On the Use of Prior and External Knowledge in Neural Sequence Models

Cong Duy Vu Hoang

Melbourne School of Engineering
The University of Melbourne

A dissertation in total fulfilment of the requirements for the award of Doctor of Philosophy (Ph.D.)

School of Computing and Information Systems    July 2019
Abstract

Neural sequence models have recently achieved great success across various natural language processing tasks. In practice, neural sequence models require massive amount of annotated training data to reach their desirable performance; however, there will not always be available data across languages, domains or tasks at hand. Prior and external knowledge provides additional contextual information, potentially improving the modelling performance as well as compensating the lack of large training data, particular in low-resourced situations. In this thesis, we investigate the usefulness of utilising prior and external knowledge for improving neural sequence models. We propose the use of various kinds of prior and external knowledge and present different approaches for integrating them into both training and inference phases of neural sequence models. The followings are main contributions of this thesis which are summarised in two major parts:

We present the first part of this thesis which is on **Training and Modelling** for neural sequence models. In this part, we investigate different situations (particularly in low resource settings) in which prior and external knowledge, such as side information, linguistic factors, monolingual data, is shown to have great benefits for improving performance of neural sequence models. In addition, we introduce a new means for incorporating prior and external knowledge based on the moment matching framework. This framework serves its purpose for exploiting prior and external knowledge as global features of generated sequences in neural sequence models in order to improve the overall quality of the desired output sequence.

The second part is about **Decoding** of neural sequence models in which we propose a novel decoding framework with relaxed continuous optimisation in order to address one of the drawbacks of existing approximate decoding methods, namely the limited ability to incorporate global factors due to intractable search.

We hope that this PhD thesis, constituted by two above major parts, will shed light on the use of prior and external knowledge in neural sequence models, both in their training and decoding phases.
Declaration

This is to certify that

(i) this PhD thesis was written by myself and comprises my original work towards the PhD degree, except where specified in the Preface,

(ii) the work has not been submitted for any professional qualification,

(iii) due acknowledgment has been made in the text to all other material used, and

(iv) the thesis has less than 100,000 words in length, exclusive of tables, figures, maps, bibliographies and appendices.

Cong Duy Vu Hoang
July 2019
Preface

Large portions presented in this thesis have already been published or reported. This applies to:

- Parts of Chapter §3 were published in:


- Parts of Chapter §4 were published in:
  Cong Duy Vu Hoang, Trevor Cohn, Gholamreza Haffari (2016). Improving Neural Translation Models with Linguistic Factors. In Proceedings of the Australasian Language Technology Association Workshop (ALTA-16), pages 7–14, Caulfield, Australia. (Best Paper Award)

- Parts of Chapter §5 were published in:

- Parts of Chapter §6 were reported in:

- Parts of Chapter §7 were published in:


  Note that I am a primary author of the above publications. I would like to thank other co-authors, including my supervisors (Trevor Cohn and Gholamreza Haffari) and my collaborators (Philipp Koehn, Marc Dymetman, and Ioan Calapodescu) — who I have fortune to work with.

  In addition, I contributed substantially to the following work:

Acknowledgements

I would like to express my deepest gratitude and appreciation to my supervisors, Prof. Trevor Cohn and Prof. Gholamreza (Reza) Haffari (Monash University), for their tremendous guidance, support and patience over the period of my PhD study. Trevor and Reza were truly experts in the field, always gave me some inspiring ideas. Reza’s fighting spirit was amazing; and he never gave up until the last minute, i.e., of paper submission. This thesis work would not have been completed without them.

I would also like to thank the University of Melbourne for their financial assistance of Melbourne International Fee Remission Scholarship and Melbourne International Research Scholarship — which gave me a great opportunity to come to Australia for broadening my knowledge with a PhD study.

I would like to acknowledge the organizers at CMU, USA for hosting The 2017 Frederick Jelinek Memorial Summer Workshop (JSALT’17) — where I have a chance to deepen my research ideas with top researchers. Also, many thanks go to Dr. Marc Dymetman and Mr. Ioan Calapodescu, who gave me an opportunity to join their team during my 3-month internship at NAVER LABS Europe, France.

I would also like to mention my friends and colleagues around, including: Mr. Tuan Doan, Ms. Ha Luong, Dr. Dat Nguyen, Dr. Long Duong, my cousin Ms. Hanh Tran — who shared many moments of both work and life.

Lastly, I am extremely grateful to my parents and my wife in particular — who have always been truly supportive of my study. Without them, I could not go as far as now.

My PhD study has lasted for 3.5 years and this journey is the most incredible and challenging experience I have ever got. I am sure that earning a PhD degree is just the beginning; and from now on, hopefully I am able to contribute more to research community with my acquired knowledge.
I would like to dedicate this thesis to my loving parents, my beloved wife who have been supporting me thoroughly and unconditionally.
# Table of contents

List of figures xvi

List of tables xviii

## 1 Introduction 1

1.1 Motivation 1

1.1.1 Deep Learning for Natural Language Processing 2

1.1.2 Word Embeddings 3

1.1.3 Neural Sequence to Sequence Learning 4

1.2 Research Questions and Scope 4

1.2.1 Training and Modelling 5

1.2.2 Decoding 6

1.2.3 Thesis Structure and Contributions 6

## 2 Background 11

2.1 Neural Sequence Models 12

2.1.1 Sequence Modelling 12

2.1.2 Sequence Generation 19

2.2 Training Neural Sequence Models 33

2.2.1 The Training Objective Function 33

2.2.2 Training Problems with Vanishing/Exploding Gradients 36

2.2.3 Hands-on Techniques on Training Neural Sequence Models 40

2.2.4 Evaluation Measures 43

2.2.5 Overview of Recent Work 46

2.3 Decoding Methods for Neural Sequence Models 51

2.3.1 Random Sampling 51

2.3.2 Greedy Decoding 52

2.3.3 Beam Search 52

2.3.4 Reranking/Rescoring 54

2.3.5 Overview of Recent Work 55
Table of contents

2.4 The Importance of Prior and External Knowledge ............................. 58
  2.4.1 Prior and External Knowledge for NLP ................................. 58
  2.4.2 Prior and External Knowledge for Neural Sequence Models ............ 59
2.5 Summary .......................................................................................... 61

Part 1 Training and Modelling ................................................................. 63

3 Neural Sequence Models with Side Information ................................. 65
  3.1 Learning Neural Language Modelling with Side Information .......... 66
    3.1.1 Recurrent Neural Network Language Model Architecture .......... 67
    3.1.2 RNNLM with Side Information ............................................. 67
    3.1.3 Experiments ...................................................................... 70
    3.1.4 Results and Analysis ......................................................... 74
    3.1.5 Discussion ...................................................................... 76
  3.2 Learning Neural Machine Translation with Side Information .......... 77
    3.2.1 Motivation ...................................................................... 77
    3.2.2 Machine Translation Data with Side Information .................... 78
    3.2.3 NMT with Side Information ................................................. 80
    3.2.4 Experiments ...................................................................... 85
    3.2.5 Connection to Related Work .............................................. 90
    3.2.6 Discussion ...................................................................... 91
  3.3 Summary ....................................................................................... 91

4 The Use of Prior Linguistics Knowledge for Neural Machine Translation 93
  4.1 Motivation .................................................................................. 94
  4.2 On the Use of Linguistic Factors .................................................. 95
  4.3 Multi-Factor Encoder-Decoder ..................................................... 96
  4.4 Multi-Factor Attention Architectures .......................................... 98
  4.5 Experiments .............................................................................. 100
    4.5.1 Datasets ........................................................................... 100
    4.5.2 Set-up and Baselines ......................................................... 101
    4.5.3 Results and Analysis ......................................................... 102
  4.6 Towards Target-factored Neural Translation Model ...................... 105
    4.6.1 Formulation ..................................................................... 106
    4.6.2 Discussion ...................................................................... 108
  4.7 Summary ..................................................................................... 108
# Table of contents

5 The Use of Monolingual Data for Neural Machine Translation 109

5.1 Iterative Back Translation for Neural Machine Translation 110

5.1.1 Related Work 110

5.1.2 The Potential Weaknesses of Back-Translation 111

5.1.3 Impact of Back-Translation Quality 112

5.1.4 The Strategy of Iterative Back-Translation 115

5.1.5 Experiments 115

5.1.6 Impact of Data Selection Methods 121

5.1.7 Analysis 123

5.1.8 Discussion 124

5.2 Iterative Back-Translation for Unsupervised Domain Adaptation 125

5.2.1 Motivation 125

5.2.2 Proposed Approach 126

5.2.3 Experiments 126

5.2.4 Discussion 129

5.3 Summary 129

6 Moment Matching Training Framework for Neural Machine Translation 131

6.1 Introduction and Motivation 132

6.1.1 Why Moment Matching? 133

6.2 Proposed Model 134

6.2.1 Moment Matching for Sequence to Sequence Models 135

6.2.2 Formulation of the MM Objective Function 135

6.2.3 Derivation of the Moment Matching Gradient 136

6.2.4 MM training vs CE training vs RL training with Policy Gradient 138

6.2.5 Computing the Moment Matching Gradient 139

6.2.6 Proof of Unbiasedness 141

6.2.7 Training with the Moment Matching Technique 143

6.2.8 Connections to Previous Work 145

6.3 Experiments 146

6.3.1 Prior Features for NMT 146

6.3.2 Datasets and Baseline 147

6.3.3 Results and Analysis 148

6.4 Moment Matching for Unsupervised Domain Adaptation 148

6.5 Summary 150
## Table of contents

**Part 2  Decoding**  

7  A Novel Decoding Framework with Relaxed Continuous Optimisation  
   7.1 Introduction and Motivation  
   7.1.1 Related Work  
   7.2 Problem Formulation  
   7.2.1 Decoding as Continuous Optimisation  
   7.3 Algorithms for Relaxed Continuous Optimisation Framework  
      7.3.1 Exponentiated Gradient (EG)  
      7.3.2 Stochastic Gradient Descent (SGD)  
   7.4 Decoding in Extended NMT  
   7.5 Experiments  
      7.5.1 Datasets  
      7.5.2 Baselines and Setup  
      7.5.3 Evaluation Metrics  
   7.6 Results and Analysis  
      7.6.1 Initialisation and Step Size  
      7.6.2 Continuous vs Discrete Costs  
      7.6.3 EG vs SGD  
      7.6.4 EG vs Reranking  
      7.6.5 Computational Efficiency  
      7.6.6 Main Results  
   7.7 Towards Constrained Decoding for Neural Sequence Models  
      7.7.1 Motivating Examples on Syntactic Parsing  
      7.7.2 Formulation of Constrained Decoding  
      7.7.3 Optimisation For Constrained Decoding  
      7.7.4 Discussion  
   7.8 Summary  

8  Conclusions and Future Work  
   8.1 Conclusions  
   8.2 Future Work  
   8.3 Summary  

References  

Appendix A  Translation Examples
List of figures

1.1 A general view of the encoder-decoder learning framework. 5
1.2 The detailed sections and chapters of this thesis. Note that the colors are associated with different parts as shown in Figure 1.3. 7
1.3 Visualisation of main parts of this thesis. 8

2.1 Research evolution in neural sequence models with main focus on language modelling, neural machine translation. 14
2.2 An overall architecture of the neural probabilistic language model. 15
2.3 An overall architecture of the recurrent neural network language model, adopted from Mikolov (2010). 17
2.4 An “unfolded view” of the RNNLM architecture over time steps. 19
2.5 A general and detailed views of the encode-decoder model with an example translating from English to Vietnamese. 20
2.6 A detailed view of an encoder-decoder model with attention, translating a sentence from English to Vietnamese. 25
2.7 The Transformer architecture. 27
2.8 Stacked LSTMs (left) vs. DGLSTMs (right). 39

3.1 Integration methods for auxiliary information, e: a) as input to the RNN, or b) as part of the output softmax layer. 69
3.2 Example of a TED talk with side information (title, description, keywords, speaker info) in addition to the main transcripts, in the TED Talks dataset. 71
3.3 A snapshot of a plenary session with side information (e.g., topic headline), in the RIE dataset. 72
3.4 An example with side information (e.g., keywords) for MT with TED Talks dataset. 78
3.5 An example with side information (e.g., author’s gender highlighted in red) for MT with personalised Europarl dataset. 79
List of figures

3.6 An example with side information (e.g., IPC highlighted in red) for MT with PatTR dataset. ......................................................... 81
3.7 Chart with individual BLEU scores for each of categories in the TEDTalks dataset. ................................................................. 88
3.8 Chart with individual BLEU scores for each of categories in the coarse PatTR-2 dataset. .......................................................... 89
4.1 Examples of linguistic factor annotations in both English and German languages. For each language, an example (1st sentence) is provided along with its annotation including: lemmatisation (2nd), word clustering (3rd), POS (4th), linearised dependency parsing (5th and 6th). .............................. 95
4.2 Proposed attention architectures of integrating linguistic factors for the NMT framework. .......................................................... 97
4.3 Analysis based on the evaluation set tst2013 in en→de translation. .......... 104
4.4 Analysis based on the evaluation set tst2014 in en→de translation. .......... 105
5.1 Creating a synthetic parallel corpus (English-German) through back-translation. 111
5.2 Iterative Back-Translation: Taking the idea of back-translation one step further. After training a system with back-translated data, it is used to create a synthetic parallel corpus for the final system. This process can be repeated until convergence. .......................................................... 114
7.1 Analysis on effects of initialisation states (uniform vs. greedy vs. beam), step size annealing, momentum mechanism from BTEC zh→en translation. **EG-400**: EG algorithm with step size $\eta = 400$ (otherwise $\eta = 50$); **EG-MOM**: EG algorithm with momentum. ............................................ 166
7.2 Comparing discrete vs continuous costs from BTEC zh→en translation, using the EG algorithm with momentum, $\eta = 50$. Each point corresponds to a sentence. .......................................................... 167
7.3 Analysis on convergence and performance comparing SOFTMAX and EG algorithms from BTEC zh→en translation. Both algorithms use momentum and step size 50. .......................................................... 168
7.4 Translation examples generated by the models. *: reranking with bidirectional (+bidir.) and bilingual (+biling.) produced the same translation string. Note that red refers to incorrect phrases/words and blue refers to correct ones. . . 171
List of tables

2.1 Applications of neural sequence models across NLP tasks. Note that we only give one representative work for each application. .......................... 13

2.2 Commonly-used non-linear activation functions ................................. 18

2.3 Update equations for GD optimisation algorithms. Note that \( \gamma \) in Momentum is called as momentum term of which value is often set around 0.9; \( \epsilon \) in Adadelta and RMSprop is a smoothing term (e.g., 10\(^{-8}\)) to avoid division by zero; \( \beta_1 \) and \( \beta_2 \) values in Adam are often set to 0.9 and 0.999, respectively. 35

2.4 Gates in LSTM and GRU. ............................................................... 37

2.5 Computation of hidden state \( \mathbf{h}_t \) at a time step \( t \) in LSTM and GRU. .............................. 38

2.6 Formulations of batch and layer normalisation methods. Note that \( \mathcal{B} \) and \( \mathcal{H} \) are batch and hidden/feature dimensions, respectively. Their equations are conceptually similar, except for the indices. .......................... 43

3.1 Statistics of the training sets, showing in each cell the number of word tokens, types, documents (talks or plenaries), and sentences; \( \text{K:} \) in thousands; \( \text{M:} \) in millions; “\#tokens”: number of word tokens. Here, “\#types” refers to word frequency thresholded at 5 and 15 for TED Talks and RIE datasets, respectively; “\#types”: number of word types; “\#docs”: number of talks and plenary sessions in TED Talks and RIE datasets, respectively; “\#sents”: number of sentences. ................................................................. 70

3.2 Perplexity scores based on the English part of the TED talks dataset, evaluating different RNNLM architectures. bold: “Statistically significant better than the 5-gram LM baseline”; ♠: “best result in the dataset” .................. 73

3.3 Perplexity scores based on the English part of the TED talks dataset, evaluating different encoding methods. +t: with “keywords” side information; bold: ‘ better than the other encoding methods” ......................... 73
3.4 **Perplexity** scores based on the **English** part of the TED talks dataset. +k, +t, +d: with keywords, title, and description as auxiliary side information respectively. **bold**: “Statistically significant better than the DGLSTM baseline”; ♠: “best result in the dataset”. .......................... 75

3.5 **Perplexity** scores based on the **French** part of TED talks dataset in IWSLT14 MT. Note that +k means with keywords in **English**. **bold**: “Statistically significant better than the best baseline”; ♠: “best result in the dataset”. ............ 76

3.6 **Perplexity** scores based on the sampled RIE dataset. +h: topic headline. **bold**: “Statistically significant better than the best baseline”; ♠: “best result in the dataset”. .......................... 77

3.7 Side information statistics for the three datasets, showing the number of types of the side information label, and the set of tokens (display truncated for PatTR-1 (deep)). .......................................................... 84

3.8 Statistics of the training & evaluation sets from datasets including TED Talks, Personalised Europarl, and PatTR; showing in each cell the count for the source language (left) and target language (right); “#types” refers to subword-segmented vocabulary sizes; “n.a.” is not applicable, for development and test sets. Note that all the “#tokens” and “#types” are approximated. ............... 86

3.9 Evaluation results with BLEU scores of various incorporation variants against the baseline; **bold**: better than the baseline, †: statistically significantly better than the baseline. .......................................................... 87

4.1 Statistics of the training & evaluation sets from IWSLT’14,15 MT track (including en↔de) showing in each cell the count for the source language (left) and target language (right). “#types” refers to filtered vocabulary with word frequency cut-off 5. .......................................................... 100

4.2 **Perplexity** scores for attentional model variants evaluated on en↔de translations, and “#param” refers to no. of model parameters (in millions). **bold**: “statistically significantly better than vanilla attentional model”, ♠: best performance. .......................................................... 101

4.3 BLEU scores for attentional models with the effects of individual linguistic factors evaluated on en↔de translations for the tst2014 testset. .................. 101

4.4 BLEU scores for attentional model variants evaluated on en↔de translations. 102

4.5 BLEU scores for attentional model variants evaluated on en→sl translations. 103
5.1 Parallel and monolingual corpora used, including English(EN)-German(DE), English-French(FR) and English-Farsi(FA). Numbers denote the number of words, and $l_2$ is the second language in each pair. The EN-DE data is from WMT 2017 (parallel) and a subset of News 2016 for monolingual side.

5.2 Results for canonical RNN-based attentional NMT systems with WMT News English↔German Translation Task, reporting cased BLEU on newstest2017, evaluating the impact of the quality of the back-translation system on the final system. Note that the back-translation systems run in the opposite direction and are not comparable to the numbers in the same row.

5.3 Results for transformer NMT systems with WMT News English↔German Translation Task, reporting cased BLEU on newstest2017, evaluating the impact of the quality of the back-translation system on the final system. Note that the back-translation systems run in the opposite direction and are not comparable to the numbers in the same row.

5.4 Results of canonical attentional NMT systems on WMT News English↔German Translation Task, reporting cased BLEU on newstest2017, for comparing the quality of different back-translation systems with different final system architectures. *Note that the quality for the back-translation system (Back) is measured in the opposite language direction. We denoted $bt$ as back-translation, $ibt+n$ as iterative back-translation with $n$ iterations.

5.5 Results of transformer NMT systems on WMT News English↔German Translation Task, reporting cased BLEU on test sets from 2014–2017, for comparing the quality of different back-translation systems with different final system architectures. *Note that the best NMT w/ $ibt$ is the system reported in Table 5.4. We denoted $bt$ as back-translation, $ibt+n$ as iterative back-translation with $n$ iterations; and bold for best results.

5.6 Low resource setting: Impact of the quality of the back-translation systems on the benefit of the synthetic parallel for the final system in a low-resource setting. Note that, we reported the single NMT systems in all numbers. We denoted $bt$ as back-translation, $ibt+n$ as iterative back-translation with $n$ iterations.

5.7 Weighting amounts of real parallel data (1) with varying amounts of synthetic data (1-3). Larger amounts of synthetic data help.

5.8 Experimental results with data selection methods for back-translation, evaluating English–German with transformer NMT systems. All BLEU scores are with single systems.
5.9 Statistics of the training & evaluation sets from (Cettolo et al., 2012) used for our experiments of unsupervised domain adaptation. ........................................... 127

5.10 Evaluation results with BLEU scores for unsupervised domain adaptation setting on the TED Talks dataset. Note that, we reported the single transformer-based NMT systems in all numbers. We denoted \( bt \) as back-translation, \( ibt+n \) as iterative back-translation with \( n \) iterations; **bold**: statistically significantly better than the baselines (\( wmt-domain-only \) and \( iwslt-domain-only \)), †: best result on the dataset. .................................................. 128

6.1 Comparing different methods for training seq2seq models. Note that we denote CE for cross-entropy, RL for reinforcement learning, PG as policy gradient, and MM for moment matching. ........................................... 139

6.2 Evaluation scores for training moment matching with length ratio between source and target sequences; **bold**: statistically significantly better than the baselines, †: best performance on dataset. .................................................. 147

6.3 Evaluation scores for training moment matching with bilingual lexical dictionary; **bold**: statistically significantly better than the baselines, †: best performance on dataset. .................................................. 147

7.1 Statistics of the training and evaluation sets; token and types are presented for both source/target languages. .................................................. 164

7.2 The BLEU evaluation results with EG algorithm against 100-best reranking on WMT evaluation dataset. .................................................. 169

7.3 The BLEU evaluation results across evaluation datasets for EG algorithm variants against the baselines; **bold**: statistically significantly better than the best greedy or beam baseline, †: best performance on dataset. .................................................. 170

7.4 Illustrating examples of linearisation for constituency and transition-based dependency parsing. .................................................. 173

7.5 Evaluation results of our baseline seq2seq models on the dependency parsing task. We denote: Unlabeled attachment Score (UAS), Labeled Attachment Score (LAS), number of errors with less shifts (LS), number of errors with more shifts (MS); and root errors (RO). .................................................. 174
Chapter 1

Introduction

Contents

1.1 Motivation .............................................................. 1
1.1.1 Deep Learning for Natural Language Processing ............ 2
1.1.2 Word Embeddings .................................................. 3
1.1.3 Neural Sequence to Sequence Learning ......................... 4
1.2 Research Questions and Scope ..................................... 4
1.2.1 Training and Modelling .......................................... 5
1.2.2 Decoding ............................................................ 6
1.2.3 Thesis Structure and Contributions .............................. 6

1.1 Motivation

In recent years, deep learning has come to dominate machine learning (ML) and artificial intelligence (AI) research due to its impressive and consistent achievements across the tasks of various fields, including computer vision, speech recognition, and natural language processing. In principle, deep learning introduces a computing mechanism originally inspired by biological architectures of a human brain (van Gerven and Bohte, 2017). The computing mechanism in deep learning relates to the network comprising artificial neurons connected through weights by which the data flow is transmitted. The way artificial neurons work is inspired by neurons in the human brain. The concept of deep learning refers to the capabilities of computer systems that are able to learn high-level abstract representations of a data “modality” (Bengio et al., 2013; LeCun et al., 2015; Schmidhuber, 2015), which can be image, video, speech or text. Conceptually, deep learning methods employ different layers
of neural networks operating via connected neurons, used for learning distributed representations of a data modality. The terminology of “Distributed representations” means that several components of the model (e.g., neurons in a neural network) are used to describe a given feature (e.g., concept in a data modality). That is, in several neurons in a deep model fire for a given type of input (e.g., a horizontal line segment). In this way, the model is more robust in the presence of noise (e.g., the line is blurred, or slightly rotated off the horizontal), which may lead to some of the inputs not firing. This capability of learning distributed representations over complex data modality is a major breakthrough technique to make deep learning models proficient across various problems. Furthermore, deep learning offers many intriguing and carefully crafted architectures, such as: deep belief networks and deep neural networks (Bengio, 2009; Glorot et al., 2011; LeCun et al., 1998; Schmidhuber, 2015), convolutional and recurrent neural networks (Hochreiter and Schmidhuber, 1997; Krizhevsky et al., 2012; Sutskever, 2013), neural Turing machines (Graves et al., 2014), generative adversarial networks (Goodfellow et al., 2014), capsule networks (Sabour et al., 2017), and neural multi-task learning (Ruder, 2017), to name a few. Recent advances in deep learning have produced impressive performance across various ML and AI tasks and domains, including computer vision (Venugopalan et al., 2015; Xu et al., 2015), speech recognition (Chorowski et al., 2015; Graves, 2013), and natural language processing (NLP) (Bahdanau et al., 2015; Sutskever et al., 2014).

1.1.1 Deep Learning for Natural Language Processing

Natural language processing (NLP) is a crucial part of AI, and is one of the most important technologies in the digital age. NLP involves computational models for automated analysis and processing of human languages. Human language, whether in the spoken or written form, is highly ambiguous and requires fine-grained meaning analysis, at the level of words, phrases, sentences, paragraphs, and higher level components in a piece of text with discourse (Jurafsky and Martin, 2009).

Over several decades, core NLP techniques largely comprised of rule-based and statistical methods (Allen, 1995; Jurafsky and Martin, 2009). These efforts have laid the basis of the fundamental concepts in human language processing, and led to the creation of abundant annotated corpora as well as the development of many well-established machine learning methods in NLP. Due to complex and ambiguous nature of structures in human languages, most successful approaches and systems in traditional NLP often require human intervention, for example, building hand-crafted patterns and features for rule-based systems in limited domains; or developing robust engineered features for statistical machine learning methods.
Over the past few years, we have witnessed an explosion on the applications of deep learning methods with remarkable performance across a wide range of NLP tasks, such as: named entity recognition (Chiu and Nichols, 2016; Lample et al., 2016; Ma and Hovy, 2016), text classification (Joulin et al., 2017; Kim, 2014; Yang et al., 2016), language modelling (Bengio et al., 2003; Merity et al., 2018a,b; Mikolov et al., 2011; Sundermeyer et al., 2012; van Gerven and Bohte, 2017); machine translation (Bahdanau et al., 2015; Gehring et al., 2017c; Lipton, 2015; Sutskever et al., 2014; Vaswani et al., 2017), question answering (Kumar et al., 2016), summarisation (Cheng and Lapata, 2016; Chopra et al., 2016; Rush et al., 2015), reading comprehension (Hermann et al., 2015), and syntactic and semantic parsing (Chen and Manning, 2014; Gardner et al., 2018; Vinyals et al., 2015a). Deep learning has been transforming the field of NLP, offering superior characteristics compared to conventional rule-based and statistical approaches. Firstly, it enables feature learning based on multiple levels of distributed representations (Bengio et al., 2013; LeCun et al., 2015), which enables reasoning over semantic relationships between words and higher level construct in a high-dimensional vector space (Mikolov et al., 2013a,b; Rong, 2014). Secondly, the end-to-end training mechanism, usually employed for deep learning models (Goodfellow et al., 2016; Schmidhuber, 2015), allows richer parameterised models to be learned over large data sets.

The success of deep learning approaches in NLP is by-and-large due to two main innovative technologies: *word embeddings* and *neural sequence to sequence learning*.

### 1.1.2 Word Embeddings

The first technology is about word embeddings (also known as distributed word representations), used as a means of converting words into numerical representations, by observing that words with similar meanings tend to appear within same context(s). Each dimension of word embeddings is responsible for a latent feature of a word, and all dimensions as a whole can represent for useful syntactic and semantic features. Word embeddings are often utilised as a very first processing step, and regarded as a key factor in the success of a deep learning model. There are two leading frameworks which have been invented for learning word embeddings, including (i) local word context(s) representation with Word2Vec (Mikolov et al., 2013a,b; Rong, 2014), and (ii) global word context(s) representation with Glove (Pennington et al., 2014). Both Word2Vec and Glove are powered by pretty simple neural models, but have revolutionized the traditional word representation with one-hot encoding or TF-IDF transformation methods. Remarkably, neural word embeddings also provide a means of mathematical interpretation of words based on vector representations, e.g., `[‘king’] - [‘man’] ≈ [‘queen’] - [‘woman’]` (where `[‘X’]` is a pre-trained word embedding vector of X). Word embeddings also provide an effective mechanism for transfer learning; for instance, pre-
trained embeddings (e.g., on a rich-resourced data) can be plugged into models of a variety of downstream tasks (e.g., sentiment analysis, text classification, tagging, translation, natural language inference) to make them more robust, thanks to the generalised and universal word and sentence representations learned on the larger data. For example, ELMo (Peters et al., 2018), UMLFIT (Howard and Ruder, 2018) and BERT (Devlin et al., 2019) and similar are robust frameworks for doing this based on pre-trained LM embeddings, resulting in state-of-the-art results across many tasks.

1.1.3 Neural Sequence to Sequence Learning

The development of neural models for sequence to sequence learning (seq2seq) has resulted in one of the most influential technologies. Neural seq2seq learning models are based on the encoder-decoder framework (Graves, 2013; Lipton, 2015; Sutskever et al., 2014); where an encoder compresses the input sequence into a distributed vector representation; and then a decoder produces the output sequence based on that representation — as shown in Figure 1.1. Both encoder and decoder are realised by neural networks, e.g., dominantly by recurrent neural networks (RNNs), and learned in an end-to-end fashion. An important characteristic of RNNs is that they provide the capability of capturing long range dependencies effectively, often superior to previous conventional modelling approaches (Goodfellow et al., 2016, Chapter 10). Specifically, Bahdanau et al. (2015) explored the usefulness of an attention mechanism for improving the encoder-decoder framework which allows richer distributed representation that is dynamic to the size of the input sequence. These capabilities make the seq2seq models robust and effective, leading to breakthroughs in a variety of NLP downstream tasks, such as: machine translation (Bahdanau et al., 2015; Gehring et al., 2017c; Lipton, 2015; Sutskever et al., 2014; Vaswani et al., 2017), summarisation (Cheng and Lapata, 2016; Chopra et al., 2016; Rush et al., 2015), and parsing (Chen and Manning, 2014; Gardner et al., 2018; Vinyals et al., 2015a).

1.2 Research Questions and Scope

A neural seq2seq model requires effective algorithms for training, modelling and decoding (a.k.a. inference). During training and modelling, model parameters are learned by optimising the training objective over the annotated training data so that the model can generalise well when unknown testing data is decoded. Although they have produced impressive results so far, neural seq2seq models have some limitations, both for training and decoding, as outlined below.
1.2 Research Questions and Scope

Fig. 1.1 A general view of the encoder-decoder learning framework.

1.2.1 Training and Modelling

The standard method for training and modelling neural sequence models comes from the exposure bias (Bengio et al., 2015; Goyal et al., 2017a) problem. This problem refers to the discrepancy between training with access to ground truth from a prior step as its input, and decoding with access limited to the model’s own predictions. The strategy of using ground truth during training is also regarded as “teacher forcing” (Bengio et al., 2015; Goodfellow et al., 2016). Although this strategy helps quickly and effectively train neural sequence models, however, it can create a mismatch between training and testing, potentially resulting in worsen model generalisation. Second, it might also have the label bias problem (Pereyra et al., 2017; Wiseman and Rush, 2016) in which the model produces overly-confident estimates towards some specific output symbols.

Further, a neural seq2seq model usually works in an end-to-end fashion, showing great benefits, i.e., without feature engineering efforts from practitioners. However, this comes with some limitations. Firstly, there often requires a massive amount of annotated training data for training a neural seq2seq model in order to produce desirable performance. However, there will not always be available data across languages, domains or tasks at hand. Prior and external knowledge provide additional contextual information, potentially improving the modelling performance as well as compensating the lack of large training data, particularly in low-resourced situations. Secondly, it is non-trivial to interpret the predictions from a neural seq2seq model. One way to address this is via the use of prior and external knowledge as shown in non-neural machine learning models (Bellare et al., 2009; Chang et al., 2007; Krupka and Tishby, 2007; Reichart and Barzilay, 2012; V. Graça et al., 2010, inter alia). However, there has been not clear evidence whether a neural seq2seq model has benefits from such prior and external knowledge.
1.2.2 Decoding

The majority of the literature on seq2seq models focuses on training, through the development of better model architectures, or better training strategies. However, the decoding problem is less well-explored. Decoding is a generative process that produces symbols in a target sequence following a sequential order, often from left-to-right. Decoding is non-trivial due to the sequential generation that causes an exponential number of possible combinations of target symbols, leading to an intractable search (Germann et al., 2001). Exact decoding for seq2seq model is unlikely, especially for structured prediction problems, e.g., in machine translation. For decoding, the most widely-used methods are approximate decoding methods, including greedy and beam search algorithms (Bahdanau et al., 2015; Sutskever et al., 2014, inter alia), sampling (Cho, 2016), and reranking (Birch, 2016; Li and Jurafsky, 2016a).

In deep neural networks, particularly with recurrent structures, they provide a means for modelling dependencies on a history of previous contexts, with no need to make a Markovian assumption. However, the models still operate using a left-to-right generation order, hence, ignoring the global dependencies of output sequence symbols during generation. Accordingly, decoding suffers from intractable search due to model’s global dependencies; hence, one is forced to use approximate decoding methods, such as greedy or beam search whose performance tends to be sub-optimal.

1.2.3 Thesis Structure and Contributions

This thesis aims to deal with these research challenges by exploring different scenarios in which using prior and external knowledge can be beneficial for both training and decoding phases of neural seq2seq models. In this thesis, we investigate the use of various kinds of prior and external knowledge, including side information, linguistic factors, bidirectional and bilingual factors, monolingual data, and prior features. We further present different approaches for integrating them into both training and decoding phases of neural sequence models.

The structure of this thesis, as illustrated in Figure 1.2, is as follows:

Chapter §2 — Background examines the neural sequence models in the literature. The following chapters are structured in three main parts, including: Modelling, Optimisation, and Theory, as shown in Figure 1.3.

---

1A Markovian property says that a future state of a stochastic process depends only on a fixed-size window of recent words, not on the sequence of words preceding it (Markov and Nagorny, 2010).
Fig. 1.2 The detailed sections and chapters of this thesis. Note that the colors are associated with different parts as shown in Figure 1.3.
In the **Training and Modelling** part, we first examine the use of different kinds of prior and external knowledge, and then explore different approaches for incorporating them into existing neural seq2seq models. This part includes the following chapters:

- **Chapter §3 — Neural Sequence Models with Side Information.** In this chapter, we examine the benefits of using external side information for neural seq2seq models. Side information has been ignored or not yet fully exploited in neural models. Side information, e.g., meta-data at a document level, which usually accompanies the main data, potentially provides extra contextual information that might be complementary to the main data. We provide a clear view in two case studies with neural language modelling and neural machine translation, which have been shown to benefits from additional side information.

- **Chapter §4 — The Use of Prior Linguistics Knowledge for Neural Machine Translation.** This chapter explores prior knowledge encoded via linguistics factors for improving existing neural translation models. We explore different kinds of linguistics factors, including lemmatisation, word clusters, part-of-speech tags, and syntactic dependency parse trees. We then propose multi-factor attention architectures for incorporating them into the existing neural translation models. We observe their benefits in the contexts of translating between low and extremely-low resourced languages.
• **Chapter §5 — The Use of Monolingual Data for Neural Machine Translation.** In this chapter, we explore the extent to which external monolingual data can be utilised for improving existing neural translation systems. We present a simple but effective iterative back translation strategy in harvesting the monolingual data. Furthermore, we apply this strategy to unsupervised domain adaptation with neural translation models. We obtain remarkable improvements in two translation tasks using this simple technique.

• **Chapter §6 — A Moment Matching Training Framework for Neural Machine Translation.** Neural sequence models have been shown to benefit from prior and external knowledge, e.g., through including this information as features or embeddings as part of the model (as shown in previous chapters). In this chapter, we propose a new means with a different point of view for incorporating prior and external knowledge based on the moment matching theory. In our approach, standard cross-entropy training is augmented with a moment matching objective that encourages statistics of the predictive distribution to match the empirical distribution. Our technique is related to policy gradient methods from reinforcement learning; however, makes different prior assumptions. We present initial results on neural machine translation, where we show how the technique can control for simple prior knowledge.

In the **Decoding** part, we focus on improving the existing decoding algorithms in the context of neural translation models. This part includes the following chapter:

**Chapter §7 — A Novel Decoding Framework with Relaxed Continuous Optimisation.** In this chapter, we focus on exploiting a new paradigm for decoding of existing neural sequence models. We address one of the drawbacks of existing approximate decoding methods, which is the limited ability to incorporate global factors (e.g., bidirectional, bilingual) due to intractable search space. We propose a novel decoding framework based on the relaxed continuous optimisation technique, allowing to exploit the inter-dependencies among the target symbols in different dimensions; hence, potentially leading to better and more diverse translations. Our method improves over the popular approximate decoding methods, including reranking, on different translation tasks;

And finally, we summarise this thesis’s findings and discuss potential future work in **Chapter 8 — Conclusions and Future Work.**
Chapter 2

Background

Contents

2.1 Neural Sequence Models ............................................. 12
   2.1.1 Sequence Modelling ........................................... 12
   2.1.2 Sequence Generation .......................................... 19
2.2 Training Neural Sequence Models .................................. 33
   2.2.1 The Training Objective Function ............................. 33
   2.2.2 Training Problems with Vanishing/Exploding Gradients .... 36
   2.2.3 Hands-on Techniques on Training Neural Sequence Models .... 40
   2.2.4 Evaluation Measures ......................................... 43
   2.2.5 Overview of Recent Work ................................... 46
2.3 Decoding Methods for Neural Sequence Models .................. 51
   2.3.1 Random Sampling ............................................ 51
   2.3.2 Greedy Decoding ............................................. 52
   2.3.3 Beam Search .................................................. 52
   2.3.4 Reranking/Rescoring ....................................... 54
   2.3.5 Overview of Recent Work ................................... 55
2.4 The Importance of Prior and External Knowledge ............... 58
   2.4.1 Prior and External Knowledge for NLP ...................... 58
   2.4.2 Prior and External Knowledge for Neural Sequence Models .... 59
2.5 Summary ..................................................................... 61
This chapter provides a thorough overview of related prior work to the thesis. Briefly, the aim of this PhD thesis is on investigating the use of prior and external knowledge for neural sequence models, in both training and decoding phases. First, in §2.1 and §2.2, we review current training and modelling approaches on neural sequence models, focusing mainly on language modelling and neural machine translation. Second, in §2.3, we outline existing decoding approaches which have been often applied for neural sequence models, and discuss the need for a new method for global decoding. Finally, in §2.4, we describe the fundamentals on how prior and external knowledge have been used in NLP in general, and discuss why such knowledge will also be beneficial for neural sequence models.

