation experiment including Question-confirmation, Question-information, Answer-confirmation, Answer-objection and Resolution in PositiveDA\textsubscript{manual}, and relegating the other 6 DAs to OtherDA\textsubscript{manual}. The results for these manually-created groupings are shown in Figure 5.5.

From Figure 5.5 we can see that the improvements in discourse parsing over the manually-chosen PositiveDA\textsubscript{manual} will lead to even greater improvements over Solvedness prediction than before, if only we can get the models to make predictions using them. Perhaps even more surprising is that our simulations predict that improvements over OtherDA\textsubscript{manual} stand to degrade Solvedness classification, for which we can only speculate the cause.

5.6 Discussion

In Section 5.4, we demonstrated that gold-standard discourse structure can improve Solvedness classification accuracy significantly. While automatically-predicted thread discourse structure can also attain improvements in Solvedness classification accuracy, the improvement is not at a level of statistical significance. However, in
Section 5.5, our simulation experimental results show that the Solvedness classification accuracy will increase disproportionately, if we can improve the F-score of thread discourse structure parsing. As a result, more research is needed to obtain better thread discourse structure parsers. Addressing this, in Section 7.2, we will discuss possible future work on improving thread discourse structure parsing.

5.7 Summary

In this chapter, we explored the task of Solvedness classification, that is the automatic prediction of whether the information need on the part of the initiator of a thread has been resolved or not, by parsing thread discourse structure in the form of a rooted directed acyclic graph over posts, with edges labelled with dialogue acts. While Solvedness classification has been shown to be very difficult in previous research (Baldwin et al. 2007), we achieved significantly better results using gold-standard discourse structure. We were also able to attain improvements in Solvedness classification accuracy using automatically-predicted thread discourse structure, although not at a level of statistical significance. However, simulations suggest that as we improve the F-score of thread discourse structure parsing, the Solvedness classification accuracy will increase disproportionately. Additionally, we showed that a particular subset of DAs is crucial to Solvedness classification accuracy, and that if we aim to improve the F-score of our discourse structure predictions over these DAs, we stand to make large gains in Solvedness classification accuracy.

Therefore, we can conclude that thread discourse structure can help thread Solvedness identification, thus help users better access information in forums. In the next
chapter, we will look into the application of forum thread retrieval, and investigate the utility of thread discourse structure in this context.
Chapter 6

Information Retrieval over Forums

6.1 Introduction

Another potential way to enhance information access and support sharing in forums is to improve information retrieval (IR) effectiveness over forum threads. To this end, Elsas (2011) amassed a forum dataset for forum thread retrieval and conducted initial experiments. In this chapter, we build on this earlier work, in exploring the hypothesis that incorporating thread discourse structure of Kim et al. (2010b), as described in Section 2.2.3, into the IR model can improve retrieval effectiveness.

Figure 6.1 shows an example threads, made up of 5 posts from 3 distinct participants, from Elsas’ Ancestry dataset (Elsas 2011), as described in Section 3.4. The example thread has its thread discourse structure annotated for the purpose of illustration. In this example, User A initiates the thread with a question (DA = Question-question) in the first post, seeking information about his/her great-grandfather. In
Figure 6.1: A snipped and annotated Ancestry thread

response, User B asks for more details about the question (DA = Question-confirmation). Then User A responds to User B to add extra information to his/her original question (DA = Question-add). Finally, User C proposes a solution to the original question (DA = Answer-answer), and User A confirms this answer to be a correct one (DA = Resolution).

Specifically in this research, we first parse thread discourse structure of the Ancestry dataset by using a parser that is trained over out-of-domain annotated data (i.e.
the CNET dataset used in Chapter 4 and the ILIAD dataset used in Chapter 5. Then we incorporate information derived from this thread discourse structure into an IR solution over forum threads, and find that thread discourse structure can, indeed, benefit thread retrieval. We also investigate the reason behind the improvements.

6.2 Thread Retrieval with Thread Contexts

Seo et al. (2009) propose a language modelling based technique for thread-level retrieval by considering a thread as a document (i.e. global representation) and incorporating local contexts (i.e. posts, pairs or dialogues). Figure 6.2 shows examples of different contexts in a thread structure. A thread's ranking score is then determined by its global representation ($GR$) and its local contexts. The ranking score for the $GR$ of a thread $T_i$ regarding query $Q$ is:

$$\Phi_{GR}(Q, T_i) = P(Q|T_i)$$

where $P(Q|T_i)$ is the query likelihood score of query $Q$ for thread $T_i$. 

**Figure 6.2: Different contexts in a thread structure**
To compute the ranking score for a thread’s local contexts (i.e. posts, threads or dialogues), pseudo-cluster selection (PCS) \cite{Seo2008} is used. The basic retrieval element for this method is the concatenation of posts from targeted local contexts of threads, in terms of a post, a pair of posts or a dialogue of posts. Given a query $Q$, first, the top $N$ targeted local contexts are retrieved. Then these $N$ local contexts are grouped into pseudo-clusters, where all the local contexts in the same pseudo-cluster belong to the same thread. That is, each pseudo-cluster represents a thread. Finally, a ranking of pseudo-clusters (i.e. threads) for the query $Q$ is calculated according to a geometric mean of scores of the top $k$ local contexts in each pseudo-cluster:

$$
\Phi_{PCS}(Q, T_i) = \left( \prod_{j=1}^{k} P(Q|L_{ij}) \right)^{1/k}
$$

where $P(Q|L_{ij})$ is the query likelihood score of the local context $L_{ij}$, which is the $jth$ local context in thread $T_i$.

If a pseudo-cluster of a thread contains less than $k$ local contexts, the following formula is used:

$$
L_{\text{min}} = \arg\min_{L_n \in L} P(Q|L_n)
$$

$$
\Phi_{PCS}(Q, T_i) = (P(Q|L_{\text{min}})^{k-m} \prod_{j=1}^{m} P(Q|L_{ij}))^{1/k}
$$

where $L$ represents all the targeted local contexts (i.e. top $k$ or $k - m$ contexts of each thread) retrieved for query $Q$, and $m$ is the number of local contexts in thread $T_i$.

After the scores for the global representation and local context are calculated, a
weighted-product of both can be derived:

$$
\Phi_{Product}(Q, T_i) = \Phi_{PCS}(Q, T_i)^{(1-\pi)} \cdot \Phi_{GR}(Q, T_i)^\pi
$$

where $\pi$ is a weighting parameter.

### 6.3 Baseline Systems and Initial Experiments

Elsas (2011) conducted a series of IR experiments over the Ancestry dataset, using Indri,$^1$ Terrier,$^2$ (Ounis et al. 2006), Zettair,$^3$ and the native Ancestry.com search facility with various configurations. The retrieval is done at the post-level, and 3 different aggregation methods are used to convert the post-level rankings to thread-level rankings. A summary of the retrieval systems with the configurations used, as well as the aggregation methods, is presented in Table 6.1. To illustrate query forms under different query formulations (i.e. BoW, DM and fielded queries) used in Indri, we use the example query “john stephen manley”, and show the corresponding query forms:

**BoW query formulation:**

`#combine(john stephen manley)`

where `#combine` computes the geometric average over the scores calculated for each component (word in this case) to form a single score.

---

1 Indri version 2.12 (Lemur version 4.12), [http://lemurproject.org/](http://lemurproject.org/)
4 Provided by Ancestry.com based on the simplified keyword queries, not the original structured queries.
DM query formulation:

\[
\text{#weight(}
\begin{align*}
0.8 \text{ #combine(john stephen manley)} \\
0.1 \text{ #combine(#1(john stephen) #1(john stephen manley) #1(stephen manley))} \\
0.1 \text{ #combine(#uw4(john stephen) #uw4(john manley) #uw4(stephen manley) #uw8(john stephen manley)))}
\end{align*}
\]

where \text{#weight} computes a weighted geometric average over the scores of each component, \text{#1(john stephen)} matches “john stephen” as an exact phrase, and \text{#uwN(john stephen)} denotes an unordered window of size \(N\), in which all terms (i.e. “john” and “stephen” in this case) must appear in any order.

Field query with linear combination:

\[
\text{#wsum(}
\begin{align*}
0.4 \text{ #weight(}
\begin{align*}
0.8 \text{ #combine(john.(post_title) stephen.(post_title) manley.(post_title))} \\
0.1 \text{ #combine(#1(john stephen).(post_title) #1(john stephen manley).(post_title) #1(stephen manley).(post_title))} \\
0.1 \text{ #combine(#uw4(john stephen).(post_title) #uw4(john manley).(post_title) #uw4(stephen manley).(post_title) #uw8(john stephen manley).(post_title)))}
\end{align*}
\end{align*}
\]
0.4 #weight(
  0.8 #combine(john.(text) stephen.(text) manley.(text))
  0.1 #combine(#1(john stephen).(text)
    #1(john stephen manley).(text)
    #1(stephen manley).(text))
  0.1 #combine(#uw4(john stephen).(text)
    #uw4(john manley).(text)
    #uw4(stephen manley).(text)
    #uw8(john stephen manley).(text)))

0.2 #weight(
  0.8 #combine(john.(subforum) stephen.(subforum) manley.(subforum))
  0.1 #combine(#1(john stephen).(subforum)
    #1(john stephen manley).(subforum)
    #1(stephen manley).(subforum))
  0.1 #combine(#uw4(john stephen).(subforum)
    #uw4(john manley).(subforum)
    #uw4(stephen manley).(subforum)
    #uw8(john stephen manley).(subforum))))

where \#wsum computes a weighted arithmetic mean over the scores of each component.

Field query with loglinear combination:
Similar to field query with linear combination as shown above, but the first line is
\#weight( rather than \#wsum, i.e. the aggregated score is calculated via a weighted
IR Systems and Configurations

<table>
<thead>
<tr>
<th>IR System</th>
<th>Configuration Used</th>
</tr>
</thead>
<tbody>
<tr>
<td>Indri</td>
<td>Bag-of-words (BoW) queries</td>
</tr>
<tr>
<td>Indri</td>
<td>Dependence Model (DM) queries (Metzler and Croft 2005), with suggested model weights</td>
</tr>
<tr>
<td>Indri</td>
<td>Fielded query with linear combination</td>
</tr>
<tr>
<td>Indri</td>
<td>Fielded query with loglinear combination</td>
</tr>
<tr>
<td>Terrier</td>
<td>(PL2) weighting model with default parameters</td>
</tr>
<tr>
<td>Terrier</td>
<td>(InL2) weighting model with default parameters</td>
</tr>
<tr>
<td>Zettair</td>
<td>Default Okapi BM25 ranking algorithm</td>
</tr>
<tr>
<td>Ancestry.com</td>
<td>The ranked boolean system used by Ancestry.com</td>
</tr>
</tbody>
</table>

Aggregation Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mean</td>
<td>Thread score is the mean of retrieved posts’ scores</td>
</tr>
<tr>
<td>Max</td>
<td>Thread score is the max score of the retrieved posts</td>
</tr>
<tr>
<td>Pseudo-Cluster Selection (PCS)</td>
<td>Thread score is the geometric mean of the top-(k) retrieved posts’ scores ((k = 5) is used)</td>
</tr>
</tbody>
</table>

Table 6.1: Summary of Elsas (2011)’s experimental setup

ageometric mean rather than a weighted arithmetic mean.

According to the experiments of Elsas (2011), Indri with bag-of-words (BoW) and dependence model (DM: Metzler and Croft (2005)) query formulation perform the best; our experiments support this conclusion. The DM used is a full dependency variant of a Markov Random Field, which assumes that all query terms are in some

---

5Poisson estimation for randomness, Laplace succession for first normalisation, and Normalisation 2 for term frequency normalisation

6Inverse document frequency model for randomness, Laplace succession for first normalisation, and Normalisation 2 for term frequency normalisation
way dependent on each other. It considers the BoW representation (with weight 0.8) of the whole query, as well as ordered representation (with weight 0.1) and unordered representation (with weight 0.1) of the subsets of the query.

We tried to reproduce the results presented in Elsas (2011) using Indri-BoW and Indri-DM for post-level retrieval with three different aggregation methods: Mean, Max and Pseudo-Cluster Selection (PCS). Our experimental results are displayed alongside the results reported in Elsas (2011) in Table 6.2. For comparability, the primary evaluation metrics used are $ppref@10$ and $mAPpref$, as described towards the end of Section 3.7.2, based on the evaluation script provided by Elsas (2011). Although there are slight differences between our results and Elsas (2011)'s results, the overall results are comparable. Because Indri-DM with PCS (Indri-DM-PCS) obtains the best results for both $mAPpref$ and $ppref@10$, it will be used as our baseline IR method.

One important parameter in Indri-DM-PCS is the $k$ for PCS, which is explained in detail in Section 6.2. $k = 5$ is used in both Elsas (2011)'s and our experiments. In order to make sure that this number is appropriate, we carry out experiments using the Indri-DM-PCS model with different $k$ values, and present the results in Table 6.3. From Table 6.3, we can see that $k = 5$ is one of the optimal values. Therefore $k = 5$ will be used throughout all the relevant experiments.