2.1 Neural Sequence Models

The focus of this thesis is on neural sequence models. We review the main milestones that have marked the recent research progress of neural sequence models — as shown in Figure 2.1. Neural sequence models are primarily concerned with sequence modelling and sequence generation. Sequence modelling, on the other hand, is related to language modelling which has been studied for a long time. We believe that the evolution of the language modelling is a main inspiration for modern models of sequence generation. Sequence generation is defined as a task of generating an output sequence conditioned on a given input. An input can be with an arbitrary modality (e.g., speech, text, image, database records) whereas an output is often a sequence, e.g., of speech or text. Note that sequence generation can be regarded as sequence-to-sequence generation, for example, text-to-text generation. Related, language modelling is described as a sequence generation of an output sequence but without conditioning on any input. Currently, sequence generation is a mainstream with a wide range of tasks and applications in NLP as illustrated in Table 2.1. In this thesis, we only focus on one representative application of neural sequence generation, namely neural machine translation — which we review in §2.1.2.

2.1.1 Sequence Modelling

2.1.1.1 Early Statistical Language Models

Here, we briefly review the development of statistical language models before describing our main focus of neural language models.

In NLP, language understanding requires a good language model (LM). The LM is an essential prerequisite of almost any NLP applications, e.g., speech recognition, machine
### 2.1 Neural Sequence Models

<table>
<thead>
<tr>
<th>Application</th>
<th>Input</th>
<th>Sequence Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tagging (Ma and Hovy, 2016)</td>
<td>a text sentence</td>
<td>a tagged sequence with syntactic structured output, e.g., named entities or part-of-speeches</td>
</tr>
<tr>
<td>Machine Translation (Bahdanau et al., 2015)</td>
<td>an input sentence in one language, e.g., French</td>
<td>a translation in another language, e.g., English</td>
</tr>
<tr>
<td>Image Captioning (Vinyals et al., 2015b)</td>
<td>an image</td>
<td>a meaningful caption describing the image content</td>
</tr>
<tr>
<td>Summarisation (Rush et al., 2015)</td>
<td>a document or a set of documents</td>
<td>a concise summary</td>
</tr>
<tr>
<td>Syntactic Parsing (Chen and Manning, 2014)</td>
<td>a natural language text</td>
<td>a syntactic analysis with either constituency or dependency structure</td>
</tr>
<tr>
<td>Semantic Parsing (Gardner et al., 2018)</td>
<td>a natural language text</td>
<td>a semantically-driven parse, e.g., in a logical form</td>
</tr>
<tr>
<td>Question Answering (Kumar et al., 2016)</td>
<td>a textual question and a textual document or an image</td>
<td>a meaningful answer (e.g., a short text or a specified piece of texts in a given document)</td>
</tr>
<tr>
<td>Speech Recognition (Chorowski et al., 2015)</td>
<td>an input speech (e.g., audible input)</td>
<td>an transcription of that speech input</td>
</tr>
<tr>
<td>Direct Speech Translation (Jia et al., 2019)</td>
<td>an input speech (audible input) in one language, e.g., French</td>
<td>an output speech outputted in another language, e.g., English</td>
</tr>
</tbody>
</table>

Table 2.1 Applications of neural sequence models across NLP tasks. Note that we only give one representative work for each application.

Translation, etc. According to Jurafsky and Martin (2009), LM is defined as a task of predicting linguistic units (e.g., words, sentences, etc.) given preceding contexts. Research in LM has evolved in count-based and continuous-based milestones, as shown in Figure 2.1 (top).

Early language modelling has been using statistical techniques (Charniak, 1993; Jurafsky and Martin, 2009). The probability of a word in a text sequence is calculated based on its occurrences collected from a large corpus. The most typical example is the \( n \)-gram LM. In a \( n \)-gram LM, predicting a word sequence \( x = x_1, x_2, \ldots, x_\ell \) (with its length \( \ell \)) is factorized into predicting individual words at a time; and each word is predicted by conditioning on a certain number of preceding words — which is the so-called Markovian assumption. In the other words, the LM probability \( p(x_1, x_2, \ldots, x_\ell) \) can be approximated as a product of word probabilities based on \( n \) – 1 preceding words, mathematically formulated as:

\[
p(x_1, x_2, \ldots, x_\ell) \approx \prod_{i=1}^{\ell} p(x_i | x_{i-n+1}, \ldots, x_{i-1}) ;
\]

where \( n \) is called as the order of \( n \)-gram LM and often has the value range, e.g., \( 1 \leq n \leq 5 \); and the estimation of word prediction probabilities is often based on maximum likelihood
**Background**

**LM Progression**

Statistical N-Gram | FF-NNLM | RNNLM | Optimised RNNLM
---|---|---|---
Bengio et al. | Mikolov et al. | deep net. + reg. & opt.

**MT Progression**

RBMT | EBMT | SMT | NMT
---|---|---|---
Direct MT | Transfer MT | Interlingua MT | enc-dec
MT | word | syntax | lstm-gru
MT | phrase | convolution | attention

**Seq2Seq Progression**

feed-forward | enc-dec | enc-dec w/ attention | enc-dec w/ convolution | transformer
---|---|---|---|---

Fig. 2.1 Research evolution in neural sequence models with main focus on language modelling, neural machine translation.

estimation (MLE) on large corpora, e.g.,

\[
p(x_t|x_{t-n+1}, \ldots, x_{t-1}) = \frac{\text{count}(x_{t-n+1}, \ldots, x_{t-1}, x_t)}{\sum_{x'} \text{count}(x_{t-n+1}, \ldots, x_{t-1}, x')} \tag{2.2}
\]

where \(\text{count}(.)\) is a count of word sequence occurrence in a large corpora. Note that these counts in Equation 2.2 may be zeros due to the data sparsity, even with large corpora. Various solutions for this were proposed, e.g., back-off and smoothing techniques (Chen and Goodman, 1996; Kneser and Ney, 1995); but they have their own deficiencies. Though successful, there are several drawbacks of \(n\)-gram LM. First, \(n\)-gram LM relies much on exact matching of \(n\)-grams on large corpora; hence, it suffers from the data sparsity problem when encountering unseen word sequences or dealing with synonyms or out-of-vocabulary (OOV) words. Second, more importantly, \(n\)-gram LM is unable to deal with modelling long distance dependencies, due to the Markovian assumption.
2.1 Neural Sequence Models

2.1.1.2 The Era of Neural Language Models

Recent development of language modelling has been dominated by the use of neural networks — which can be regarded as neural language modelling (or NLM for short). Several main models of NLM have been proposed to address two main weaknesses of statistical language modelling.

Neural Probabilistic Language Model. First, Bengio et al. (2003) proposed a neural probabilistic language model (NPLM) of which overall architecture is shown in Figure 2.2. The core idea of NPLM is to learn the conditional probability of generating next word given a history of $n - 1$ previous words $p(x_t | x_{t-n+1}, \ldots, x_{t-1})$ through parameterising words as distributed word vectors or so-called word embeddings. A given word at position $t$ as $x_t$ is formulated as:

$$ x_t \equiv W_e I_{x_t}, $$

(2.3)
where \( \mathbf{x}_t \) (in bold) indicates the embedding of the word at position \( t \) in the input sequence \( \mathbf{x} \) with length \( \ell \); and \( \mathbf{W}_e \in \mathbb{R}^{H \times |V|} \) is a word embedding weight matrix added as additional model parameters which will be learned along with the model; \( |V| \) is a vocabulary size and \( H \) is the predefined dimension for word embedding. Also, \( \mathbf{1}_{x_t} \in \mathbb{R}^{|V|} \) refers to one-hot vector indicating the given word \( x_t \) in the vocabulary \( V \). Then, these word embeddings will be fed as inputs to a feed-forward neural network. Here, a feed-forward neural network is expressed as a non-linear transformation function \( g \) (e.g., \( \tanh \)) over the representations of the words in context, e.g., \( g(\mathbf{x}_{t-n+1}, \ldots, \mathbf{x}_{t-1}) \). As a result, the conditional probability \( p(x_t | x_{t-n+1}, \ldots, x_{t-1}) \) is computed via a softmax activation function over the resulting output of \( g(.) \). In comparison to the n-gram LMs, one important benefit of using distributed representations of the words with a neural network is to allow for better modelling of the relationships between words as well as the prediction of synonyms; hence, reducing the data sparsity problem since smoothing can be implicitly performed. Two major weaknesses of the NPLM approach include the issue of long training requirement and the use of fixed length context. For handling long training time, several speed-up techniques were proposed, e.g., representing the words in the vocabulary as a binary hierarchical tree built using external knowledge (Morin and Bengio, 2005).

**Recurrent Neural Network Language Model.** Similar to n-gram LMs, the second weakness of the NPLM approach is on the use of fixed length history context, e.g., the \( n-1 \) previous words. Mikolov (2010, 2012) addressed this weakness by using recurrent connections in a neural network, regarded as a recurrent neural network language model (RNNLM) — as illustrated in Figure 2.3. Note that recurrent neural networks, also known as RNNs, refer to class of neural networks where outputs from previous hidden states to be used as additional inputs to current hidden states when generating the output symbols.\(^1\) Such recurrent connections essentially allow the RNNs to exhibit temporal dynamic behavior over time; hence, being considerably suitable for modelling sequential information. Adopted from the idea of RNNs, in the RNNLM proposed by Mikolov (2010, 2012), the contextual information of history (i.e., previous words) is implicitly encoded and internally captured inside these recurrent connections. This makes RNNLM suitable for modelling the long-term dependencies of contexts in arbitrary length sequential inputs. In more details, the RNNLM proposed by Mikolov (2010) comprises three layers, including *input*, *hidden* and *output* layers.

The *input* layer can be referred to a word embedding layer, e.g., Equation 2.3 — identical to the NPLM as described earlier. Recall that an input sequence \( \mathbf{x} \) comprises \( \ell \) words

\(^1\)In the context of translation and language modelling, a symbol typically refers to a token (separated by a white space, e.g., in English, French), a word (not separated by a white space but an external word segmenter, e.g., Chinese, Vietnamese) or a sub-word (separated by a sub-word segmenter).
2.1 Neural Sequence Models

Fig. 2.3 An overall architecture of the recurrent neural network language model, adopted from Mikolov (2010).

\[ x_1, \ldots, x_\ell \]. Each sequence is wrapped with special start-of-sequence and end-of-sequence sentinels, e.g., \( x_0 = \langle s \rangle \) and \( x_{\ell+1} = \langle /s \rangle \). The goal is to provide additional contexts for the first decision with start-of-sequence and for allowing “termination” criteria with end-of-sequence; hence, the model is able to describe sequences with arbitrary lengths. We have a vocabulary \( V \) of fixed size \( |V| \). Due to its fixed size, any out-of-vocabulary words are marked as “(unk)” symbol.\(^2\) We denote their word embedding vectors as

\[
x_0, x_1, \ldots, x_\ell, x_{\ell+1} \in \mathbb{R}^{H_I};
\]

where \( H_I \) refers to predefined input dimension.

\(^2\)Limiting a vocabulary size is required for working with neural networks since a large vocabulary increases network size, which often leads to computational overhead in training and decoding.
### Function Name | Equation
--- | ---
Sigmoid | \[ f(z) = \frac{1}{1 + \exp^{-z}} \]
Tanh | \[ f(z) = \frac{\exp^z - \exp^{-z}}{\exp^z + \exp^{-z}} \]
RELU | \[ f(z) = \max(0, z) \]
SoftMax | \[ f(z_j) = \frac{\exp^{z_j}}{\sum_{k=1}^{K} \exp^{z_k}}, j = 1, \ldots, K \]

Table 2.2 Commonly-used non-linear activation functions

The hidden layer is expressed via a recurrent unit (RU). For each time step \( t \), the current hidden state \( h_t \) can be computed as:

\[
h_t = RU(x_t, h_{t-1}) = f(W^{(hh)}h_{t-1} + W^{(ih)}x_t + b^{(h)}).,
\]

where \( x_t \) is word embedding of word \( x_t \) as defined earlier; and \( h_{t-1} \) is the vector expressing the previous hidden state from the RU. Here, the RU introduces additional coefficients \( W^{(hh)} \in \mathbb{R}^{H \times H}, W^{(ih)} \in \mathbb{R}^{H \times H_I}, b^{(h)} \in \mathbb{R}^H \) that will be shared temporally and learned along with the model; and \( H \) is predefined hidden dimension, and \(|V|\) is the size of the vocabulary. For simplicity, in practice, one often defines the same value for both input and hidden dimensions, e.g., \( H_I = H \). A standard RU defines a linear transformation which takes as its inputs \( x_t \) and \( h_{t-1} \) followed by a non-linear activation function \( f \). There are different options for the non-linear activation function \( f \) used in practice, e.g., sigmoid, tanh, rectified linear unit (or relu) — which can be seen in Table 2.2. In practice, one may want to add multiple layers of RUs, leading to a deeper network (Arisoy et al., 2012) — which might or might not lead to improved performance, depending on the complexity of the problem. However, there will be an increase in training and decoding time, and potential propensity to “overfit” since a deeper network will attempt to learn the training data really well, but it may not be able to generalise to unseen test data.
Next, in the output layer, the next output symbol $x_{t+1}$ is drawn from the probability distribution via a softmax activation function (see Table 2.2), computed based on the output of current hidden state $h_t$ as:

$$x_{t+1} \sim \text{softmax} \left( W^{(ho)} h_t + b^{(o)} \right) ;$$

where the output layer introduces additional model parameters $W^{(ho)} \in \mathbb{R}^{V \times H}$ and $b^{(o)} \in \mathbb{R}^V$ to be learned by the model. We describe an “unfolded view” of the RNNLM architecture over time steps in Figure 2.4. As seen in this figure, RNNLM introduces its parameters, e.g., matrix weights from input layer to hidden layer $W^{(ih)}$, from hidden layer to hidden layer $W^{(hh)}$ (recurrent connection) and from hidden layer to output layer $W^{(ho)}$. More importantly, all these parameters are shared across all time steps.

2.1.2 Sequence Generation

Now we turn to sequence to sequence generation (seq2seq), which is an application of sequence modelling. In terms of a general definition, seq2seq refers to a model that is capable of generating an output sequence given an input sequence.

Here, we focus on the application to neural machine translation (or NMT). The progression of neural sequence generation is closely tied with the development of NMT. Although NMT has just appeared recently (since 2014), it has opened a new era in MT for both research and industry — as shown in Figure 2.1 (bottom).

2.1.2.1 Encoder-Decoder Model

According to our best knowledge, Devlin et al. (2014) is the first work of leveraging neural networks for machine translation. In this work, they successfully incorporated contextual
Fig. 2.5 A general and detailed views of the encode-decoder model with an example translating from English to Vietnamese.

features learned by feed-forward neural networks applied in the context of SMT (Koehn, 2010). This work indeed sheds light on a new era for neural approaches in machine translation.

Shortly after that, in late 2014, Sutskever et al. (2014) proposed a means of learning pure neural networks for machine translation (MT). In their work, they proposed an encoder-decoder architecture where an encoder consumes the entire input sequence into a vector
representation of a fixed dimensionality (also called as a context); and then another decoder produces the output sequence (e.g., a text sentence in a target language) based on that context vector. Here, in the MT task, the input and output sequences are text sentences in the source and target languages, respectively. Figure 2.5(a) shows a general view of an encoder-decoder architecture. Note that both encoder and decoder in this architecture are governed by the LSTM-based RNNs. Given the success of RNNs in sequence modelling as discussed earlier in §2.1.1, Sutskever et al. (2014) employed these components as building blocks in NMT models. The idea is simple, and it has remarkable characteristics: i) it requires minimal knowledge about the structures of the input and output sequences; ii) it offers an end-to-end learning method without engineering efforts, unlike conventional MT approaches, e.g., Rule-based (Nirenburg et al., 1986), Example-based (Hutchins, 2005), or Statistical (Koehn, 2010) MT — which are all riddled with much more complexity (Jurafsky and Martin, 2009, Ch. 25, p. 879).

The detail of an encoder-decoder model illustrated as in Figure 2.5(b). Here, the conditional probability of an output sequence $y = (y_1, \ldots, y_{\ell_y})$ is modelled given an input sequence $x = (x_1, \ldots, x_{\ell_x})$ as $p(y|x)$ ; where $\ell_x$ and $\ell_y$ are token-based lengths of input and output sequences, respectively. Note that each sequence requires to be wrapped with special start-of-sentence ($x_0$ and $y_0$ refer to “(s)”) and end-of-sentence ($x_{\ell_x+1}$ and $y_{\ell_y+1}$ refer to “(s)” symbols, allowing the model to manipulate sequences with arbitrary lengths. All the words in the input and output sequences belong to source and target vocabularies of fixed sizes, denoted as $V_x$ and $V_y$, respectively. Out-of-vocabulary words in the parallel data will be marked using “\texttt{\langle unk\rangle}” symbol.\footnote{Later, we will discuss the Byte Pair Encoding (BPE) method (Sennrich et al., 2016d) that effectively handles the problem of out-of-vocabulary words in NMT.} The conditional probability is formulated as follows:

$$p(y|x) = p(\ldots, y_i; x_0, x_1, \ldots, x_{\ell_x})$$

Each factor $p(y_{i+1}|y_0, \ldots, y_i; x_0, \ldots, x_{\ell_x+1})$ is defined as:

$$p(y_{i+1}|y_0, \ldots, y_i; x_0, \ldots, x_{\ell_x+1}) = p(y_{i+1}|y_0, \ldots, y_i; c_x).$$

Equivalently, we can say that this is a sequence modelling problem (which we’ve discussed in §2.1.1) but with a condition, where a RNNLM over the target sequence $y$ is conditioned
on the representation of the input sequence \( \mathbf{x} \) denoted as \( \mathbf{c}_x \). On the other hand, \( \mathbf{c}_x \) also plays a role as a context vector for generating the output symbols in the target sequence \( \mathbf{y} \).

**Word Embeddings.** As for the RNNLM, the first step is to convert the discrete words in input and output sequences into word embeddings. Here, we denote

\[
(\mathbf{x}_0, \mathbf{x}_1, \ldots, \mathbf{x}_{\ell_x}, \mathbf{x}_{\ell_x+1}) = (W_{e_x} \mathbf{1}_x(\mathbf{s}), W_{e_x} \mathbf{1}_{x_1}, \ldots, W_{e_x} \mathbf{1}_{x_{\ell_x}}, W_{e_x} \mathbf{1}_{(\mathbf{s})})
\]

and

\[
(\mathbf{y}_0, \mathbf{y}_1, \ldots, \mathbf{y}_{\ell_y}, \mathbf{y}_{\ell_y+1}) = (W_{e_y} \mathbf{1}_y(\mathbf{s}), W_{e_y} \mathbf{1}_{y_1}, \ldots, W_{e_y} \mathbf{1}_{y_{\ell_y}}, W_{e_y} \mathbf{1}_{(\mathbf{s})})
\]

as embeddings of words in the sequences \( \mathbf{x} \) and \( \mathbf{y} \), respectively. Here, we have two word embedding weight matrices \( W_{e_x} \in \mathbb{R}^{H \times |\mathcal{V}_x|} \) and \( W_{e_y} \in \mathbb{R}^{H \times |\mathcal{V}_y|} \) (where \( H \) is a pre-defined embedding dimension; and \( |\mathcal{V}_x| \) and \( |\mathcal{V}_y| \) are source and target vocabulary sizes, respectively) — which will be learned along the model. For simplicity, we only use one \( H \) value to represent the dimensions of both source and target word embeddings.

**Encoder.** Now, we proceed to compute the context vector \( \mathbf{c}_x \) in Equation 2.7. Sutskever et al. (2014) used a standard LSTM-based RNNLM to encode the input sequence into a distributed vector of fixed dimensionality, formulated as:

\[
\mathbf{c}_x = h_{\ell_x+1} = \text{LSTM-RNN}^W_{\text{enc}}(h_{\ell_x}, \mathbf{x}_{\ell_x+1}) ;
\]

where LSTM-RNN$^W_{\text{enc}}$ is a LSTM-based RNNLM with its own model parameters \( \psi \), specified for the encoder; \( h_{\ell_x+1} \in \mathbb{R}^H \) is the hidden state at the last position \( \ell_x+1 \) (a.k.a the end-of-sentence symbol \( \langle s \rangle \)), obtained by running the LSTM-RNN through the entire sequence \( \mathbf{x} = (x_0, \ldots, x_{\ell_x+1}) \). Note that the input sequence is best processed in reversed order (Sutskever et al., 2014). This reversal facilitates easier correspondences between the input and output sequences due to the introduction of some short-term dependencies, e.g., the first few words in the input sequence move closer to the first few words in the output sequence. Also note that one often can choose a shallow or deep architecture for their LSTM option depending on the available resources. Deep LSTMs in principle outperform shallow LSTM (Sutskever et al., 2014); however, models with deep LSTMs are more difficult for training (Pascanu et al., 2013).

**Decoder.** Given the context vector \( \mathbf{c}_x \), the goal of a decoder is to generate the output symbols sequentially, from a left-to-right direction. A decoder performs as another LSTM-based
RNNLM (denoted as LSTM-RNN$^\phi_{\text{dec}}$ which has its own model parameters $\phi$) over the target sequence $y$, but takes as its extra input the context vector $c_x$. In practice, this context vector $c_x$ will be concatenated with the representation of each output symbols to form the new inputs at each time step. Each next output symbol $y_{t+1}$ is drawn from a probability distribution $p(y_{t+1}|y_0, \ldots, y_t; c_x)$ computed via a softmax over the current hidden state of the LSTM-RNN$^\phi_{\text{dec}}$, formulated as:

$$g_t = \text{LSTM-RNN}^\phi_{\text{dec}}(g_{t-1}, \left[ y_t \atop c_x \right])$$

$$p(y_{t+1}|y_0, \ldots, y_t; c_x) = \text{softmax}(W^{(ho)}g_t + b^{(o)})$$

$$y_{t+1} \sim p(y_{t+1}|y_0, \ldots, y_t; c_x);$$

where $g_t$ refers to the hidden state of the LSTM-RNN$^\phi_{\text{dec}}$ at a time step $t$; $W^{(ho)} \in \mathbb{R}^{V_y \times H}$ and $b^{(o)} \in \mathbb{R}^{V_y}$ are respective parameters for weight matrix and bias term to be learned along the model.

### 2.1.2.2 Attention-based Encoder-Decoder Model

One of the weaknesses of the encoder-decoder model is the use of a fixed-size context vector representation ($c_x$ in Figure 2.5b). It is too much for one context vector to carry all important information from the input sequence, particularly for long sequences, the model is prone to forgetting the information of distant words in the sequence due to the recurrent structure of the encoder, even with the use of LSTM or GRU. Bahdanau et al. (2015) addressed this weakness by introducing two major improvements for the encoder-decoder model.

**Encoder.** The first improvement is to use bidirectional (forward and backward) RNNs for encoding the input sequence. The forward and backward RNNs take as their inputs by reading the input sequence in the forward and backward directions, respectively; and as a result produce the two hidden states for each input symbol. This bidirectional encoding scheme captures not only the position specific information, but also the richer information coming from both left and right contexts. Another way to represent this encoder is in the Transformer architecture, which we review in §2.1.2.3.
Bidirectional RNNs (biRNN) encoders are formulated as follows:

\[
\mathbf{h}_j = \text{biRNN}^\psi_{\text{enc}} \left( x_j, \left[ \mathbf{h}_{j-1}, \mathbf{h}_{j+1} \right] \right)
\]

\[
= \begin{bmatrix}
\text{forwardRNN}^\psi_{\text{enc}} (x_j, \mathbf{h}_{j-1}) \\
\text{backwardRNN}^\psi_{\text{enc}} (x_j, \mathbf{h}_{j+1})
\end{bmatrix};
\] (2.10)

where \( \mathbf{h}_j \in \mathbb{R}^{2H} \) is the annotation for the input symbol at a specific position \( j \); \( \psi = [\mathbf{\bar{w}}; \mathbf{\bar{v}}] \) are the model parameters of respective forward and backward RNNs running over the entire input sequence \( \mathbf{x} \) with its length \( \ell_x + 2 \) (wrapped by \( \langle s \rangle \) and \( \langle /s \rangle \) symbols as discussed earlier); and \( x_j \in \mathbb{R}^H \) is the word embedding at position \( j \in [1, \ell_x] \); and \( \mathbf{h}_j \) and \( \mathbf{\bar{h}}_j \) are the hidden states of forward and backward RNNs, respectively.

**Attention and Decoder.** Bahdanau et al. (2015) also proposed the use of attention over the input sequence. The main idea is to use a dynamic mechanism for the context vector \( \mathbf{c}_i \) that enables the decoder to attend to different parts of the input sequence at each step of generating the output sequence. In that sense, the model uses a dynamic context vector denoted as \( \mathbf{c}_i \) for each position \( i \) in the output sequence \( \mathbf{y} \) with its length \( \ell_y + 2 \) (also wrapped by \( \langle s \rangle \) and \( \langle /s \rangle \) symbols as discussed earlier).\(^4\) Then, each context vector \( \mathbf{c}_i \) is computed as weighted linear combination of the hidden states produced by the bidirectional (forward and backward) RNNs of the encoder, formulated as:

\[
\mathbf{c}_i = \sum_{j=1}^{\ell_x} \alpha_{ij} \mathbf{h}_j; \] (2.11)

where \( \mathbf{h}_j \) is computed as in Equation 2.10; and \( \alpha_i \in \mathbb{R}^{\ell_y} \) is a probability distribution expressing correlations between the current output symbol at a position \( i \) and all the input symbols in the input sequence \( \mathbf{x} \). Note that in the context of machine translation, one also regards this probability \( \alpha_{i} \) as a “soft” alignment probability between the input and output symbols at specific positions.

Here, Bahdanau et al. (2015) proposed that the probability \( \alpha_{i} \) is computed via a single Multi-Layer Perceptron (MLP) that consumes the previous hidden state \( g_{i-1} \) of the decoder.

\(^4\)Here, we use \( i \) instead of \( t \) as used previously to distinguish between time steps in the input and output RNNs.
Fig. 2.6 A detailed view of an encoder-decoder model with attention, translating a sentence from English to Vietnamese.

\[ \alpha_i = \text{softmax} (e_i) = \frac{\exp (e_i)}{\sum_{k=1}^{t} \exp (e_{ik})} \]

\[ e_{ij} = \text{MLP} \left( g_{i-1}, h_j \right) \]

\[ = v_a^\top \tanh (U_a g_{i-1} + V_a h_j + b_a) ; \]

where \( e_{ij} \in \mathbb{R} \) is an unnormalised alignment probability between the input symbol at a position \( j \) and the output symbol at a position \( i \). Here, the MLP is parameterised by a feedforward neural network with a tanh activation function which uses additional model parameters, including \( v_a \in \mathbb{R}^H, U_a \in \mathbb{R}^{H}, V_a \in \mathbb{R}^{H \times 2H}, b_a \in \mathbb{R}^H \). All these parameters will be jointly learned with other parameters of the encoder-decoder model.
Similarly, in an encoder-decoder model with attention, a decoder operated by another RNN is used to predict the output symbols in the target sequence \( y \) sequentially from a left-to-right direction, formulated as:

\[
g_i = \text{RNN}^\phi_{\text{dec}} \left( g_{i-1}, y_{i-1}, c_i \right) \\
g'_i = f' \left( c_i, y_{i-1}, g_i \right) \\
y_{t+1} \sim \text{softmax} \left( W^{(ho)} g'_t + b^{(o)} \right) ;
\]  

(2.13)

where \( \text{RNN}^\phi_{\text{dec}} \) is denoted as a decoder RNN with model parameters \( \phi \) operating over the output sequence. Another difference when compared to the encoder-decoder model without attention as discussed in §2.1.2.1 is the use of an intermediate representation denoted as \( g'_i \) which serves as a deep output layer (via a function \( f' \)) before making predictions over the output symbols. This deep output layer can be a single MLP (Pascanu et al., 2013) or a multilayer network with a single maxout (Goodfellow et al., 2013). We illustrate a detailed view of this encoder-decoder model with attention as in Figure 2.6.

**Discussion.** Overall, the encoder-decoder model with attention mechanism provides a novel architecture, setting a new milestone for neural machine translation. In their benchmarks in translating English to French, Bahdanau et al. (2015) showed that this model reached state-of-the-art results, and were comparable to the existing dominating phrase-based SMT systems. We believe this was a striking achievement given the fact that the development of neural machine translation had just appeared. This idea has strongly influenced not only the MT field but the other fields in NLP as well.

However, Koehn and Knowles (2017) empirically pointed out six main challenges with the encoder-decoder model with or without attention, including: a) rare words; b) domain mismatch; c) the amount of training data; d) long sentences; e) word alignment; and f) beam search. Some of these will be discussed in §2.2, for instance, addressing rare words in §2.2.5.1; improving long sentences in §2.3.3.2 and word alignment in §2.2.5.3. In this thesis, we aim to address some of these challenges, including utilising monolingual data presented in §5.1; handling domain mismatch in §5.2; and decoding with global factors for greedy and beam search algorithms in §7.

### 2.1.2.3 Encoder-Decoder Model with Transformers

Conventionally, both encoder and decoder models, e.g., without attention (§2.1.2.1) and with attention (§2.1.2.2) are dependent on the use of RNNs. Encoder RNNs sequentially read the input sequence into a context representation; then decoder RNNs sequentially generate
2.1 Neural Sequence Models

This is my PhD thesis.

An English sentence

Multi-Head Self-Attention
Add & Layer-Normalise
Add & Layer-Normalise
Feed Forward Network

Feed Forward Network
Multi-Head Source Attention
Add & Layer-Normalise
Multi-Head Self-Attention

Fig. 2.7 The Transformer architecture.

the output sequence. There are two limitations with these approaches. The first limitation is about the sequential property of RNNs. For example, computing the hidden states $h_j$ (in the encoder) and $g_i$ (in the decoder) requires the previous hidden states $h_{j-1}$ and $g_{i-1}$, respectively. Therefore, it is impossible to process the inputs for both encoder and decoder RNNs in parallel. This is particularly important for speeding up the training of deep neural networks, especially when dealing with large-scale data. Also, another limitation of RNN-based sequence methods is on learning long range relationships among the symbols in both input and output sequences. Currently, the relationships between symbols in sequences rely solely on the recurrent structures of encoder and decoder RNNs; and encoder and decoder RNNs are connected through a single attention mechanism (Bahdanau et al., 2015). This still remains an obstacle for modelling more fine-grained and complicated relationships between symbols.

Vaswani et al. (2017) introduced a novel architecture, namely the Transformer. In general, the Transformer still follows the philosophy of the encoder-decoder architecture of NMT models (Sutskever et al., 2014). However, there are major differences in the encoder and decoder such as: a) the complete replacement of the RNNs by using residual deep net-
works for the encoder and decoder; b) employing more fine-grained attention mechanism. Such changes lead to some advantages, better parallel model training, and better modelling of complicated relationships between symbols in the sequences.

Overall, the Transformer model architecture are illustrated in Figure 2.7, comprising the following components:

**Embeddings.** Similar to the (attentional) encoder-decoder models (Bahdanau et al., 2015; Sutskever et al., 2014), both source and target sequences (with their lengths $\ell_x$ and $\ell_y$) will go through embedding layers to produce the word embedding representation of the same dimensionality (e.g., $H_d$), denoted as $E^w_x \in \mathbb{R}^{H_d \times \ell_x}$ and $E^w_y \in \mathbb{R}^{H_d \times \ell_y}$:

$$E^w_x = [W^s_{pi} \mathbb{1}_x \ldots W^s_{pi} \mathbb{1}_x]$$
$$E^w_y = [W^t_{pi} \mathbb{1}_y \ldots W^t_{pi} \mathbb{1}_y].$$

(2.14)

The Transformer aims at parallelisation in computation. Unlike in RNN-based models, it does not align positions of words to steps in computation time. In order to preserve the sequential nature, additional positional encodings for both source and target sequences are used, denoted as $E^p_x \in \mathbb{R}^{H_d \times \ell_x}$ and $E^p_y \in \mathbb{R}^{H_d \times \ell_y}$, respectively. There are two equivalent methods for positional encodings. The first method (Gehring et al., 2017c) introduces an additional weight matrices $W^s_{pi} \in \mathbb{R}^{H_d \times \ell_{smax}}$ and $W^t_{pi} \in \mathbb{R}^{H_d \times \ell_{tmax}}$ to be learned alongside the model, formulated as:

$$E^p_x = [W^s_{pi} \mathbb{1}_x \ldots W^s_{pi} \mathbb{1}_x]$$
$$E^p_y = [W^t_{pi} \mathbb{1}_y \ldots W^t_{pi} \mathbb{1}_y].$$

(2.15)

where $\mathbb{1}_p$ is one-hot coding representation of the position $p$ in the respective sequence; $\ell_{smax}$ and $\ell_{tmax}$ are the maximum source and target sequence length to be chosen. Note that these values can be chosen differently.

The second method proposed by Vaswani et al. (2017) is a sinusoid-based fixed positional encoding, formulated as:

$$E^p_x = [f_s(0), \ldots, f_s(p), \ldots, f_s(\ell_x)]$$
$$E^p_y = [f_s(0), \ldots, f_s(p), \ldots, f_s(\ell_y)];$$

(2.16)

where the subscript “s” refers to sinusoid; $f_s(p) \in \mathbb{R}^{H_d}$ is a positional encoding vector at a position $p$ in the sequence, computed based on sine and cosine functions of different
frequencies — similar to wavelengths with a geometric progression from $2\pi$ to $10000 \times 2\pi$:

$$f_s(p) = \begin{bmatrix} \vdots \\ PE(p, 2i) \\ PE(p, 2i + 1) \\ \vdots \end{bmatrix}$$

(2.17)

$$PE(p, 2i) = \sin \left( \frac{p}{10000 \frac{2\pi}{H_d}} \right)$$

$$PE(p, 2i + 1) = \cos \left( \frac{p}{10000 \frac{2\pi}{H_d}} \right);$$

where $i \in \left[0, \frac{H_d-1}{2}\right]$. Note that the advantage of the sinusoid positional encoding method is that it does not introduce additional model parameters, and more importantly it can handle sequences with arbitrary lengths.

Then, the total source and target embeddings $E_x$ and $E_y$ are computed as:

$$E_x = E_x^w + E_x^p$$

$$E_y = E_y^w + E_y^p.$$  

(2.18)

**Transformer Encoder and Decoder Stacks.** In the Transformer, an encoder comprises a stack of $N$ identical encoder layers (denoted as EncoderLayer). Each EncoderLayer has two main components, including multi-head self attention (denoted as MultiHeadSelfAttn) sub-layer and position-wise fully-connected feed-forward network (denoted as pFFN) sub-layer. Residual connections (He et al., 2016b) and layer normalisation (Ioffe and Szegedy, 2015) (denoted as LayerNorm) are employed around MultiHeadSelfAttn and pFFN sub-layers. The computation of an EncoderLayer given an input $X^\ell$ at layer $\ell$, denoted as EncoderLayer ($X^\ell$) is as follows:

$$\text{EncoderLayer} \left( X^\ell \right) = \text{LayerNorm} \left( X_{mha_{self}}^\ell + \text{pFFN} \left( X_{mha_{self}}^\ell \right) \right)$$

(2.19)

$$X_{mha_{self}}^\ell = \text{LayerNorm} \left( X^\ell + \text{MultiHeadSelfAttn} \left( X^\ell \right) \right);$$

where LayerNorm is defined as discussed in §2.2.3.

Overall, the output from an encoder is as follows:

$$X_{enc} = \underbrace{\text{EncoderLayer} \left( \ldots \left( \text{EncoderLayer} \left( E_x \right) \right) \right)}_{N \text{ encoder layers}}.$$  

(2.20)
Similar to an encoder, a decoder comprises a stack of $M$ decoder layers (denoted as $DecoderLayer$). A $DecoderLayer$ has also two sub-layers: $MultiHeadSelfAttn$ and $pFFN$. However, the main difference here is a decoder has an additional sub-layer, namely multi-head source attention (denoted as $MultiHeadSrcAttn$). This $MultiHeadSrcAttn$ sits in between $MultiHeadSelfAttn$ and $pFFN$ sub-layers. On the other hand, the computation of an $DecoderLayer$ given an input $Y^ℓ$ at layer $ℓ$, denoted as $DecoderLayer(Y^ℓ|X^enc)$, is as follows:

$$
DecoderLayer(Y^ℓ|X^enc) = \text{LayerNorm} \left( Y^ℓ_{mha,src} + \text{pFFN} \left( Y^ℓ_{mha,src} \right) \right)
$$

$$
Y^ℓ_{mha,src} = \text{LayerNorm} \left( Y^ℓ_{mha,src} + \text{MultiHeadSrcAttn} \left( Y^ℓ_{mha,src}, X^enc \right) \right)
$$

$$
Y^ℓ_{mha,src} = \text{LayerNorm} \left( Y^ℓ + \text{MultiHeadSelfAttn} \left( Y^ℓ \right) \right).
$$

(2.21)

Overall, the output from a decoder is as follows:

$$
Y_{dec} = \underbrace{\text{DecoderLayer} \left( \ldots \left( \text{DecoderLayer} \left( E_y | X^enc \right) \right) \ldots \right)}_{M \text{ decoder layers}}.
$$

(2.22)

One important note is that the $MultiHeadSelfAttn$ sub-layer in the $DecoderLayer$ needs to mask out positions from attending to later positions. This masking trick is to make sure that the attentions for a specific position can depend only on the previous known positions given that decoding generates from left to right; hence, this self-attention mechanism works consistently both in training and decoding/inference. Also note that each of $EncoderLayer$ and $DecoderLayer$ has distinct parameter sets in order to express different levels of abstraction over the respective source and target sequence representations.

**Multi-Head Attention.** The major difference in the Transformer is the introduction of multi-head attention mechanism. First, the Transformer adopts the definition of an attention function as “mapping a query and a set of key-value pairs to an output” where query, keys, values are sets of vectors. This definition applies to both conventional attentional model (Bahdanau et al., 2015) and the Transformer, in which keys (denoted as $K$) and values (denoted as $V$) correspond to hidden states from an encoder (e.g., $\{h_j\}_{j \in [0,ℓ_x]}$ in an attentional encoder-decoder model as discussed earlier), both with dimension $K, V \in \mathbb{R}^{H \times ℓ_x}$, whereas queries (denoted as $Q \in \mathbb{R}^{H \times ℓ_y}$) are hidden states (e.g., $\{g_i\}_{i \in [0,ℓ_y]}$) from a decoder. The Transformer employs the dot-product attention from Luong et al. (2015a) with a scaling factor, computed as:

$$
\text{ScaledDotProdAttn} (Q, K, V) = V \text{softmax} \left( \frac{K^T Q}{S_{\text{attn}}} \right);
$$

(2.23)
where $S_{\text{att}}$ is a scaling factor. Later we will discuss how to specify its value. Note that the advantage of the dot-product attention over additive attention computation (Bahdanau et al., 2015) is on its matrix calculation which can be greatly accelerated, e.g., on GPU(s); hence, it facilitates the parallelisation in computation.