We also experimented with Terrier but could not reproduce the results reported in Elsas (2011), as shown in Table 6.4. This is mainly due to the lack of

---

7Indri version 5.3 is used in our experiments, which is a more recent version than the version 2.12 used by Elsas (2011).
8Available at https://github.com/jelsas/Pairwise-Preference-Evaluation
9Terrier version 3.5 is used in our experiments.
Chapter 6: Information Retrieval over Forums

Table 6.2: Elsas (2011)’s IR results (Original) and our reproduced results (Reproduced) over the Ancestry dataset using Indri. Retrieval is performed at the post-level, and evaluation is conducted at the thread-level. Three aggregation methods are used for each system to transform post-level scores to thread-level scores. The best results for each column are bold-faced.

<table>
<thead>
<tr>
<th>System</th>
<th>Aggregation Method</th>
<th>( mAP_{\text{pref}} ) Orginal</th>
<th>Reproduced</th>
<th>( ppref_{@10} ) Orginal</th>
<th>Reproduced</th>
</tr>
</thead>
<tbody>
<tr>
<td>Indri-BoW</td>
<td>Mean</td>
<td>.542</td>
<td>.533</td>
<td>.492</td>
<td>.501</td>
</tr>
<tr>
<td></td>
<td>Max</td>
<td>.599</td>
<td>.591</td>
<td>.561</td>
<td>.556</td>
</tr>
<tr>
<td></td>
<td>PCS</td>
<td>.656</td>
<td>.650</td>
<td>.640</td>
<td>.633</td>
</tr>
<tr>
<td>Indri-DM</td>
<td>Mean</td>
<td>.549</td>
<td>.536</td>
<td>.506</td>
<td>.510</td>
</tr>
<tr>
<td></td>
<td>Max</td>
<td>.608</td>
<td>.597</td>
<td>.571</td>
<td>.568</td>
</tr>
<tr>
<td></td>
<td>PCS</td>
<td>.661</td>
<td>.657</td>
<td>.646</td>
<td>.664</td>
</tr>
</tbody>
</table>

configuration details provided by Elsas (2011).

6.4 Discourse Parsing for Thread IR

In order to investigate different ways of using thread discourse structure in the context of thread information retrieval (IR), the discourse structure of all the threads in the Ancestry dataset is needed. However, it is not practical for us to manually annotate the discourse structure of the whole Ancestry dataset nor just the portion of the dataset retrieved by the different IR systems. Rather, we opt to use automatically-predicted discourse structure. To build a discourse parser for Ancestry threads, we randomly selected and annotated 50 threads from the whole dataset to use in parameter tuning.
<table>
<thead>
<tr>
<th>$k$</th>
<th>mAPref</th>
<th>ppref@10</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>.548</td>
<td>.516</td>
</tr>
<tr>
<td>2</td>
<td>.644</td>
<td>.628</td>
</tr>
<tr>
<td>3</td>
<td>.656</td>
<td>.655</td>
</tr>
<tr>
<td>4</td>
<td>.656</td>
<td>.657</td>
</tr>
<tr>
<td>5</td>
<td>.657</td>
<td>.664</td>
</tr>
<tr>
<td>6</td>
<td>.657</td>
<td>.664</td>
</tr>
<tr>
<td>7</td>
<td>.657</td>
<td>.664</td>
</tr>
<tr>
<td>8</td>
<td>.657</td>
<td>.664</td>
</tr>
</tbody>
</table>

Table 6.3: The effect of $k$ in Indri-DM-PCS

Discourse parsing, as discussed in Section 4.2, can be addressed in several ways. If a structured classification approach, such as a conditional random field (CRF), is used, we can either classify the links (Link) and dialogue act (DA) separately and compose them afterwards (denoted as Composition), or classify the combined Link and DA (e.g. treat $0+$Question-question as a single label) directly (denoted as Combined). Another approach is to treat discourse parsing as a dependency parsing problem (Kübler et al. 2009).

One additional step which is worth noting is that the parsers need the posts in a thread to be presented in chronological order, which is not recorded in the Ancestry dataset. While timestamps of posts are available in the data, the precision of the timestamps is at the day level. In order to recover the chronological orders of Ancestry posts in threads, we try to scrape precise timestamps of the posts from the Ancestry forums and use the precise timestamps to order the posts chronologically in a thread. If one or more posts’ precise timestamps in a thread are unrecoverable, the posts in this thread are presented in preorder of the thread tree.

For discourse parsing, we follow our work described in Section 4.2. All ex-
Table 6.4: Elsas (2011)’s IR results (Original) and our reproduced results (Reproduced) over the Ancestry dataset using Terrier. Retrieval is performed at the post-level, and evaluation is conducted at the thread-level. Three aggregation methods are used for each system to transform post-level scores to thread-level scores. The best results for each column are bold-faced.

<table>
<thead>
<tr>
<th>System</th>
<th>Aggregation Method</th>
<th>( mAP_{pref} )</th>
<th>( ppref@10 )</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Original</td>
<td>Reproduced</td>
</tr>
<tr>
<td>Terrier-(PL2)</td>
<td>Mean</td>
<td>.529</td>
<td>.441</td>
</tr>
<tr>
<td></td>
<td>Max</td>
<td>.582</td>
<td>.513</td>
</tr>
<tr>
<td></td>
<td>PCS</td>
<td>.648</td>
<td>.560</td>
</tr>
<tr>
<td>Terrier-(InL2)</td>
<td>Mean</td>
<td>.541</td>
<td>.470</td>
</tr>
<tr>
<td></td>
<td>Max</td>
<td>.583</td>
<td>.535</td>
</tr>
<tr>
<td></td>
<td>PCS</td>
<td>.655</td>
<td>.585</td>
</tr>
</tbody>
</table>

Experiments were carried out based on stratified 10-fold cross-validation, stratifying at the thread level to ensure that all posts from a given thread occur in a single fold. Additionally, we augment the training data in each of the 10-fold cross-validation experiments with the CNET and ILIAD datasets. The results are evaluated using post-level micro-averaged F-score (\( \beta = 1 \)). All three discourse parsing methods described above were tested in our experiments, using CRFSGD (Bottou 2011) and MaltParser (Nivre et al. 2007). For features, we experimented with all the features presented in Section 4.2, as well as many new features. We found that using CRFSGD with a simple feature indicating whether a post’s author is the initiator of the thread and the Combined approach achieves the highest Link and DA joint (LinkDA) F-scores, as shown in Table 6.5. Because the availability of annotated discourse structure data cannot always be assumed, we decided to use only out-of-
Train dataset setup | LinkDA | Link | DA  
--- | --- | --- | --- 
CNET | .359/.388 | .681/.717 | .435/.460 
ILIAD | .529/.524 | .801/.852 | .569/.524 
Ancestry+CNET | .427/.441 | .711/.724 | .501/.514 
Ancestry+ILIAD | .539/.495 | .827/.852 | .569/.524 
CNET+ILIAD | .406/.442 | .688/.717 | .478/.514 
Ancestry+CNET+ILIAD | .488/.452 | .730/.728 | .563/.523 

Table 6.5: Discourse structure parsing F-scores by applying CRFSGD/MaltParser with Initiator feature using the Combined approach over different training dataset setups (The best result for each column is **bold-faced**)

domain data to train the discourse parsers. Therefore, only the configurations of CNET, ILIAD and CNET+ILIAD are used in later experiments.