Next, the innovation in the Transformer architecture is that rather than computing the attention once as in Bahdanau et al. (2015), the attention can be executed multiple times in parallel. This is so-called multi-head attention. The resulting multiple attentions can independently capture information from different representations at different positions. It can be regarded as a form of attention ensembling. The multi-head attention can be formulated as:

$$\text{MultiHeadAttn} (Q, K, V) = W^O \text{Concat} (\text{head}_1; \ldots; \text{head}_i; \ldots; \text{head}_h); \quad (2.24)$$

where: $\text{head}_i = \text{ScaledDotProdAttn} \left( W^Q_i Q, W^K_i K, W^V_i V \right)$ and $\{ W^Q_i, W^K_i, W^V_i \}_{i=1}^h \in \mathbb{R}^{d_k \times H}$ and $W^O \in \mathbb{R}^{H \times H}$ are additional parameter matrices to be learned alongside the model. Note that $h$ is the number of heads or parallel independent attentions; and $d_k = \frac{H}{h}$ is a column-wise dimension for each head attention matrix. The division by $h$ means that total computation cost of multi-head attention is similar to that of single-head attention with full $H$. In this case, the scaling factor in the multi-head attention in Equation 2.23 will be computed as $S_{\text{att}} = \sqrt{d_k}$.

The Transformer expresses its attention mechanism in three ways of multi-head attentions:

- **Multi-Head Source Attention** is defined as an attention of each DecoderLayer over the output of the encoder. In this attention, the queries $Q$ come from the previous DecoderLayer and the set of keys $K$ and values $V$ come from the output of the encoder. It is formulated as:

$$\text{MultiHeadSrcAttn} \left( Y^流失_{mha_self}^\ell, X^\text{enc} \right) = \text{MultiHeadAttn} \left( Y^流失_{mha_self}^\ell, X^\text{enc}, X^\text{enc} \right) \in \mathbb{R}^{H \times \ell_y}. \quad (2.25)$$

- **Multi-Head Self Encoder Attention** is defined as an attention of the encoder to itself. In this attention, the queries $Q$ and the set of keys $K$ and values $V$ all come from the same input of EncoderLayer, formulated as:

$$\text{MultiHeadSelfAttn} \left( X^\ell_{enc} \right) = \text{MultiHeadAttn} \left( X^\ell_{enc}, X^\ell_{enc}, X^\ell_{enc} \right) \in \mathbb{R}^{H \times \ell_y}. \quad (2.26)$$
- **Multi-Head Self Decoder Attention** is defined as an attention of the decoder to itself. In this attention, the queries $Q$ and the set of keys $K$ and values $V$ all come from the same input of DecoderLayer formulated as:

$$\text{MultiHeadSelfAttn}(Y^{\text{dec}}_\ell) = \text{MultiHeadAttn}(Y^{\text{dec}}_\ell, Y^{\text{dec}}_\ell, Y^{\text{dec}}_\ell) \in \mathbb{R}^{H \times \ell_y}. \tag{2.27}$$

Note that the above equations 2.25, 2.27 elaborate the definitions of MultiHeadSrcAttn(.) and MultiHeadSelfAttn(.) in equations 2.19 and 2.21. Furthermore, the masking trick will be applied appropriately for each of the computation inside MultiHeadAttn(.) functions.

**Feed Forward Network.** Each of the EncoderLayer and DecoderLayer uses a position-wise fully connected feed-forward network pFFN, defined a two-step linear transformation with a RELU activation in between.

$$\text{pFFN}(X) = W_{ffn}^{2} \text{RELU}(0, W_{ffn}^{1} X + b_{ffn}^{1}) + b_{ffn}^{2}; \tag{2.28}$$

where $W_{ffn}^{2}, W_{ffn}^{1} \in \mathbb{R}^{H \times H}$ and $b_{ffn}^{1}, b_{ffn}^{2} \in \mathbb{R}^{H}$ are additional parameters defined specifically for the feed-forward network to be learned alongside the model.

Similar to the conventional attentional encoder-decoder models (Bahdanau et al., 2015), the output from the final DecoderLayer (denoted as $Y^{\text{dec}}$) will be used to produce the final predictions of the symbols in the target sequence, e.g.,

$$P = \text{softmax}(W^{(hoa)} Y^{\text{dec}} + b^{(o)}); \tag{2.29}$$

Note that $P \in \mathbb{R}^{|V_{y}| \times \ell_y}$ is the prediction matrix of the target symbols. Each of the column from this matrix corresponds to the probability of predicting the target symbol at a specific position.

**Discussion.** Overall, the encoder-decoder model with the Transformer architecture sheds light on applying deep neural networks for developments of neural sequence models. Inspired by this architecture, many exciting sequence models, such as Universal Sentence Encoder (Cer et al., 2018), BERT (Devlin et al., 2019) and others, have been proposed and applied to various language understanding tasks. In this thesis, we applied this architecture for neural machine translation in some parts of this thesis, i.e., §3.2, §5, and §6.
2.2 Training Neural Sequence Models

In this section, we will discuss current approaches for training neural sequence models, to learn the model parameters at training time.

2.2.1 The Training Objective Function

Neural sequence models as presented in §2.1 requires training with a cross entropy loss function. A cross entropy loss is defined as the number of bits we need to encode output symbols from ground truths given model predictions. Suppose we are predicting an output sequence \( y \) with its length \( \ell_y \), then its cross-entropy loss is calculated based on the individual losses at each time step of generating output symbols, formulated as:

\[
\mathcal{L}_{\text{ent}}(\Theta) = \sum_{t=0}^{\ell_y} \mathcal{L}(y_{t+1}, \hat{y}_{t+1}) ;
\]  

(2.30)

where: \( \Theta \) refers to all model parameters and \( \mathcal{L}(y_{t+1}, \hat{y}_{t+1}) \) is a cross-entropy loss of generating next output symbol, denoted as \( \hat{y}_{t+1} \), given the ground truth \( y_{t+1} \).

We derive \( \mathcal{L}(y_{t+1}, \hat{y}_{t+1}) \) as follows:

\[
\mathcal{L}(y_{t+1}, \hat{y}_{t+1}) = -\mathbf{1}_{y_{t+1}}^T \cdot \log \hat{y}_{t+1} ;
\]  

(2.31)

where: \( \mathbf{1}_{y_{t+1}}^T \in \mathbb{R}^{1 \times |V|} \) is transpose of one-hot vector indicating the ground truth \( y_{t+1} \) in the vocabulary \( V \); and \( \hat{y}_{t+1} \in \mathbb{R}^{|V|} \) is defined as \( \hat{y}_{t+1} = p(y_{t+1}|y_{\preceq t}, x) = \text{softmax}(f_{\Theta}^{nn}(y_{t+1}|y_{\preceq t}, x)) \), where \( f_{\Theta}^{nn} \) is defined specific to a model, such as: an RNNLM (in sequence modelling in which \( x \) is not used), or an encoder-decoder model (sequence generation in which \( x \) is an input sequence) — as presented in §2.1.2.

It is important to note that the cross entropy loss function defined above is non-convex and non-linear due to the nature of using deep neural network \( f_{\Theta}^{nn} \). Here, the optimisation problem is to find an “optimal” parameter set that minimises the above loss function, formulated as:

\[
\arg\min_{\Theta} \mathcal{L}_{\text{ent}}(\Theta) .
\]  

(2.32)

During training, the model is learned end-to-end using the standard backpropagation algorithm (Lecun, 1988; Rumelhart et al., 1986) with stochastic gradient descent methods by optimising the training objective function as discussed above. At each time step \( T \), backpropagation is performed through time and the derivative of the loss function \( \mathcal{L} \) in Equation 2.30

\footnote{Recall that the output symbol is wrapped with sentinels, e.g., \( y_0 = \langle s \rangle \) and \( y_{\ell+1} = \langle /s \rangle \).}
with respect to all model parameters (denoted as $\Theta$) is computed as follows:

$$\frac{\partial \mathcal{L}^{(T)}_{\text{xent}}}{\partial \Theta} = \left. \sum_{t=1}^{T} \frac{\partial \mathcal{L}^{(T)}_{\text{xent}}}{\partial \Theta} \right|_{\theta_t}.$$  \hfill (2.33)

There are different gradient descent optimisation methods used for this purpose — which we will review next.

### 2.2.1.1 Gradient Descent Optimisation

Gradient Descent (GD) optimisation provides an elegant and effective way for optimising the cross entropy loss function. The main idea of GD is that it updates model parameters in an opposite direction of their gradients to find the steepest decline in the loss function value. In general, GD performs a number of iterations until convergence to finding a local minimum of a given loss function.

Given a model parameter $\theta$ at a timestep $t$, GD updates it as follows:

$$\theta_{t+1} = \theta_t - \eta \nabla_\theta \mathcal{L}_{\text{xent}} \left( \theta_t \right);$$  \hfill (2.34)

where: $\theta_{t+1}$ is a model parameter for next timestep $t+1$; $\nabla_\theta \mathcal{L}_{\text{xent}} \left( \theta_t \right)$ is first order gradient of the given cross-entropy loss function; and $\eta$ is a learning rate to indicate the pace that model parameter will get updated. Note that learning rate should be chosen carefully. If a learning rate is too large, we might go past the expected minimum, diverge and never reach it. If a learning rate is too small, the learning might take too long to reach it.

In principle, there are two main types of GD optimisation, including: Batch GD, Stochastic GD (SGD), and Mini-batch GD. Batch GD computes its gradient update by using all training instances. SGD computes this update using only one training instance whereas mini-batch GD uses a small batch of training instances. In practice, SGD and mini-batch GD are faster in terms of model convergence, and often lead to better model performance.

### 2.2.1.2 Algorithms for Gradient Descent Optimisation

Various algorithms have been proposed for GD optimisation. According to our best knowledge, these algorithms can be divided into two groups: non-adaptive and adaptive methods. Note that these algorithms only differ in their update equations. For the non-adaptive methods, the learning rate is either a constant or changed according to an external scheduler. Representatives for this group include: Momentum (Qian, 1999), Nesterov Accelerated Gradient (Nesterov, 1983). For the adaptive methods, learning rate is often changed with respect to
<table>
<thead>
<tr>
<th>Method</th>
<th>Update Equation</th>
</tr>
</thead>
<tbody>
<tr>
<td>SGD</td>
<td>( \Delta \theta_t = -\eta \cdot g_t ) ( \theta_t = \theta_t + \Delta \theta_t )</td>
</tr>
<tr>
<td>Momentum</td>
<td>( \Delta \theta_t = -\gamma v_{t-1} - \eta g_t; v_t = \gamma v_{t-1} + \eta g_t )</td>
</tr>
<tr>
<td>NAG</td>
<td>( \Delta \theta_t = -\gamma v_{t-1} - \eta \nabla_{\theta} L_{xent} (\theta_t - \gamma v_{t-1}) )</td>
</tr>
<tr>
<td>Adagrad</td>
<td>( \Delta \theta_t = -\frac{\eta}{\sqrt{G_t} + \epsilon} \odot g_t; G_t = \text{diag} \left( \sum_{j=1}^{i} g_j g_j^T \right) )</td>
</tr>
<tr>
<td>Adadelta</td>
<td>( \Delta \theta_t = -\frac{\text{RMS}[\Delta \theta_{t-1}] g_t}{\text{RMS}[g_t]} ) ( \text{RMS}[\Delta \theta_t] = \sqrt{E[\Delta \theta_t^2]} + \epsilon ) ( E[\Delta \theta_t^2] = \gamma E[\Delta \theta_{t-1}^2] + (1 - \gamma) \Delta \theta_t^2 )</td>
</tr>
<tr>
<td>RMSprop</td>
<td>( \Delta \theta_t = -\frac{\eta}{\sqrt{E[g_t^2]} + \epsilon} g_t )</td>
</tr>
<tr>
<td>Adam</td>
<td>( \Delta \theta_t = -\frac{\eta}{\sqrt{\hat{v}_t} + \epsilon} \hat{m}_t ) ( \hat{m}<em>t = \frac{m_t}{1 - \beta_1^t}; m_t = \beta_1 m</em>{t-1} + (1 - \beta_1) g_t ) ( \hat{v}<em>t = \frac{v_t}{1 - \beta_2^t}; v_t = \beta_2 v</em>{t-1} + (1 - \beta_2) g_t^2 )</td>
</tr>
</tbody>
</table>

Table 2.3 Update equations for GD optimisation algorithms. Note that \( \gamma \) in Momentum is called as momentum term of which value is often set around 0.9; \( \epsilon \) in Adadelta and RMSprop is a smoothing term (e.g., \( 10^{-8} \)) to avoid division by zero; \( \beta_1 \) and \( \beta_2 \) values in Adam are often set to 0.9 and 0.999, respectively.

the changes in parameters at each iteration. Representatives include: AdaGrad (Duchi et al., 2011), AdaDelta (Zeiler, 2012), RMSprop,⁶ and Adam (Kingma and Ba, 2014). Table 2.3 presents a summary of GD algorithms.

Adaptive methods seem to be better in terms of efficiency and performance, particularly useful for sparse features; and are possibly more suited to complex non-convex optimisation. However, it is hard to indicate the best or optimal algorithm, depending on a specific optimisation problem or data complexity. Adam (Kingma and Ba, 2014) is arguably the most popular method used in training neural sequence models.

**Practical Tricks.** There are useful tricks used with GD optimisation in practice, including:

- GD needs regular shuffling (for avoiding learning biases), learning rate annealing (for escaping poor local minima) or curriculum learning (for addressing easy cases first, harder ones later).

GD requires early stopping (e.g., stops training if no improvement is observed on the development data) to avoid over-fitting. GD may need batch normalisation (Ioffe and Szegedy, 2015) to reduce covariance shift of hidden state units during training; hence, potentially helps to prevent over-fitting.

Sometimes, gradient noise (e.g., via Gaussian noise) can be added to the gradient update to make the GD more robust to poor initialisation (Welling and Teh, 2011).

Other advanced techniques can be applied to make GD more robust, e.g., altering between different GD algorithms (Wu et al., 2016); using larger batch size (Smith et al., 2018); using with restarts (Loshchilov and Hutter, 2017); applying snapshot ensembles (Huang et al., 2017); or learning to learn (Bello et al., 2017; Li and Malik, 2017).

### 2.2.2 Training Problems with Vanishing/Exploding Gradients

As discussed in §2.1, neural sequence models are along with the RNNLM architecture. We now turn our discussion on its advantages and disadvantages. In comparison with conventional count-based n-gram LMs, RNNLM offers some unique characteristics. First, it provides a possibility of processing the input sequence of arbitrary length, and more importantly, the model size is not proportional with the length of the input sequence. In contrast, classical count-based n-gram LMs are semi-parametric, relying on an order of grams \( n \) in texts and often leading to a bigger model size when training on large data (Siivola and Pellom, 2005). Second, RNN has great potential of capturing long-term dependencies in sequences thanks to its recurrent connections; as well as identifying synonyms thanks to its distributed representations. Though promising, RNNLM still has some drawbacks. It is quite difficult to train a RNNLM (Pascanu et al., 2012). Second, its computation in training is relatively slow, especially on large-scale data. It requires modern computation utilities such as GPUs (Graphical Processing Units) for fast processing of linear algebraic operations. Also, RNNLM still operates from left-to-right; hence, it cannot process any future input, e.g., leveraging right contexts (Chen et al., 2017c; Serdyuk et al., 2018).

In the context of RNNLM, the vanishing and exploding gradient problems are often encountered during training due to gradient-based learning methods with backpropagation (Pascanu et al., 2012). In the RNNLM, there are many multiplicative gradient updates during backpropagation that can be exponentially decreasing (vanishing gradient) or increasing (exploding gradient), a problem that is exacerbated with an increasing number of recurrent layers. A computationally efficient method to address the exploding gradient problem is to use gradient clipping technique (Mikolov, 2012; Pascanu et al., 2012). Gradient clipping is regarded as a method of rescaling the gradient \( g \) based on its norm \( \|g\| \) according to a particular threshold \( \eta \), e.g., if \( \|g\| > \eta \) then \( g \leftarrow \frac{\eta g}{\|g\|} \). The benefit of doing this is to prevent
2.2 Training Neural Sequence Models

<table>
<thead>
<tr>
<th>Type of gate</th>
<th>Description</th>
<th>RU</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Input gate:</strong></td>
<td>$i_t = \sigma \left( W_i^{(hh)} h_{t-1} + W_i^{(ih)} x_t + b_i^{(h)} \right)$</td>
<td>to convey the current inputs</td>
</tr>
<tr>
<td><strong>Forget gate:</strong></td>
<td>$f_t = \sigma \left( W_f^{(hh)} h_{t-1} + W_f^{(ih)} x_t + b_f^{(h)} \right)$</td>
<td>whether to erase a state?</td>
</tr>
<tr>
<td><strong>Output gate:</strong></td>
<td>$o_t = \sigma \left( W_o^{(hh)} h_{t-1} + W_o^{(ih)} x_t + b_o^{(h)} \right)$</td>
<td>How much to reuse a “cell” state?</td>
</tr>
</tbody>
</table>

Table 2.4 Gates in LSTM and GRU.

an exponential increase in the norm of the gradients; hence, reducing the exploding gradient problem in practice.

2.2.2.1 Recurrent Structures with LSTMs and GRUs

In order to resolve the *vanishing gradient* problem encountered by standard RNNLM, two common types of RU (as mentioned earlier in Equation 2.4) have been proposed, including Long Short-Term Memory (LSTM) (Hochreiter and Schmidhuber, 1997) and Gated Recurrent Unit (GRU) (Cho et al., 2014a). The key idea of LSTM and GRU is to effectively use specific gates to control the information flow (here, referring to the inputs $x_t$ and previous hidden states $h_{t-1}$) in such a way that the important information should be kept through time steps. Table 2.4 summarises the gates and their formulations used in LSTM and GRU. In this table, note that weight matrices $\{ W_i^{(hh)} \}, \{ W_i^{(ih)} \}$, and $\{ b_i^{(h)} \}$ are defined specific to the respective input, forget, and output gates of LSTM and GRU and $\sigma(\cdot)$ is a *sigmoid* function as defined in Table 2.2.

In principle, an LSTM uses three gates for a given time step including input, forget, and output gates ($i_t, f_t, o_t$, respectively), and defines a new cell state $c_t$ to store the temporal information whose recurrence follows a linear scaling instead of a multiplicative and non-linear update. GRU simplifies LSTM by only using two gates including input and forget gates ($i_t$ and $f_t$, respectively), and does not have the cell state $c_t$. The computation of hidden state for each time step can be found in Table 2.5. Note that LSTM has its own weight matrices $W_g^{(hh)}, W_g^{(ix)}, b_g^{(h)}$ whereas GRU has $W_f^{(hh)}, W_f^{(ih)}, b_f^{(h)}$, $\odot$ denotes the element-wise multiplication operator between two vectors; and $i_t, f_t,$ and $o_t$ are input, forget, and output gates, respectively; and their computations can be found in Table 2.4. The use of intermediate cell state (in LSTM) as well as flexible gates for transforming important information and ignoring unnecessary information makes LSTM and GRU more robust than a standard RU. Thanks
Table 2.5 Computation of hidden state $h_t$ at a time step $t$ in LSTM and GRU.

to that, both LSTM- and GRU-based RNNLM increase the capability of memorising long term dependencies and have superior performance than both standard RNNLM and n-gram LM (Cho et al., 2014a); hence, become most popular choices for subsequent applications. Note that Chung et al. (2014) shows that LSTM and GRU are comparable in terms of the modelling performance; however, GRU is slightly less complex (e.g., less parameters and computation) and accordingly is more efficient in practice.

2.2.2.2 Deeper Networks

We now turn to discuss deep LSTM architectures in which several LSTM hidden layers build up progressively and potentially enhance the power of neural networks in modelling sequences. A simple stacked LSTM networks comprise of at least two or more LSTM hidden layers in which the output of previous LSTM hidden layer is fed as the input of the upper one. Further, Yao et al. (2015) proposed Depth-Gated LSTM (DGLSTM) which comprises a stack of LSTMs with extra connections between memory cells in deep layers to regulate information flow across layers. DGLSTM can be regarded as being a generalisation of LSTM networks recurrence to both time and depth dimensions. Such deep recurrent structure may capture long distance patterns at their most general, and the language modelling experiments in (Yao et al., 2015) show the robustness of DGLSTM compared to others (e.g., GRU, LSTM, stacked LSTM). Figure 2.8 compares the similarities and differences between these stacked LSTMs and DGLSTMs. As can be seen in the figure, DGLSTMs introduce
new temporal depth gates for storing depth recurrent connections from lower layer LSTM cell state to upper layer one (see blue connections Figure 2.8). Such gates play a similar role as residual connections in deep neural networks (He et al., 2016b), but are applied specifically for recurrent LSTM networks. Mathematically, the depth gate $d_{i}^{L+1}$ at layer $L + 1$ at time step $t$ is formulated as:

$$
d_{i}^{L+1} = \sigma \left( W_{xd}^{(L+1)} x_{i}^{(L+1)} + W_{cd}^{(L+1)} \odot c_{i-1}^{(L+1)} + W_{id}^{(L+1)} \odot c_{i}^{(L)} + b_{d}^{(L+1)} \right),
$$

(2.35)

where $W_{xd}^{(L+1)}$ and $W_{cd}^{(L+1)}$ are new weight vectors for weighting the past cell state of next LSTM layer $L + 1$ and the cell state at current LSTM layer $L$, respectively; and $b_{d}^{(L+1)}$ is a bias term. Then, this depth gate is added in computing the cell state at upper layer $L + 1$ as follows:

$$
c_{i}^{(L+1)} = d_{i}^{(L+1)} \odot c_{i}^{(L)} + f_{i}^{(L+1)} \odot c_{i-1}^{(L+1)} + i_{i}^{(L+1)} \odot g_{i}^{(L+1)}. 
$$

(2.36)
Other deep architectures for LSTM networks exist such as Grid-LSTM (Kalchbrenner et al., 2015), which is based on similar idea to the DGLSTM.

Despite powerful, RNNLMs based on LSTM networks are hard to train since they have a lot of model parameters as well as hyper-parameters that need to be tuned in an effective and efficient way. There are many research works that attempted to make the RNNLM training more robust and stable; however, according to our best knowledge, there are two out-of-the-box techniques, including regularisation and optimisation. The most representative works are Merity et al. (2018a, b) — which show a collection of best practices based on regularisation and optimisation that are put together to train state-of-the-art LSTM language models. For instance, for regularisation, they proposed a new way for recurrent regularisation with a novel weight-dropped LSTM which applies DropConnect strategy (Wan et al., 2013) on hidden weights of the LSTM to prevent overfitting. Also, for optimisation, they explored a variant of stochastic gradient descent (SGD) optimisation methods called NT-ASGD (Non-monotonically Triggered Average-SGD), which encourages switching to gradient averaging mode based on observing the losses on a validation data. These techniques often lead to better convergence and stabilisation for training RNNLM models.

2.2.3 Hands-on Techniques on Training Neural Sequence Models

Various techniques have been proposed to improve the efficacy of training neural sequential models (Neishi et al., 2017), including:

2.2.3.1 Mini-batching

Mini-batching is one powerful technique to speed up the training of neural sequential models. The main idea behind mini-batching is to calculate the gradients for multiple examples at one time, leveraging the powerful architecture of modern GPUs. Mathematically, it can be formulated as:

$$\nabla_{\Theta} \sum_{(x,y) \text{ in minibatch}} \log p(y|x);$$  \hspace{1cm} (2.37)

where $\Theta$ is a collection of model parameters. Note that the $\sum$ above can be done in parallel; and small batches are arguably used in practice (Neubig, 2017). Overall, training with mini-batching often helps to make the updates to model parameters more stable, hence, leading to significantly faster training convergence for neural sequential models.

---

7also see this blog: https://danijar.com/tips-for-training-recurrent-neural-networks/.
2.2.3.2 Early Stopping

One simple but popular technique of preventing overfitting when training with neural sequential models is early stopping (Yao et al., 2007). The idea behind this technique is to evaluate the model’s performance on a separate development dataset, and then stop training early if the model does not get improved on this dataset. Implicitly, early stopping often has a regularisation effect to prevent the model training (often with gradient descent methods) overfitting on the training data but increasing the generalisation error.

2.2.3.3 Regularisation with Dropout

Another simple technique to prevent neural sequential models from overfitting is dropout (Srivastava et al., 2014). The core idea is to stochastically disable the effects of some units in neural network layers in an appropriate way such that it can help improve the regularisation of the neural network training. Note that during decoding, dropout will not be applied.

In applying to neural sequence modelling, dropout can be utilised in various ways. For example, Zaremba et al. (2014) suggested that dropout should be only applied to the non-recurrent connections. As opposed to Zaremba et al. (2014), Gal and Ghahramani (2016) proposed variational dropout by applying dropout for all inputs, outputs, and recurrent layers at each of the same time step. Further, Semeniuta et al. (2016) applied dropout for the recurrent connections of RNNs to regularise the recurrent weights. Recently, Merity et al. (2018b) proposed AWD-LSTM (termed as ASGD Weight-Dropped LSTM) by applying DropConnect (Wan et al., 2013) on the recurrent hidden to hidden weight matrices combined with variational dropout for all other dropout operations following Gal and Ghahramani (2016). This AWD-LSTM has achieved the state-of-the-art language modelling performance.

Dropout is also very helpful in the context of neural sequence generation (e.g., dominantly by NMT). For conventional attentional encoder-decoder models (Bahdanau et al., 2015; Sutskever et al., 2014), dropout can be applied either to embedding layers or within recurrent connections. For the Transformer model (Vaswani et al., 2017), residual dropout has been applied effectively, e.g., to the output of each sub-layer within each encoder and decoder layers. Though simple, dropout has been proved to be very effective in practice, and is an indispensable part in most of the recent state-of-the-art NMT models.

2.2.3.4 Label Smoothing

In addition to dropout, another effective regularisation technique is label smoothing (Goodfellow et al., 2016; Szegedy et al., 2016). The idea behind label smoothing is to encourage the model to avoid over-confidence towards some output symbols (or labels). Conceptually,
label smoothing is equivalent to injecting noise on the output labels. For example, label smoothing is based on an assumption that ground-truth label \( y \) given an input \( x \) is correct with the probability \( 1 - \epsilon \) where \( \epsilon \) is a pre-defined small constant. As a result, label smoothing computes the so-called “soft” target probability \( p'(y|x) \) on a probability softmax \( p(y|x) \) with \( K \) output labels by imposing the 0 and 1 target classification with targets of \( \frac{\epsilon}{K - 1} \) and \( 1 - \epsilon \), respectively; formulated as:

\[
\log p'(y|x) = (1 - \epsilon) \log p(y|x) + \frac{\epsilon}{K - 1} \sum_{i=1}^{K} \log p(y_i|x);
\]  

(2.38)

Note that label smoothing is applied only to training for improving the model generalisation; and during decoding, it is not required. For example, label smoothing is proved to be effective in the Transformer model (Vaswani et al., 2017).

### 2.2.3.5 Batch and Layer Normalisation

In the context of deep learning models, one important concern is the change in the distribution of the inputs at each layer that forces the intermediate layers to be adaptive to their changing inputs. This is so-called “covariate shift” problem in machine learning. Batch normalisation (Ioffe and Szegedy, 2015) minimises covariate shift by transforming the distribution of the inputs to strictly follow zero mean \( \mu \) and unit variance \( \sigma \). This helps each layer to be able to learn based on more stable distribution of the inputs, hence accelerating the neural network learning process.

The key drawback of batch normalisation is its dependence on the mini-batch on its computation. Ba et al. (2016) introduced an adaptive version of batch normalisation called “layer normalisation” that does not depend on batch computation. The idea behind layer normalisation is that it normalises the inputs across their neuron units within a hidden layer. Ba et al. (2016) showed that layer normalisation is effective for improving the generalisation of recurrent neural networks (RNNs). Table 2.6 compares the formulations of batch and layer normalisation methods.

In the context of the Transformer model (Vaswani et al., 2017) as presented in §2.1.2.3, layer normalisation plays an indispensable role stabilising the interactions between sub-layers in both Transformer encoder and decoders.

### 2.2.3.6 Model Ensemble

Another useful technique used in practice for better modeling is ensemble method (Neubig, 2017). The idea is to leverage different models that can be obtained from different training
2.2 Training Neural Sequence Models

Batch Normalisation | Layer Normalisation
---|---
\[ \mu_j = \frac{1}{B} \sum_{i=1}^{B} x_{i,j} \] & \[ \mu_i = \frac{1}{H} \sum_{j=1}^{H} x_{i,j} \]  
\[ \sigma^2_j = \frac{1}{B} \sum_{i=1}^{B} (x_{ij} - \mu_j)^2 \] & \[ \sigma^2_i = \frac{1}{H} \sum_{j=1}^{H} (x_{ij} - \mu_i)^2 \]  
\[ \hat{x}_{ij} = \frac{x_{ij} - \mu_j}{\sqrt{\sigma^2_j + \epsilon}} \] & \[ \hat{x}_{ij} = \frac{x_{ij} - \mu_i}{\sqrt{\sigma^2_i + \epsilon}} \]

Table 2.6 Formulations of batch and layer normalisation methods. Note that \( B \) and \( H \) are batch and hidden/feature dimensions, respectively. Their equations are conceptually similar, except for the indices.

checkpoints, architectures or selection of training instances. Often, ensemble methods lead to significantly better translation results; however, they are much slower in decoding; hence often impractical in real-time translation systems.

2.2.4 Evaluation Measures

Now we proceed to discuss evaluation methods for neural sequence models which have been presented in §2.1.1 and §2.1.2, including:

2.2.4.1 Evaluation of Language Models

A good language model (LM) often predicts higher probabilities for sequences which are real or actually occur in the observed data than the ones that rarely occur. The best way to evaluating a LM’s quality is using extrinsic evaluation, e.g, measuring through downstream tasks such as speech recognition or machine translation. However, this way requires annotated data, and often is time-consuming. An approximation is to use an intrinsic evaluation with the perplexity measure (denoted as PP), which is commonly used for evaluating language models. Given a LM, a perplexity can be explained as a function of probability of
predicting all words in a given sequence, and is formulated as:

$$PP = p(x_1, \ldots, x_N)^{-\frac{1}{N}}$$

$$= \sqrt[N]{\frac{1}{p(x_1, \ldots, x_N)}} \approx \sqrt[3]{\sqrt[N]{\prod_{t=1}^{N} \frac{1}{p(x_t|x_{t-n+1}, \ldots, x_{t-1})}}}$$

$$= \exp \left( \frac{\sum_{t=1}^{N} \log \left( \frac{1}{p(x_t|x_{t-n+1}, \ldots, x_{t-1})} \right)}{N} \right)$$

(2.39)

where a sequence \(x = x_1, x_2, \ldots, x_N\) has its word length \(N\). In this sense, minimising the perplexity is equivalent to maximising the probability due to the inversion; likely indicating a better language model. Another interpretation of perplexity is about the size of a vocabulary that would imply equivalent performance when using a uniform model. For example, the ideal perplexity of 1 means that the model is certain about predicting the next word; but a perplexity of 1000 indicates uniform choice among 1000 words. Also, out-of-vocabulary (OOV) word rate must be reported when evaluating the quality of LMs. It is unfair to compare language models between neural LMs with count-based \(n\)-gram LMs: i) when the evaluation is computed over different vocabulary sizes, i.e., \(n\)-gram LMs with full vocabularies and neural LMs with fixed size ones.; or ii) when evaluating over a “pre-UNKed” data, which means that the back-off and smoothing techniques (Chen and Goodman, 1996; Kneser and Ney, 1995) are not fairly estimated, as they need to know, e.g., counts for singleton events.

### 2.2.4.2 Evaluation of Neural Machine Translation Models

An extrinsic measure for evaluating the quality of a LM is through a machine translation task, e.g., with BLEU (Bilingual Evaluation Understudy) (Papineni et al., 2002). BLEU is the most popular method used for evaluating the quality of machine translation systems (including NMT). The idea behind BLEU is to compare the matches of up to \(n\)-grams of the translation candidate with the ones in the translation reference(s). Intuitively, the more \(n\)-gram matches the candidate translation has, the better quality it shows. Mathematically,

---

8By limiting the data with a fixed size vocabulary and applying “UNK” for out-of-vocabulary words.
corpus-based BLEU is formulated as:

$$\text{BLEU} = \text{BP} \cdot \exp \left( \sum_{n=1}^{N} w_n \log p_n \right)$$

$$\text{BP} = \begin{cases} 1 & \text{if } c > r \\ \exp(1-r/c) & \text{if } c \leq r \end{cases}$$

(2.40)

$$p_n = \frac{\sum_{C \in \{\text{Candidates}\}} \sum_{n\text{-gram}\in C} \text{Count}_{\text{clip}} (n\text{-gram})}{\sum_{C' \in \{\text{Candidates}\}} \sum_{n\text{-gram}'\in C'} \text{Count} (n\text{-gram}')}$$

where BP is the brevity penalty which is used to penalise translation candidates which are longer than translation references (where \(c\) refers to the length of the translation candidate and \(r\) is the effective reference corpus length); \(p_n\) is the modified \(n\)-gram precision; \(\text{Count}_{\text{clip}}(.)\) is used to clip the total count of each word in the candidate translation with its maximum count in the references, calculated as:

$$\text{Count}_{\text{clip}} (n\text{-gram}) = \min (\text{count}, \max_{\text{ref}_\text{count}}) .$$

The weights \(w_n\) are positive and must be summed to one, and are often chosen uniformly; and \(N\) is used with 4 (hence, so-called as BLEU-4). Note that BLEU always has the range of \([0, 1]\) of which 1 score indicates the perfect translation against the human reference. Evaluating over multiple references often leads to higher BLEU scores.

A statistical significance test (Koehn, 2004) should be done along with BLEU evaluation to provide a reliable evaluation score. A statistical significance test indicates a significant difference in true translation quality for comparing two given translation systems.

Although BLEU correlates well with human judgments (Papineni et al., 2002), it still has certain subjectivity. There are different cases that an improved BLEU score does not sufficiently reflect in terms of translation quality (Callison-Burch et al., 2006). Other alternatives for BLEU include NIST (based on BLEU with some alterations), METEOR (Banerjee and Lavie, 2005), TER (Snover et al., 2006) and so on. Overall, each metric has greater or lesser degree of expressing the translation quality in terms of accuracy, semantics or meanings. Further, Post (2018) proposed SacreBLEU to call for clarity in the reported BLEU scores published in the MT research. SacreBLEU is intended to address a lack of standardisation in comparisons among published BLEU scores due to different tokenisation and normalisation schemes applied to both translation hypotheses and references.
2.2.5 Overview of Recent Work

Over the past few years, we have witnessed rapid progress in neural sequence models, particularly for NMT. Here, we highlight some of recent trends in the developments of the NMT models.

2.2.5.1 Addressing Rare and Unseen Word Problems

One of the key drawbacks of the NMT models is that they are unlikely to generate word distributions to the full extent of the target vocabulary — as in conventional SMT approaches (Koehn et al., 2007, 2003).

Early work such as Jean et al. (2015); Luong et al. (2015b) addressed the problem of limited vocabulary by suggesting that out-of-word vocabulary (OOV) (e.g., rare and unseen) words can be predicted by using pre and post-processing steps based on the alignments extracted from the attention mechanism.

Towards the NMT model with open vocabulary, Sennrich et al. (2016d) proposed to use the sequences of sub-word units instead. Here, the main idea is to apply the sub-word segmentation as a preprocessing step for all sequences both in training and decoding. The sub-word segmentation is a very fast unsupervised algorithm based on Byte Pair Encoding (BPE) scheme (Gage, 1994) — adapted from a text compression literature. Sennrich et al. (2016d) adapted the BPE algorithm for NMT as follows. The BPE algorithm first performs the splitting of the whole sequence into individual sets of characters; then join the consecutive pairs of characters that occur most frequently; and this process is repeated until a BPE vocabulary with pre-defined size is obtained. Next, this vocabulary is then used in a greedy algorithm for segmenting a given sequence. As a result, the vocabulary at a sub-word level can address various cases of rare and unknown words such as: names, compounds, foreign or loan words, etc. Though simple, this BPE scheme is one of the influential techniques used in the existing NMT models, leading to significant boost in end translation performance across challenging datasets. This technique has been used for most subsequent research works in NMT. Further, Kudo (2018) improved the robustness of the NMT model by making use of multiple invariant instances retrieved from a sub-word segmentation algorithm via a regularisation scheme (or called as sub-word regularisation) during training. This sub-word regularisation can be served as a data augmentation method that makes the NMT model exposed to the ambiguities of the sub-word segmentation.

Also, other work suggested that the NMT models can be alternated in more fine-grained levels of word units, e.g., via a hybrid NMT model in which a word model is backed-off by a character model (Luong and Manning, 2016); via fully character-level NMT model (Chung
2.2 Training Neural Sequence Models

et al., 2016; Lee et al., 2017); or via a multiple attention mechanism from different levels of word granularity (Chen et al., 2018a).

2.2.5.2 Model Architecture

As described in Figure 2.1, there are three main milestones of existing NMT architectures, including: RNN-based NMT (Bahdanau et al., 2015; Sutskever et al., 2014), convolution-based NMT (Gehring et al., 2017a,d) and Transformer-based NMT (Vaswani et al., 2017). We have presented in details for the RNN-based and Transformer-based NMT architectures in §2.1.2. In this PhD thesis, we decided to choose these twos for our extensions. However, it is worth noting the convolution-based NMT architecture proposed by Gehring et al. (2017a,d). Similar in spirit to the Transformer (Vaswani et al., 2017), Gehring et al. (2017a,d) removed the reliance on the recurrent structures by employing the multi-layer convolution-based neural networks (CNNs). Due to the characteristics of CNNs, the main advantage of convolution-based NMT is to facilitate parallel training with modern GPU computation. Although convolution-based NMT architecture achieved impressive results compared to the RNN-based models; however, it was later outperformed by the Transformer (Vaswani et al., 2017) in terms of both training speed and performance.