### 6.5 Augment Thread IR with Discourse Structure

One idea for using the discourse structure to enhance existing IR systems is to use either links (Links) or dialogue acts (DAs) to modify the document ranking. For example, in the framework of pseudo-cluster selection (PCS), one could imagine that a retrieved Answer-answer (i.e. an independent answer to a question) post should be weighted higher than Other posts (including irrelevant posts), and thus contribute more to the thread ranking score. Under this assumption, we examined all the correctly predicted instances from the parsers described in Section 6.4 over our Ancestry development set, and found that the correctly predicted set only contains 5 dialogue acts, namely: Question-question (Qq), Question-add (Qadd), Answer-answer (Aa), Answer-add (Aadd), and Resolution (Res). Therefore, only predictions for
these 5 dialogue acts are considered. Building on the Indri-DM-PCS system, our system (Indri-DM-LD) modifies the post-level rankings based on the predicted DA types of the posts. If a post’s predicted DA type belongs to the selected DA subset (DASubset), it is considered to be more important than other posts, and its score is increased/promoted by a certain factor. In addition to the 5 dialogue acts (DAs+ALL), we experiment with omitting one DA at a time (e.g. DAs–Qq = the five DAs minus Question-question predictions), to gauge the impact of each DA on the overall results.

Table 6.6 presents the mAPpref/npref@10 results for our Indri-DM-LD system with all DAs+ALL DASubset ablations and promotion factors (i.e. 30–70%). We test for statistical significance over the Indri-DM-PCS baseline with the two-tailed $t$-test ($p < 0.05$).\footnote{The two-tailed $t$-test was used by Elsas (2011), and is used instead of randomised estimation here for reasons of comparability.}
Table 6.6: The \( mAP_{pref} \) / \( ppref@10 \) scores from Indri-DM-LD when training the discourse parser over different training data sets (CNET, ILIAD or CNET+ILIAD), and with different promotion factors for the selected DAs; boldface signifies a better result than the Indri-DM-PCS baseline at a level of statistical significance (\( p < 0.05 \)). The baseline results are \( mAP_{pref} = .657 \) and \( ppref@10 = .664 \).

From Table 6.6, we can see that our system outperforms the Indri-DM-PCS baseline system (\( mAP_{pref} = .657 \) and \( ppref@10 = .664 \)) in most cases, demonstrating the superiority of our method. Our best results (\( mAP_{pref} = .674 \) and \( ppref@10 = .678 \)), which are better than the baseline at a level of statistical significance (\( p < 0.05 \)), are achieved using the combined CNET and ILIAD datasets for discourse parser training,
the DASubset of DAs–Qq, and a DA promotion factor of 50%. The intuition behind Question-question posts not warranting promotion is that they contain question and not answer data, and are less likely to contain information relevant to the resolution of a query. It is important to note that the discourse structure information used in these experiments was derived automatically based on out-of-domain data.

Another way to run the above experiments is using cross-validation to learn the DA promotion factor automatically. To do so, we run 10-fold cross-validation, and evaluate over the combined results from all test folds. We use the same setup as displayed in Table 6.6 and present the 10-fold cross-validation results in Table 6.7, along with all the learned DA promotion factors and their corresponding standard deviations ($\delta$).
Chapter 6: Information Retrieval over Forums

Table 6.7: The mAPpref/pref@10 scores from Indri-DM-LD when training the discourse parser over different training data sets (CNET, ILIAD or CNET+ILIAD), based on 10-fold cross-validation experiments. The table also shows the promotion factors learned and their corresponding standard deviations (δ). **Boldface** signifies a better result than the Indri-DM-PCS baseline at a level of statistical significance (p < 0.05). The baseline results are mAPpref = .657 and pref@10 = .664.

<table>
<thead>
<tr>
<th>DA training</th>
<th>DA Subset</th>
<th>mAPpref</th>
<th>pref@10</th>
<th>Promotion factors (δ)</th>
</tr>
</thead>
<tbody>
<tr>
<td>DAs + ALL</td>
<td></td>
<td>.668</td>
<td>.673</td>
<td>[.4 .4 .4 .4 .4 .4 .4 .4 .4 .4] (.00)</td>
</tr>
<tr>
<td></td>
<td>– Qq</td>
<td>.673</td>
<td>.678</td>
<td>[.5 .5 .5 .5 .5 .5 .5 .5 .5 .5] (.00)</td>
</tr>
<tr>
<td></td>
<td>– Qadd</td>
<td>.669</td>
<td>.672</td>
<td>[.4 .4 .4 .4 .4 .4 .5 .4 .4 .4] (.03)</td>
</tr>
<tr>
<td></td>
<td>– Aa</td>
<td>.655</td>
<td>.655</td>
<td>[.6 .6 .3 .6 .3 .3 .3 .3 .3 .7] (.16)</td>
</tr>
<tr>
<td></td>
<td>– Aadd</td>
<td>.668</td>
<td>.673</td>
<td>[.4 .4 .4 .4 .4 .4 .4 .4 .4 .4] (.00)</td>
</tr>
<tr>
<td></td>
<td>– Res</td>
<td>.667</td>
<td>.668</td>
<td>[.5 .4 .5 .4 .5 .4 .4 .5 .4 .5] (.05)</td>
</tr>
</tbody>
</table>

| ILIAD       |           | .668    | .673    | [.4 .4 .4 .4 .4 .4 .4 .4 .4 .4] (.00) |
|             | – Qq      | .673    | .670    | [.4 .5 .5 .5 .5 .5 .4 .5 .5 .5] (.04) |
|             | – Qadd    | .669    | .671    | [.5 .5 .5 .5 .5 .5 .5 .5 .5 .5] (.00) |
|             | – Aa      | .667    | .660    | [.3 .3 .7 .3 .3 .3 .7 .3 .3 .3] (.16) |
|             | – Aadd    | .660    | .656    | [.4 .4 .3 .4 .4 .4 .4 .4 .4 .5] (.04) |
|             | – Res     | .668    | .673    | [.4 .4 .4 .4 .4 .4 .4 .4 .4 .4] (.00) |

| CNET + ILIAD|           | .668    | .673    | [.4 .4 .4 .4 .4 .4 .4 .4 .4 .4] (.00) |
|             | – Qq      | .674    | .678    | [.5 .5 .5 .5 .5 .5 .5 .5 .5 .5] (.00) |
|             | – Qadd    | .669    | .672    | [.5 .5 .5 .5 .5 .5 .5 .5 .5 .5] (.00) |
|             | – Aa      | .655    | .658    | [.6 .6 .3 .6 .6 .3 .3 .3 .3 .3] (.15) |
|             | – Aadd    | .668    | .673    | [.4 .4 .4 .4 .4 .4 .4 .4 .4 .4] (.00) |
|             | – Res     | .668    | .671    | [.4 .4 .4 .4 .4 .4 .4 .4 .4 .4] (.00) |

From Table 6.7, we can see that the standard deviations of the promotion factors for most systems are very small, especially for the top systems which often have the same promotion factors for each experiment in the 10-fold cross-validation. This indicates that the systems are stable and the promotion factor is not hard to tune.