Recently, Chen et al. (2018b) combined the strengths of both RNN and Transformer architectures in a so-called RNMT+ architecture. The key idea is that RNMT+ still uses the recurrent structures in encoder and decoder layers but adopts a multi-head attention interleaved between them.

2.2.5.3 Improving Attention Mechanism

Attention mechanism is one of the breakthrough techniques and plays a key role in the success of the attentional encoder-decoder architectures. The early impressive work by Luong et al. (2015a) explored more variants for the attention mechanism from Bahdanau et al. (2015), suggesting global and local attention mechanisms. In this work, the global attention is computed in a more flexible multiplicative space rather than additive one; whereas the local attention concentrates on a smaller centering area of word contexts rather the full sequence — as in Bahdanau et al. (2015). This work also proposed a simple but effective input feeding strategy that allows the model to be aware of the input sequence at every generation step. Similar in spirit to this, Feng et al. (2016) made use of additional RNN structure for the attention mechanism, hence likely capturing long range dependencies between the attention contexts. Adapted from the alignment models (Brown et al., 1993; Och and Ney, 2000; Vogel et al., 1996) of the SMT literature (Koehn, 2010; Koehn et al., 2007), Cohn et al. (2016) en-
riched the attention mechanism via structural biases; whereas Mi et al. (2016b) integrated the additional alignments (e.g., from statistical alignment models) into the attention mechanism via a supervised learning setting. Also, Tu et al. (2016) proposed a so-called coverage vector to trace the attention history for flexibly adjusting future attentions. Further, Miculicich et al. (2018a) added a self-attention mechanism into the decoder of the existing NMT models (Bahdanau et al., 2015) although this self-attention technique has been invented separately in the Transformer model (Vaswani et al., 2017).

Other work has focused more on improving the attention mechanism in the Transformer architecture (Vaswani et al., 2017). Shaw et al. (2018) proposed an extension of self-attention mechanism that takes the relative relationships between the words in sequences into consideration. These relative relationships are represented by additional parameters to be learned to capture how effective words interact each other.

One of the key drawbacks of the Transformer model is its relatively slower decoding, compared to RNNs and CNNs models. The hurdle is due to the self-attention mechanism that forces the decoding process to re-compute the attention at every generation step. Zhang et al. (2018a) addressed this drawback by proposing an Average Attention Network (AAN) that effectively replaces the self-attention computation in the Transformer decoder. In this AAN, two additional layers are added, including an average layer (to compute the cumulative attention up to a current generation step) and a gating layer (to control how much information should be preserved or filtered for generating a current target symbol). As a result, this AAN technique helps improve the decoding latency of the Transformer up to 4 times without the loss of translation performance (see also Gu et al. (2018), as reviewed in section §2.3.5).

Another drawback of the Transformer model is that increasing the number of layers will make the model training and decoding unstable. Often, the state-of-the-art NMT systems based on the Transformer model have 6-8 encoder and decoder layers (Bojar et al., 2018). Bapna et al. (2018) presented an interesting idea that facilitates the training of up to 2-3 times deeper Transformer models. The key idea here is on the use of transparent attention mechanism which applies residual connections (He et al., 2016b) with trainable weights along the encoder layers.

Further, Domhan (2018) presented a comprehensive analysis on the use of attention mechanism in various NMT architectures. They also introduced a so-called Architecture Definition Language (ADL) that allows practitioners to easily specify the configurations of NMT models suitable with their needs.
2.2 Training Neural Sequence Models

2.2.5.4 Addressing Exposure Bias via Structured Losses

The existing NMT models suffer from the exposure bias problem. During decoding, the NMT models are exposed to their own predictions of target sequences rather than the ground-truth sequences as in training, also regarded as “teacher forcing” strategy (Bengio et al., 2015; Goodfellow et al., 2016). Bengio et al. (2015) addressed this issue by using a scheduler to force the model to gradually use its own predictions at later stages of training. It means: the model still uses the ground truth inputs at the beginning of training process; but at a certain stage controlled by a scheduler, the model should be exposed with its own predictions. Further, Wiseman and Rush (2016) proposed a training technique that defines an additional end-to-end loss that exposes to the potential errors made by beam search decoding during training. Other techniques take the sequence level evaluations (e.g., with BLEU (Papineni et al., 2002)) into consideration and then use these evaluations as additional signals to reinforce the model training (Edunov et al., 2018b; Ranzato et al., 2015; Shen et al., 2016; Wu et al., 2018).

2.2.5.5 Semi-Supervised and Unsupervised Learning

Several attempts applied semi-supervised methods to enhance the robustness of the existing NMT models by leveraging the external monolingual data. Sennrich et al. (2016c) proposed a simple technique called “back translation” that allows the model to learn from additional monolingual data. The main idea is to first use a pre-trained reverse (target-to-source) NMT model to “back-translate” some monolingual data, and then use this new synthetic parallel data combined with the real parallel data to re-train the model. Other work also elaborated the back translation approach (Sennrich et al., 2016c), e.g., by focussing on the translations of difficult words only (Fadaee and Monz, 2018); or investigating the effect of decoding methods used for back-translations (Edunov et al., 2018a); or extending for low-resource settings (Fadaee et al., 2017). He et al. (2016a) further generalised the back translation approach under a principled dual learning framework based on a reinforcement learning with policy gradient. In this framework, views from reverse NMT model and additional language models (as dual models) are used to evaluate the corresponding back translations and form the reinforced training signals for the main NMT model (as primal model). Wang et al. (2019) further extended the dual learning framework with an exposure to more views of primal and dual models.

Recently, unsupervised learning has emerged as a new research direction in the context of NMT. The key idea is to completely remove the use of parallel data, instead uses the training signals extracted from monolingual data only. For instance, several attempts pro-
posed to leverage the signals from pre-trained unsupervised cross-lingual word embedding mappings (Artetxe et al., 2018) or from induced unsupervised bilingual dictionaries (Lample et al., 2018a,b) and the application of online iterative back translation to train an unsupervised NMT model. Overall, the unsupervised learning methods are very useful for practical situations of translating between languages without any parallel data, but inferior to models with available parallel corpora.

### 2.2.5.6 Multi-Modal Models

Several attempts have investigated the incorporation of multiple modalities into the existing NMT models. Zoph and Knight (2016) proposed the use of multiple source sequences (e.g., in different languages) translated into a target sequence in another language. The key technique here is to modify the attention mechanism that takes multiple source sequences into consideration. Other attempts, such as Calixto and Liu (2017a); Calixto et al. (2017); Delbrouck and Dupont (2017); Hewitt et al. (2018), proposed to leverage a modality other than text, e.g., an image which is associated with a given source sequence when translating into a target sequence. They leveraged spatial visual features over images, which were often pre-trained using convolutional neural networks (Vinyals et al., 2015b), and then added additional attention based on these features. Luong et al. (2016) proposed to jointly learn different tasks that leverage different sources of information such as machine translation and syntactic parsing. Further, side information has been exploited for improving NMT models (Jehl and Riezler, 2018; Michel and Neubig, 2018). Overall, the common key property of these approaches is on the effective use of representation of other modalities (e.g., visual features; or external knowledge such as side information) in order to enhance the attention mechanism.

### 2.2.5.7 Document-level Models

In practice, people usually want to translate a whole document instead of a single sentence. Obviously, the existing NMT models currently do not have the mechanism to make use of the document property. Different approaches have been proposed to make use of the document context in the existing NMT models, e.g., via hierarchical attention networks (Miculicich et al., 2018b); via memory networks (Maruf and Haffari, 2018); via a continuous cache (Tu et al., 2018). Further, Läuibli et al. (2018) showed an empirical analysis on evaluating the quality of translations (by the NMT models) in the scope of single sentence and entire document. They suggested a document-level evaluation measure needed for better measuring the improvements of any existing NMT models. Similar in spirit to this, it has been showed
that adding document contexts helps the NMT model learn coherence (Kuang et al., 2018), anaphora resolution (Voita et al., 2018) and discourse phenomena (Bawden et al., 2018).

2.3 Decoding Methods for Neural Sequence Models

In §2.1, we have presented the training part of neural sequence models. In this section, we aim to lay the ground of decoding methods for neural sequence models – which is a starting point for our proposed framework presented in Chapter §7.

Overall, the goal of decoding process is to find the target sequence \( y = y_1, \ldots, y_\ell \) that involves the following optimisation problem:

\[
y = \arg \max_{y_1, \ldots, y_\ell} p(y_1, \ldots, y_\ell | x).
\] (2.41)

Based on the above optimisation, there are different decoding strategies, including:

### 2.3.1 Random Sampling

A very simple decoding method is random sampling. The idea of random sampling is to pick the next word with its softmax probability distribution from the model output according to a random distribution. Its simple pseudo-code is presented in Algorithm 1.

One noticeable characteristic of random sampling is that its generated samples are diverse, often too diverse; and appear highly random. One trick to prevent this is to use a softmax temperature to control the randomness of the random sampling. We do this by changing the normal softmax in Algorithm 1, e.g., to softmax \((f(.)/\sigma)\) (where a temperature \(\sigma \in (0, 1]\), and \(\sigma = 1\) used for normal softmax).

In RNNLM, random sampling is often used for observing the outputs during training (technically, in this case the sequence \( x \) does not exist in Algorithm 1). In practice, it is rarely used for NMT decoding due to the randomness in the translation outputs. However, Edunov et al. (2018a) recently showed out that random sampling is very useful in creating diverse
Algorithm 2 Greedy Viterbi algorithm for NMT decoding

1: **Input:** a pre-trained model \( \Theta \); an input sequence in source language \( x \)
2: Set \( t = 0, y_t = <s> \)
3: while \( <s> \) symbol not reached for \( y_t \) do
4: \( t = t + 1 \).
5: Compute \( y_t = p_\Theta (y_t | x, y_{<t}) \).
6: Set \( y_t = \arg \max_{Wid=0}^{|V|-1} (y_t[Wid]) \)
7: return a generated sequence \( y \).

back-translations if used during decoding process of back translation strategy (Sennrich et al., 2016c).

### 2.3.2 Greedy Decoding

Another decoding method is called as greedy decoding. Greedy decoding is simple and very fast, and is the default method used in practice. Unlike random sampling, the idea of greedy decoding is to pick the most likely word at each decoding step — as presented in Algorithm 2.

In practice, greedy decoding is very fast, often generates sufficient translations. However, the main drawback of greedy decoding is that it is only able to generate one translation only. Also, it is impossible that the algorithm looks back if initial translation part already goes wrong. An usually-observed phenomenon of greedy decoding is that some parts of generated translations are unexpectedly repeated sometimes.

### 2.3.3 Beam Search

The most popular decoding method in practice is beam search. In principle, beam search decoding overcomes the limitation of greedy decoding by keeping track of possible word translations at each decoding step (the “beam”) and then moving forward by expanding from that beam. The overall pseudo-code of beam search algorithm is presented in Algorithm 3.

Note that if the beam size \( B \) is set to one, the beam search decoding becomes greedy decoding. Beam search decoding mostly produces significantly better in terms of translation quality than greedy one; however, there is a trade-off for decoding speed and usage memory. Depending on a pre-defined beam size \( B \) (often set to a standard value, e.g., 5-10), beam search decoding is often \( B \)-times slower than greedy one. In practice, with modern computing architectures, this trade-off is widely acceptable, and thus beam search decoding is the most widely used method for NMT decoding.
Algorithm 3 Beam search decoding algorithm for NMT decoding

1: **Input**: a pre-trained model \( \Theta \); an input sequence in source language \( x \); a beam size \( B \);
a maximum number of decoding steps \( T_{\text{max}} \).
2: Set \( t = 0 \) and list of completed hypotheses \( L_c \) empty.
3: Create initial hypothesis with only \(<s>\), add to beam of previous hypotheses \( L_b \).
4: **while** \( t < T_{\text{max}} \) **do**
  5: Set \( t = t + 1 \).
  6: **for** hypo\( _{\text{prev}} \) ∈ \( L_b \) **do**
  7: Compute \( y_t = p_{\Theta} (y_t | x, y_{<t} \in \text{hypo}_{\text{prev}}) \).
  8: Find top B likely word indices based on the above computed \( y_t \) as \( I_B \).
  9: **for** wid ∈ \( I_B \) **do**
  10: Create a new hypothesis hypo\( _{\text{new}} \) based on \( \text{wid} \) and hypo\( _{\text{prev}} \).
  11: **if** \( \text{wid} \) is an index of \(<s>\) **then**
  12: Update \( L_c \).
  13: **else**
  14: Update hypo\( _{\text{new}} \) to \( L_b \) accordingly.
  15: **if** \( L_b \) has more than \( B \) candidates **then**
  16: Keep only top \( B \) candidates according to their scores.
17: **return** a generated sequence \( y_{\text{best}} \).

2.3 Decoding Methods for Neural Sequence Models

2.3.3.1 Curse of Beam Search

Increasing the beam size in beam search algorithm can lead to worse translation results. Yang et al. (2018) showed that with increased beam size, the beam search algorithm has more possible candidates to explore its translation space. However, the beam search algorithm may select candidates emitting the end-of-sentence sentinel (i.e., \(<s>\)) early, meaning producing much shorter translations. This is related to the label bias problem in the NMT model where the probability of producing the \(<s>\) sentinel is overestimated (Murray and Chiang, 2018). This can be corrected by employing length normalisation schemes — which we discuss next.

2.3.3.2 Practical Techniques

In order to improve the stability and effectiveness of beam search, one may consider the following practical techniques, including:

**Length Normalisation.** In order to compare translation hypotheses with different lengths, a refinement is required for beam search decoding, e.g., length normalisation is applied to line 17 in Algorithm 3. Each translation hypothesis can be normalised based on a function
of its word length, and is formulated as:

\[
\text{score}_{\text{hypo}}^{\text{normalised}} = \frac{\text{score}_{\text{hypo}}}{\left(\alpha + \frac{\ell_{\text{hypo}}}{b}\right)\alpha},
\]

(2.42)

where: \(\ell_{\text{hypo}}\) is the word length of current hypothesis; \(\alpha\) is a softener hyper-parameter and its value is often in a range of 0 and 1. When \(\alpha = 0\), the decoding algorithm falls back to using a pure probability (without a length normalisation) in search. Note that the values of \(a\) and \(b\) can be chosen in different ways. If \(a = 0\) and \(b = 1\) and \(\alpha = 1\), then only the length is used, i.e.,

\[
\text{score}_{\text{hypo}}^{\text{normalised}} = \frac{\text{score}_{\text{hypo}}}{\ell_{\text{hypo}}}. \tag{2.43}
\]

Sennrich et al. (2017b) proposed to use \(a = 0\) and \(b = 1\) and \(\alpha\) to be fine-tuned, i.e.,

\[
\text{score}_{\text{hypo}}^{\text{normalised}} = \frac{\text{score}_{\text{hypo}}}{\ell_{\text{hypo}}^a}. \tag{2.44}
\]

In Google’s NMT system (Wu et al., 2016), the values are chosen as follows: \(a = 5\) and \(b = 6\) and \(\alpha\) to be fine-tuned, i.e.,

\[
\text{score}_{\text{hypo}}^{\text{normalised}} = \frac{\text{score}_{\text{hypo}}}{\left(5 + \frac{\ell_{\text{hypo}}}{6}\right)^{\alpha}}. \tag{2.45}
\]

In practice, this simple technique often leads to improved beam search algorithm.

### 2.3.4 Reranking/Rescoring

The drawback of beam search decoding is that it only takes left contexts into consideration, hence ignoring the right context due to its unavailability during run-time. Also, it is difficult to incorporate the global context (evaluating the target sequence as a whole). One possible solution to overcome this drawback is the reranking (or rescoring) method. The main idea of reranking is that instead of decoding with only one model, one may use various kinds of models that are complementary each other and use them in a combination manner (Neubig et al., 2015). The reranking pipeline method includes the following steps:

- **Step 0:** suppose we have \(N - 1\) additional pre-trained NMT models in addition to the convention NMT model.
- **Step 1:** use conventional NMT model with beam search decoding method to produce \(k\)-best list of translation candidates.
• Step 2: apply the other $N - 1$ models to get the scores for each candidate in the above $k$-best list.

• Step 3: use an additional development data set (a gold parallel data) to estimate the model weights for the above models, e.g., using MERT (Och, 2003) or MIRA (Cherry and Foster, 2012) training criteria.

• Step 4: re-rank the $k$-best list according to the weighted combination (from step 3) of all available scores, and return the candidate given its best combined score.

The additional pre-trained NMT models may include: right-to-left models, reverse (target-to-source) models. For example, most of the top systems in the WMT 2018 competition use right-to-left re-ranking methods (Bojar et al., 2018). Also, there are other ways, e.g., Shen et al. (2004) proposed to effectively learn a classifier to select candidates with higher BLEU (Papineni et al., 2002) outputs.

2.3.5 Overview of Recent Work

Recently, decoding for NMT models has attracted more attentions in the literature. Different models have been proposed to improve the weaknesses of current decoding methods, including (but not limited):

**Diverse Decoding.** Often, $k$-best lists of translation candidates produced by standard beam search decoding method are not very diverse, i.e., having only slightly different translations. Li and Jurafsky (2016b) proposed to diversify the $k$-best list of translation candidates by measuring the degree of diversity before feeding them into re-ranking. For example, the measure of diversity is based on distinct uni-grams and bi-grams given each translation candidate. Similarly, Vijayakumar et al. (2018) explored a novel diversity-augmented decoding objective for standard beam search. The augmented term is based on a dissimilarity measure for promoting the diversity among translation candidates in the $k$-best list. He et al. (2018) enhanced the translation diversity and quality by employing a mixture of translation models that are learned over specified training data, and the best translation will be chosen among that mixture during decoding — similar in spirit to ensemble method. Also, Schulz et al. (2018) added latent variables to the NMT model to account for lexical and syntactic variants in translations.

---

9http://matrix.statmt.org/matrix/systems_list/1881
Addressing Evaluation Metric Mismatch. Another obvious drawback of neural sequence models is the mismatch of evaluation metrics. Specifically, cross-entropy objective is evaluated during training; however, BLEU is used during decoding to evaluate the end quality of generated translations. Here, the possibility is that BLEU may not correlate consistently well with the cross-entropy objective, and therefore training must be stopped at a poor stage.

In order to address the above problem, He et al. (2017) added a trainable value network that predicts the long-term value of a given partial target translation according to an evaluation measure (BLEU). This predicted value will then be added to the total objective during decoding. The benefit of doing this way is to make the model more robust with an exposure to what it has to generate towards an end quality with BLEU.

Multi-Pass Decoding. Xia et al. (2017) proposed a two-pass decoding method called deliberation network where the first pass generates a draft version of translation, and then the second pass will attempt to polish that draft. This deliberation network includes one encoder with two separate decoders. The encoder and first decoder have the same architecture as in the standard attentional encoder-decoder models (Bahdanau et al., 2015). The difference is that the second decoder will take as its inputs both source representation as well as the output from the first-pass decoder. The advantage of this deliberation network is that taking the draft translation into consideration can provide additional useful contextual information. Akin to Xia et al. (2017), Zhang et al. (2018c) used the same two-pass decoding method but with a right-to-left decoder for the first-pass instead.

Also following this trend, Geng et al. (2018) further extended to an adaptive multi-pass decoding framework for decoding NMT models. The idea is to use an additional trainable policy network to determine suitable actions, e.g., given a current pass from a decoder, either continue with a pass to another decoder or halt at that point. The flexibility of using this policy network is that it can automatically decide the decoder depth (e.g., how many decoders are really needed) for generating a good translation.

Noise Injection Methods. Cho (2016) pointed out an interesting property of the existing deep NMT models that the hidden state space can be minimally perturbed without affecting its modelling power. Based on this property, this work proposed to inject small noise to the transition of the hidden states in the recurrent neural networks during decoding. Next, they run some independent processes of such noise-injected decoding steps in parallel; then pick the decoded sequence with the highest probabilities.

Gu et al. (2017) extended the work of Cho (2016) with a novel framework for transforming the greedy decoding algorithm trainable under an arbitrary objective function, in addition
to maximum likelihood estimation (MLE) training. The unstructured additive noise for manipulating the transition of RNNs (Cho, 2016) is replaced by a parametric function which is then approximated by a deterministic policy gradient algorithm. This parametric function is conditioned on the current hidden states of the neural networks in the NMT model. Further, Chen et al. (2018c) instead proposed to use a small deterministic neural network called an actor to be trained for manipulating the hidden states of the previously-trained decoder.

**Non-Autoregressive Methods.** Existing decoding approaches generate the translation output sequentially from a left to right direction, meaning that each target output word is dependent on prior decisions of generated words. This is regarded as *auto-regressive* property. This property prevents the existing decoding methods for NMT models from parallelisation. One particular example of the NMT model which is beneficial from parallelisation is the Transformer model (as discussed in §2.1.2.3) — of which training is an order of magnitude faster than the conventional RNN-based NMT models (as discussed in §2.1.2.1 and §2.1.2.2). However, its decoding process is still sequential.

Gu et al. (2018) extended the existing Transformer model (Vaswani et al., 2017) in an attempt to achieve the non-autoregressive property (so-called Non-Autoregressive Transformer or NAT for short) — which enables the decoding of output sequence in parallel towards an order of magnitude faster decoding speed. In the NAT model, a *fertility* prediction model for each source word is introduced on the top of the output of the last layer in the Transformer encoder stack and is represented by a set of latent variables. Note that fertility refers to a tendency of a word in a source language to be translated as a consistent number of words in the target language. Based on this fertility model, all the target words can be derived and predicted in parallel; hence, it can be regarded as a replacement for the factorised conditional distribution, e.g., $p(y_t|y_{<t}, x)$ now becomes $p(y_t|f, x)$ (where $f$ refers to the output of the fertility model). This NAT model exhibits a remarkable speed-up (up to 16 times faster); however, it suffers a degradation in accuracy compared to the non-autoregressive counterpart.

Lee et al. (2018) also investigated the non-autoregressive NMT problem based on the idea of “iterative refinement”. During training, the objective is to optimise a joint function of latent variable models (LVM) and denoising auto-encoders (DAE). The LVM aims to implicitly capture the conditional dependencies of target symbols, towards without autoregressive property; whereas the DAE serves as a reconstruction loss by first corrupting then resampling the output. During decoding, the process is as follows: given the input $x$, a

\[^{10}\text{Fertility refers to the ratio of how many foreign words translate a single source word (Brown et al., 1993).}\]
target length $\ell_y$ is predicted, then the generated output is iteratively refined. As a result, this proposed model leads to its high efficiency in decoding.

Also following this trend, Libovický and Helcl (2018) recently proposed an end-to-end architecture based on Connectionist Temporal Classification (CTC) scheme (Graves et al., 2006), allowing the decoder to generate all output words in parallel.

**Constrained Decoding.** Recent work has explored the possibilities of constrained decoding (or also called as guided decoding) for beam search. Hasler et al. (2018) proposed to perform external constraints (e.g., artificial terminologies) via finite state automaton to guide the decoding process to pay attention to such constraints. Hokamp and Liu (2017) extended the standard beam search by using a dynamic programming on a grid of hypotheses to allow the inclusion of arbitrary lexical constraints in the context of interactive MT and domain adaptation for MT. However, this approach can in turn increase the latency of beam search algorithm. Similar in spirit to Hokamp and Liu (2017), Post and Vilar (2018) explored a principled algorithm for constrained decoding, but with constant latency proportional to the number of integrated constraints. The main idea is to dynamically allocate the beam slots across the constraints at each decoding step; hence, allowing the incorporation of arbitrary constraints on the target side with only minor changes in the standard beam search algorithm. Also, Zhang et al. (2018b) proposed an approach which guides the NMT decoding process given external knowledge of similar translation pieces (e.g., words or phrases) retrieved from a parallel training data; and showed that it helps in translation tasks in which repetitiveness in the target sequences occurs frequently.

### 2.4 The Importance of Prior and External Knowledge

In this section, we discuss the important roles of prior and external knowledge in NLP, and also benefits for neural sequence models.

#### 2.4.1 Prior and External Knowledge for NLP

Given a task in NLP, we have a lot of *unlabeled* data and we would like to apply a supervised learning method. Our first option is to annotate the data sufficiently, e.g., by hiring linguists or skilled annotators.

Note that in order to obtain a productive annotation process, one requires prior human knowledge when annotating the unlabeled data, particularly in creating annotation guidelines. For example, in document classification, the prior knowledge is the label (or category)
distribution when a word token appears. In Part-of-Speech (POS) induction or tagging, people determine the POS tag sets based on human knowledge. Doing annotations, we know that most sentences should contain main verbs; and each word type should be assigned to a relatively small number of POS tags. In dependency parsing, words with noun type are usually dependent on ones with verb type. In word alignment of machine translation, alignment should refer to (mostly) one-to-one translational equivalence (so-called bijectivity property); or source to target and target to source alignments should be with an agreement (so-called symmetry property). Brown et al. (1993) encoded such prior knowledge in their IBM models — one of the earliest statistical models successfully applied for machine translation.

Other advanced approaches for data annotation are based on human computation frameworks (Wang et al., 2013), particularly Games with a Purpose (GWAP) or Amazon Mechanical Turk (AMT). The core idea is to utilise the crowds to do the annotation tasks. GWAP uses prior human knowledge to encode the annotations inside an entertaining game; whereas, AMT defines tasks with a specific goals and human knowledge is used in order to effectively validate the annotations from the crowds.

Overall, the annotation process always requires significant human efforts and usually large amounts of money. Also, it might not be scalable to every new task as well as domain of interests. This motivates researchers to encode prior knowledge for developing machine learning methods in many ways, e.g., via constraining feature functions (Bellare et al., 2009; Chang et al., 2007; Krupka and Tishby, 2007; Reichart and Barzilay, 2012; V. Graça et al., 2010); via priors or posteriors on model parameters (Ganchev et al., 2010); or via additional latent variables (Ganchev et al., 2009). In fact, if we have much annotated data, we may need less prior knowledge for learning models. However, prior knowledge is particularly useful for scenarios in which we have limited data.

2.4.2 Prior and External Knowledge for Neural Sequence Models

One of the most successful characteristics of deep learning models is their ability to learn without feature engineering. In principle, what we need is to select network architecture, set up training configurations; and let deep learning models learn by themselves. Deep learning models often learn powerful abstractions and features on their own, automatically from the data. However, this often comes with a requirement of massive amount of annotated data — which is not always available across tasks and domain of interests. Using prior and external knowledge can be a viable solution, potentially compensating for a lack of annotated data. A research question is that how prior and external knowledge can be used for improving deep learning models. Recent work has explored this line of research, by incorporating prior knowledge via different kinds of constraints into deep learning models. Here, our focus is
on the applications for neural sequence models, particularly in NMT. Prior knowledge can be encoded via different kinds of constraints, including word, phrase, syntactic, or semantics constraints.

**Word and Phrase Constraints.** Standard architectures for NMT are based on encoder-decoder models which do not constrain on prior knowledge of word or phrases in the source and target sequences. Several work attempted to incorporate such knowledge into NMT models. For instance, Arthur et al. (2016) and Nguyen and Chiang (2018) proposed different mechanisms to incorporate discrete translation lexicons obtained externally from a statistical translation model into NMT models (Bahdanau et al., 2015), leading to improved translation quality in the contexts of translating resource-poor languages. Zhang et al. (2017) adopted the idea of posterior regularisation (Ganchev et al., 2010) in the context of NMT, and proposed to incorporate simple prior knowledge, such as source and target length ratio, statistical bilingual dictionary, and coverage constraints. Also, Ugawa et al. (2018) leveraged named entity information in the source sequence as an external knowledge; and Wang et al. (2018a,b) addressed the pronoun-dropping phenomenon\(^\text{11}\) for translating conversational domain. The commonality of these approaches is that they require certain modification on the use of attention mechanism or the objective function to signal the presence of prior or external knowledge.

**Syntactic and Semantics Constraints.** Other advanced constraints such as syntactic and semantics have also been taken into considerations. In order to encourage more linguistic structures in the attention mechanism, Cohn et al. (2016) proposed to make use of structural alignment biases inspired from conventional statistical alignment models, such as word alignment models inspired from IBM alignment models 1, 2 (Brown et al., 1993), including positional bias inspired from IBM alignment model 2 and (Dyer et al., 2013); Markov conditioning inspired from HMM alignment model (Vogel et al., 1996); word fertility inspired from IBM alignment models 3; and joint bidirectional alignment agreement training inspired from symmetry heuristics in the phrase-based SMT model (Koehn, 2010; Koehn et al., 2007). Note that I significantly contributed to this work (Cohn et al., 2016) although this paper is not included in the thesis.

Several work attempted to incorporate syntactic information into existing NMT models, such as syntactic tree-based NMT (Aharoni and Goldberg, 2017; Currey and Heafield, 2018; Eriguchi et al., 2016); dependency-based NMT (Chen et al., 2017b; Hashimoto and

\(^{11}\)https://en.wikipedia.org/wiki/Pro-drop_language
Tsuruoka, 2017; Wu et al., 2017); and forest-based NMT (Ma et al., 2018; Zaremoodi and Haffari, 2018a).

Instead of directly incorporating linguistics knowledge into existing NMT models, other work proposed the use of multi-task learning (MTL) mechanism (Collobert and Weston, 2008; Ruder, 2017) that jointly learn the NMT task together with other linguistics tasks; for instance: NMT with POS tagging (Niehues and Cho, 2017) and with syntactic parsing (Kiperwasser and Ballesteros, 2018; Luong et al., 2016); or NMT with other NLP tasks including named entity recognition, syntactic parsing, semantic parsing (Zaremoodi et al., 2018; Zaremoodi and Haffari, 2018b). In principle, the successes of all these approaches are motivated by the intelligent selection of multiple complementary tasks, and the degree of parameter sharing.

Discussion. As discussed above, prior pre-neural and neural methods have been beneficial learning from prior and external knowledge. In this thesis, we further extend this line of research by investigating unexplored prior and external knowledge that potentially can be beneficial for improving neural sequence models, including: side and meta information, linguistics knowledge, monolingual data; and applied to both training and decoding phases.

2.5 Summary

This chapter presented background materials in details relevant to this PhD thesis. We first presented the research evolution of neural sequence models, including the training and decoding processes, the evaluation, as well as the optimisation. Then, we discussed the importance of prior and external knowledge in the context of NLP in general and of neural sequence models in particular. This thesis aims to extend this line of research, by further exploring various situations in which prior and external knowledge are useful for neural sequence models, applied to both training and decoding processes.
Part 1

Training and Modelling
Chapter 3

Neural Sequence Models with Side Information

Contents

3.1 Learning Neural Language Modelling with Side Information . . . . . . . . . 66
  3.1.1 Recurrent Neural Network Language Model Architecture . . . . . . 67
  3.1.2 RNNLM with Side Information . . . . . . . . . . . . . . . . . . . . . . . . . 67
  3.1.3 Experiments . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 70
  3.1.4 Results and Analysis . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 74
  3.1.5 Discussion . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 76
3.2 Learning Neural Machine Translation with Side Information . . . . . . . . 77
  3.2.1 Motivation . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 77
  3.2.2 Machine Translation Data with Side Information . . . . . . . . . . 78
  3.2.3 NMT with Side Information . . . . . . . . . . . . . . . . . . . . . . . . . . 80
  3.2.4 Experiments . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 85
  3.2.5 Connection to Related Work . . . . . . . . . . . . . . . . . . . . . . . . . . 90
  3.2.6 Discussion . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 91
3.3 Summary . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 91

In this chapter, we aim at exploring the usefulness of external knowledge such as metadata or side information that can be leveraged to improving the neural sequence models. We will focus on two case studies in neural sequence models, including sequence modelling with neural language modelling (in §3.1) and sequence generation with neural machine translation (in §3.2). We study various kinds of side information, and then propose the appropriate
Neural Sequence Models with Side Information

modifications for neural network architectures to take such information into consideration. Our extensive experiments show that side information indeed helps boost the performance of neural sequence models across several datasets and benchmarks.

Parts of this chapter were published in:


3.1 Learning Neural Language Modelling with Side Information

Neural network approaches to language modelling (LM) have made remarkable performance gains over traditional count-based $n$-gram LMs as described in §2.1.1. They offer several desirable characteristics, including the capacity to generalise over large vocabularies through the use of a vector space representation, and — for recurrent models (Mikolov et al., 2011) — the ability to encode long distance dependencies that are impossible to include with limited contexts used in conventional $n$-gram LMs. These early papers have spawned a cottage industry in neural LM based applications, where text generation is a key component, including conditional language models for image captioning (Kiros et al., 2014; Vinyals et al., 2015b) and neural machine translation (Bahdanau et al., 2015; Kalchbrenner and Blunsom, 2013; Sutskever et al., 2014).

Inspired by these works for conditioning LMs on complex side information, such as images and foreign text, in this paper we investigate the possibility of improving LMs in a more traditional setting, that is when applied directly to text documents. Typically corpora include rich side or meta information, such as document titles, authorship, time stamp, keywords and so on, although this information is usually discarded when applying statistical models. However, this information can be highly informative, for instance, keywords, titles or descriptions, often include central topics which will be helpful in modelling or understanding the document text. We propose mechanisms for encoding this side information into
3.1 Learning Neural Language Modelling with Side Information

a vector space representation, and means of incorporating it into the generating process in a RNNLM framework. Evaluating on two corpora and two different languages, we show consistent perplexity reductions over the state-of-the-art RNNLM models.

3.1.1 Recurrent Neural Network Language Model Architecture

First, we briefly review the RNNLM architecture (Mikolov et al., 2011) before describing our extension in §3.1.2. The details of the RNNLM architecture have been presented in §2.1.1.

Typically, the standard RNNLM consists of three main layers: an input layer where each input word has its embedding via one-hot vector coding; a hidden layer consisting of recurrent units where a state is conditioned recursively on past states; and an output layer where a target word will be predicted. RNNLM has an advantage over conventional n-gram language model in modelling long distance dependencies effectively.

In general, an RNN operates from left-to-right over the input word sequence; i.e.,

\[
h_t = RU (x_t, h_{t-1}) = f (W^{(hh)} h_{t-1} + W^{(ih)} x_t + b^{(h)})
\]

\[
x_{t+1} \sim \text{softmax} (W^{(ho)} h_t + b^{(o)}) ;
\]

where \( f(.) \) in Equation 3.1b is a non-linear function, e.g., tanh (see Table 2.2), applied element-wise to its vector input; \( h_t, x_t \in \mathbb{R}^{H_h} \) and \( h_t, x_t \in \mathbb{R}^{H_i} \) are the current RNN hidden state and input word embedding representation at time-step \( t \); and matrices \( W^{(hh)} \in \mathbb{R}^{H_h \times H_h}, W^{(ih)} \in \mathbb{R}^{H_h \times H_i} \) and vectors \( b^{(h)} \in \mathbb{R}^{H_h}, b^{(o)} \in \mathbb{R}^{|V|} \) are learnable model parameters; \( H_h \) and \( H_i \) are hidden and input dimensions, respectively. Note that we observed that using the same \( H_h \) and \( H_i \) values yields no difference in perplexities on the development dataset. Also, a recurrent unit (RU) in Equation 3.1a can be adopted from LSTM, GRU, or deeper structures, e.g. DGLSTM as we have discussed earlier in §2.1.1. The model is trained using gradient-based methods to optimise a (regularised) training objective, e.g. the likelihood function of predicting individual words in the word sequence.

3.1.2 RNNLM with Side Information

Many corpora are archived with side information or contextual meta-data; however, they have often ignored or rarely used in most of statistical machine learning models. In this chapter, we investigate that such information can be useful for language modelling (and presumably for other downstream NLP tasks, see §3.2). By providing this auxiliary information directly to the RNNLM, we stand to boost language modelling performance.
The first question in using side information is how to encode these unstructured inputs, \( \{x'\} \), into a vector representation, denoted \( e \in \mathbb{R}^{H_a} \) where \( H_a \) is dimension of a vector representation of auxiliary inputs. Without the loss of generality, suppose we have a single unstructured input \( x' \) with token length \( l_s \). We discuss several possible methods for encoding the vector representation of this auxiliary input:

**BOW**

additive bag of words, \( e = \sum_{t=1}^{l_s} x'_t \), and

**average**

the average embedding vector, \( e = \frac{1}{l_s} \sum_{t=1}^{l_s} x'_t \), both inspired by Hermann and Blunsom (2014);

**bigram**

convolution with sum-pooling, \( e = \sum_{t=1}^{l_s} \tanh (x'_{t-1} + x'_t) \) inspired by Hermann and Blunsom (2014); and

**RNN**

a recurrent neural network over the word sequence \( x' \) as proposed by (Sutskever et al., 2014), using the final hidden state(s) as \( e \).

From the above methods, **BOW** and **average** methods have sheer advantage since it is applicable for any unstructured inputs whereas other methods like **bigram**, **RNN** require sequential inputs. For that reason, we argue that **BOW** and **average** methods are more flexible in many practical contexts. In our experiment, we will show that **BOW** worked consistently well, outperforming the other methods, and moreover lead to a simpler model with faster and more stable training.

Note that when using multiple auxiliary inputs, we use a weighted combination, formulated as \( e = \sum_{i=1}^{M} W^{(ai)} e^{(i)} \), where \( W^{(ai)} \in \mathbb{R}^{H_x \times H_a} \) and \( M \) is a number of auxiliary inputs. Note that \( e \) must have the same vector dimension with the input word embedding representation \( x \).

The next step is the integration of \( e \) into the RNNLM. We consider two integration methods: as input to the hidden state (denoted **input**), and connected to the output softmax layer (**output**), as shown in Figure 3.1 a and b, respectively. In both cases, we compare experimentally the following integration strategies:

**add**

adding the vectors together, e.g., using \( x_i + e \) as the input to the RNN, such that \( h_i = RU (x_i + e, h_{i-1}) \);
3.1 Learning Neural Language Modelling with Side Information

Fig. 3.1 Integration methods for auxiliary information, $e$: a) as input to the RNN, or b) as part of the output softmax layer.

**stack**

concatenating the vectors, e.g., using $[x_t^T e^T]^T$ for generating the RNN hidden state, such that $h_t = RU \left( \begin{bmatrix} x_t \\ e \end{bmatrix}, h_{t-1} \right)$; and

**mlp**

feeding both vectors into an extra perceptron with single hidden layer, using a tanh non-linearity and projecting the output to the required dimensionality; i.e.,

$$h'_t = \tanh \left( W^{(hh')} h_t + W^{(he)} e + b^{(h')} \right) \quad (3.2a)$$

$$x_{t+1} \sim \text{softmax} \left( W^{(ho)} h'_t + b^{(o)} \right) ; \quad (3.2b)$$

where additional matrices $W^{(hh')} \in \mathbb{R}^{H_h \times H_h}$, $W^{(he)} \in \mathbb{R}^{H_h \times H_a}$ and bias vector $b^{(h')} \in \mathbb{R}^{H_h}$ in non-linear function, tanh, are model parameters.

Note that add requires the vectors to be the same dimensionality, while the other two methods do not. The stack method is costly, given that it increases the size of several matrices, either in the recurrent unit (for input) or the output mapping for word generation. This is a problem in the latter case: given the large size of the vocabulary, the matrix $W^{(ho)}$ is already very large this doubles in size and we observed a sizeable effect on training time (and also an increased propensity to over-fit). The output+stack method does however have a compelling interpretation as a jointly trained product model between a RNNLM and a
Table 3.1 Statistics of the training sets, showing in each cell the number of word tokens, types, documents (talks or plenaries), and sentences; K: in thousands; M: in millions; “#tokens”: number of word tokens. Here, “#types” refers to word frequency thresholded at 5 and 15 for TED Talks and RIE datasets, respectively; “#types”: number of word types; “#docs”: number of talks and plenary sessions in TED Talks and RIE datasets, respectively; “#sents”: number of sentences.

<table>
<thead>
<tr>
<th>Dataset-Language</th>
<th>tokens (M)</th>
<th>types (K)</th>
<th>docs</th>
<th>sents (K)</th>
</tr>
</thead>
<tbody>
<tr>
<td>TED-en</td>
<td>4.0</td>
<td>18.3</td>
<td>1414</td>
<td>179</td>
</tr>
<tr>
<td>TED-fr</td>
<td>4.3</td>
<td>22.6</td>
<td>1414</td>
<td>179</td>
</tr>
<tr>
<td>RIE-en</td>
<td>13.7</td>
<td>15.0</td>
<td>200</td>
<td>460</td>
</tr>
<tr>
<td>RIE-fr</td>
<td>14.9</td>
<td>19.4</td>
<td>200</td>
<td>460</td>
</tr>
</tbody>
</table>

unigram model conditioned on the side information, where both models are formulated as softmax classifiers. Considered as a product model (Hinton, 2002; Pascanu et al., 2013), the two components can concentrate on different aspects of the problem where the other model is not confident, and allowed each model the ability to ‘veto’ certain outputs, by assigning them a low probability.

3.1.3 Experiments

In order to validate our hypothesis about the benefits of side information, we conducted our experiments on two datasets with different genres in two languages. As the first dataset, we use the IWSLT2014 data collection on TED Talks due to its self-contained rich auxiliary information, including: title, description, keywords, and author related information. We chose two languages including English and French for our experiments. Figure 3.2 shows an example of TED Talks side information including the title, description, and keywords. The statistics of the training set is shown in Table 3.1. We used dev2010 (7 talks/817 sentences) for early stopping of training neural network models. For evaluation, we used different testing sets over years, including tst2010 (10 talks/1587 sentences), tst2011 (7 talks/768 sentences), tst2012 (10 talks/1083 sentences).

As the second dataset, we crawled the debate transcripts from European Parliament website, focusing on plenary sessions. Such sessions contain useful structural information with multilingual texts divided into speaker and committee sessions and topics. Each speaker or committee member aims at talking about a specific topic (or so-called topic headline). Fig-

---

2Our method can be easily applied to other languages as well.
3http://www.europarl.europa.eu/
Fig. 3.2 Example of a TED talk with side information (title, description, keywords, speaker info) in addition to the main transcripts, in the TED Talks dataset.
Fig. 3.3 A snapshot of a plenary session with side information (e.g., topic headline), in the RIE dataset.
3.1 Learning Neural Language Modelling with Side Information

<table>
<thead>
<tr>
<th>Method</th>
<th>test2010</th>
<th>test2011</th>
<th>test2012</th>
</tr>
</thead>
<tbody>
<tr>
<td>5-gram LM</td>
<td>79.9</td>
<td>77.4</td>
<td>89.9</td>
</tr>
<tr>
<td>RNNLM</td>
<td>65.8</td>
<td>63.9</td>
<td>73.0</td>
</tr>
<tr>
<td>LSTM</td>
<td>54.1</td>
<td>52.2</td>
<td>58.4*</td>
</tr>
<tr>
<td>DGLSTM</td>
<td>53.1*</td>
<td>52.1*</td>
<td>58.8</td>
</tr>
</tbody>
</table>

Table 3.2 Perplexity scores based on the English part of the TED talks dataset, evaluating different RNNLM architectures. **bold**: “Statistically significant better than the 5-gram LM baseline”; ♠: “best result in the dataset”.

<table>
<thead>
<tr>
<th>Method</th>
<th>test2010</th>
<th>test2011</th>
<th>test2012</th>
</tr>
</thead>
<tbody>
<tr>
<td>DGLSTM</td>
<td>53.1</td>
<td>52.1</td>
<td>58.8</td>
</tr>
<tr>
<td>output+add+t w/ BOW encoding</td>
<td>52.3</td>
<td>53.5</td>
<td>58.3</td>
</tr>
<tr>
<td>output+add+t w/ average encoding</td>
<td>52.6</td>
<td>53.4</td>
<td>58.6</td>
</tr>
<tr>
<td>output+add+t w/ bigram encoding</td>
<td>101.3</td>
<td>103.5</td>
<td>120.8</td>
</tr>
<tr>
<td>output+add+t w/ RNN encoding</td>
<td>80.8</td>
<td>88.7</td>
<td>95.4</td>
</tr>
</tbody>
</table>

Table 3.3 Perplexity scores based on the English part of the TED talks dataset, evaluating different encoding methods. +t: with “keywords” side information; **bold**: ‘ better than the other encoding methods”

Figure 3.3 shows an example of plenary session with side information as topic headline. Such information has not yet been utilised before. We believe that those kinds of information are interesting and beneficial for language modelling tasks. Our dataset contains 724 plenary sessions over 12.5 years until June 2011 with multilingual texts in 22 languages. This is a mix of transcripts and translations depending on languages of speakers. Note that we decided to ignore the period from June 2011 onwards for consistency purpose, since that the EU Parliament website has stopped creating manual human translations but using automatic machine translation systems instead. We refer to this dataset by RIE⁴ (Rich Information Europarl). We randomly selected 200/5/30 plenary sessions as the training/development/test sets, respectively. We believe that this new data including side information is suitable for the proposed task. Furthermore, the sizes of our working datasets are an order of magnitude larger than the standard Penn Treebank set which is often used for evaluating neural language models.

⁴Due to its copyright, we are only able to release our source code which can be used to re-crawl this dataset. The code is available at https://github.com/duyvuleo/RIE.
3.1.3.1 Set-up and Baselines

We have used DyNet\(^5\) to implement our recurrent neural network language models. We use the same configurations for all neural models: 512 input embedding and hidden layer dimensions, 2 hidden layers, and vocabulary sizes after standard tokenisation with Moses scripts\(^6\), as given in Table 3.1. Note that we applied the same vocabulary for the auxiliary side information and modelled main text. We trained a conventional 5-gram language model using modified Kneser-Ney smoothing, with the KenLM toolkit Heafield (2011). We used the Wilcoxon signed-rank test (Wilcoxon, 1945) to measure the statistical significance (\(p < 0.05\)) on differences between sentence-level perplexity scores of improved models compared to the best baseline. We used sentence markers including \(\langle s \rangle\), \(\langle /s \rangle\), and \(\langle unk \rangle\) to represent for beginning, ending and out-of-vocabulary words. Throughout our experiments, punctuation and stop words are filtered out in all auxiliary inputs. If words in the auxiliary inputs do not exist in the vocabulary, they will be ignored. We observed that this filtering was required for BOW and average encoding methods to work reasonably well. For each model, the best perplexity score on development set is used for early stopping of training models, which was obtained after 2-5 and 2-3 epochs on TED Talks and RIE datasets, respectively.

3.1.4 Results and Analysis

We performed the first experiments to evaluate the performances of different RNNLM architectures — which can be found in Table 3.2. We observed that all RNNLM variants consistently achieve substantially better perplexities compared to the conventional 5-gram language model baseline. For a fair comparison, when computing the perplexity with the 5-gram LM, we excluded all test words marked as \(\langle unk \rangle\) (i.e., with low counts or OOVs) from consideration. Of the basic RNNLM models (middle), the DGLSTM works consistently better than both the standard RNN and the LSTM. Possibly, this might be due to better interactions of memory cells in hidden layers. This concurs with the finding in Yao et al. (2015), who showed that DGLSTM produced the state-of-the-art results over Penn Treebank dataset. Since the DGLSTM outperformed others, we used it for all subsequent experiments.

Next, we would like to evaluate the behaviors of different encoding methods as discussed earlier. We chose “title” side information and the output+add+t incorporation method as an anchor for evaluation. Note that the encoding methods such as bigram and RNN require sequential inputs, hence “keywords” side information is not applicable. Also, the more complex side information such “description” requires hierarchical representation, e.g., as in Li

\(^5\)https://github.com/clab/dynet/
\(^6\)https://github.com/moses-smt/mosesdecoder/tree/master/scripts
3.1 Learning Neural Language Modelling with Side Information

<table>
<thead>
<tr>
<th>Method</th>
<th>test2010</th>
<th>test2011</th>
<th>test2012</th>
</tr>
</thead>
<tbody>
<tr>
<td>DGLSTM</td>
<td>53.1</td>
<td>52.1</td>
<td>58.8</td>
</tr>
<tr>
<td>input+add+k</td>
<td>52.9</td>
<td>52.1</td>
<td>57.5</td>
</tr>
<tr>
<td>input+mlp+k</td>
<td>53.3</td>
<td>51.5</td>
<td>57.3</td>
</tr>
<tr>
<td>input+stack+k</td>
<td>53.7</td>
<td>51.9</td>
<td>58.1</td>
</tr>
<tr>
<td>output+mlp+k</td>
<td>51.7</td>
<td>50.6</td>
<td>55.8</td>
</tr>
<tr>
<td>output+mlp+t</td>
<td>52.3</td>
<td>53.5</td>
<td>58.3</td>
</tr>
<tr>
<td>output+mlp+d</td>
<td>52.0</td>
<td>49.8</td>
<td>56.3</td>
</tr>
<tr>
<td>output+mlp+k+t</td>
<td>51.4</td>
<td>51.1</td>
<td>56.8</td>
</tr>
<tr>
<td>output+mlp+k+d</td>
<td>51.2</td>
<td>49.7*</td>
<td>55.1*</td>
</tr>
<tr>
<td>output+mlp+t+d</td>
<td>52.6</td>
<td>51.5</td>
<td>57.2</td>
</tr>
<tr>
<td>output+mlp+k+t+d</td>
<td>51.1*</td>
<td>50.6</td>
<td>56.3</td>
</tr>
</tbody>
</table>

Table 3.4 Perplexity scores based on the English part of the TED talks dataset. +k, +t, +d: with keywords, title, and description as auxiliary side information respectively. **Bold**: “Statistically significant better than the DGLSTM baseline”; ♠: “best result in the dataset”.

et al. (2015a), to represent the inter-sentence dependencies — which is out of scope of our current study. As expected, the BOW and average works reasonably well, outperforming the others, although there is no significant difference between the two, as shown in Table 3.3. Also, though much simpler, using both BOW and average methods lead to lesser additional model parameters and much faster in terms of the convergence. For simplicity, we will report only results for the BOW encoding in subsequent experiments.

We continued our experiments on evaluating the incorporation of the side information into RNNLM models. The perplexity results on TED Talks dataset are presented in Table 3.4 and 3.5. For TED Talks dataset, there are three kinds of side information, including keywords, title, description. We attempted to inject those into different RNNLM layers, resulting in model variants as shown in Table 3.4. First, we chose “keywords” (+k) information as an anchor to figure out which incorporation method works well; hence, others such as “title” (+t) and “description” (+d) were not used though. Comparing input+add+k, input+mlp+k and input+stack+k, the largest decrease is obtained by input+mlp+k consistently across all test sets (as well as development sets observed during training, not shown here). Possibly, incorporating the side formation into the output layer is more effective since it has a more direct affect to the word prediction. We further evaluated the addition of other side information (e.g., “description” (+d), “title” (+t)), finding that +d has similar effect as +k whereas +t has a mixed effect, being detrimental for one test set (test2011). We suspect that it is due to short sentences of titles resulted after our filtering step, lead to a shortage of useful information fed into neural network learning. Interestingly, the best performance
Table 3.5 Perplexity scores based on the French part of TED talks dataset in IWSLT14 MT. Note that +k means with keywords in English. **bold**: “Statistically significant better than the best baseline”; ♠: “best result in the dataset”.

<table>
<thead>
<tr>
<th>Method</th>
<th>test2010</th>
<th>test2011</th>
<th>test2012</th>
</tr>
</thead>
<tbody>
<tr>
<td>5-gram LM</td>
<td>65.1</td>
<td>60.3</td>
<td>64.8</td>
</tr>
<tr>
<td>RNNLM</td>
<td>53.6</td>
<td>49.7</td>
<td>53.4</td>
</tr>
<tr>
<td>LSTM</td>
<td>45.0</td>
<td>42.5</td>
<td>44.0</td>
</tr>
<tr>
<td>DGLSTM</td>
<td>44.0</td>
<td>41.9</td>
<td>43.0</td>
</tr>
<tr>
<td>output+mlp+t</td>
<td>42.1</td>
<td>40.6</td>
<td>42.5</td>
</tr>
<tr>
<td>output+mlp+d</td>
<td>40.9</td>
<td>38.9</td>
<td>40.3</td>
</tr>
<tr>
<td>output+mlp+t+d</td>
<td>41.7</td>
<td>39.8</td>
<td>42.8</td>
</tr>
<tr>
<td>output+mlp+k</td>
<td>40.8</td>
<td>38.3♠</td>
<td>39.7</td>
</tr>
<tr>
<td>output+mlp+d+k</td>
<td>40.2♠</td>
<td>38.3♠</td>
<td>39.4♠</td>
</tr>
</tbody>
</table>

is obtained when incorporating both +k and +d, showing that there is complementary information in the two auxiliary inputs. Overall, side information such as “keywords” and “descriptions” are slightly more useful than “title”, standing to an obvious reason — title is for attentive grabbing, but not for showing the detailed information.

Further, we also achieved the similar results in the counterpart of English part (in French language) using output+mlp with both +t and +d as shown in Table 3.5. Note that input incorporation method does not work well in the English part, we did not proceed with the experiments for French data. Remarkably, in the French data, the “keywords” side information is unavailable. For this reason, we run additional experiments by injecting English keywords as side information into neural models of French. Interestingly, we found that “keywords” side information in English effectively improves the modelling of French texts as shown in Table 3.5, serving as a new form of cross-lingual language modelling.

We further achieved similar results by incorporating the topic headline in the RIE dataset. We obtained the consistently-improved results (see Table 3.6), demonstrating the robustness of the proposed output+mlp approach.

### 3.1.5 Discussion

We have proposed an effective approach to boost the performance of RNNLM using auxiliary side information (e.g. keywords, title, description, topic headline) of a textual utterance. We provided an empirical analysis of various ways of injecting such information into a distributed representation, which is then incorporated into either the input, hidden, or output layer of RNNLM architecture. Our experiments show consistent improvements of RNNLMS
3.2 Learning Neural Machine Translation with Side Information

<table>
<thead>
<tr>
<th>Method</th>
<th>test (en)</th>
<th>test (fr)</th>
</tr>
</thead>
<tbody>
<tr>
<td>5-gram LM</td>
<td>55.7</td>
<td>38.5</td>
</tr>
<tr>
<td>RNNLM</td>
<td>45.8</td>
<td>31.9</td>
</tr>
<tr>
<td>LSTM</td>
<td>40.3</td>
<td>28.5</td>
</tr>
<tr>
<td>DGLSTM</td>
<td>36.4</td>
<td>25.4</td>
</tr>
<tr>
<td>output+mlp+h</td>
<td><strong>33.3</strong></td>
<td><strong>24.0</strong></td>
</tr>
</tbody>
</table>

Table 3.6 Perplexity scores based on the sampled RIE dataset. +h: topic headline. **bold**: “Statistically significant better than the best baseline”; •: “best result in the dataset”.

Using side information over the baselines for two different datasets and genres in two languages. Interestingly, we found that side information in a foreign language can be highly beneficial in modelling texts in another language, serving as a form of cross-lingual language modelling.

### 3.2 Learning Neural Machine Translation with Side Information

As shown in §3.1, the use of side information helps boost the performance of recurrent neural language models in an unconditional case, now we further investigate whether such information is still helpful in the case of conditional language models, in the downstream task of neural machine translation.

#### 3.2.1 Motivation

Neural machine translation is the task of generating a target language sequence given a source language sequence, framed as a neural network (Bahdanau et al., 2015; Sutskever et al., 2014, *inter alia*). Most research efforts focus on inducing more prior knowledge (Cohn et al., 2016; Mi et al., 2016a; Zhang et al., 2017, *inter alia*), incorporating linguistics factors (García-Martínez et al., 2017; Sennrich and Haddow, 2016) or changing the network architecture (Elbayad et al., 2018; Gehring et al., 2017b,c; Vaswani et al., 2017) in order to better exploit the source representation. Consider a different direction, situations in which there exists a modality other than a text of a source sentence. For instance, the WMT 2017 campaign\(^7\) proposed to use additional information obtained from *images* to enrich the neural MT models, as in Calixto and Liu (2017b); Calixto et al. (2017); Matusov et al. (2017). This task, also

\(^7\)[http://www.statmt.org/wmt17/multimodal-task.html]
known as multi-modal translation, seeks to leverage images which can contain cues representing the perception of the image in source text, and potentially can contribute to resolve ambiguity (e.g., lexical, gender), vagueness, out-of-vocabulary terms, and topic relevancy.

Inspired from the idea of multi-modal translation, in our work, we propose the use of another modality, namely metadata or side information. As presented earlier in §3.1, we have shown the usefulness of side information for neural language models. This work will investigate the potential usefulness of side information for NMT models. In our work, we target towards unstructured and heterogeneous side information which potentially can be found in practical applications. Specifically, we investigate different kinds of side information, including topic keywords, personality information and topic classification. Then we study different methods with minimal efforts for incorporating such side information into existing NMT models.

### 3.2.2 Machine Translation Data with Side Information

First, let’s explore some realistic scenarios in which the side information is potentially useful for NMT.

**TED Talks** The TED Talks website\(^8\) hosts technical videos from influential speakers around the world on various topics or domains, such as: education, business, science, technology, creativity, etc. Thanks to users’ contributions, most of such videos are subtitled in multiple languages. Based on this website, Cettolo et al. (2012) created a parallel corpus for the MT

---

\(^8\)[https://www.ted.com/talks]
3.2 Learning Neural Machine Translation with Side Information

**English Sentence:** Accordingly, I consider it essential that both the identification of cattle and the labelling of beef be introduced as quickly as possible on a compulsory basis.

**German Sentence:** Entsprechend halte ich es auch für notwendig, daß die Kennzeichnung möglichst schnell und verpflichtend eingeführt wird, und zwar für Rinder und für Rindfleisch.

**Meta Info:** EUROID="2209" NAME="Schierhuber" LANGUAGE="DE" GENDER="FEMALE" DATE_OF_BIRTH="31 May 1946" SESSION_DATE="97-02-19" AGE="50"

**English Sentence:** Can the Commission say that it will seek to have sugar declared a sensitive product?

**German Sentence:** Kann die Kommission sagen, dass sie danach streben wird, Zucker zu einem sensiblen Produkt erklären zu lassen?

**Meta Info:** EUROID="22861" NAME="Ó Neachtain (UEN)." LANGUAGE="EN" GENDER="MALE" DATE_OF_BIRTH="22 May 1947" SESSION_DATE="03-09-02" AGE="56"

**English Sentence:** For example, Brazil has huge concerns about the proposals because the poor and landless there will suffer if sugar production expands massively, as is predicted.

**German Sentence:** So hegt beispielsweise Brasilien bezüglich der Vorschläge enorme Bedenken, denn wenn die Zuckerproduktion, wie vorhergesagt, massiv expandiert, wird das die Not der Armen und Landlosen dort noch verstärken.

**Meta Info:** EUROID="28115" NAME="McGuinness (PPE-DE )." LANGUAGE="EN" GENDER="FEMALE" DATE_OF_BIRTH="13 June 1959" SESSION_DATE="05-02-22" AGE="45"

**English Sentence:** The European citizens’ initiative should be seen as an opportunity to involve people more closely in the EU’s decision-making process.

**German Sentence:** Die Europäische Bürgerinitiative ist als Chance zu werten, um die Menschen stärker in den Entscheidungsprozess der EU miteinzubeziehen.

**Meta Info:** EUROID="96766" NAME="Ernst Strasser" LANGUAGE="DE" GENDER="MALE" DATE_OF_BIRTH="29 April 1956" SESSION_DATE="10-12-15-010" AGE="54"

Fig. 3.5 An example with side information (e.g., author’s gender highlighted in red) for MT with personalised Europarl dataset.

Research community. Inspired by this, Chen et al. (2016) further customised this dataset and included an additional sentence-level topic information. We consider such topic information as side information. Figure 3.4 illustrates some examples of this dataset. As can be seen, the keywords (second column, treated as side information) contain additional contextual information that can provide complementary cues so as to better guide the translation process. Let’s take an example in Figure 3.4 (TED Video Id 172), the keyword “art” provides cues for words and phrases in target sequence such as: “place, design”; whereas the keyword “tech” refers to “Media Lab, computer science”.

**Personalised Europarl** For the second dataset, we evaluate our proposed idea in the context of personality-aware MT. Mirkin et al. (2015) explored whether translation preserves personality information (e.g., demographic and psychometric traits) in statistical MT (SMT); and further Rabinovich et al. (2017) found that personality information like author’s gender is an obvious signal in source text, but it is less clear in human and machine translated texts. As a result, they created a new dataset for personalised MT partially based on the original Europarl.

---


The personality such as author’s gender will be regarded as side information in our setup. An excerpt of this dataset is shown in Figure 3.5. As can be seen from the figure, there exist many kinds of side information pertaining to authors’ traits, including identification (ID, name), native language, gender, date of birth/age, and plenary session date. Here, we will focus on the “gender” trait and evaluate whether it can have any benefits in the context of NMT complementing the work of Rabinovich et al. (2017) attempted a similar idea as part of a SMT, rather than NMT, system.

**Patent MT Collection** Another interesting data is patent translation which includes rich side information. PatTR\(^{11}\) is a sentence-parallel corpus which is a subset of the MAREC Patent Corpus (Wäschle and Riezler, 2012a). In general, PatTR contains millions of parallel sentences collected from all patent text sections (e.g., title, abstract, claims, description) in multiple languages (English, French, German) (Simianer and Riezler, 2013; Wäschle and Riezler, 2012b). An appealing feature of this corpus is that it provides a labelling at a sentence level, in the form of IPC (International Patent Classification) codes. The IPC codes explicitly provide a hierarchical classification of patents according to various different areas of technology to which they pertain. This kind of side information can provide a useful signal for MT task – which has not yet been fully exploited. Figure 3.6 gives us an illustrating excerpt of this corpus. We can see that each of sentence pair in this corpus is associated with any number of IPC label(s) as well as other metadata, e.g., patent ID, patent family ID, publication date. In this work, we consider only the IPC labels. The full meaning of all IPC labels can be found on the official IPC website,\(^{12}\) however we provide in Figure 3.6 the glosses for each referenced label. Note that those IPC labels form a WordNet style hierarchy (Fellbaum, 1998), and accordingly may be useful in many other deep models of NLP.

### 3.2.3 NMT with Side Information

Now, we investigate different ways of incorporating side information into the NMT model(s).

#### 3.2.3.1 Encoding of Side Information

In this work, we propose the use of unstructured heterogeneous side information, which is often available in practical datasets. Due to the heterogeneity of side information, our techniques are based on a bag-of-words (BOW) representation of the side information, an approach which has been shown to be effective as presented in §3.1. Each element of the

---

\(^{11}\)http://www.cl.uni-heidelberg.de/statnlpgroup/pattr/

\(^{12}\)http://www.wipo.int/classifications/ipc/en/
3.2 Learning Neural Machine Translation with Side Information

![Image of Fig. 3.6 An example with side information (e.g., IPC highlighted in red) for MT with PatTR dataset.](image)

Side information (a label, or word) is embedded using a matrix $W_e^s \in \mathcal{R}^{H_s \times |V_s|}$, where $|V_s|$ is the vocabulary of side information and $H_s$ the dimensionality of the hidden space. These embedding vectors for the input to several different neural architectures, which we now outline.

### 3.2.3.2 NMT Model Formulation

Recall the general formulation of NMT (Bahdanau et al., 2015; Sutskever et al., 2014, inter alia) as a conditional language model in which the generation of target sequence is conditioned on the source sequence (Bahdanau et al., 2015; Sutskever et al., 2014, inter alia), formulated as:

$$y_{t+1} \sim p_\Theta (y_{t+1} | y_{<t}, x) = \text{softmax} \left( f_\Theta (y_{<t}, x) \right) ; \quad (3.3)$$

where the probability $p_\Theta(\cdot)$ of generating the next target word $y_{t+1}$ is conditioned on the previously generated target words $y_{<t}$ and the source sequence $x$; $f$ is a neural network which can be framed as an encoder-decoder model (Sutskever et al., 2014) and can use an attention mechanism (Bahdanau et al., 2015; Luong et al., 2015a). In this model, the encoder encodes
Neural Sequence Models with Side Information

the information of the source sequence; whereas, the decoder decodes the target sequence sequentially from left-to-right. The attention mechanism controls which parts of the source sequence where the decoder should attend to in generating each symbol of target sequence. Later, advanced models have been proposed with modifications of the encoder and decoder architectures, e.g., using the 1D (Gehring et al., 2017b,c) and 2D (Elbayad et al., 2018) convolutions; or a transformer network (Vaswani et al., 2017). These advanced models have led to significantly better results in terms of both performance and efficiency via different benchmarks (Elbayad et al., 2018; Gehring et al., 2017b,c; Vaswani et al., 2017).

Regardless of the NMT architecture, we aim to explore in which case side information can be useful, as well as the effective and efficient way of incorporating them with minimal modification of the NMT architecture. Mathematically, we formulate the NMT problem given the availability of side information $e$ as follows:

$$y_{t+1} \sim p_{\Theta} (y_{t+1}| y_{<t}, x, e) = \text{softmax}(f_{\Theta} (y_{<t}, x, e)) ; \quad (3.4)$$

where $e$ is the representation of additional side information we would like to incorporate into NMT model.

### 3.2.3.3 Conditioning on Side Information

Keeping in mind that we would like a generic incorporation method so that only minimal modification of NMT model is required, we propose and evaluate different approaches.

**Side Information as Source Prefix/Suffix** The most simple way to include side information is to add the side information as a string prefix or suffix to the source sequence, and letting the NMT model learn from this modified data. This method requires no modification of the NMT model. This method was firstly proposed by Sennrich et al. (2016a) who added the side constraints (e.g., honorifics) as suffix of the source sequence for controlling the politeness in translated outputs.

**Side Information as Target Prefix** Alternatively, we can add the bag of words as a target prefix, inspired from Johnson et al. (2017) who introduces an artificial token as a prefix for specifying the required target language in a multilingual NMT system. Note that this method leads to additional benefits in the following situations: a) when the side information exists, the model takes them as inputs and then does its translation task as normal; b) when
the side information is missing, so the model first generates the side information itself and subsequently uses it to proceed with translation.

**Output Layer**  Similar to our findings in §3.1 – in which side information is considered in the model focusing on the output side which worked well in LM, this method involves in two phases. First, it transforms the representation of the side information into a *summed* vector representation, \( e = \sum_{m \in [1, M]} e_{w}^m \). We also tried the *average* operator in our preliminary experiments but observed no difference in end performance compared to the baseline.

Next, the side representation vector, \( e \), is added to the *output layer* before the softmax transformation of the NMT model, e.g.,

\[
y_{t+1} \sim \text{softmax} \left( W_o \cdot f^{dec}_t (\ldots) + b_e + b_o \right)
\]

where there is one bias vector here denoted as \( b_o \) while the term \( b_e \) is the representation of the side information \( e \) weighted by additional weight matrix \( W_e \). Also, \( W_e \in \mathcal{R}^{k_{y_{t}} \times H} \) is an additional weight matrix (learnable model parameters) for linear projection of side information representation onto the target output space (\( H \) is a predefined dimension for embedding side information). The rationale behind this method is to let the model learn to control the importance of the existing side information contributed to the generation. The function \( f^{dec}_t (\ldots) \) is specific to our chosen network reparameterisation, based on RNN (Bahdanau et al., 2015; Luong et al., 2015a; Sutskever et al., 2014), or convolution (Elbayad et al., 2018; Gehring et al., 2017b,c), or transformer (Vaswani et al., 2017). Although we make an effort for modification of the NMT model, we believe that it is minimally simple, and generic to suit many different styles of NMT model.

**Multi-task Learning**  Consider the case where we would like to use existing side information to improve the main NMT task. We can define a generative model \( p(y, e|x) \), formulated as:

\[
p(y, e|x) = \underbrace{p(y|x, e)}_{\text{translation model}} \cdot \underbrace{p(e|x)}_{\text{classification model}}
\]

where \( p(y|x, e) \) is a translation model conditioned on the side information as explained earlier; \( p(e|x) \) can be regarded as a classification model – which predicts the side information given the source sentence. Note that side information can often be represented as individual words – which can be treated as labels, making the classification model feasible.

Importantly, the above formulation of a generative model would require summing over “\( e \)” at test/decode time, which might be done by decoding for all possible label combinations,
Table 3.7 Side information statistics for the three datasets, showing the number of types of the side information label, and the set of tokens (display truncated for PatTR-1 (deep)).

<table>
<thead>
<tr>
<th>Dataset</th>
<th>No. of labels</th>
<th>Examples</th>
</tr>
</thead>
<tbody>
<tr>
<td>TED Talks</td>
<td>10</td>
<td>tech business arts issues education health env recreation politics others</td>
</tr>
<tr>
<td>Personalised Europarl</td>
<td>2</td>
<td>male female</td>
</tr>
<tr>
<td>PatTR-1 (deep)</td>
<td>651</td>
<td>G01G G01L G01N A47F F25D C01B …</td>
</tr>
<tr>
<td>PatTR-2 (shallow)</td>
<td>8</td>
<td>G A F C H B E D</td>
</tr>
</tbody>
</table>

then reporting the sentence with the highest model score. This may be computationally infeasible in practice. We resort this by approximating the NMT model as $p(y|\mathbf{x}, e) \approx p(y|\mathbf{x})$, resulting in

$$p(y, e|\mathbf{x}) \approx p(y|\mathbf{x}) \cdot p(e|\mathbf{x});$$

and thus force the model to encode the shared information in the encoder states.

Our formulation in Equation 3.7 gives rise to multi-task learning (MTL). Here, we propose the joint learning of two different but related tasks: NMT and multi-label classification (MLC). Here, the MLC task refers to predicting the labels that possibly represent words of the given side information. This is interesting in the sense that the model is capable of not only generating the translated outputs, but also explicitly predicting what the side information is. Here, we adopt a simple instance of MTL for our case, called soft parameter sharing similar to (Duong et al., 2015; Yang and Hospedales, 2016). In our MTL version, the NMT and MLC tasks share the parameters of the encoders. The difference between these two is at the decoder part. In the NMT task, the decoder is kept unchanged. For the MLC task, we define its objective function (or loss), formulated as:

$$\mathcal{L}_{MLC} = - \sum_{m=1}^{M} \mathbb{I}_{w_m} \log p_s;$$

where $p_s$ is the probability of predicting the presence or absence of each element in the side information, formulated as:

$$p_s = \text{sigmoid}\left(W_s \left[ \frac{1}{|\mathbf{x}|} \sum_i g'(x_i) \right] + b_s \right);$$

where $\mathbf{x}$ is the source sequence, comprising $x_1, \ldots, x_i, \ldots, x_{|\mathbf{x}|}$ words. Here, we denote a generic function term $g'(\cdot)$ which refers to a vectorised representation of a specific word.
depending on designing the network architecture, e.g., stacked bidirectional (forward and backward) networks over the source sequence (Bahdanau et al., 2015; Luong et al., 2015a); or a convolutional encoder (Gehring et al., 2017b,c) or a transformer encoder (Vaswani et al., 2017).

Further, $W_s \in \mathbb{R}^{|V_s| \times H_x}$ and $b_s \in \mathbb{R}^{|V_s|}$ are two additional model parameters for linear transformation of the source sequence representation (where $H_x$ is a dimension of the output of the $g'(\cdot)$ function, it will differ from network architectures as discussed earlier).

Now, we have two objective functions at the training stage, including the NMT loss $L_{NMT}$ and the MLC loss $L_{MLC}$. The total objective function of our joint learning will be:

$$L = L_{NMT} + \lambda L_{MLC};$$

(3.10)

where: $\lambda$ is the coefficient balancing the two task objectives, whose value is fine-tuned based on the development data to optimise for NMT accuracy measured using BLEU (Papineni et al., 2002).

The idea of MTL applied for NLP was firstly explored by Collobert and Weston (2008), later attracts increasing attentions from the NLP community (Ruder, 2017). Specifically, the idea behind MTL is to leverage related tasks which can be learned jointly — potentially introducing an inductive bias (Feinman and Lake, 2018). An alternative explanation of the benefits of MTL is that joint training with multiple tasks acts as an additional regulariser to the model, reducing the risk of overfitting (Collobert and Weston, 2008; Ruder, 2017, inter alia).

3.2.4 Experiments

3.2.4.1 Datasets

We conducted our experiments using three different datasets including TED Talks (Chen et al., 2016), Personalised Europarl (Rabinovich et al., 2017), and PatTR (Simianer and Riezler, 2013; Wäschle and Riezler, 2012b), translating from German (de) to English (en). The statistics of the training and evaluation sets can be shown in Table 3.8. For the TED Talks and Personalised Europarl datasets, we followed the same sizes of data splits since they are made available on the authors’ github repository and website. For the PatTR dataset, we used the Abstract sections for patents from 2008 or later, and the development and test sets are constructed to have 2000 sentences each, similar to (Simianer and Riezler, 2013; Wäschle and Riezler, 2012b).

---

13Here, to avoid repeating the materials which have been presented in the Background chapter, we will not elaborate their formulations.
Neural Sequence Models with Side Information

Table 3.8 Statistics of the training & evaluation sets from datasets including TED Talks, Personalised Europarl, and PatTR; showing in each cell the count for the source language (left) and target language (right); “#types” refers to subword-segmented vocabulary sizes; “n.a.” is not applicable, for development and test sets. Note that all the “#tokens”’ and “#types”’ are approximated.

<table>
<thead>
<tr>
<th>Dataset</th>
<th># tokens (M)</th>
<th># types (K)</th>
<th># sents</th>
<th># length limit</th>
</tr>
</thead>
<tbody>
<tr>
<td>TED Talks de→en</td>
<td>3.73</td>
<td>3.75</td>
<td>19.78</td>
<td>14.23</td>
</tr>
<tr>
<td>train</td>
<td>0.02</td>
<td>0.02</td>
<td>4.03</td>
<td>3.15</td>
</tr>
<tr>
<td>dev</td>
<td>0.03</td>
<td>0.03</td>
<td>6.07</td>
<td>4.68</td>
</tr>
<tr>
<td>test</td>
<td></td>
<td></td>
<td></td>
<td>n.a.</td>
</tr>
<tr>
<td>.</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Personalised Europarl de→en</td>
<td>8.46</td>
<td>8.39</td>
<td>21.15</td>
<td>14.04</td>
</tr>
<tr>
<td>train</td>
<td>0.16</td>
<td>0.16</td>
<td>14.67</td>
<td>9.83</td>
</tr>
<tr>
<td>dev</td>
<td>0.16</td>
<td>0.16</td>
<td>14.76</td>
<td>9.88</td>
</tr>
<tr>
<td>test</td>
<td></td>
<td></td>
<td></td>
<td>n.a.</td>
</tr>
<tr>
<td>.</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PatTR de→en</td>
<td>33.07</td>
<td>32.52</td>
<td>24.97</td>
<td>13.28</td>
</tr>
<tr>
<td>train</td>
<td>0.13</td>
<td>0.13</td>
<td>13.50</td>
<td>6.88</td>
</tr>
<tr>
<td>dev</td>
<td>0.13</td>
<td>0.12</td>
<td>13.35</td>
<td>6.89</td>
</tr>
<tr>
<td>test</td>
<td></td>
<td></td>
<td></td>
<td>n.a.</td>
</tr>
</tbody>
</table>

It is important to note the labeling information for side information. We extracted all kinds of side information from three aforementioned datasets in the form of individual words or labels. This makes our the label embeddings much easier. Their relevant statistics and examples can be found in Table 3.7.

We preprocessed all the data using Moses’s training scripts\(^{14}\) with standard steps: punctuation normalisation, tokenisation, truecasing. For training sets, we set word-based length thresholds for filtering long sentences since they will not be useful when training the seq2seq models as suggested in the NMT literature (Bahdanau et al., 2015; Luong et al., 2015a; Sutskever et al., 2014, \textit{inter alia}). We chose 80, 80, 150 length thresholds for TED Talks, Personalized Europarl, and PatTR datasets, respectively. Note that the 150 threshold indicates that the sentences in the PatTR dataset is in average much longer than in the others. For better handling the OOV problem, we segmented all the preprocessed data with subword units using byte-pair-encoding (BPE) method proposed by Sennrich et al. (2016d). We found that languages such English and German share an alphabet, hence learning BPE on the concatenation of source and target languages (hence called shared BPE) increases the consistency of the segmentation. We applied 32000 operations for learning the shared BPE by using the open-source toolkit.\(^{15}\) Also, we used dev sets for tuning model parameters and

\(^{14}\)https://github.com/moses-smt/mosesdecoder/tree/master/scripts

\(^{15}\)https://github.com/rsennrich/subword-nmt
Table 3.9 Evaluation results with BLEU scores of various incorporation variants against the baseline; **bold**: better than the baseline, †: statistically significantly better than the baseline.