To investigate the mechanics behind our system, we conducted error analysis...
over Indri-DM-PCS vs. Indri-DM-LD. In Figure 6.3, there are two threads, namely Thread1 and Thread2, which relate to Query 38 (*jacob lazarus; great synagogue, dukes place, london*). In the gold-standard annotation, Thread1 is preferred to Thread2. The posts retrieved by Indri-DM system are posts 3, 4 and 9 for Thread1 and posts 2, 7 and 12 for Thread2. Under the Indri-DM-PCS baseline system, Thread2 is ranked higher than Thread1. However, with Indri-DM-LD and DAs–Qq, the correct ordering of Thread1 and Thread2 is predicted, as the DA of post 12 in Thread2 is Question-question while the DA of all other posts is in DAs–Qq. As a consequence, the relative promotion of Thread1 is greater than Thread2, and the correct ranking is derived.

During our experiments, we demonstrated that making use of discourse structure of forum threads can boost retrieval effectiveness. As an alternative to full discourse parsing, we experimented with simply promoting all non-first posts (under the assumption that first posts are most likely to be Question-question posts). The best results achieved for this simple method are $mAP_{pref} = .667$ and $ppref@10 = .670$. Although the $mAP_{pref}$ score is significantly better than the baseline, the $ppref@10$ score is not (and both results are slightly below the best results achieved with discourse parsing, of $mAP_{pref} = .674$ and $ppref@10 = .678$). Nevertheless it shows the potential of using a lighter-weight version of discourse structure to improve IR effectiveness.
Figure 6.3: An example ranking comparison from the Indri-DM-PCS and Indri-DM-PCS systems

6.6 Summary

In this chapter, we explored the hypothesis that IR over forum threads can be improved by incorporating thread discourse structure. Specifically, we used simple information derived from thread discourse structure to augment IR systems. When compared to previous research conducted over the Ancestry dataset, we achieved significantly better results using automatically-predicted thread discourse structure. This demonstrates the utility of thread discourse structure in forum retrieval and
again shows that thread discourse structure can help improve information access over forums.

This chapter concludes the main content of the thesis. In Chapter 4, we proposed generalised and robust methods to effectively parse thread discourse structure. Then in Chapter 5 and this chapter, we demonstrated that the thread discourse structure can improve thread Solvedness prediction and help forum thread retrieval, and as such, improve solution sharing and information access over forum data. In the following final chapter, we will present a brief summary of this thesis and our contribution, as well as possible extensions of this research, in terms of future work.
Chapter 7

Conclusion

The overarching theme of this thesis is to tackle the challenge of improving information access and solution sharing over forum data. To this end, we first conducted an extensive review over relevant published literature in Chapter 2. Then, based on the belief that thread discourse structure can help users better access information in forums, in Chapter 4, we proposed generalised methods to parse the thread discourse structure, and demonstrated the robustness of the methods over dynamically evolving threading. Furthermore, we demonstrated the utility of thread discourse structure with respect to improved information access in two contexts, namely thread Solvedness classification in Chapter 5 and thread IR in Chapter 6. In the following two subsections, we first summarise the content and findings of each chapter, and then propose future work.
7.1 Chapter Summary

Chapter 2: In this chapter, we reviewed two areas of forum-related research. The first one targets the recovery of metadata from forum data, such as reply-to links between posts, post-level dialogue acts, post quality, and thread-level characteristics. The review showed that much of the previous research has focused on feature engineering, and that forum-specific features are often the most effective ones. The second area of forum-related research is on forum-related tasks, including forum IR, thread summarisation and knowledge base construction. Research in this area demonstrates the fact that forum data is distinct from traditional text documents, and forum-specific features such as structural features can help forum-related tasks significantly.

Chapter 3: Chapter 3 detailed the resources used in this thesis. We used three datasets, namely CNET, ILIAD and Ancestry datasets. The CNET dataset contains discussions over technical topics that are related to operating systems, software, hardware and web development. The ILIAD dataset is made up of both general Linux-related troubleshooting and specific technical discussions over certain Linux distributions. The Ancestry dataset consists of threads on genealogy-related discussions. For experiments, we used MaltParser (for dependency parsing) and CRFSGD (for conditional random fields) intensively. Many of our experiments were managed in a declarative classification framework named Hydrat. In addition, we also reviewed the core empirical methodologies that underpin MaltParser, CRFSGD and Hydrat.
Chapter 4: In this chapter, we described our generalised approaches for parsing thread discourse structure, which involves the joint classification of both the inter-post links (Links) and the dialogue acts (DAs). We used two sequence learners, namely CRFs and dependency parsing, for this task. Dependency parsing natively handles the combination of Link and DA, and its usage for thread discourse structure parsing is novel to this thesis. For CRFs, we tested two basic approaches to joint classification: (1) classifying the Link and DA separately, and composing the predictions to form the joint classification; and (2) combining the Link and DA labels into a single class, and applying CRFs over the posts with the combined class. The proposed approaches achieve significantly better results than a strong heuristic baseline, with dependency parsing achieving the best results. This is partially because it is hard for CRFs to handle long distance links well due to data sparsity, while dependency parsing is better equipped to handle long distance links. Additionally, the dependency parsing package we used, namely MaltParser, is highly configurable, and was tuned extensively in our experiments. We also carried out in-situ classification, which establishes a way to analyse parsing performance over dynamically evolving threads. We found that our parsing methods can robustly handle growing threads, and achieve similar results over partial threads compared to complete threads.

Chapter 5: Next, we explored the utility of thread discourse structure for thread Solvedness prediction. We used features from both gold standard discourse structure and automatically parsed discourse structure. We found that simple
features derived from the gold standard discourse structure can greatly boost the accuracy of Solvedness classification, while the automatically parsed discourse structure can improve Solvedness classification, but not at a level of statistical significance. However, through simulation experiments, we demonstrated that by improving the accuracy of thread discourse structure parsing, thread Solvedness classification accuracy increases accordingly. Additionally, we discovered that, to help Solvedness classification, it is sufficient to focus on improving the parsing accuracy of only a subset of the dialogue acts, which can be identified automatically.