3.2.4.2 Baselines and Setups

Recall that our method for incorporating the additional side information into the NMT models is generic; hence, it is applicable to any NMT architecture. We chose the transformer architecture as discussed in details in §2.1.2.3 for all our experiments since it arguably is currently the most robust NMT models compared to RNN and convolution based architectures. We re-implemented the transformer - based NMT system using the C++ Neural Network Library - DyNet\textsuperscript{16} as our deep learning backend toolkit. Our re-implementation results in the open source toolkit.\textsuperscript{17}

In our experiments, we use the same configurations for all transformer models and datasets, including: 2 encoder and decoder layers; 512 input embedding and hidden layer dimensions; sinusoid positional encoding; dropout with 0.1 probability for source and target embeddings, sub-layers (attention + feedforward), attentive dropout; and label smoothing with weight 0.1. For training our neural models, we used early stopping based on development perplexity, which usually occurs after 20-30 epochs.\textsuperscript{18}

We conducted our experiments with various incorporation methods as discussed in Section 3.2.3.1. We denote the system variants as follows:

base refers to the baseline NMT system using the transformer without using any side information.

\textsuperscript{16}https://github.com/clab/dynet/
\textsuperscript{17}https://github.com/duyvuleo/Transformer-DyNet/tree/master/src/research
\textsuperscript{18}The training process of transformer models is much faster than the RNN and convolution - based ones, but requires more epochs for convergence.
**Fig. 3.7** Chart with individual BLEU scores for each of categories in the TEDTalks dataset.

*si-src-prefix* and *si-src-suffix* refer to the NMT system using the side information as respective prefix or suffix of the source sequence Jehl and Riezler (2018), applied to both training and decoding.

*si-trg-prefix* refers to the NMT system using the side information as prefix of the target sequence. We investigated two following variants. The first variant is called as “*si-trg-prefix-p*” in which the side information is firstly generated by the model and is then used as additional contexts for generating the output sequence. During decoding, this is done using a language model output to generate the side information as a sequence of symbols, whereas during training, teacher forcing (Bengio et al., 2015; Goodfellow et al., 2016) is used to condition on ground truth contexts. Another variant is “*si-trg-prefix-h*” indicating that the side information is given at decoding run-time.

*output-layer* refers to the method of incorporating side information in the final output layer.  

*mtl* refers to the multi-task learning method.

Note that the dimensional value for the *output-layer* method was fine-tuned over the development set, using the value range of \{64, 128, 256, 512\}. Similarly, the balancing weight in the *mtl* method is fine-tuned using the value range of \{0.001, 0.01, 0.1, 1.0\}. For evaluation, we measured the end translation quality with case-sensitive BLEU (Papineni et al., 2002). We averaged two runs for each of the method variants.
3.2.4.3 Results and Analysis

The experimental results can be seen in Table 3.9. Overall, we obtained limited success for the method of adding side information as prefix or suffix for TED Talks and Personalised Europarl datasets. On the PatTR dataset, small improvements (0.1-0.2 BLEU) are observed. We experimented two sets of side information in the PatTR dataset, including PatTR-1 (651 deep labels) and PatTR (8 shallow labels).\(^\text{19}\) The possible reason for this phenomenon is that the multi-head attention mechanism in the transformer may have some confusion given the existing side information, either in source or target sequences. In some ambiguous cases, the multi-head attention may not know where it should pay more attention to. Another possible reason is that the implicit ambiguity of side information that may exist in the data.

Contrary to these variants, the output-layer variant was more consistently successful, obtaining the best results across datasets. In the TED Talks and PatTR datasets, this method also provides the statistically significant results compared to the baselines. Additionally, we conducted another experiment by splitting the TED Talks and coarse PatTR-2 datasets by the meta categories, then observed the individual effects when incorporating the side information with output-layer variant, as shown in Figure 3.7 and 3.8. In the TED Talks

\(^{19}\)The shallow setting takes the first character of each label code, which denotes the highest level concept in the type hierarchy, e.g., G01P (measuring speed) → G (physics), with definitions as shown in Fig 3.6.
dataset, we observed improvements for most categories, except for “business, education”. In the coarse PatTR-2 dataset, the improvements are obtained across all categories. The key behind this success of the output-layer variant is that the representation of existing side information is added in the final output layer and controlled by additional learnable model parameters. In that sense, it results in a more direct effect on lexical choice of the NMT model. This resembles the success in the context of language modelling as presented in §3.1.

Further, we also obtained the promising results for the mtl variant although we did implement a very simple instance of MTL with a sharing mechanism and no side information given at a test time. For a fair comparison with the output-layer method, we added an additional experiment in which the output-layer method does not have the access of side information. As expected, its performance has been dropped, as can be seen in the second last row in Table 3.9. In this case, the mtl method without the side information at a test time performs better. We believe that more careful design of the mtl variant can lead to even better results. We also think that the hybrid method combining the output-layer and mtl variants is also an interesting direction for future research, e.g., relaxing the approximation as shown in Equation 3.7.

Given the above results, we can find that the characteristics of side information play an important role in improving the NMT models. Our empirical experiments show that topical information (as in the TED Talks and PatTR datasets) is more useful than the personal traits (as in the Personalised Europarl dataset). However, sometimes it is still good to reserve the personal traits in the target translations (Rabinovich et al., 2017) although their BLEU scores are not necessarily better.

3.2.5 Connection to Related Work

The proposed work is mainly inspired from §3.1 in which we presented the use of side information for boosting the performance of recurrent neural network language models. We further apply this idea for a downstream task of neural machine translation. We have adapted different methods in the literature for this specific problem and evaluated using different datasets with different kinds of side information.

Our methods for incorporating side information as suffix, prefix for either source or target sequences have been adapted from (Johnson et al., 2017; Sennrich et al., 2016a). Also working on the same patent dataset, Jehl and Riezler (2018) proposed to incorporate document meta information as special tokens, similar to our source prefix/suffix method, or by concatenating the tag with each source word. They reported an improvement, consistent with
our findings, although the changes they observe are larger, of about 1 BLEU point, albeit from a lower baseline.

Also, Michel and Neubig (2018) proposed to personalise neural MT systems by taking the variance that each speaker speaks/writes on his own into consideration. They proposed the adaptation process which takes place in the “output” layer, similar to our output layer incorporation method.

The benefit of our proposed MTL approach is not surprising, resembling from existing work, e.g., jointly training translation models from or to multiple languages (Dong et al., 2015); jointly training the encoders (Zoph and Knight, 2016) or both encoders and decoders (Johnson et al., 2017).

3.2.6 Discussion

In this work, we have presented various situations to which extent the side information can boost the performance of the NMT models. We have studied different kinds of side information (e.g. topic information, personal trait) as well as present different ways of incorporating them into the existing NMT models. Though being simple, the idea of utilising the side information for NMT is indeed feasible and we have proved it via our empirical experiments. Our findings will encourage practitioners to pay more attention to the side information if exists. Such side information can provide valuable external knowledge that compensates for the learning models.

3.3 Summary

We have presented our findings on the use of external knowledge such as side information for neural sequence models. We explored different kinds of side information, and then proposed appropriate approaches for incorporating them into use cases of neural sequence models, including recurrent neural network language model and neural machine translation. Our findings will encourage further research on the use of external knowledge which has not been fully exploited yet. Such potentially useful knowledge, on the other hand, might further provide extra benefits in learning neural models. Further, we believe that this idea is not limited to the context of neural LM or NMT, but it may be applicable to other NLP tasks such as summarisation, parsing, reading comprehension, and so on.
Chapter 4

The Use of Prior Linguistics Knowledge for Neural Machine Translation

Contents

4.1 Motivation .................................................... 94
4.2 On the Use of Linguistic Factors ............................ 95
4.3 Multi-Factor Encoder-Decoder ............................. 96
4.4 Multi-Factor Attention Architectures ...................... 98
4.5 Experiments .................................................. 100
4.5.1 Datasets .................................................. 100
4.5.2 Set-up and Baselines .................................... 101
4.5.3 Results and Analysis .................................... 102
4.6 Towards Target-factored Neural Translation Model .... 105
4.6.1 Formulation .............................................. 106
4.6.2 Discussion ............................................... 108
4.7 Summary ..................................................... 108

In chapter §3, we have presented how external knowledge such as side information can help boost the performance of sequence models. In this chapter, we will study a different kind of external knowledge by utilising word-level linguistic factors; and discuss how they can be integrated into NMT models in an effective and efficient way.

In general, machine translation does not primarily involve linguistic operations. Automated machine translation attempts to learn translational equivalence between languages by using statistical methods. Adding linguistic factors may be of great benefits to learning
of MT models, potentially reducing language ambiguity or alleviating the data sparseness problem. Koehn and Hoang (2007) explored a successful factored translation framework for incorporating linguistic factors into conventional statistical machine translation (SMT) models. This factored framework allows to incorporate arbitrary word-level linguistics features into the SMT framework (Koehn et al., 2003), leading to not only improved translation quality but also better grammatical coherence. We believe that such linguistic factors might also have similar effects for NMT models.

In this chapter, we further explore another kind of external knowledge by utilising different linguistic annotations at the word level, such as: lemmatisation, word clusters, Part-of-Speech tags, and labeled dependency relations — as shown in §4.2. In §4.3 and §4.4, we then propose different neural attention architectures to integrate these additional factors into the NMT framework. In §4.5, evaluating on translation between English and German in two directions with a “low” resource setting and English to Slovenian with an “extremely low” resource setting, in the domain of TED talks, we obtain promising results in terms of both perplexity reductions and improved BLEU scores over baseline methods.

Parts of this chapter were published in:

(i) Cong Duy Vu Hoang, Trevor Cohn, Gholamreza Haffari (2016). Improving Neural Translation Models with Linguistic Factors. In Proceedings of the Australasian Language Technology Association Workshop (ALTA-16), pages 7-14, Caulfield, Australia. (Best Paper Award)

4.1 Motivation

Though promising, the NMT models still lack of the ability of modelling deeper semantic and syntactic aspects of the language. Koehn and Hoang (2007) proposed a factored translation model to address this issue for the conventional SMT framework (Koehn et al., 2007), where the model incorporates various linguistic annotations for the surface level words. Particularly for low-resource conditions, these extra annotations can lead to better translation of OOVs (or low-count words) and resolve ambiguities, hence increase the generalisation capabilities of the model. In this chapter, we will investigate utilizing extra syntactic and semantic linguistic factors in the context of the NMT framework. Adding such extra factors may be of great benefits to NMT models, potentially reducing language ambiguity and alleviating data sparseness further. We explore different linguistic factor annotations, including: lemmatisation, word clusters, Part-of-Speech tagging, and syntactic dependency parse trees. We then propose neural attention architectures to integrate these additional factors into the NMT framework.
4.2 On the Use of Linguistic Factors

Linguistic factors can be bundles of features, e.g., stems, roots, lemmas, morphological classes, data-driven clusters, syntactic analyses (part-of-speeches, constituency parsing, dependency parsing). In this work, we will explore incorporating of some of these factors into the context of the NMT framework.

First, we examine lemmatisation, a shallow morphological analysis reducing the inflected words, and word clustering, a data-driven induction based on distributional similarity. Such kinds of factors can provide extra dimensions for data sparseness problem as shown in earlier work (Rishøj and Søgaard, 2011; Wuebker et al., 2013; Zhang and Sumita, 2007).
Second, to alleviate semantic ambiguities in translation, we utilize rich syntactic information available in language, including: grammatical role of words with Part-of-Speech (POS) and word dependency relations with labelled dependency. Here, the POS tags can form an extra sequence of grammatical roles on words in a sequence; and we transform the dependency representation where each word is annotated with the dependency label between each word and its head (together with its direction, i.e. left or right) in the dependency parse tree.

As a result, we have additional layers of word annotations with linguistic knowledge and also note that such annotations form sequences of the same lengths as the word sequence. Figure 4.1 shows an example of a given sentence in English and German annotated with annotations of different linguistic factors.

### 4.3 Multi-Factor Encoder-Decoder

We investigate different ways of incorporating the annotations of linguistic factors into the NMT model(s).

**Encoder.** First, to encode the source-side information, we first run each layer of linguistic annotations through bidirectional (forward and backward) RNNs (biRNN) for dynamically representing the sequence embeddings, i.e.,

\[
\begin{align*}
    h^\ell_j &= \text{biRNN}^{\ell,\psi_{\text{enc}}}(x^\ell_j, [\overrightarrow{h^\ell_{j-1}}, \overleftarrow{h^\ell_{j+1}}]) \\
    &= \begin{bmatrix}
        \text{forwardRNN}^{\ell,\psi_{\text{enc}}}(x^\ell_j, \overrightarrow{h^\ell_{j-1}}) \\
        \text{backwardRNN}^{\ell,\psi_{\text{enc}}}(x^\ell_j, \overleftarrow{h^\ell_{j+1}})
    \end{bmatrix}
\end{align*}
\]

where \(\psi_{\ell} = [\overrightarrow{\psi_{\ell}}, \overleftarrow{\psi_{\ell}}]\) are the model parameter sets of respective forward and backward RNNs running over the sequence annotation \(\ell^\ell\), \(x^\ell_j \in \mathbb{R}^{H^\ell}\) is the word embedding at position \(j\) in sequence layer \(\ell\), and \(\overrightarrow{h^\ell_j}\) and \(\overleftarrow{h^\ell_j}\) are the hidden states of forward and backward RNNs, respectively. Here, RNN can be with LSTM (Hochreiter and Schmidhuber, 1997) or GRU (Cho et al., 2014b). This bidirectional encoding scheme captures not only the position specific information, but also the information coming from the left and right contexts.
Fig. 4.2 Proposed attention architectures of integrating linguistic factors for the NMT framework.
Decoder. Next, a decoder operated by another RNN is used to predict the target $y$ sequentially, from left to right:

$$
g_i = \text{RNN}^\phi (c_i, y_{i-1}, g_{i-1})$$

$$y_i \sim \text{softmax} \left( W_o \cdot \text{MLP} \left( c_i, y_{i-1}, g_i \right) + b_o \right) ;$$

where MLP is a single hidden layer neural network with tanh activation. The model parameters include $\phi$ the weight matrix $W_o \in \mathbb{R}^{V_y \times H}$ and the bias $b_o \in \mathbb{R}^{V_y}$, with $V_y$ and $H$ denoting the target vocabulary size and hidden dimension size, respectively.

Note that the state of the decoder $g_i$ is conditioned on its previous state $g_{i-1}$, the previously generated target word $y_{i-1}$, and the source side context $c_i$ summarizing the areas of the source sentence which is attended to. Finally, the model is trained end-to-end by minimizing the cross-entropy loss over the target sequence $y_1, y_2, \ldots, y_T$ and stochastic gradient descent (SGD) is used for optimizing the model parameters.

In what follows, we explore various attention mechanisms for our case where the input sentence is annotated with multiple linguistic factors, and show how the source context $c_i$ is constructed.

### 4.4 Multi-Factor Attention Architectures

In this paper, we explore various attention mechanisms of integrating linguistic factors as briefly summarized in Figure 4.2, including Global Attention, Local Attention, and hybrid Global-Local Attention.

Global Attention. Our first approach has one shared attention vector for all the annotation layers, forcing each layer to attend to the same positions. This essentially means stacking the representations of the embeddings of all the layers $\{x^\ell_j\}_{\ell=1}^L$ into one vector, i.e.,

$$x^g_j = \begin{bmatrix} x^0_j \\ x^1_j \\ \vdots \\ x^L_j \end{bmatrix}.$$  

This stacked representation is used in place of only word embedding $x_j$ to encode the input position (eqn 4.1) to $h^g_j$. It is then used to construct the source context for
the decoder, using \( c_j = \sum_{j=1}^{|x|} \alpha_{ij} h^s_j \) with

\[
\alpha_i = \text{softmax}(e_i) \quad ; \quad e_{ij} = \text{MLP} \left( g_{i-1}, h^s_j \right) \\
h^s_j = \text{biRNN}^\theta_{\text{enc}} \left( x^s_j, \begin{bmatrix} h^s_{j-1} \\ h^s_{j+1} \end{bmatrix} \right),
\]

where scalar \( e_{ij} \) denotes the unnormalized alignment probability between the source word annotation \( j \) and target word \( i \), which is produced by single hidden layer neural network with tanh activation.

**Local Attention.** The model may benefit from different attentions learned for different layers. Thus, the second idea is to have multiple attentions for linguistic layers independently, and compute layer-specific context vectors \( \{e^\ell_i\}_{\ell=0}^L \) and stack them up:

\[
c_i = [c^0_i, \ldots, c^L_i]^T \quad ; \quad c^\ell_i = \sum_{j=1}^{T_N} \alpha^\ell_{ij} h^\ell_j \\
\alpha^\ell_i = \text{softmax}(e^\ell_i) \quad ; \quad e^\ell_{ij} = \text{MLP} \left( g_{i-1}; h^\ell_j \right)
\]

where \( e^\ell_{ij} \) denotes the alignment score between the annotation at layer \( \ell \) and the target word. The MLP for each layer has a different parameterisation.

**Global-Local Attention.** Finally, we consider a hybrid global-local attention mechanism which makes use of the *global* hidden representation \( h^s \) across all of the layers in generating the *local* attentions, formulated as:

\[
e^\ell_{ij} = \text{MLP} \left( g_{i-1}, h^s_j \right).
\]

In contrast to the local attention the attention for layer \( \ell \) depends on the global encoding, \( h^s \), rather than the local encoding for that layer, \( h^\ell \).

In training, we *encourage* the model to have similar attentions across the layers by adding a penalty term to the cross-entropy training objective,

\[
\lambda \sum_{n=1}^N \sum_{i=1}^{|y^{(n)}|} \sum_{\ell=0}^L \left\| \alpha_{i}^{(n),\ell} - \bar{\alpha}_i^{(n),\ell} \right\|_2^2
\]

where \( \alpha_{i}^{(n),\ell} \) is the attention to the layer \( \ell \) when generating the target word \( i \), \( \lambda \) is a weight for controlling the importance of the penalty term, and we define \( \bar{\alpha}_i^{(n),\ell} := \frac{1}{L+1} \sum_{\ell=0}^L \alpha_{i}^{(n),\ell} \) as the
average attention across all layers. Essentially, our regularizer penalizes parameters which induce layer-specific attentions deviating from the average attention.

### 4.5 Experiments

#### 4.5.1 Datasets

We conducted our experiments on TED Talks datasets (Cettolo et al., 2012) and translate between English (en) ↔ German (de), and English (en) → Slovenian (sl). For training, we used about 200K and 17K parallel sentences for English–German and English–Slovenian language pairs, respectively; and used tst2010 for tuning model parameters (phrase-based SMT) and early stopping (NMT). We evaluated on the official test sets tst2013 and tst2014, following Cettolo et al. (2014). For English–German language pair, we chose a word frequency cut-off of $\geq 5$ for limiting the vocabulary when training neural models, resulting in around 19K and 26K word types for English and German, respectively. For English–Slovenian, due to its extremely small size, we chose a word frequency cut-off of $\geq 2$, resulting in around 6K and 9K word types for English and Slovenian, respectively. All details of data statistics can be found in Table 4.1.

As linguistic factors, we annotated the source sentences with lemmas using NLTK,$^1$ word clusters with Brown clustering toolkit$^2$ and POS tagging. We also annotated with the labelled dependency, i.e. by taking the dependency label between each word and its head together with its direction, i.e. left (encoded as “L”) or right (encoded as “R”) in the dependency parse

---

4.5 Experiments

<table>
<thead>
<tr>
<th>Configuration</th>
<th>tst2013</th>
<th>tst2014</th>
<th>#param (M)</th>
</tr>
</thead>
<tbody>
<tr>
<td>en→de</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Vanilla Attentional Model</td>
<td>8.20</td>
<td>10.98</td>
<td>47.80</td>
</tr>
<tr>
<td>w/ glo+all-factors</td>
<td>7.84</td>
<td>10.35</td>
<td>50.88</td>
</tr>
<tr>
<td>w/ loc+all-factors</td>
<td>8.02</td>
<td>10.80</td>
<td>52.06</td>
</tr>
<tr>
<td>w/ glo-loc+all-factors (w/o regularisation penalty)</td>
<td>7.81</td>
<td>10.28</td>
<td>57.52</td>
</tr>
<tr>
<td>w/ glo-loc+all-factors (w/ regularisation penalty)</td>
<td>7.48*</td>
<td>10.15*</td>
<td>57.52</td>
</tr>
<tr>
<td>de→en</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Vanilla Attentional Model</td>
<td>8.76</td>
<td>11.81</td>
<td>44.46</td>
</tr>
<tr>
<td>w/ glo+all-factors</td>
<td>8.50</td>
<td>11.26</td>
<td>47.58</td>
</tr>
<tr>
<td>w/ loc+all-factors</td>
<td>8.50</td>
<td>11.48</td>
<td>48.76</td>
</tr>
<tr>
<td>w/ glo-loc+all-factors (w/o regularisation penalty)</td>
<td>8.33</td>
<td>11.22</td>
<td>54.22</td>
</tr>
<tr>
<td>w/ glo-loc+all-factors (w/ regularisation penalty)</td>
<td>8.29*</td>
<td>10.95*</td>
<td>54.22</td>
</tr>
</tbody>
</table>

Table 4.2 Perplexity scores for attentional model variants evaluated on en→de translations, and “#param” refers to no. of model parameters (in millions). **bold**: “statistically significantly better than vanilla attentional model”, ♠: best performance.

<table>
<thead>
<tr>
<th>Configuration</th>
<th>Perplexity</th>
<th>BLEU</th>
<th>Configuration</th>
<th>Perplexity</th>
<th>BLEU</th>
</tr>
</thead>
<tbody>
<tr>
<td>en→de</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Vanilla Attentional Model</td>
<td>10.98</td>
<td>20.96</td>
<td>w/ glo+word-classes</td>
<td>10.71</td>
<td>21.51</td>
</tr>
<tr>
<td>w/ glo+POS</td>
<td>10.56</td>
<td>21.60</td>
<td>w/ glo+POS</td>
<td>11.66</td>
<td>24.99</td>
</tr>
<tr>
<td>w/ glo+morph</td>
<td>10.74</td>
<td>21.08</td>
<td>w/ glo+morph</td>
<td>11.78</td>
<td>24.90</td>
</tr>
<tr>
<td>w/ glo+dep</td>
<td>10.57</td>
<td>21.73</td>
<td>w/ glo+dep</td>
<td>11.58</td>
<td>25.01</td>
</tr>
<tr>
<td>w/ glo+all-factors</td>
<td>10.38*</td>
<td>22.15*</td>
<td>w/ glo+all-factors</td>
<td>11.26*</td>
<td>25.30*</td>
</tr>
</tbody>
</table>

| de→en         |          |      |              |            |      |
| Vanilla Attentional Model | 11.81    | 24.84 | w/ glo+word-classes | 11.71     | 24.90 |
| w/ glo+POS    | 11.66     | 24.99 | w/ glo+POS    | 11.66      | 24.99 |
| w/ glo+morph  | 11.78     | 24.90 | w/ glo+morph  | 11.78      | 24.90 |
| w/ glo+dep    | 11.58     | 25.01 | w/ glo+dep    | 11.58      | 25.01 |
| w/ glo+all-factors | 11.26*    | 25.30* | w/ glo+all-factors | 11.26*    | 25.30* |

Table 4.3 BLEU scores for attentional models with the effects of individual linguistic factors evaluated on en↔de translations for the tst2014 testset.

tree — as shown in Figure 4.1. Also note that the POS tags and dependency parse trees were extracted from parsing results produced by Stanford Parser\(^3\) and ParZu.\(^4\) Note that we do not have the lemma analyser, part-of-speech tagger, as well as dependency parser available for Slovenian language, so we ignore our experiments for Slovenian to English translation direction though.

### 4.5.2 Set-up and Baselines

We used the DyNet library\(^5\) for our implementation of proposed models. All neural models were configured with 512 input embedding and hidden layer dimensions, and 384 alignment

\(^3\)http://nlp.stanford.edu/software/lex-parser.shtml (en)

\(^4\)https://github.com/rsennrich/ParZu (de)

\(^5\)https://github.com/clab/dynet
<table>
<thead>
<tr>
<th>Configuration</th>
<th>tst2013</th>
<th>tst2014</th>
</tr>
</thead>
<tbody>
<tr>
<td>en→de</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Moses baseline</td>
<td>21.31</td>
<td>19.16</td>
</tr>
<tr>
<td>Vanilla Attentional Model</td>
<td>25.03</td>
<td>20.96</td>
</tr>
<tr>
<td>w/ glo+all-factors</td>
<td>25.43</td>
<td>22.15</td>
</tr>
<tr>
<td>w/ loc+all-factors</td>
<td>25.04</td>
<td>21.24</td>
</tr>
<tr>
<td>w/ glo-loc+all-factors (w/o regularisation penalty)</td>
<td>25.06</td>
<td>21.29</td>
</tr>
<tr>
<td>w/ glo-loc+all-factors (w/ regularisation penalty)</td>
<td>25.92</td>
<td>21.84</td>
</tr>
<tr>
<td>de→en</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Moses baseline</td>
<td>29.96</td>
<td>25.13</td>
</tr>
<tr>
<td>Vanilla Attentional Model</td>
<td>29.85</td>
<td>24.84</td>
</tr>
<tr>
<td>w/ glo+all-factors</td>
<td>29.63</td>
<td>25.30</td>
</tr>
<tr>
<td>w/ loc+all-factors</td>
<td>29.32</td>
<td>24.40</td>
</tr>
<tr>
<td>w/ glo-loc+all-factors (w/o regularisation penalty)</td>
<td>29.78</td>
<td>24.70</td>
</tr>
<tr>
<td>w/ glo-loc+all-factors (w/ regularisation penalty)</td>
<td>30.45</td>
<td>24.72</td>
</tr>
</tbody>
</table>

Table 4.4 BLEU scores for attentional model variants evaluated on en↔de translations.

dimension, with 1 and 2 hidden layers in the source and target, respectively. We employed LSTM recurrent structure (Hochreiter and Schmidhuber, 1997) for both source and target RNN sequences. For the phrase-based SMT baseline, we used the Moses toolkit (Koehn et al., 2007) with its standard configuration. To encode the linguistic factors, we used 128, 64, 64, 64 embedding dimensions for each of lemma, word cluster, Part-of-Speech (POS), and labelled dependency sequences, respectively. For training our neural models, the best perplexity scores on tuning sets were used for early stopping of training, which was usually between 5-8 epochs. For decoding, we used a simple greedy algorithm with length normalisation. For evaluation of translations, we applied bootstrapping re-sampling (Koehn, 2004) to measure the statistical significance ($p < 0.05$) of BLEU score differences between translation outputs of proposed models compared to the baselines.

4.5.3 Results and Analysis

We report our experimental results based on standard perplexity and BLEU (Papineni et al., 2002) scores, as shown in Table 4.3, 4.2 and 4.4. First, we would like to know the individual effects of linguistic factors during learning, which can be found in Table 4.3. As can be seen, the part-of-speech and dependency parse information are more useful factors than others in any cases. There is complementary information in these factors that leads to bigger impacts than each alone in learning the model. Next, Table 4.2 shows that the attentional model with our extensions is noticeably better than the vanilla NMT in terms of perplexity. Among the three attention architectures, the glo-loc attention outperformed others, giving
significant improvement compared to the vanilla model. The use of the \textit{loc} attention did not give much improvement. We suspect that the learned model itself has difficulties deciding which factors to attend to. The drawback of the \textit{glo} attention is that it enforces only one attention mechanism for all of the layers. This may cause the loss of individual effects that potentially exist in each of layers. The \textit{glo-loc} attention aims at taking advantage of \textit{glo} attention and solving the limitation of \textit{loc} attention with the penalty term, hence giving better performance. We also obtain similar but more consistent improvements in an extremely case of low resource setting for English to Slovenian translation direction - where we found that both extensions with the \textit{glo} and \textit{glo-loc} attentions are effective.

Table 4.4 shows the BLEU score results. Compared to Moses baseline, the vanilla attentional model is superior for \textit{en$\rightarrow$de} and comparable for \textit{de$\rightarrow$en} translation tasks. It is noticeable that the attentional model is capable of working remarkably well, despite the relatively small amounts of parallel data. However, table 4.4 shows the inconsistency, compared to the respective perplexity scores in Table 4.2. For \textit{en$\rightarrow$de}, both \textit{glo} and \textit{glo-loc} attention architectures worked competitively well, giving significantly better BLEU scores than the vanilla attentional model. Compared to \textit{glo}, the \textit{glo-loc} attention is superior in tst2013, but slightly detrimental in tst2014 (although its respective perplexity scores are better). These results show that reductions in perplexity scores do not guarantee improved BLEU scores, which are particularly true for \textit{de$\rightarrow$en} translation.

For the analysis, we further investigate the improvement of the translation quality versus sentence complexity. This would show the extent to which the extra linguistic layers have been helpful in resolving ambiguities of source sentences in translation. We formalize sentence complexity by taking either its length or the depth of its parse tree into consideration. Figure 4.3 and 4.4 plot the BLEU score versus these two measures of complexity in two evaluation sets. As seen, the extra linguistic layers has helped the translation quality of more

<table>
<thead>
<tr>
<th>Configuration</th>
<th>tst2013</th>
<th>tst2014</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Perplexity</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Vanilla Attentional Model</td>
<td>28.98</td>
<td>31.01</td>
</tr>
<tr>
<td>\textit{w/ glo}+all-factors</td>
<td>26.90</td>
<td>29.57</td>
</tr>
<tr>
<td>\textit{w/ glo-loc}+all-factors (w/ regularisation penalty)</td>
<td>26.45$^\bullet$</td>
<td>28.99$^\bullet$</td>
</tr>
<tr>
<td><strong>BLEU</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Moses baseline</td>
<td>10.61</td>
<td>9.22</td>
</tr>
<tr>
<td>Vanilla Attentional Model</td>
<td>7.27</td>
<td>6.76</td>
</tr>
<tr>
<td>\textit{w/ glo}+all-factors</td>
<td>8.04</td>
<td>7.62</td>
</tr>
<tr>
<td>\textit{w/ glo-loc}+all-factors (w/ regularisation penalty)</td>
<td>8.50$^\bullet$</td>
<td>7.80$^\bullet$</td>
</tr>
</tbody>
</table>

Table 4.5 BLEU scores for attentional model variants evaluated on \textit{en$\rightarrow$sl} translations.
Fig. 4.3 Analysis based on the evaluation set tst2013 in en→de translation.

complex sentences compared to the vanilla attentional model. This in turn proves that our proposed attention architectures worked reasonably well.
4.6 Towards Target-factored Neural Translation Model

We have presented the source factored neural translation model in the previous section. The next step is to come up with the idea of neural translation model with the factorisation operating over target language. This idea will be of great benefits for situations where there is
availability of linguistic factors in target language (e.g. translating from other languages to English). We regard this model as target factored neural translation model (TF-NTM). In the source factored neural translation model (SF-NTM) as discussed earlier, the linguistic factors will be always available both in training and inference (or decoding). However, unlike the SF-NTM, in the TF-NTM, the linguistic factors will be partially observed in training and completely unobserved in decoding, meaning that only factors of previously-generated tokens are observed. Thus, we have to develop a factorisation mechanism such that the model can be trained with the ability of jointly predicting the linguistic factors and using those predictions effectively during decoding.

Here, we discuss our initial proposal on a factorisation mechanism such that the model can be trained with the ability of jointly predicting the linguistic factors and using those predictions effectively during decoding. Our initial results on this idea were inconclusive (thus, not reported here), however we believe the method is promising, and leave evaluation and further development for future work.

4.6.1 Formulation

First, for simplicity (without the loss of generalisation), we will ignore the linguistic annotations of source language, but focusing mainly on the annotations of target language.

Assume that we have $L$ layers of linguistic factor annotations operating over the target language. We are provided training data consisting of $N$ training parallel sentences, e.g.,

$$D = \{ x^{(n)}, \{ y^{(n,\ell)} \}_{\ell=0}^{L} \}_{n=1}^{N},$$

where the word target sequence of the $n$th sentence-pair is denoted in the layer zero $y^{(n,0)}$, its length is denoted by $|y^{(n)}|$, its $L$ layers of annotations are denoted by $\{ y^{(n,\ell)} \}_{\ell=1}^{L}$; and the source sequence is denoted as $x^{(n)}$.

We formulate the idea of the TF-NTM by building a probabilistic generative model over all layers of linguistic factor annotations over the target language as follows:

$$P_\theta \left( y_1^0, \ldots, y_{|y|}^0, \{ y_1^\ell, \ldots, y_{|y|}^\ell \}_{\ell=1}^{L} | x \right) = \prod_{\ell=1}^{L} P_\theta \left( y_1^\ell, \ldots, y_{|y|}^\ell | x \right)$$

(4.3)
where we define the conditional probabilities over the linguistic factor sequences as:

\[
P_\theta^\ell (y_1^\ell, \ldots, y_{|y|}^\ell | x) = \prod_{i=1}^{|y|} P_\theta^\ell (y_i^\ell | y_1^\ell, \ldots, y_{i-1}^\ell, x);
\]

(4.4)

and over the target word sequence as:

\[
P_\theta (y_0^0, \ldots, y_{|y|}^0 | \{y_1^\ell, \ldots, y_{|y|}^\ell\}_{\ell=1}^L, x) = \prod_{i=1}^{|y|} \left[ P_\theta (y_i^0 | y_1^0, \ldots, y_{i-1}^0, \{y_1^\ell, \ldots, y_{|y|}^\ell\}_{\ell=1}^L, x) + \sum_{v' \in V^\ell} \exp \left( g\left( h_i^0, v', x, \{y_1^{\ell-1}, \ldots, y_{|y|}^{\ell-1}\}_{\ell=1}^L \right) \right) \right] ;
\]

(4.5)

where \(\theta\) and \(\theta^\ell\) refer to model parameters of generative model for target word sequence and linguistic factor sequences, respectively. Also note that the conditional probability \(P_\theta (y_i^0 | y_1^0, \ldots, y_{i-1}^0, \{y_1^{\ell-1}, \ldots, y_{|y|}^{\ell-1}\}_{\ell=1}^L)\) is a variant of factored language model (Bilmes and Kirchhoff, 2003) which is additionally conditioned on the source language.

Here, we can observe that the factorized conditional probabilities in equations 4.4 and 4.5, \(P_\theta^\ell (y_i^\ell | y_1^\ell, \ldots, y_{i-1}^\ell, x)\) and \(P_\theta (y_i^0 | y_1^0, \ldots, y_{i-1}^0, \{y_1^{\ell-1}, \ldots, y_{|y|}^{\ell-1}\}_{\ell=1}^L)\) can be exclusively modeled using the attentional encoder-decoder framework as discussed earlier in Section 2.1.2. Concretely, they can be approximately parametrized via a softmax RNN, i.e.,

\[
P_\theta^\ell (y_i^\ell = v^\ell | y_1^\ell, \ldots, y_{i-1}^\ell, x) = \frac{\exp \left( g\left( h_i^\ell, v^\ell, x \right) \right)}{\sum_{v'' \in V^\ell} \exp \left( g\left( h_i^\ell, v'', x \right) \right)}
\]

\[
P_\theta (y_i^0 = v | y_1^0, \ldots, y_{i-1}^0, x, \{y_1^{\ell-1}, \ldots, y_{|y|}^{\ell-1}\}_{\ell=1}^L) = \frac{\exp \left( g\left( h_i^0, v, x, \{y_1^{\ell-1}, \ldots, y_{|y|}^{\ell-1}\}_{\ell=1}^L \right) \right)}{\sum_{v' \in V^0} \exp \left( g\left( h_i^0, v', x, \{y_1^{\ell-1}, \ldots, y_{|y|}^{\ell-1}\}_{\ell=1}^L \right) \right)};
\]

(4.6)

where \(h_i^\ell\) and \(h_i^0\) refer to hidden states at time-step \(t\) of word target and linguistic factor RNN sequences, respectively. \(V^\ell (l = [0, L])\) is vocabulary of the linguistic factor at layer \(\ell\).

**Objective.** Here, we have to define the training objective that takes multiple predictions into consideration, i.e.,

\[
\mathcal{L} (\Theta, D) = -\frac{1}{N} \sum_n \sum_{i=1}^{T_{i(n)}} \left[ \log P\left( y_i^{(n)} | y_{<i}^{(n)}, x^{(n)} \right) + \sum_{\ell} \log P\left( y_i^{(n,\ell)} | y_{<i}^{(n,\ell)}, x^{(n)} \right) \right] + \frac{1}{2} \| \Theta \|^2;
\]

(4.7)

where \(\Theta\) refers to all model parameters.
4.6.2 Discussion

The fully-factored neural translation model (FF-NTM) will be a combination of both the SF-NTM and TF-NTM as discussed earlier. This can be done thanks to the independent characteristics of SF-NTM and TF-NTM. Note that the FF-NTM also has potential in improving the existing NMT framework. We leave this for future work.

4.7 Summary

In this chapter, we show that external knowledge such as linguistic factors can provide benefits for improving existing NMT models. We explored different annotations of linguistic factors including word clusters, part-of-speech, lemma and dependency parse tree; then proposed different architectures for incorporating them in existing NMT models. We evaluated our proposed approaches in low-resourced settings on translations between English-German and English-Slovenian directions; and obtained improvements in terms of both perplexity and BLEU.
Chapter 5

The Use of Monolingual Data for Neural Machine Translation

Contents

5.1 Iterative Back Translation for Neural Machine Translation .......... 110
  5.1.1 Related Work .................................................. 110
  5.1.2 The Potential Weaknesses of Back-Translation .................. 111
  5.1.3 Impact of Back-Translation Quality ............................ 112
  5.1.4 The Strategy of Iterative Back-Translation ...................... 115
  5.1.5 Experiments .................................................... 115
  5.1.6 Impact of Data Selection Methods ................................ 121
  5.1.7 Analysis ....................................................... 123
  5.1.8 Discussion ...................................................... 124

5.2 Iterative Back-Translation for Unsupervised Domain Adaptation .... 125
  5.2.1 Motivation ..................................................... 125
  5.2.2 Proposed Approach ........................................... 126
  5.2.3 Experiments .................................................... 126
  5.2.4 Discussion ...................................................... 129

5.3 Summary .......................................................... 129

In this chapter, we will present the extent to which external knowledge such as monolingual data can be used in an effective and efficient way for improving neural machine translation systems. In §5.1, we present a simple but robust strategy (called *iterative back*)
translation) for utilising the monolingual data to improve the NMT systems. In §5.2, we apply the above iterative back translation in the context of unsupervised domain adaptation for NMT models.