Chapter 6: This chapter investigated ways to use thread discourse structure in thread IR. The basic approach is to first conduct post retrieval, then weight up post-level rankings based on their predicted dialogue act labels, and finally aggregate post rankings to thread rankings. The intuition behind this approach is that certain types of posts (e.g. Answer-answer) are more important than others (e.g. Other). The experimental results support this hypothesis — i.e. even with predicted thread discourse structure, the proposed approach can improve thread retrieval significantly, when compared to previously published results.

7.2 Future Work

As shown in Chapters 5 and 6, in order to use thread discourse structure to help users better access information, we need parsers which can parse the thread discourse structure more accurately, and also be able to adapt to new domains. In
this section, we will briefly discuss possible future work from these two aspects. In addition, we briefly discussed the possibility of exploiting non-traditional forum data to aid research over traditional forums.

**Improving Thread Discourse Structure Parsing**

In Section 4.2, we showed the effectiveness of using dependency parsing for thread discourse structure parsing. However, as described in Section 4.2.2, there are a few challenges when applying dependency parsing directly over forum data. The multi-headedness and disconnected sub-graph problems are the most difficult ones.

Regarding multi-headedness, one potential way to tackle this issue is to do sub-post linking and tagging (e.g. at the sentence or paragraph level), by trying to identify sections in a post which should be linked to different preceding posts, with separate DA labels; or trying to identify sections in a post which are linked to the same preceding posts, but with different DA labels. A bottom-up approach can be used for this sub-post linking and tagging, where we can first identify which preceding post each sentence is linked to and the DA label of that link, based on the assumption that each sentence can only be linked back to one post with one DA label. Then we can merge adjacent sentences, which are linked to the same preceding post and with the same DA label, into sections. Sub-post thread discourse parsing can not only ameliorate the multi-headedness problem, but is also an interesting future research direction in itself. By distilling the most useful sections from thread discussions, it has the potential to further improve information access and
support sharing in forums. Sub-post linking and tagging introduce additional complexity. For example, linking may appear among sentences in one post, with the possibility of both forward and backward directions. Identifying the boundaries of sections in a post is also a separate research question in itself. However, the thread discourse structure approaches proposed in this thesis can be adopted for sub-post thread discourse structure with minimum modification. While the sub-post analysis approach is a practical way to tackle the multi-headedness problem, another more research-oriented approach is a top-down one, which explores dependency parsing algorithms which can handle multi-headedness (Henderson et al. 2008; Sagae and Tsujii 2008). We can first identify the possible preceding posts the current post is likely to be linked to and the corresponding DAs. Then we can try to identify sections which are responsible to each link, by using techniques from discourse disentanglement research, as described in Section 2.2.3.

The disconnected sub-graph problem is a relatively minor issue in our research to date (e.g. only 2% of the threads in CNET dataset contain disconnected sub-graphs). This is partially because CNET forums are relatively heavily moderated. Therefore, cases of thread-hijacking, spamming, and off-the-topic discussions are relatively rare. Another possible reason is that the community culture of CNET forums tends to generate shorter and more concise threads with mainly informative posts. However, in many other forums, such as Apple Discussions\(^1\) and World of Warcraft Forums\(^2\), long informative threads with many potential disconnected sub-graphs occur more often. This poses challenges to our current discourse parsing

\(^{1}\)https://discussions.apple.com
\(^{2}\)http://us.battle.net/wow/en/forum/
methodologies, which cannot handle disconnected sub-graphs properly, and suffers from error propagation in long threads. One potential way to tackle this challenge is to partition threads with disconnected sub-graphs into independent segments, and then conduct discourse parsing over each segment. There is already research that has explored discussion partition (Kim et al. 2005), as described in Section 2.2.3, which we can borrow ideas from.

Domain Adaptation

Domain adaptation is a more ambitious direction to take, and represents a significant departure from the main contributions of this thesis. In this research, we annotated three forum datasets with thread discourse structure. These datasets can be used as the basis for investigating domain adaptation. That is, given datasets from one or more domains, we wish to train parsers which can effectively parse datasets from a target domain, which may be a new domain or may be one of the training domains.

There are several possible directions to pursue regarding domain adaptation for thread discourse structure parsing. The first one is to build a parsing model for each training domain and combine them using a domain-aware linear regression model which weights each component model based on domain similarity. In tasks of sentence constituency parsing, this method has been shown to be effective over target domains which are in training domains or totally new (McClosky et al. 2010). A recent shared task (Petrov and McDonald 2012) on parsing web text from the Google Web Treebank, which consists of data from blogs, emails, reviews, forums,
and question answering sites, has also shown that domain-aware models achieve the best sentence constituency parsing and dependency parsing results.

Another direction to take is semi-supervised learning, such as self-training (also know as bootstrapping). In self-training, a parser trained over labelled data is first used to parse unlabelled data. Then, predictions with high confidence are treated as additional training data to improve the parser. Although previous research (Charniak 1997; Steedman et al. 2003; McClosky et al. 2006) has shown that this method alone usually only leads to relatively small improvements, it has been shown that combining semi-supervised methods with supervised models, e.g. using domain-aware models, can further improve sentence constituency parsing and dependency parsing (Petrov and McDonald 2012).

Additionally, we can take a feature augmentation approach (Daumé III 2007; Finkel and Manning 2009a) to tackle the domain adaptation problem. The basic idea is that different features should be weighted differently for datasets from different domains, i.e. each domain should have its own domain-specific parameter for each feature. For example, we may want to weight source domain specific features and target domain specific features differently, and weight domain independent features the same. This feature augmentation can be done as a preprocessing step over features before learning (Daumé III 2007), or via a Bayesian approach with domain-specific parameters tied to the prior (Finkel and Manning 2009a).
Exploiting Rich Metadata in Non-traditional Forums

One way to alleviate the problem of the lack of annotated data for forum research is to exploit the existing rich metadata, which may appear in non-traditional forums. Stack Overflow\(^3\) and Stack Exchange\(^4\) are such examples. While they are officially categorised as question and answer sites, the rich and well-maintained metadata on these sites can be valuable to research in traditional forums. For example, the comments linked to each answer in a thread can be considered similar to Answer-add (often from a non-initiator) or Answer-confirmation (often from the initiator) in our DA set, and can be extracted as additional training data. Moreover, the best answer in a thread can often be easily identified, based on the initiator’s acceptance and other reader ratings. Analysis of these best answers can potentially help identify Resolution posts in datasets like ILIAD, where a thread often contains many answers with little interactions. Also, duplicated questions are often manually identified and linked together by the moderators in Stack Overflow and Stack Exchanges sites. This data can be used to train models to identify duplicated threads in a forum or across forums. Additionally, questions in Stack Overflow and Stack Exchange cover a wide range of domains, and these questions are often manually categorised and tagged. Such data can be used to facilitate domain adaption experiments mentioned in the previous section.