Parts of this chapter were published in:


5.1 Iterative Back Translation for Neural Machine Translation

In this section, we present iterative back-translation, a method for generating increasingly better synthetic parallel data from monolingual data to train neural machine translation systems. Our proposed method is very simple yet effective and highly applicable in practice.

The exploitation of monolingual training data for neural machine translation is an open challenge. One successful method is back-translation (Sennrich et al., 2016c), whereby an NMT system is trained in the reverse translation direction (target-to-source), and is then used to translate target-side monolingual data back into the source language (in the backward direction, hence the name back-translation). The resulting sentence pairs constitute a synthetic parallel corpus that can be added to the existing training parallel data to learn a source-to-target model. Figure 5.1 illustrates this idea. As can be seen from the figure, a system in the reverse direction (German→English) is trained and then used to translate monolingual data (German) from the target side backward into the source side (English), to be used in training the final system (English→German).

5.1.1 Related Work

The idea of back-translation dates back at least to statistical machine translation, where it has been used for semi-supervised learning (Bojar and Tamchyna, 2011), or self-training (Goutte et al., 2009, ch.12, p.237). In modern NMT research, (Sennrich et al., 2017a) reported significant gains on the WMT and IWSLT shared tasks. They showed that even simply duplicating the monolingual target data into the source was sufficient to get some benefits. Currey
et al. (2017) reported similar findings for low resource conditions, showing that even poor translations can be beneficial. Gwinnup et al. (2017) mentioned in their system description iteratively applying back-translation, but did not report successful experiments.

A more refined idea of back-translation is the dual learning approach of He et al. (2016a) which integrates training on parallel data and training on monolingual data via round-tripping. We have to admit that we extensively experimented with an implementation of this approach, but did not achieve any gains.

An alternative way to make use of monolingual data is the integration of a separately trained language model into the neural machine translation architecture (Gülçehre et al., 2015), but this has not yet proven to be as successful as back-translation.

Lample et al. (2018b) explored the use of back-translated data generated by neural and statistical machine translation systems, aided by denoising with a language model trained on the target side. Their focus was mainly on unsupervised translation setting.

### 5.1.2 The Potential Weaknesses of Back-Translation

The main motivation of this chapter is to study the back-translation strategy in a deeper perspective. First, we highlight two of its potential weaknesses, and then investigate possible methods to improve upon it. We aim to address two following questions:
Table 5.1 Parallel and monolingual corpora used, including English(EN)-German(DE), English-French(FR) and English-Farsi(FA). Numbers denote the number of words, and $l_2$ is the second language in each pair. The EN-DE data is from WMT 2017 (parallel) and a subset of News 2016 for monolingual side.

<table>
<thead>
<tr>
<th></th>
<th>EN-DE</th>
<th>EN-FR$_{100K}$</th>
<th>EN-FR$_{1M}$</th>
<th>EN-FA</th>
</tr>
</thead>
<tbody>
<tr>
<td>parallel en</td>
<td>141 280 704</td>
<td>2 651 040</td>
<td>26 464 159</td>
<td>2 233 688</td>
</tr>
<tr>
<td>parallel $l_2$</td>
<td>134 638 256</td>
<td>2 962 318</td>
<td>29 622 370</td>
<td>2 473 608</td>
</tr>
<tr>
<td>mono en</td>
<td>322 529 936</td>
<td>2 154 175 053</td>
<td>2 154 175 053</td>
<td>2 154 175 053</td>
</tr>
<tr>
<td>mono $l_2$</td>
<td>301 736 163</td>
<td>766 646 932</td>
<td>766 646 932</td>
<td>65 585 281</td>
</tr>
</tbody>
</table>

Does the quality of back-translations matter?

In this question, we aim to show that the quality of back-translation matters and propose iterative back-translation, where back-translated data is used to build better translation systems in forward and backward directions, which in turn is used to re-back-translate monolingual data. This process can be “iterated” several times. This is a form of co-training (Blum and Mitchell, 1998) where the two models over both translation directions are used to train one another. We show that iterative back-translation leads to improved results over simple back-translation, under both high and low resource conditions, improving over the state of the art.

Does back-translation need the non-random sampling of monolingual data?

Current back-translation (Sennrich et al., 2016c) employed random process when selecting data from the monolingual corpora. We believe that back-translation can be also improved by employing an effective data selection strategy which potentially diversifies the quality and quantity of monolingual data selected for the back-translation process. It is interesting to explore whether state-of-the-art data selection methods are applicable to back-translation for NMT.

5.1.3 Impact of Back-Translation Quality

Our work is inspired by the intuition that a better back-translation system will lead to a better synthetic corpus, hence producing a better final system. To empirically validate this hypothesis and measure the correlation between back-translation system quality and final system quality, we use a set of machine translation systems of differing quality (trained in the reverse “back-translation” direction), and check how this effects the final system quality.
5.1 Iterative Back Translation for Neural Machine Translation

Table 5.2 Results for canonical RNN-based attentional NMT systems with WMT News English→German Translation Task, reporting cased BLEU on newstest2017, evaluating the impact of the quality of the back-translation system on the final system. Note that the back-translation systems run in the opposite direction and are not comparable to the numbers in the same row.

<table>
<thead>
<tr>
<th>German–English</th>
<th>Back</th>
<th>Final</th>
<th>English–German</th>
<th>Back</th>
<th>Final</th>
</tr>
</thead>
<tbody>
<tr>
<td>no back-translation</td>
<td>-</td>
<td>29.6</td>
<td>no back-translation</td>
<td>-</td>
<td>23.7</td>
</tr>
<tr>
<td>10k iterations</td>
<td>10.6</td>
<td>29.6 (+0.0)</td>
<td>10k iterations</td>
<td>14.5</td>
<td>23.7 (+0.0)</td>
</tr>
<tr>
<td>100k iterations</td>
<td>21.0</td>
<td>31.1 (+1.5)</td>
<td>100k iterations</td>
<td>26.2</td>
<td>25.2 (+1.5)</td>
</tr>
<tr>
<td>convergence</td>
<td>23.7</td>
<td>32.5 (+2.9)</td>
<td>convergence</td>
<td>29.6</td>
<td>25.9 (+2.2)</td>
</tr>
</tbody>
</table>

Table 5.3 Results for transformer NMT systems with WMT News English→German Translation Task, reporting cased BLEU on newstest2017, evaluating the impact of the quality of the back-translation system on the final system. Note that the back-translation systems run in the opposite direction and are not comparable to the numbers in the same row.

<table>
<thead>
<tr>
<th>German–English</th>
<th>Back</th>
<th>Final</th>
<th>English–German</th>
<th>Back</th>
<th>Final</th>
</tr>
</thead>
<tbody>
<tr>
<td>no back-translation</td>
<td>-</td>
<td>33.0</td>
<td>no back-translation</td>
<td>-</td>
<td>27.7</td>
</tr>
<tr>
<td>10k iterations</td>
<td>4.6</td>
<td>33.0 (+0.0)</td>
<td>10k iterations</td>
<td>3.5</td>
<td>27.7 (+0.0)</td>
</tr>
<tr>
<td>100k iterations</td>
<td>25.6</td>
<td>36.0 (+3.0)</td>
<td>100k iterations</td>
<td>31.3</td>
<td>28.8 (+1.1)</td>
</tr>
<tr>
<td>convergence</td>
<td>27.7</td>
<td>36.5 (+3.5)</td>
<td>convergence</td>
<td>33.0</td>
<td>29.4 (+1.7)</td>
</tr>
</tbody>
</table>

We carried out experiments on the high-resource WMT German↔English news translation tasks (Bojar et al., 2017). For these tasks, large parallel corpora are available from related domains. In addition, in-domain monolingual news corpora are provided as well, in much larger quantities. We sub-sampled the 2016 news corpus (see Table 5.1) for about twice as large as the parallel training corpus.

Following Sennrich et al. (2016c), a synthetic parallel corpus is created from the in-domain news monolingual data, in equal amounts to the existing real parallel corpus. The systems used to translate the monolingual data are canonical RNN-based attentional NMT systems (Bahdanau et al., 2015) and advanced transformer NMT systems (Vaswani et al., 2017). Our setup is very similar to Edinburgh’s submission to the WMT 2016 evaluation campaign (Sennrich et al., 2016b), but uses the fast Marian toolkit (Junczys-Dowmunt et al., 2018) for training. We trained three different back-translation systems, namely:

---

1 EU Parliament Proceedings, official EU announcements, news commentaries, and web crawled data.
2 With true-casing and 50,000 BPE operations (Sennrich et al., 2016d) as pre-processing steps.
Fig. 5.2 Iterative Back-Translation: Taking the idea of back-translation one step further. After training a system with back-translated data, it is used to create a synthetic parallel corpus for the final system. This process can be repeated until convergence.

**10k iterations**

Training a neural translation model on the parallel corpus, but stopping after 0.15 epochs;

**100k iterations**

As above, but stopping after 1½ epochs; and

**convergence**

As above, but training until convergence (10 epochs, 3 GPU days).

Given these three different systems, we create three synthetic parallel corpora of different quality and train systems on each. Table 5.2 and 5.3 show the quality of the final systems. For both directions, the quality of the back-translation systems differs vastly. The **10k iteration** systems perform poorly, and their synthetic parallel corpus provides no benefit over a baseline that does not use any back-translated data. Note that the transformer NMT systems performed very poorly for the first 10k (see the “Back” columns in Table 5.3) compared to the canonical RNN-based attentional NMT ones. The longer trained systems have much better translation quality, and their synthetic parallel corpora apparently prove to be
5.1 Iterative Back Translation for Neural Machine Translation

Algorithm 4 Iterative Back-Translation

**Input:** parallel data $D^p$, monolingual source, $D^s$, and target $D^t$ text

1: Let $\overline{T} = D^p$
2: repeat
3: Train target-to-source model $\overline{\Theta}$ on $\overline{T}$
4: Use $\overline{\Theta}$ to create $S = \{(\hat{s}, t)\}$, for $t \in D^t$
5: Let $\overline{T} = D^p \cup S$
6: Train source-to-target model $\overline{\Theta}$ on $\overline{T}$
7: Use $\overline{\Theta}$ to create $S' = \{(s, \hat{t})\}$, for $s \in D^s$
8: Let $\overline{T} = D^p \cup S'$
9: until convergence condition reached

**Output:** newly-updated models $\overline{\Theta}$ and $\overline{\Theta}$

beneficial. The back-translation system trained for 100k iteration already provides tangible benefits (e.g., +1.5 BLEU for both directions in RNN-based systems; and +3.0 for German–English and +1.1 for English-German in transformer systems), while the converged system yields even bigger improvements (+2.9 for German–English, and +2.2 for English–German in RNN-based systems; and +3.5 for German–English, and +1.7 for English–German in transformer systems). These results indicate that the quality of the back-translation system is a significant factor for the success of the approach.

5.1.4 The Strategy of Iterative Back-Translation

We now take the idea of back-translation one step further. If we can build a better system with the back-translated data, then we can continue repeating this process: Use this better system to back-translate the data, and use this data in order to build an even better system. See Figure 5.2 for an illustration of this **iterated-back-translation** process (repeated back-translation). See Algorithm 4 for the details of this **iterated** back-translation process. The final system benefits from monolingual data in both the source and target languages.

We do not have to stop at one iteration of repeated back-translation. We can iterate training the two back-translation systems multiple times. We refer this process to **iterative back-translation**.

5.1.5 Experiments

In our experiments, we validate our approach under both high-resource and low-resource conditions. Under high-resource conditions, we improve the state of the art with iterative
Table 5.4 Results of canonical attentional NMT systems on WMT News English↔German Translation Task, reporting cased BLEU on newstest2017, for comparing the quality of different back-translation systems with different final system architectures. *Note that the quality for the back-translation system (Back) is measured in the opposite language direction. We denoted \( bt \) as back-translation, \( ibt+n \) as iterative back-translation with \( n \) iterations.

<table>
<thead>
<tr>
<th>DE→EN</th>
<th>Back*</th>
<th>Shallow</th>
<th>Deep</th>
<th>Ensemble</th>
</tr>
</thead>
<tbody>
<tr>
<td>( bt )</td>
<td>23.7</td>
<td>32.5</td>
<td>35.0</td>
<td>35.6</td>
</tr>
<tr>
<td>( ibt+1 )</td>
<td>27.9</td>
<td>33.6</td>
<td>36.1</td>
<td>36.5</td>
</tr>
<tr>
<td>( ibt+2 )</td>
<td>27.8</td>
<td>33.6</td>
<td>36.0</td>
<td>36.3</td>
</tr>
<tr>
<td>Best WMT 2017</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>35.1</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>EN→DE</th>
<th>Back*</th>
<th>Shallow</th>
<th>Deep</th>
<th>Ensemble</th>
</tr>
</thead>
<tbody>
<tr>
<td>( bt )</td>
<td>29.1</td>
<td>25.9</td>
<td>28.3</td>
<td>28.8</td>
</tr>
<tr>
<td>( ibt+1 )</td>
<td>34.8</td>
<td>27.0</td>
<td>29.0</td>
<td>29.3</td>
</tr>
<tr>
<td>( ibt+2 )</td>
<td>35.0</td>
<td>26.9</td>
<td>28.8</td>
<td>29.1</td>
</tr>
<tr>
<td>Best WMT 2017</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>28.3</td>
</tr>
</tbody>
</table>

5.1.5.1 Experiments on High Resource Scenario

In §5.1.3 we demonstrated that the quality of the back-translation system has a significant impact on the effectiveness of the back-translation approach under high-resource data conditions such as WMT 2017 German–English. Here we ask: how much additional benefit can be realised for repeating this process? Also, do the gains for state-of-the-art systems that use deeper models, i.e., more layers in encoder and decoder (Miceli Barone et al., 2017) still apply in this setting?

Experimental settings. For the experiments in the German–English high-resource scenario, we used the Marian toolkit (Junczys-Dowmunt et al., 2018) for training and for back-translation. The shallow systems (also used for the back-translation step) match the setup of Edinburgh’s WMT 2016 system (Sennrich et al., 2016b). It is an attentional RNN (default Marian settings) with drop-out of 0.2 for the RNN parameters, and 0.1 otherwise. Training is smoothed with moving average. It takes about 2–4 days.

The deep system we used matches the setup of Edinburgh’s WMT 2017 system (Sennrich et al., 2017a). It uses 4 encoder and 4 decoder layers (Marian setting best-deep) with LSTM cells. Drop-out settings are the same as above. Decoding during test time is done
with a beam size of 12, while back-translation uses only a beam size of 2. This difference is reflected in the reported BLEU score for the deep system after back-translation (35.0 for German–English, 28.3 for English–German) and the score reported for the quality of the back-translation system (34.8 (−0.2) and 27.9 (−0.4), respectively) in Table 5.4.

For the transformer-based NMT systems, we used the following setting: 6 encoder and decoder layers; 512 input embedding and hidden layer dimensions; sinusoid positional encoding; dropout with 0.2 probability for source and target embeddings, sub-layers (attention + feedforward), attentive dropout; label smoothing with weight 0.1; and swish (Ramachandran et al., 2017) feed-forward activation.

For all experiments, the true-casing model and the list of BPE operations is left constant. Both were learned from the original parallel training corpus.

We evaluate on German–English and English–German, under the same data conditions as in Section 5.1.3. We experiment with both shallow and canonical attentional NMT systems with deep stacked-layer encoders/decoders, as well as the transformer architectures.

**The base translation system**

is trained on the parallel data only. We train a shallow system using 4-checkpoint ensembling (Chen et al., 2017a). The system is used to translate the monolingual data using a beam size of 2.

**The first back-translation system (bt)**

is trained on the parallel data and the synthetic data generated by the base translation system. For better performance, we train a deep model with 8-checkpoint ensembling; again we use a beam size of 2.

**The second (ibt+1) and third (ibt+2) back-translation systems**

are trained using several different systems: a shallow architecture, a deep architecture, and an ensemble system of 4 independent training runs.

**Experimental results.** First, let’s consider the results from the canonical attentional NMT systems from Table 5.4. Across the board, the systems with more iterations of back-translation outperform the final systems with simple back-translation, by a margin of 0.5–1.1 BLEU. Notably, the final deep systems trained by 2nd-iterative back-translation outperform the state-of-the-art established at the WMT 2017 evaluation campaign for these language pairs, by a margin of about 1 BLEU point. These are the best published results for this dataset, to the best of our knowledge. Note that the third iteration of back-translation do not lead to
The Use of Monolingual Data for Neural Machine Translation

Table 5.5 Results of transformer NMT systems on WMT News English→German Translation Task, reporting cased BLEU on test sets from 2014–2017, for comparing the quality of different back-translation systems with different final system architectures. *Note that the best NMT w/ ibt is the system reported in Table 5.4. We denoted bt as back-translation, ibt+n as iterative back-translation with n iterations; and bold for best results.

<table>
<thead>
<tr>
<th></th>
<th>newstest2014</th>
<th>newstest2015</th>
<th>newstest2016</th>
<th>newstest2017</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>DE→EN</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>NMT baseline</td>
<td>32.08</td>
<td>32.23</td>
<td>38.06</td>
<td>33.01</td>
</tr>
<tr>
<td>bt</td>
<td>34.05</td>
<td>34.60</td>
<td>42.38</td>
<td>36.53</td>
</tr>
<tr>
<td>ibt+1</td>
<td>34.41</td>
<td>34.80</td>
<td>42.64</td>
<td>36.83</td>
</tr>
<tr>
<td>ibt+2</td>
<td><strong>34.64</strong></td>
<td><strong>34.88</strong></td>
<td><strong>42.94</strong></td>
<td><strong>36.91</strong></td>
</tr>
<tr>
<td>Best NMT w/ ibt*</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>36.5</td>
</tr>
<tr>
<td>Best WMT 2017</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>35.1</td>
</tr>
<tr>
<td><strong>EN→DE</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>NMT baseline</td>
<td>26.37</td>
<td>29.45</td>
<td>34.08</td>
<td>27.67</td>
</tr>
<tr>
<td>bt</td>
<td>28.69</td>
<td>30.65</td>
<td>35.27</td>
<td>29.35</td>
</tr>
<tr>
<td>ibt+1</td>
<td>29.08</td>
<td>30.11</td>
<td>35.35</td>
<td>29.82</td>
</tr>
<tr>
<td>ibt+2</td>
<td><strong>29.13</strong></td>
<td><strong>30.86</strong></td>
<td><strong>36.20</strong></td>
<td><strong>30.03</strong></td>
</tr>
<tr>
<td>Best NMT w/ ibt*</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>29.3</td>
</tr>
<tr>
<td>Best WMT 2017</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>28.3</td>
</tr>
</tbody>
</table>

better performance. We believe after reaching the certain performances, the canonical attentional NMT systems might reach its convergence on this dataset; hence learning more from monolingual data will not provide further benefits. Note that it remains a research question on how many back-translation iterations is sufficient and whether we are able to computationally value it.

Next, we conducted our experiments with the transformer NMT systems and the results can be found in Table 5.5. For experimental settings, we used the same datasets for training and preprocessing steps as in previous settings for canonical NMT systems. However, for transformer NMT systems, we only reported one single system (with best checkpoint on development set) and used beam search with beam size of 5. As expected, our observations on the results reveal that back-translation with more iterations has been proved to be indeed useful. Remarkably, the third iterations of back-translation leads to best result which even outperforms the best result for the canonical NMT systems.

---

3 Despite that, we will suggest that practitioners should evaluate the idea of iterative back-translation on other data and domains.

4 We believe that experiments with ensemble systems probably lead to even much better results. Due to limited time, we ignore them in our experiments.
5.1 Iterative Back Translation for Neural Machine Translation

### Setting

<table>
<thead>
<tr>
<th>Setting</th>
<th>FR→EN 100K</th>
<th>FR→EN 1M</th>
<th>EN→FR 100K</th>
<th>EN→FR 1M</th>
<th>FA→EN 100K</th>
<th>EN→FA 100K</th>
</tr>
</thead>
<tbody>
<tr>
<td>NMT baseline</td>
<td>16.7</td>
<td>24.7</td>
<td>18.0</td>
<td>25.6</td>
<td>21.7</td>
<td>16.4</td>
</tr>
<tr>
<td>bt</td>
<td>22.1</td>
<td>27.8</td>
<td>21.5</td>
<td>27.0</td>
<td>22.1</td>
<td>16.7</td>
</tr>
<tr>
<td>ibt+1</td>
<td>22.5</td>
<td>28.9</td>
<td>22.7</td>
<td>28.5</td>
<td>22.7</td>
<td>17.1</td>
</tr>
<tr>
<td>ibt+2</td>
<td><strong>22.6</strong></td>
<td><strong>29.1</strong></td>
<td><strong>22.6</strong></td>
<td><strong>28.8</strong></td>
<td><strong>22.6</strong></td>
<td><strong>17.2</strong></td>
</tr>
<tr>
<td>bt (w/ Moses)</td>
<td>23.7</td>
<td>27.9</td>
<td>23.5</td>
<td>27.3</td>
<td>21.8</td>
<td>16.8</td>
</tr>
</tbody>
</table>

Table 5.6 Low resource setting: Impact of the quality of the back-translation systems on the benefit of the synthetic parallel for the final system in a low-resource setting. Note that, we reported the single NMT systems in all numbers. We denoted $bt$ as back-translation, $ibt+n$ as iterative back-translation with $n$ iterations.

#### 5.1.5.2 Experiments on Low Resource Scenario

NMT is a data-hungry approach, requiring a large amount of parallel data to reach reasonable performance (Koehn and Knowles, 2017). In a low-resource setting, only small amount of parallel data exists. Previous work has attempted to incorporate prior or external knowledge to compensate for the lack of parallel data, e.g. injecting inductive bias via linguistic constraints (Cohn et al., 2016) or linguistic factors (as discussed in Chapter §4). However, it is much cheaper and easier to obtain monolingual data in either the source or target language. An interesting question is whether the (iterative) back-translation can compensate for the lack of parallel data in such low-resource settings.

To explore this question, we conducted experiments on two datasets: A simulated low-resource setting with English–French, and a more realistic setting with English–Farsi. For the English–French dataset, we used the original WMT dataset, sub-sampled to create smaller sets of 100K and 1M parallel sentence pairs. For English–Farsi, we used the available datasets from LDC and TED Talks, totaling about 100K sentence pairs. For detailed statistics see Table 5.1.

Following the same experimental setup as in high-resource setting,\(^5\) we obtain similar patterns of improvement of translation quality (Table 5.6).

**Back-translation.** Generally, it is our expectation that the back-translation approach still improves the translation accuracy in all language pairs with a low-resource setting. In the English–French experiments, large improvements over the baseline are observed in both directions, with +3.5 BLEU for English to French and +5.4 for French to English in 100K

---

\(^5\)The difference here is on the NMT toolkit used — we opted to use Amazon’s Sockeye Hieber et al. (2018). We used Sockeye’s default configuration with dropout 0.5.
The Use of Monolingual Data for Neural Machine Translation

<table>
<thead>
<tr>
<th>English–Farsi</th>
<th>100K</th>
<th>Farsi-English</th>
<th>100K</th>
</tr>
</thead>
<tbody>
<tr>
<td>back-translation 1:1</td>
<td>16.7</td>
<td>back-translation 1:1</td>
<td>22.1</td>
</tr>
<tr>
<td>back-translation 1:2</td>
<td>16.8</td>
<td>back-translation 1:2</td>
<td>22.4</td>
</tr>
<tr>
<td>back-translation 1:3</td>
<td>16.9</td>
<td>back-translation 1:3</td>
<td>22.4</td>
</tr>
</tbody>
</table>

Table 5.7 Weighting amounts of real parallel data (1) with varying amounts of synthetic data (1-3). Larger amounts of synthetic data help.

setting. In 1M setting, we also obtained a similar pattern of BLEU gains, albeit of a smaller magnitude, i.e., +1.4 BLEU for English to French and +3.1 for French to English.⁶ Note that the large gains here may be due to the fact that the monolingual data is a similar domain to the test data. Our inspections of the resulting translations show that the lexical choice has been improved significantly. In English–Farsi experiments shown in Table 5.6, we also observed BLEU gains, albeit more modest in size: +0.3 BLEU for English to Farsi and +0.4 for Farsi to English. The smaller gains may be because Farsi translation is much more difficult than French; or a result of the diverse mix of domains in the parallel training data (news with LDC and technical talks with TED) where the domain in monolingual data is entirely news, leading to much lower quality than the other datasets. Measuring the impact of iteratedly back-translated data in relation to varying domain mismatch between parallel and monolingual data, is a very interesting problem which we will explore in future work; but is out of the scope for this work.

**Balance of real and synthetic parallel data.** In all our experiments with back-translation, in order to create synthetic parallel data, a small amount of monolingual data is randomly sampled from the big monolingual data (Table 5.1). As pointed out by Sennrich et al. (2016c), the balance between the real and synthetic parallel data matters. However, there is no obvious evidence about the affect of the sample size, hence we further studied this by choosing a ratio between the real and synthetic parallel data. We opt to use different ratio (e.g., 1(real):2(synthetic) and 1(real):3(synthetic)) in our experiments. Our results in Table 5.7 show that more synthetic parallel data seems to be useful (though not obvious), e.g., gains from 16.7 to 16.9 in English to Farsi and gain from 22.1 to 22.4 in Farsi to English.

**Iterative back-translation.** For iterative back-translation, we obtained consistent results with the earlier findings from §5.1.5.1. In English–French tasks, we see more than +1 BLEU

---

⁶All the scores are statistically significant with \( p < 0.01 \).
from a further iteration of back-translations, with little difference between 1 or 2 additional iterations. However, in English–Farsi tasks, gains are much smaller.

**Comparison to back-translation with Moses.** We now consider the utility of creating synthetic parallel data from different sources, e.g., from a phrase-based SMT models produced by Moses (Koehn et al., 2007), a considerably faster and more scalable system than modern NMT techniques. As can be seen in Table 5.6, this has mixed results, being better for English–French, and worse in English–Farsi, than using neural models, although in all cases the results are not far apart.

**Quality of the sampled monolingual data.** Back-translation is dependent much on the quality of back-translated synthetic data. In our paper, repeating the back-translation process in 2-3 times can lead to improved translation. However, this can be different in other language pairs and domains. Also, in our work, we sampled the monolingual data uniformly at random, so sentences may be used more than once in subsequent rounds. It is quite likely that other techniques for data sampling and selection, e.g., non-uniform sampling like transductive selection or active learning - which potentially diversifies the quality and quantity of monolingual data - would lead further improvements in translation performance. We leave this for our future work.

**5.1.6 Impact of Data Selection Methods**

Let’s recall that we aim to address our second question whether non-random data selection process may have any benefits on back-translation approach. Data selection methods have usually been investigated in the scope of domain adaptation for MT (SMT in particular). The notable methods are from Axelrod et al. (2011); Moore and Lewis (2010) - which proposed to filter potentially bad parallel candidates from out-of-domain parallel data based on the cross entropy difference criterion. We adopted this idea in the context of back-translation approach by marking the monolingual data as out-of-domain and the real parallel data as in-domain, then applying both monolingual and bilingual cross-entropy variants for data filtering.

Specifically, the monolingual cross-entropy difference was proposed by Moore and Lewis (2010) where the cross-entropy is defined as:

\[
\mathcal{H}(P_{LM}) = \frac{1}{l} \sum_{i=1}^{l} \log P_{LM}(x_i|x_1, \ldots, x_{i-1});
\]  

(5.1)
The Use of Monolingual Data for Neural Machine Translation

Table 5.8 Experimental results with data selection methods for back-translation, evaluating English–German with transformer NMT systems. All BLEU scores are with single systems.

<table>
<thead>
<tr>
<th>Setting</th>
<th>newstest2016</th>
<th>newstest2017</th>
</tr>
</thead>
<tbody>
<tr>
<td>bt w/ random</td>
<td>35.27</td>
<td>29.35</td>
</tr>
<tr>
<td>bt w/ XenC-m</td>
<td>31.56</td>
<td>25.48</td>
</tr>
<tr>
<td>bt w/ XenC-b</td>
<td>33.02</td>
<td>27.64</td>
</tr>
<tr>
<td>bt w/ BSE</td>
<td>35.03</td>
<td>28.89</td>
</tr>
</tbody>
</table>

Table 5.8 Experimental results with data selection methods for back-translation, evaluating English–German with transformer NMT systems. All BLEU scores are with single systems.

where $P_{LM}$ is the probability of a language model (LM) for the word sequence $x$ comprising $l$ words. This LM can be formulated as either statistical n-gram LM or neural probabilistic LM (Bengio et al., 2003; Mikolov et al., 2011). In this method, we trained two LMs: $H_O$ from a random subset of the monolingual data (says in German) and $H_I$ from the target side of real parallel data (also in German); and both of them have the same vocabulary. Next, the sentences $s_1, \ldots, s_M$ from monolingual data will then be ranked in a descending order by the monolingual cross-entropy difference defined as $H_I(s_i) - H_O(s_i)$; and the top ranked sentences will be used for back-translation approach.

The second method is based on the bilingual cross-entropy difference as proposed by Axelrod et al. (2011). The idea is to take the bilingual languages into consideration. The monolingual cross-entropy difference for each language is computed as described in Equation 5.1, then the bilingual cross-entropy difference for a given sentence pair $(s_{src}, s_{trg})$ is defined as:

$$\left[ H^{src}_I(s_{src}) - H^{src}_O(s_{src}) \right] + \left[ H^{trg}_I(s_{trg}) - H^{trg}_O(s_{trg}) \right].$$

In this method, we first sample a large monolingual data for back-translating and then select sentences pairs according to the bilingual cross-entropy difference.

Further, we also attempted to evaluate the method based on bilingual sentence embeddings (BSE) adapted from Hassan et al. (2018). The idea behind this method is to train a single bilingual NMT system by incorporating a language identification marker at the beginning of each source sentence, as proposed by Johnson et al. (2017). Given this single system, we can utilise the contextual vector representations of the given source and target sentences and compute their cosine similarity. All sentence pairs produced by back-translation will be ranked and filtered by their similarity scores.

Experiments. We conducted some preliminary experiments to validate our hypothesis on the data selection for back-translation. For monolingual and bilingual cross-entropy
differences (XenC-m and XenC-b, respectively), we used the open source toolkit XenC\(^7\) (Rousseau, 2013). For the BSE method, we used the Marian toolkit (Junczys-Dowmunt et al., 2018) for our experiments. We used the same datasets as well as configurations as in previous sections.

The results are as shown in Table 5.8. Unfortunately, our experiments with data selection were not successful. All the experimented methods performed worse than the random one. The XenC\(^m\) and XenC\(^b\) approaches performed very poorly, and further diminished the robustness of back-translation for NMT. This can be understood by the fact that these data selection methods have preferences to select shorter sentences from the monolingual data. The NMT systems trained on many short sentences can suffer from under-generalisation, e.g., it cannot handle longer sentences seen in the test sets. Also, not as expected, the BSE method also does not provide any advantage over the random method - which is puzzling us. As a result, though being simple, the random method is still very robust, often leading to satisfying results. However, we believe that further investigation should be made to figure out a robust strategy for data selection method for improving the back-translation approach.

5.1.7 Analysis

5.1.7.1 Efficacy on iterative back-translation

The efficiency of the NMT toolkits we used (sockeye, marian) is excellent. Both support batch decoding for fast translation, e.g., with a batch-size of 200 (beam-size 5) Marian can achieve over 5000 words per second on one GPU (less than 1 day for translating 4M sentences)\(^8\); and also this scales linearly to the number of GPUs. Alternatively, we can split the monolingual data into smaller parts and distribute these parts over different GPUs. This can greatly speed up the back-translation process. This leaves the problem of training the model in each iteration, which we do 2-3 times. Overall the computational complexity is not a big deal (even with larger dataset), and the iterative back-translation is quite feasible with existing modern GPU servers.

5.1.7.2 The Insights of Iterative Back-Translation

In this work, we proposed the iterative back-translation strategy and empirically show that it helps to improve the state-of-the-art NMT systems. However, it is also interesting to understand the principled interpretation behind this idea. Concurrent to our work, Cotterell and

\(^7\)https://github.com/antho-rousseau/XenC
\(^8\)https://marian-nmt.github.io/features/
Kreutzer (2018) took one step further and interpreted iterative back-translation as approximate inference in a generative model for learning from bilingual texts. They conceptualised it into the *Wake-Sleep* variational scheme proposed by Hinton et al. (1995). The *Wake-Sleep* scheme explains the iterative back-translation in two phases: the *Wake* phase and the *Sleep* phase. This scheme is in principle similar to our Algorithm 4; however, the *Wake-Sleep* algorithm further defines the implicit language model in order to generate samples in the sleep phase (as in their work (Cotterell and Kreutzer, 2018)) or for back-translation process (as defined in our paper). However, further work may be required to understand and exploit the role of this implicit language model.

Although the iterative back-translation as presented so far works well in practice, it is not a principled approach, requiring offline engineering efforts from practitioners. He et al. (2016a) proposed a novel framework (so-called round tripping)\(^9\) for utilising the monolingual data to improve neural machine translation models. The idea behind their framework is based on a “communication-based” model, where bidirectional (forward and backward) translation models are learned jointly to improve each other by forth- and back-translating monolingual data and then using them as reward signals to reinforce the models using the policy gradient updates. This approach can be regarded as a principled generalisation of iterative back-translation. Although success has been claimed by He et al. (2016a), after many attempts, we could not reproduce their results; hence, further study is needed.

### 5.1.8 Discussion

We have presented the iterative back-translation strategy for improving the existing NMT systems using monolingual data. In order to validate our approach, we performed our experiments in both rich and low resource settings, and showed its effectiveness. We also addressed our doubt about data selection for back-translation. This study will further provide practical advice for practitioners on how to utilise the monolingual data for improving NMT systems.

\(^9\)In this thesis, we call it round tripping instead of dual learning as in their work due to its characteristics.
5.2 Iterative Back-Translation for Unsupervised Domain Adaptation

Algorithm 5 Iterative Back-Translation for Unsupervised Domain Adaptation

**Input:** general-domain parallel data $D_o^p$, in-domain monolingual source, $D_i^s$, and target $D_i^t$ text

1: Get $\bar{T}$ from $D_o^p$
2: Get $\bar{T}$ from $D_o^p$
3: Train source-to-target model $\Theta$ on $\bar{T}$
4: Train target-to-source model $\bar{\Theta}$ on $\bar{T}$
5: **repeat**
6: Use $\bar{\Theta}$ to create $S_i = \{ (\hat{s}, t) \}, \text{for } t \in D_i^t$
7: Fine-tune the source-to-target model $\bar{\Theta}$ on $S_i$ until convergence
8: Use $\bar{\Theta}$ to create $S_i' = \{ (s, \hat{t}) \}, \text{for } s \in D_i^s$
9: Fine-tune the target-to-source model $\bar{\Theta}$ on $S_i'$ until convergence
10: **until** convergence condition reached

**Output:** newly-updated models $\bar{\Theta}$ and $\bar{\Theta}$

5.2 Iterative Back-Translation for Unsupervised Domain Adaptation

5.2.1 Motivation

We apply the back-translation approach to unsupervised domain adaptation for NMT. Unsupervised domain adaptation for NMT is very challenging due to the fact that there is no (or very limited) parallel data in the target domain (also called in-domain). To the best of our knowledge, no similar work in the literature has been done in this direction. In domain adaptation for NMT, there often exists (huge/big) general-domain parallel data, and a little in-domain parallel data. The most popular approaches are fine-tuning in which the NMT system is trained over general-domain data and then is fine-tuned over the in-domain data. More details and relevant references can be read further in this survey (Chu and Wang, 2018).

We go one step further by investigating a practical problem in domain adaptation for NMT where the challenge is that the in-domain data is very limited, hence the fine-tuning approaches cannot be applied. In our work, we assume that we have access to abundant in-domain monolingual data in both directions. Such monolingual data can be very cheap to obtain. One practical example is to develop a domain-specific NMT translation engine for translating multilingual customer/product reviews (e.g., in e-commerce websites such as: Booking.com, Alibaba, Amazon). In this situation, we can imagine that there is no existing

---

\(^{10}\) at least > 100K sentence pairs

\(^{11}\) One may argue that creating more in-domain parallel data for fine-tuning, e.g., by hiring human translators, will be much easier. However, this approach is very expensive and time-consuming.
parallel data for the review domain(s). However, one already has developed NMT engine for translating in general domain (e.g., news) - which exists in WMT campaigns.\footnote{http://www.statmt.org/wmt18/}

5.2.2 Proposed Approach

We apply iterative back-translation for unsupervised domain adaptation. Our approach is straightforward. First, we begin with the existing NMT models in both directions, say English–German and German–English. Also, we are provided with in-domain monolingual data in both English and German, and importantly there does not exist the parallel connection between them (our strict assumption). Then, we use the existing NMT systems to translate the above monolingual data into the respective source and target languages. As a result, we obtain in-domain “pseudo” parallel data which can be used to “fine-tune” the previous NMT models (Freitag and Al-Onaizan, 2016; Luong and Manning, 2015). As mentioned, the starting NMT models are trained based on the general-domain parallel data. This process can be repeated several times as figured out in the idea of iterative back-translation. Note that the convergence of this process could be based on BLEU (Papineni et al., 2002) evaluated on a development set. Overall, we present the algorithm of our proposed approach for unsupervised domain adaptation in Algorithm 5.

5.2.3 Experiments

5.2.3.1 Datasets

In order to validate our approach, we conducted our experiments on a “simulated” setting.\footnote{We believe that our approach can be generalised to a realistic situation if data can be available.} We used the general domain in news with the high-resource data: WMT 2017 German–English. We re-used the existing pre-trained baseline NMT systems from the previous part - which serve as starting models as mentioned in Algorithm 5. We used the technical talks from the TED talks dataset (Cettolo et al., 2012) as our in-domain data. For simulating the domain adaptation scenario, we simply removed the parallel connections of in-domain parallel data, hence obtaining in-domain monolingual data for source and target languages. Table 5.9 shows the detailed statistics of in-domain data used in our experiments.
5.2 Iterative Back-Translation for Unsupervised Domain Adaptation

<table>
<thead>
<tr>
<th>dataset</th>
<th># tokens (K)</th>
<th># types (K)</th>
<th># sents</th>
<th># docs</th>
</tr>
</thead>
<tbody>
<tr>
<td>TED Talks</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>train</td>
<td>4384.68</td>
<td>4161.58</td>
<td>19.42</td>
<td>1597</td>
</tr>
<tr>
<td>tune-tst2010</td>
<td>35.13</td>
<td>33.42</td>
<td>3.29</td>
<td>1565</td>
</tr>
<tr>
<td>test1-tst2014</td>
<td>27.40</td>
<td>26.44</td>
<td>3.21</td>
<td>1305</td>
</tr>
<tr>
<td>test2-tst2015</td>
<td>20.81</td>
<td>19.75</td>
<td>2.55</td>
<td>1080</td>
</tr>
</tbody>
</table>

Table 5.9 Statistics of the training & evaluation sets from (Cettolo et al., 2012) used for our experiments of unsupervised domain adaptation.