\(^3\)\url{http://stackoverflow.com/} \(^4\)\url{http://stackexchange.com/}
Chapter 7: Conclusion

The Utility of Thread Linking Structure

In our research, the joint parsing of linking structure and dialogue acts in Chapter 4 shows that the thread linking structure can help identify the dialogue acts. Our experiments with MaltParser, which jointly parses the linking and dialogue act structure, lead to better dialogue act identification than benchmark methods from Kim et al. (2010b), where the dialogue acts are parsed independently. However, our experiments over Solvedness classification and thread information retrieval in Chapter 5 and Chapter 6 did not show the recovered thread linking structure to have empirical utility. It seems that only the dialogue acts are useful in these applications.

Nevertheless, as reviewed in Section 2.2.3, it has been shown that thread linking structure can derive features to help forum post-level retrieval (Xi et al. 2004), as well as help build better language models for forum post-level retrieval (Duan and Zhai 2011; Seo et al. 2009) and thread-level retrieval (Bhatia and Mitra 2010; Seo et al. 2009). As a future research direction, it would be interesting to investigate the utility of thread linking structure recovered by our methodologies, to see whether it can bring about empirical improvements in some of these applications. It would also be worth testing the utility of thread linking structure in other applications such as discussion summarisation (Wang and Rosé 2010), discussion visualisation for easier navigation, and automatic discussion monitoring for better forum administration.
Chapter 7: Conclusion

7.3 Summary

Web user forums (or simply “forums”) are a valuable means for users to resolve specific information needs, both interactively for the participants and statically for users who search and browse over historical thread data. However, the complex structure of forum threads can make it difficult for users to extract relevant information. Addressing this problem, we proposed to parse thread discourse structure of forum threads for the purpose of enhancing information access and solution sharing over web user forums.

The discourse structure of a forum thread is modelled as a rooted directed acyclic graph (DAG), and each post in the thread is represented as a node in this DAG. The reply-to relations between posts are then denoted as directed edges (LINKs) between nodes in the DAG, and the type of a reply-to relation is defined as a dialogue act (DA). To parse the discourse structure of threads, both LINKs and DAs need to be identified. The first method we proposed uses conditional random fields to either classify the LINK and DA separately and compose them afterwards, or classify the combined LINK and DA directly. Another technique we adopted is to treat this discourse structure parsing as a dependency parsing problem, because the joint classification of LINK and DA is a natural fit for dependency parsing. Our parsing methods not only perform significantly better than a strong heuristic baseline, but also can robustly handle growing threads, and achieve similar results over partial threads compared to complete threads. Additionally, we explored unsupervised approaches for LINK classification by using lexical chaining.

Then, we explored ways of using thread discourse structure information to im-
prove information access and solution sharing over web user forums. Specifically, we first demonstrated that the proposed discourse structure can help thread solvedness identification (i.e. automatically identify whether the question asked in a forum thread is resolved or not). The basic idea is using features derived from thread discourse structure to help solvedness classification. For example, the last reply-to LINK and its DA type can be indicative of whether the asked question is resolved or not. Experimental results showed that simple features derived from thread discourse structure can greatly boost the accuracy of solvedness classification, which has been shown to be very difficult in previous research.

We also investigated the utility of discourse structure in forum thread IR. The proposed method first parses the discourse structure of targeted threads, then uses information from the parsed discourse structure to augment existing IR systems. For instance, if a post is linked to a question post with a DA type of an answer, more weight should be given to this post during retrieval. Experimental results demonstrated that exploiting the characteristics of discourse structure of forum threads can benefit IR, when compared to previously-published state-of-the-art IR methods.

Finally, we briefly discussed possible future work in terms of improving thread discourse structure parsing and domain adaption. Regarding the former, we focused on the multi-headedness and disconnected sub-graph problems. With respect to domain adaption, we discussed several possible directions, including domain-aware linear regression, semi-supervised learning and feature augmentation.


Blum, Avrim, and Tom Mitchell. 1998. Combining labeled and unlabeled data


——, Amit Singh, Rashmi Gangadharaih, Dinesh Raghu, and Karthik


Cong, Gao, Long Wang, Chin-Yew Lin, Young-In Song, and Yueheng Sun.


Dahlmeier, Daniel, Hwee Tou Ng, and Tanja Schultz. 2009. Joint learning of preposition senses and semantic roles of prepositional phrases. In Proceed-


—–, —–, —–, and —–. 2006c. Learning to detect conversation focus of


Huang, Jizhou, Ming Zhou, and Dan Yang. 2007. Extracting chatbot knowledge from online discussion forums. In *Proceedings of the 20th International Joint Conference on Artificial Intelligence (IJCAI'07)*, 423–428, Hyderabad, India.


———. 2002. Optimizing search engines using clickthrough data. In *Proceedings of
the Eighth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD 2002), 133–142, Edmonton, Canada.


Kim, Jihie, Grace Chern, Donghui Feng, Erin Shaw, and Eduard Hovy. 2006. Mining and assessing discussions on the web through speech act analysis. In
Proceedings of the ISWC’06 Workshop on Web Content Mining with Human Language Technologies, Athens, USA.


of the 15th International Conference on Machine Learning (ICML’98), 296–304, Madison, USA.


Language Learning (EMNLP-CoNLL 2007), 122–131, Prague, Czech Republic.


In *Proceedings of the 8th International Workshop on Parsing Technologies (IWPT 03)*, 149–160, Nancy, France.


Pei, Jian, Jiawei Han, Behzad Mortazavi-Asl, and Helen Pinto. 2001. Pre-fixspan: Mining sequential patterns efficiently by prefix-projected pattern growth. In Proceedings of the 2001 International Conference on Data Engineering (ICDE’01), 215–224, Heidelberg, Germany.


Ranganath, Rajesh, Dan Jurafsky, and Dan McFarland. 2009. It’s not you, it’s me: detecting flirting and its misperception in speed-dates. In Proceedings of the


Steedman, Mark, Miles Osborne, Anoop Sarkar, Stephen Clark, Rebecca Hwa, Julia Hockenmaier, Paul Ruhlen, Steven Baker, and Jeremiah Crim.


— —, and — —. 2007. An introduction to conditional random fields for relational


Wang, Hongning, Chi Wang, ChengXiang Zhai, and Jiawei Han. 2011a. Learning online discussion structures by conditional random fields. In Proceedings of the 34th Annual International ACM SIGIR Conference (SIGIR 2011), 435–444, Beijing, China.


Yang, Jiang-Ming, Rui Cai, Yida Wang, Jun Zhu, Lei Zhang, and Wei-Ying Ma. 2009a. Incorporating site-level knowledge to extract structured data from


——, and ——. 2006. On the summarization of dynamically introduced informa-

Appendix A

Cohen’s Kappa Calculation

A.1 Standard Cohen’s Kappa

Cohen's Kappa (Cohen 1960) is conventionally used to calculate the agreement between two annotators:

\[ \kappa = \frac{P(a) - P(e)}{1 - P(e)} \]

where \( P(a) \) refers to the relative observed agreement between the annotators, and \( P(e) \) is the hypothetical probability of chance agreement between the annotators.

In order to illustrate the calculation of the \( \kappa \) value, a set of hypothetical annotation statistics is presented in Table A.1. In this hypothetical example, there are two annotators (‘S’ and ‘L’), three classes (‘A’, ‘B’, and ‘C’), and 5 threads. To calculate \( P(a) \) and \( P(e) \), a confusion matrix of the class set (i.e. [A, B, C] in this example) needs to be built, with the two axes representing the two annotators’ choices. The confusion matrix for this example is illustrated in Table A.2. As we can see, for each thread, according to the choices made by the two annotators, we increment
Table A.1: An hypothetical annotation example

<table>
<thead>
<tr>
<th>Annotator</th>
<th>Thread</th>
<th>Class label</th>
</tr>
</thead>
<tbody>
<tr>
<td>S</td>
<td>1</td>
<td>A</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>B</td>
</tr>
<tr>
<td></td>
<td>3</td>
<td>C</td>
</tr>
<tr>
<td></td>
<td>4</td>
<td>A</td>
</tr>
<tr>
<td></td>
<td>5</td>
<td>B</td>
</tr>
<tr>
<td>L</td>
<td>1</td>
<td>A</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>B</td>
</tr>
<tr>
<td></td>
<td>3</td>
<td>C</td>
</tr>
<tr>
<td></td>
<td>4</td>
<td>B</td>
</tr>
<tr>
<td></td>
<td>5</td>
<td>A</td>
</tr>
</tbody>
</table>

Table A.2: Confusion matrix for the hypothetical example

\[
\begin{array}{ccc}
\text{L} & \text{A} & \text{B} & \text{C} \\
\text{A} & 1 & 1 & 0 \\
\text{B} & 1 & 1 & 0 \\
\text{C} & 0 & 0 & 1 \\
\end{array}
\]

the relevant position in the confusion matrix by one. For example, for thread 4, annotator ‘S’ labeled it with class ‘A’ while annotator ‘L’ labeled it with class ‘B’. Therefore, one is added to position \((A, B)\) in the confusion matrix.

After the confusion matrix is created, \(P(a)\) and \(P(e)\) are calculated by:

\[
P(a) = \frac{\sum_{i} n_{i,i}}{N}, \quad P(e) = \frac{\sum_{i} (\sum_{j} n_{i,j} \times \sum_{j} n_{j,i})}{N^2}
\]

where \(n_{i,j}\) depicts the value at position \((i, j)\) in the confusion matrix, and \(N\) refers to the total value summed across the whole confusion matrix.
A.2 Improved Cohen’s Kappa for Multi-class Annotation

The standard formulation of Cohen’s Kappa cannot be used for multi-class annotation tasks. An extended method for calculating $\kappa$ was proposed by Wang (2009) to address this problem. In order to explain this scheme, a set of hypothetical post annotation statistics is generated and listed in Table A.3. In this hypothetical example, there are, once again, two annotators (‘S’ and ‘L’), three classes (‘A’, ‘B’, and ‘C’), and 3 posts.

As described in Section A.1, the most important step for computing the $\kappa$ value is calculating the cumulative counts for each position in the confusion matrix based on the two annotators’ choices. In the proposed scheme, the link information for each class label is taken into account. Firstly, for a post, the choices (i.e. classes) made by the two annotators are partitioned according to their link labels: classes

<table>
<thead>
<tr>
<th>Annotator</th>
<th>Post</th>
<th>Class label</th>
<th>Link label</th>
</tr>
</thead>
<tbody>
<tr>
<td>S</td>
<td>2</td>
<td>A</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td></td>
<td>B</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>3</td>
<td>B</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td></td>
<td>C</td>
<td>2</td>
</tr>
<tr>
<td></td>
<td>4</td>
<td>A</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td></td>
<td>B</td>
<td>3</td>
</tr>
<tr>
<td>L</td>
<td>2</td>
<td>A</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td></td>
<td>B</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>3</td>
<td>B</td>
<td>2</td>
</tr>
<tr>
<td></td>
<td>4</td>
<td>A</td>
<td>2</td>
</tr>
<tr>
<td></td>
<td></td>
<td>B</td>
<td>3</td>
</tr>
</tbody>
</table>

Table A.3: An hypothetical post annotation example
Figure A.1: Calculating the counts distribution for Post 2

Figure A.2: Calculating the counts distribution for Post 3

that have the same link will be put into one group. Then, within each group, the two annotator’s choices are combined only when they are the same, and the count is incremented by two. Otherwise, their choices are each combined with a hypothetical ‘NULL’ class and incremented by one. At last, these counts for the pairs are added into the their relevant positions in the confusion matrix. This approach implicitly extends the size of the confusion matrix. In the example shown in Table A.3, there are three possible classes (‘A’, ‘B’, and ‘C’) and three possible links (‘1’, ‘2’, and ‘3’). Hence, each axis of the confusion matrix has 10 dimensions (i.e. $3 \times 3 + 1$). The structure of the confusion matrix is shown in Table A.4. Examples for calculating the count distributions for Post 2 and Post 3 using this method are given in Figure A.1 and Figure A.2, respectively. The final confusion matrix is presented in Table A.4 (including counts from Post 2, Post 3, and Post 4).

After the confusion matrix is created, the $\kappa$ value can be calculated using the method described in Section A.1.
Appendix A: Cohen’s Kappa Calculation

<table>
<thead>
<tr>
<th></th>
<th>1A</th>
<th>2A</th>
<th>3A</th>
<th>1B</th>
<th>2B</th>
<th>3B</th>
<th>1C</th>
<th>2C</th>
<th>3C</th>
<th>NULL</th>
</tr>
</thead>
<tbody>
<tr>
<td>1A</td>
<td>2</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>2A</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>3A</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1B</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>2</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>S</td>
<td>2B</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>3B</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>2</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1C</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>2C</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>3C</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>NULL</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

Table A.4: Confusion matrix for the hypothetical post annotation example

This improved $\kappa$ value calculation scheme still has some problems. Take Post3 in Table A.3, for example. The two annotators have a certain amount of agreement because both of them annotated the post with class ‘B’. However, under our calculation presented in Figure A.2, no agreement counts are generated. Therefore, there is still room for improvement with this scheme.
Minerva Access is the Institutional Repository of The University of Melbourne

Author/s:
WANG, LI

Title:
Knowledge discovery and extraction of domain-specific web data

Date:
2014

Persistent Link:
http://hdl.handle.net/11343/45174

File Description:
Knowledge discovery and extraction of domain-specific web data