5.2.3.2 Baseline and Setups

Regarding the settings for our NMT systems, we used the Transformer model (Vaswani et al., 2017) in all our experiments. For fair comparisons, we provided different baselines systems, e.g.,

IWSLT (base): the simple baseline system reported in the IWSLT competition (Cettolo et al., 2015).

IWSLT (best): the best system reported in the IWSLT competition (Cettolo et al., 2015).

iwslt-domain-only: the NMT system trained only on the IWSLT parallel data.

fine-tune-parallel-data: the baseline NMT system was trained on the WMT parallel data, then was fine-tuned with the IWSLT parallel data. Note that we report two systems (iwslt-domain-only and fine-tune-parallel-data) to know how well the NMT systems perform if the in-domain parallel data exist.

wmt-domain-only: the baseline NMT systems trained only on the WMT parallel data as reported in Section 5.1.

fine-tune-bt+n: our proposed NMT systems enhanced by the iterative back-translation strategy. “+n” refers to the number of iterations for performing the back-translation cycle as shown in Algorithm 5.

5.2.3.3 Results and Analysis

Our experimental results can be found in Table 5.10. First, our baseline NMT systems with the transformer architecture are on par with the best reported IWSLT systems (Cettolo et al., 2014) We believe that the similar pattern of improvements can be undoubtedly obtained with other NMT architectures.
The Use of Monolingual Data for Neural Machine Translation

<table>
<thead>
<tr>
<th></th>
<th><strong>EN→DE</strong></th>
<th></th>
<th><strong>DE→EN</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td><strong>test2014</strong></td>
<td><strong>test2015</strong></td>
<td><strong>test2014</strong></td>
</tr>
<tr>
<td>IWSLT (base)</td>
<td>18.49</td>
<td>20.08</td>
<td>17.99</td>
</tr>
<tr>
<td>IWSLT (best)</td>
<td>27.58</td>
<td>30.85</td>
<td>26.18</td>
</tr>
<tr>
<td>iwslt-domain-only</td>
<td>26.24</td>
<td>28.82</td>
<td>31.49</td>
</tr>
<tr>
<td>fine-tune-parallel-data</td>
<td>29.82</td>
<td>32.38</td>
<td>36.56</td>
</tr>
<tr>
<td>wmt-domain-only</td>
<td>27.31</td>
<td>29.16</td>
<td>30.14</td>
</tr>
<tr>
<td>fine-tune-bt</td>
<td>27.68</td>
<td><strong>30.00</strong></td>
<td><strong>33.46</strong></td>
</tr>
<tr>
<td>fine-tune-ibt+1</td>
<td><strong>28.15</strong>†</td>
<td><strong>30.66</strong>†</td>
<td><strong>33.76</strong></td>
</tr>
<tr>
<td>fine-tune-ibt+2</td>
<td><strong>28.03</strong></td>
<td><strong>30.31</strong></td>
<td><strong>33.96</strong>†</td>
</tr>
</tbody>
</table>

Table 5.10 Evaluation results with BLEU scores for unsupervised domain adaptation setting on the TED Talks dataset. Note that we reported the single transformer-based NMT systems in all numbers. We denoted \( bt \) as back-translation, \( ibt+n \) as iterative back-translation with \( n \) iterations; **bold**: statistically significantly better than the baselines (wmt-domain-only and iwslt-domain-only), †: best result on the dataset.

This again gives us further convincing evidence that the transformer NMT systems worked much better than the RNN-based ones, apparently shown in the line of wmt-domain-only - which was trained without any in-domain parallel data. As expected, the back-translation approach provides large improvements over the baselines, particularly for the DE→EN direction with > 3.5 BLEU scores. For the EN→DE direction, we obtained a similar pattern of BLEU gains, albeit of a smaller magnitude. It is interesting to see that without any parallel data, we are still able to perform domain adaptation with the help of back-translation. Performing the back-translation for more iterations also leads to further improvements, in both directions. This again validates the effectiveness of our proposed iterative back-translation strategy as in our intuition. We obtained the best scores with two iterations of back-translations, and more importantly we are filling the gaps with the performance of the NMT system fine-tuned on the “existing parallel” data (the fine-tune-parallel-data line in Table 5.10). This will be very useful in a realistic situation in which we do not have the access to the in-domain parallel data.

Note that in our experiments, the monolingual sentences in German and English sides are taken from the two sides of the real parallel data. In the future work, it could be more interesting to see what happens if the two monolingual corpora are unrelated.

---

15Here, we only referred to the systems reported in the IWSLT’15 competition.
5.2.4 Discussion

In this part, we have provided further evidence for the effectiveness of our proposed iterative back-translation applied to an unsupervised domain adaptation setting. Although we performed our experiments in a simulated condition; however, we believe that our approach is applicable to realistic situation where the in-domain parallel data do not exist, but the in-domain monolingual data can be accessible.

5.3 Summary

This chapter provides a further study on back-translation on utilising monolingual data for improving the NMT models. We proposed a simple but effective iterative back-translation strategy which improves the performance of existing NMT models in both rich- and low-resourced settings. We also give some additional analyses observed on the behaviors of iterative back-translation. Further, we applied this idea for unsupervised domain adaptation setting for NMT. Overall, this study gives a better understanding of the back-translation approach in different situations.
Chapter 6

Moment Matching Training Framework for Neural Machine Translation

Contents

6.1 Introduction and Motivation .............................................. 132
  6.1.1 Why Moment Matching? ........................................... 133
6.2 Proposed Model .......................................................... 134
  6.2.1 Moment Matching for Sequence to Sequence Models ............. 135
  6.2.2 Formulation of the MM Objective Function ....................... 135
  6.2.3 Derivation of the Moment Matching Gradient ................. 136
  6.2.4 MM training vs CE training vs RL training with Policy Gradient . 138
  6.2.5 Computing the Moment Matching Gradient ..................... 139
  6.2.6 Proof of Unbiasedness ............................................ 141
  6.2.7 Training with the Moment Matching Technique ................. 143
  6.2.8 Connections to Previous Work ................................ 145
6.3 Experiments ............................................................... 146
  6.3.1 Prior Features for NMT ........................................... 146
  6.3.2 Datasets and Baseline ............................................. 147
  6.3.3 Results and Analysis ............................................ 148
6.4 Moment Matching for Unsupervised Domain Adaptation ............ 148
6.5 Summary ................................................................. 150
As shown in previous chapters (e.g., §3-5), neural sequence models have been shown great benefits to prior and external knowledge, e.g., through including such information as features or embeddings as part of the model. In this chapter, we propose a different point of view on how to incorporate prior and external knowledge in a principled way, using a moment matching framework (§6.1). In this approach, the standard local cross-entropy training of the sequential model is combined with a moment matching training mode that encourages the equality of the expectations of certain predefined features between the model distribution and the empirical distribution. In particular, we show how to derive unbiased estimates of some stochastic gradients that are central to the training, and compare our framework with a formally related one: policy gradient training in reinforcement learning, pointing out some important differences in terms of prior assumptions in both approaches (§6.2). Our initial results (§6.3) are promising, showing the effectiveness of our proposed framework.

Parts of this chapter were reported in:


\section{6.1 Introduction and Motivation}

Standard training of neural sequence to sequence (seq2seq) models (as discussed in §2.1) requires the construction of a cross-entropy loss (Lipton, 2015; Sutskever et al., 2014). This loss normally manipulates at the level of generating individual tokens in the target sequence (i.e., sequential predictions), hence, potentially suffering from label or observation bias (Pereyra et al., 2017; Wiseman and Rush, 2016). Thus, it might be difficult for neural seq2seq models to capture the semantics at sequence level. This may be detrimental when the desired generated sequence may be missing or lacking some desired properties, for example, avoiding repetitions, preserving the consistency between source and target length ratio, or satisfying biasedness upon some external evaluation measures such as ROUGE (Lin, 2004) and BLEU (Papineni et al., 2002) in summarisation and translation tasks, respectively; or avoiding omissions and additions of semantic materials in natural language generation; etc. Sequence properties, on the other hand, may be associated with prior knowledge about the sequence the model aims to generate.

The cross-entropy loss with sequence constraints is intractable. In order to inject such prior knowledge into seq2seq models, methods in reinforcement learning (RL) (Sutton and Barto, 1998) emerge as reasonable choices. In principle, RL is a general-purpose frame-
work applying for sequential decision making processes. In RL, an agent interacts with an environment $\mathcal{E}$ over a certain number of discrete timesteps (Sutton and Barto, 1998). The ultimate goal of the agent is to select any action according to a policy $\pi$ that maximises a future cumulative reward. This reward is the objective function of RL guided by the policy $\pi$, and is defined specifically for the application task. Considering sequential prediction problems in our case, an action of choosing the next word prediction is guided by a stochastic policy and receives a sequence-specific reward with a real value return via observations (Ross et al., 2011). The agent tries to maximise the expected reward for $T$ timesteps, e.g., $\mathcal{R} = \sum_{t=1}^{T} r_t$.

The idea of RL has recently been applied to a variety of neural seq2seq tasks. Ranzato et al. (2015) applied this idea to abstractive summarisation with neural seq2seq models, using the ROUGE evaluation measure (Lin, 2004) as a reward. Similarly, some success also has been achieved for neural machine translation. For instance, Ranzato et al. (2015) and He et al. (2017) used BLEU score (Papineni et al., 2002) as a reward function in their RL setups whereas He et al. (2016a) used a reward interpolating the probabilistic scores from reverse translation and language models. Edunov et al. (2018b) revisited different kinds of structured losses and discussed how the RL framework can be applied to sequence to sequence learning tasks, particularly in summarisation and translation.

### 6.1.1 Why Moment Matching?

The main motivation of moment matching (MM) is to inject prior knowledge into the model which takes the properties of whole sequences into consideration. We aim to develop a generic method that is applicable for any seq2seq models.

Inspired from the method of moments in statistics,\footnote{https://en.wikipedia.org/wiki/Method_of_moments_(statistics)} we propose the following moment matching approach. The underlying idea of moment matching is to seek optimal parameters reconciling two distributions, namely: one from the samples generated by the model and another one from the empirical data. Those distributions aim to evaluate the generated sequences as a whole, via the use of feature functions or constraints that one would like to behave similarly between the two distributions, based on the encoding of the prior knowledge about sequences. Note that this proposed moment matching technique is not stand-alone, but to be used in alternation or combination with standard cross-entropy training. This is similar to the way RL is typically applied in seq2seq models (Ranzato et al., 2015).

Here, we will discuss some important differences with RL, then we will present the details on how the MM technique works in the next sections.
The first difference is that RL assumes that one has defined some reward function $\mathcal{R}$, which is done quite independently of what the training data tells us. By contrast, MM only assumes that one has defined certain features that are deemed important for the task, but one then relies on the actual training data to tell us how to use these features. One could say that the “arbitrariness” in MM is just in the choice of the features to focus on, while the arbitrariness in RL is that we want the model to get a good reward, even if that reward is not connected to the training data at all.

Suppose that we are in the context of NLG and are trying to reconcile several objectives at the same time, such as (1) avoiding omissions of semantic material, (2) avoiding additions of semantic material, (3) avoiding repetitions (Agarwal and Dymetman, 2017). In general, in order to address this kind of problem in an RL framework, we need to “invent” a reward function based on certain computable features of the model outputs which in particular means inventing a formula for combining the different objectives we have in mind into a single real number. This can be a rather arbitrary process, and potentially it does not guarantee any fit with actual training data. The point of MM is that the only arbitrariness is in choosing the features to focus on, but after that it is actual training data that tells us what should be done.

The second difference is that RL tries to maximize a reward, and is only sensitive to the rewards of individual instances, while MM tries to maximize the fit of the model distribution with that of the empirical distribution, where the fit is on specific features.

For instance, this difference is especially clear in the case of language modelling where RL will try to find a model that is strongly peaked on the $x$ which has the strongest reward (assuming no ties in the rewards), while MM will try to find a distribution over $x$ which has certain properties in common with the empirical distribution, e.g., for generating diverse outputs. For language modelling, RL is a strange method, because language modelling requires the model to be able to produce different outputs; for MT, the situation is a bit less clear, in case one wanted to argue that for each source sentence, there is a single best translation; but in principle, the observation also holds for MT, which is a conditional language model.

6.2 Proposed Model

In this section, we will describe our formulation of moment matching for seq2seq modelling in detail.
6.2 Proposed Model

6.2.1 Moment Matching for Sequence to Sequence Models

Recall the sequence-to-sequence problem whose goal is to generate an output sequence given an input sequence. In the context of neural machine translation – which is our main focus here, the input sequence is a source language sentence, and the output sequence is a target language sentence.

Suppose that we are modelling the target sequence \( y = y_1, \ldots, y_t, \ldots, y_{|y|} \) given a source sequence \( x = x_1, \ldots, x_t, \ldots, x_{|x|} \), using a sequential process \( p_\Theta(y|x) \). This sequential process can be implemented via a neural mechanism, e.g., recurrent neural networks within an (attentional) encoder - decoder framework (Bahdanau et al., 2015) or a transformer framework (Vaswani et al., 2017) as discussed earlier in Chapter 2.1. Regardless of its implementation, such a neural mechanism depends on model parameters \( \Theta \).

Our proposal is that we would like our sequential process to satisfy some moment constraints. Such moment constraints can be modeled based on features that encode prior (or external) knowledge or semantics about the generated target sentence. Mathematically, features can be represented through vectors, e.g., \( \Phi(y|x) \equiv (\phi_1(y|x), \ldots, \phi_j(y|x), \ldots, \phi_m(y|x)) \), where \( \phi_j(y|x) \) is the \( j^{th} \) conditional feature function of a target sequence \( y \) given a source sequence \( x \), and \( m \) is number of features or moment constraints. Considering a simple example where the moment feature is for controlling the length of a target sequence - which would just return a number of elements in that target sequence.

6.2.2 Formulation of the MM Objective Function

In order to incorporate such constraints into the seq2seq learning process, we introduce a new objective function, namely the moment matching loss \( J_{MM} \). Generally speaking, given a vector of features \( \Phi(y|x) \), the goal of moment matching loss is to encourage the identity of the model average estimate,

\[
\hat{\Phi}_n(\Theta) \equiv \mathbb{E}_{y \sim p_\Theta(\cdot|x_n)} \left[ \Phi \left( y | x_n \right) \right]
\]

with the empirical average estimate,

\[
\bar{\Phi}_n \equiv \mathbb{E}_{y \sim p_D(\cdot|x_n)} \left[ \Phi \left( y | x_n \right) \right]; \quad (6.1)
\]

where \( D \) is the training data; \( x, y \in D \) are source and target sequences, respectively; \( n \) is the data index in \( D \). This can be formulated as minimising a squared distance between the
two distributions with respect to model parameters $\Theta$:

$$
\mathcal{J}_{MM} (\Theta) = \frac{1}{N} \sum_{n=1}^{N} \| \hat{\Phi}_n(\Theta) - \bar{\Phi}_n \|^2_2 \\
= \frac{1}{N} \sum_{n=1}^{N} \left\| \mathbb{E}_{y \sim p_\theta(.|x_n)} [\Phi(y|x_n)] - \mathbb{E}_{y \sim p_\mathcal{D}(.|x_n)} [\Phi(y|x_n)] \right\|^2_2 .
$$

(6.2)

To be more elaborate, $\hat{\Phi}_n(\Theta) \equiv \mathbb{E}_{y \sim p_\theta(.|x_n)} [\Phi(y|x_n)]$ is the model average estimate over the samples which are drawn i.i.d. from the model distribution $p_\theta(.)$ given the source sequence $x_n$, and $\bar{\Phi}_n \equiv \mathbb{E}_{y \sim p_\mathcal{D}(.|x_n)} [\Phi(y|x_n)]$ is the empirical average estimate given the $n^{th}$ training instance, where our data are drawn i.i.d. from the empirical distribution $p_\mathcal{D}(.|x)$.

### 6.2.3 Derivation of the Moment Matching Gradient

We now show how to compute the gradient of $\mathcal{J}_{MM}$ in the equation 6.2, denoted as $\nabla_\Theta \mathcal{J}_{MM}$, which will be required in optimisation. We first define:

$$
\Gamma_{\Theta,n} \equiv \nabla_\Theta \left( \| \Delta_n \|^2_2 \right) ,
$$

where $\Delta_n \equiv \Phi_n(\Theta) - \bar{\Phi}_n$, then the gradient $\nabla_\Theta \mathcal{J}_{MM}$ can be computed as:

$$
\nabla_\Theta \mathcal{J}_{MM} = \frac{1}{N} \sum_n \Gamma_{\Theta,n} .
$$

(6.3)

Next, we need to proceed with the computation of $\Gamma_{\Theta,n}$. By derivation, we have the following:

$$
\Gamma_{\Theta,n} = 2 \sum_y p_\theta(y|x_n) \langle \hat{\Phi}_n(\Theta) - \bar{\Phi}_n, \Phi(y|x_n) - \bar{\Phi}_n \rangle \nabla_\Theta \log p_\theta(y|x_n) \\
= 2 \mathbb{E}_{y \sim p_\theta(.|x_n)} [\langle \hat{\Phi}_n(\Theta) - \bar{\Phi}_n, \Phi(y|x_n) - \bar{\Phi}_n \rangle \nabla_\Theta \log p_\theta(y|x_n)] .
$$

(6.4)

**Proof.** Mathematically, we can say that $\Gamma_{\Theta,n}$ is the gradient of the composition $F \circ G$ of two functions $F(.) = \| . \|^2_2 : \mathbb{R}^m \to \mathbb{R}$ and $G(.) = \Phi_n(.) - \bar{\Phi}_n : \mathbb{R}^{\Theta} \to \mathbb{R}^m$.

Noting that the gradient $\nabla_\Theta (F \circ G)$ is equal to the Jacobian $\mathcal{J}_{F \circ G}[\Theta]$, and applying the chain rule for Jacobians, we have:

$$
\mathcal{J}_{F \circ G}[\Theta] = \left( \nabla_\Theta \| \hat{\Phi}_n(\Theta) - \bar{\Phi}_n \|^2_2 \right) [\Theta] = \mathcal{J}_F [G(\Theta)] \cdot \mathcal{J}_G [\Theta]
$$

(6.5)
Next, we need the computation for $\mathcal{F}_F[G(\Theta)]$ and $\mathcal{F}_G[\Theta]$ in Equation 6.5. First, we have:

$$
\mathcal{F}_F[G(\Theta)] = 2 \left( \hat{\Phi}_n(\Theta) - \bar{\Phi}_n \right)
= 2 \left( \hat{\Phi}_{n,1}(\Theta) - \bar{\Phi}_{n,1}, \ldots, \hat{\Phi}_{n,j}(\Theta) - \bar{\Phi}_{n,j}, \ldots, \hat{\Phi}_{n,m}(\Theta) - \bar{\Phi}_{n,m} \right),
$$

(6.6)

where $\hat{\Phi}_n(\Theta)$ and $\bar{\Phi}_n$ are vectors of size $m$. And we also have:

$$
\mathcal{F}_G[\Theta] = \left[
\frac{\partial \mathbb{E}_{y \sim p_{\Theta}(.|x_n)}[\phi_j(y|x_n) - \bar{\phi}_{n,j}]}{\partial \Theta_i} \ldots \frac{\partial \mathbb{E}_{y \sim p_{\Theta}(.|x_n)}[\phi_1(y|x_n) - \bar{\phi}_{n,1}]}{\partial \Theta_i}
\ldots \frac{\partial \mathbb{E}_{y \sim p_{\Theta}(.|x_n)}[\phi_m(y|x_n) - \bar{\phi}_{n,m}]}{\partial \Theta_i}
\right] = \mathcal{M}
$$

(6.7)

A key part of these identities in Equation 6.7 is the value of $\frac{\partial \mathbb{E}_{y \sim p_{\Theta}(.|x_n)}[\phi_j(y|x_n) - \bar{\phi}_{n,j}]}{\partial \Theta_i}$ which can be expressed as:

$$
\mathcal{M}_{ji} = \frac{\partial \mathbb{E}_{y \sim p_{\Theta}(.|x_n)}[\phi_j(y|x_n) - \bar{\phi}_{n,j}]}{\partial \Theta_i} = \frac{\partial \sum_y p_{\Theta}(y|x_n) (\phi_j(y|x_n) - \bar{\phi}_{n,j})}{\partial \Theta_i}
= \sum_y (\phi_j(y|x_n) - \bar{\phi}_{n,j}) \frac{\partial p_{\Theta}(y|x_n)}{\partial \Theta_i}
$$

(6.8)

Next, using the well-known “log-derivative trick”:

$$
p_{\Theta}(y|x) \frac{\partial \log p_{\Theta}(y|x)}{\partial \Theta_i} = \frac{\partial p_{\Theta}(y|x)}{\partial \Theta_i}
$$

from the Policy Gradient technique in reinforcement learning (Sutton et al., 2000), we can rewrite the equation 6.8 as follows:

$$
\mathcal{M}_{ji} = \sum_y (\phi_j(y|x_n) - \bar{\phi}_{n,j}) p_{\Theta}(y|x_n) \frac{\partial \log p_{\Theta}(y|x_n)}{\partial \Theta_i}
= \mathbb{E}_{y \sim p_{\Theta}(.|x_n)} \left[ (\phi_j(y|x_n) - \bar{\phi}_{n,j}) \frac{\partial \log p_{\Theta}(y|x_n)}{\partial \Theta_i} \right].
$$

(6.9)

Combining Equations 6.8, 6.9, we have:

$$
\frac{\partial \mathbb{E}_{y \sim p_{\Theta}(.|x_n)}[\phi_j(y|x_n) - \bar{\phi}_{n,j}]}{\partial \Theta_i} = \mathbb{E}_{y \sim p_{\Theta}(.|x_n)} \left[ (\phi_j(y|x_n) - \bar{\phi}_{n,j}) \frac{\partial \log p_{\Theta}(y|x_n)}{\partial \Theta_i} \right],
$$
so in turn we obtain the computation of $\mathcal{J}_G [\Theta]$. The expectation $\mathbb{E}_{y \sim p_\Theta (\cdot | x_n)}$ is easy to sample and the gradient $\frac{\partial \log p_\Theta (y | x_n)}{\partial \theta_j}$ is easy to evaluate as well. Also note that the explicit probability allows an intractable sum to be effectively approximated via sampling.

Since we already have the computations of $\mathcal{J}_F [G (\Theta)]$ and $\mathcal{J}_G [\Theta]$, we can finalise the gradient computation $\Gamma_{\Theta, n}$ as follows:

$$
\Gamma_{\Theta, n} = 2 \left( \hat{\phi}_{n, 1} (\Theta) - \bar{\phi}_{n, 1}, \ldots, \hat{\phi}_{n, j} (\Theta) - \bar{\phi}_{n, j}, \ldots, \hat{\phi}_{n, m} (\Theta) - \bar{\phi}_{n, m} \right)
$$

$$
\cdot \mathbb{E}_{y \sim p_\Theta (\cdot | x_n)} \left[ \left( \phi_j (y | x_n) - \tilde{\phi}_{n, j} \right) \frac{\partial \log p_\Theta (y | x_n)}{\partial \theta_j} \right]
$$

$$
= 2 \left( \hat{\phi}_{n, 1} (\Theta) - \bar{\phi}_{n, 1}, \ldots, \hat{\phi}_{n, j} (\Theta) - \bar{\phi}_{n, j}, \ldots, \hat{\phi}_{n, m} (\Theta) - \bar{\phi}_{n, m} \right)
$$

$$
\cdot \sum_y p_\Theta (y | x_n) \left( \left( \phi_j (y | x_n) - \tilde{\phi}_{n, j} \right) \frac{\partial \log p_\Theta (y | x_n)}{\partial \theta_j} \right)
$$

$$
= 2 \sum_y p_\Theta (y | x_n) \left( \hat{\phi}_{n, 1} (\Theta) - \bar{\phi}_{n, 1}, \ldots, \hat{\phi}_{n, j} (\Theta) - \bar{\phi}_{n, j}, \ldots, \hat{\phi}_{n, m} (\Theta) - \bar{\phi}_{n, m} \right)
$$

$$
\cdot \left( \left( \phi_j (y | x_n) - \tilde{\phi}_{n, j} \right) \frac{\partial \log p_\Theta (y | x_n)}{\partial \theta_j} \right)
$$

$$
= 2 \sum_y p_\Theta (y | x_n) \left( \hat{\Phi}_n (\Theta) - \bar{\Phi}_n, \Phi (y | x_n) - \tilde{\Phi}_n \right) \nabla_\theta \log p_\Theta (y | x_n)
$$

$$
= 2 \mathbb{E}_{y \sim p_\Theta (\cdot | x_n)} \left[ \left( \hat{\Phi}_n (\Theta) - \bar{\Phi}_n, \Phi (y | x_n) - \tilde{\Phi}_n \right) \nabla_\theta \log p_\Theta (y | x_n) \right].
$$

By the reasoning just made, we can obtain the computation of $\Gamma_{\Theta, n}$ which is the central formula of the proposed moment matching technique.

### 6.2.4 MM training vs CE training vs RL training with Policy Gradient

Based on equation 6.4, and ignoring the constant factor, we can use as our gradient update, for each pair $(x_n, y_j \sim p_\Theta (\cdot | x_n)) (j \in [1, J])$ the value

$$
\left( \hat{\Phi}_{n, y} (\Theta) - \bar{\Phi}_{n}, \Phi (y | x_n) - \tilde{\Phi}_n \right) \nabla_\theta \log p_\Theta (y | x_n),
$$

| multiplicative score | standard gradient update |
6.2 Proposed Model

<table>
<thead>
<tr>
<th>Method</th>
<th>Formulation</th>
<th>Note</th>
</tr>
</thead>
<tbody>
<tr>
<td>CE</td>
<td>$\nabla_\theta \log p_\theta (y)$</td>
<td>$y \sim \mathcal{D}$</td>
</tr>
<tr>
<td>RL w/ PG</td>
<td>$\mathcal{R} (y) \nabla_\theta \log p_\theta (y)$</td>
<td>$y \sim p_\theta (.)$</td>
</tr>
<tr>
<td>MM</td>
<td>$\langle \hat{\phi} (\theta) - \Phi (y) - \hat{\phi} \rangle \nabla_\theta \log p_\theta (y)$</td>
<td>$y \sim p_\theta (.)$</td>
</tr>
</tbody>
</table>

Unconditional Case

<table>
<thead>
<tr>
<th>Method</th>
<th>Formulation</th>
<th>Note</th>
</tr>
</thead>
<tbody>
<tr>
<td>CE</td>
<td>$\nabla_\theta \log p_\theta (y</td>
<td>x)$</td>
</tr>
<tr>
<td>RL w/ PG</td>
<td>$\mathcal{R} (y</td>
<td>x) \nabla_\theta \log p_\theta (y</td>
</tr>
<tr>
<td>MM</td>
<td>$\langle \hat{\phi} (\theta) - \Phi (y</td>
<td>x) - \hat{\phi} \rangle \nabla_\theta \log p_\theta (y</td>
</tr>
</tbody>
</table>

Conditional Case

Table 6.1 Comparing different methods for training seq2seq models. Note that we denote CE for cross-entropy, RL for reinforcement learning, PG as policy gradient, and MM for moment matching.

where $\hat{\phi}_n$, the empirical average of $\Phi (\cdot | x_n)$, can be estimated through the observed value $y_n$, i.e. $\hat{\phi}_n \approx \Phi (y_n | x_n)$.

Note that the above gradient update draws a very close connection to RL with policy gradient method (Sutton et al., 2000) where the “multiplication score” plays a similar role to the reward $\mathcal{R} (y | x)$; however, unlike RL training using a predefined reward which is identical to expected risk minimisation if considering $\mathcal{R} (y | x)$ to be a cost function, the major difference in MM training is that MM’s multiplication score does depend on the model parameters $\Theta$ and looks at what the empirical data tells the model via using explicit prior features. Table 6.1 compares the differences among three methods, namely CE, RL with Policy Gradient (PG) and our proposal MM, for neural seq2seq models in both unconditional (e.g., language modelling) and conditional (e.g., NMT, summarisation) cases.

6.2.5 Computing the Moment Matching Gradient

We have derived the gradient of moment of matching loss as shown in Equation 6.4. In order to compute it, we still need to have the evaluation of two estimates, namely the model average estimate $\hat{\phi}_n (\Theta)$ and the empirical average estimate $\hat{\phi}_n$.

**Empirical Average Estimate.** First, we need to estimate the empirical average $\hat{\phi}_n$. In the general case, given a source sequence $x_n$, suppose there are multiple target sequences $y \in \mathcal{Y}$ associated with $x_n$, then $\hat{\phi}_n \equiv 1/|\mathcal{Y}| \sum_{y \in \mathcal{Y}} \phi (y | x_n)$. Specifically, when we have only one reference sequence $y$ per source sequence $x_n$, then $\hat{\phi}_n \equiv \phi (y | x_n)$ — which is the standard case in the context of neural machine translation training.
Model Average Estimate. In practice, it is impossible to obtain a full computation of $\Gamma_{\Theta,n}$ due to intractable search of $y$. Therefore, we resort to estimate $\Gamma_{\Theta,n}$ by a sampling process. There are possible options for doing this.

The simplistic approach to that would be to:

First, estimate the model average $\hat{\Phi}_n(\Theta)$ by sampling $y^{(1)}, y^{(2)}, \ldots, y^{(K)}$ and then estimating:

$$\hat{\Phi}_n(\Theta) \approx \frac{1}{K} \sum_{k \in [1,K]} \Phi(y^{(k)}|x_n).$$

Next, estimate the expectation $\mathbb{E}_{y \sim p_\Theta}$ in Equation 6.4 by independently sampling $J$ second values of $y$, and then estimate:

$$\Gamma_{\Theta,n} \approx \frac{1}{J} \sum_{j \in [1,J]} \langle \hat{\Phi}_n(\Theta) - \tilde{\Phi}_n, \Phi(y^{(j)}|x_n) - \tilde{\Phi}_n \rangle \nabla_\Theta \log p_\Theta(y^{(j)}|x_n).$$

Note that two sets of samples are separate and $J \neq K$. This would provide an unbiased estimate of $\Gamma_{\Theta,n}$, but at the cost of producing two independent sample sets of sizes $K$ and $J$, used for two different purposes — which would be computationally wasteful.

A more economical approach might consist in using the same sample set of size $J$ for both purposes. However, this would produce a biased estimate of $\Gamma_{\Theta,n}$. This can be illustrated by considering the estimate case with $J = 1$. In this case, the dot product $\langle \hat{\Phi}_n(\Theta) - \tilde{\Phi}_n, \Phi(y^{(1)}|x_n) - \tilde{\Phi}_n \rangle$ in $\Gamma_{\Theta,n}$ is strictly positive since the value of $\hat{\Phi}_n(\Theta)$ is equal to $\Phi(y^{(1)}|x_n)$; hence, in this case, the current sample $y^{(1)}$ to be systematically discouraged by the model.

Here, we proposed a better approach, resulting in an unbiased estimate of $\Gamma_{\Theta,n}$, formulated as follows:

First, we sample $J$ values of $y$: $y^{(1)}, y^{(2)}, \ldots, y^{(J)}$ with $J \geq 2$, then:

$$\Gamma_{\Theta,n} \approx \frac{1}{J} \sum_{j \in [1,J]} \langle \hat{\Phi}_{n,j}(\Theta) - \tilde{\Phi}_n, \Phi(y^{(j)}|x_n) - \tilde{\Phi}_n \rangle \nabla_\Theta \log p_\Theta(y^{(j)}|x_n),$$

where

$$\hat{\Phi}_{n,j}(\Theta) \equiv \frac{1}{J-1} \sum_{j' \in [1,J], j' \neq j} \Phi(y^{(j')}|x_n).$$

We can then prove that this computation provides an unbiased estimate of $\Gamma_{\Theta,n}$ (see §6.2.6). Note that here, we have exploited the same $J$ samples for both purposes but have
taken care of not exploiting the exact same $y^{(j)}$ for both — akin to a Jackknife resampling estimator.\footnote{\url{https://en.wikipedia.org/wiki/Jackknife_resampling}}

### 6.2.6 Proof of Unbiasedness

For simplicity, we will consider here the “unconditional case”, e.g., $p(.)$ instead of $p(.|x)$, but the conditional case follows easily.

**Lemma 6.2.1.** Let $p(.)$ be a probability distribution over $y$, and let $\zeta(y)$ be any function of $y$ and $\Phi(y)$ is a feature vector over $y$.

We wish to compute the quantity $\mathcal{A} = \mathbb{E}_{y \sim p(.)} [\langle \hat{\Phi}, \Phi(y) \rangle \zeta(y)]$, where $\hat{\Phi} \equiv \mathbb{E}_{y \sim p(.)} [\Phi(y)]$.

Let us sample $J$ sequences $y^{(1)}, \ldots, y^{(J)}$ (where $J$ is a pre-defined number of generated samples) independently from $p(.)$, and let us compute:

$$\mathcal{B}(y^{(1)}, \ldots, y^{(J)}) = \frac{1}{J} \left[ \langle \hat{\Phi}^{(-1)}, \Phi(y^{(1)}) \rangle \zeta(y^{(1)}) + \ldots + \langle \hat{\Phi}^{(-J)}, \Phi(y^{(J)}) \rangle \zeta(y^{(J)}) \right].$$

where $\hat{\Phi}^{(-i)}$ is formulated as:

$$\hat{\Phi}^{(-i)} = \frac{1}{J-1} \left[ \Phi(y^{(1)}) + \ldots + \Phi(y^{(i-1)}) + \Phi(y^{(i+1)}) + \ldots + \Phi(y^{(J)}) \right].$$

Then we have:

$$\mathcal{A} = \mathbb{E}_{\{y^{(i)}\}_{i=1}^{J} \sim p(.)} [\mathcal{B}(y^{(1)}, \ldots, y^{(J)})],$$

in the other words, $\mathcal{B}(y_1, \ldots, y_J)$ provides an unbiased estimate of $\mathcal{A}$.

**Proof.** Let us define:

$$\mathcal{E} = \mathbb{E}_{\{y^{(i)}\}_{i=1}^{J} \sim p(.)} [\mathcal{B}(y^{(1)}, \ldots, y^{(J)})]$$

$$= \sum_{y^{(1)}, \ldots, y^{(J)}} p(y^{(1)}) \ldots p(y^{(J)}) \frac{1}{J} \sum_{i=1}^{J} \left[ \langle \hat{\Phi}^{(-i)}, \Phi(y^{(i)}) \rangle \zeta(y^{(i)}) \right]$$

$$= \frac{1}{J} \sum_{i=1}^{J} \sum_{y^{(1)}, \ldots, y^{(J)}} p(y^{(1)}) \ldots p(y^{(J)}) \langle \hat{\Phi}^{(-i)}, \Phi(y^{(i)}) \rangle \zeta(y^{(i)})$$

$$= \frac{1}{J} \sum_{i=1}^{J} \mathcal{B}(y^{(1)}, \ldots, y^{(J)}) = \mathcal{A} \quad (6.12)$$
For a given value of $i$, we have:

$$
\sum_{y^{(1)}, \ldots, y^{(J)}} p(y^{(1)}) \cdots p(y^{(J)}) \left( \Phi(y^{(i)}) , \Phi(y^{(i)}) \right) \zeta(y^{(i)})
$$

$$
= \sum_{y^{(i)}} p(y^{(i)}) \zeta(y^{(i)}) \sum_{y^{(1)}, y^{(i-1)}, y^{(i+1)}, \ldots, y^{(J)}} p(y^{(1)}) \cdots p(y^{(i-1)}) p(y^{(i+1)}) \cdots p(y^{(J)}) \frac{1}{J-1} 
\langle \left[ \Phi(y^{(1)}) + \cdots + \Phi(y^{(i-1)}) + \Phi(y^{(i+1)}) + \cdots + \Phi(y^{(J)}) \right], \Phi(y^{(i)}) \rangle
$$

$$
= \sum_{y^{(i)}} p(y^{(i)}) \zeta(y^{(i)}) \frac{1}{J-1} \left[ \sum_{y^{(1)}, y^{(i-1)}, y^{(i+1)}, \ldots, y^{(J)}} p(y^{(1)}) \cdots p(y^{(i-1)}) p(y^{(i+1)}) \cdots p(y^{(J)}) \langle \Phi(y^{(1)}) , \Phi(y^{(i)}) \rangle + \right.
\cdots \right.
$$

$$
+ \sum_{y^{(i-1)}} p(y^{(i-1)}) \langle \Phi(y^{(i-1)}) , \Phi(y^{(i)}) \rangle 
$$

$$
+ \sum_{y^{(i+1)}} p(y^{(i+1)}) \langle \Phi(y^{(i+1)}) , \Phi(y^{(i)}) \rangle 
\cdots 
+ \sum_{y^{(J)}} p(y^{(J)}) \langle \Phi(y^{(J)}) , \Phi(y^{(i)}) \rangle 
\right]
$$

$$
= \sum_{y^{(i)}} p(y^{(i)}) \zeta(y^{(i)}) \frac{1}{J-1} \left[ \sum_{y^{(i)}} p(y^{(i)}) \langle \Phi(y^{(1)}) , \Phi(y^{(i)}) \rangle + \cdots + \langle \Phi(y^{(i)}) \rangle \right] 
$$

$$
= \sum_{y^{(i)}} p(y^{(i)}) \zeta(y^{(i)}) \langle ̂\Phi, \Phi(y^{(i)}) \rangle
$$

$$
= \mathbb{E}_{y \sim p(.)} \langle ̂\Phi, \Phi(y) \rangle \zeta(y).
$$

(6.13)
Finally, by collecting the results for the $J$ values of the index $i$, we obtain:

$$
\mathcal{E} = \frac{1}{J} \cdot J \cdot \mathbb{E}_{y \sim p(y)} \langle \hat{\Phi}, \Phi(y) \rangle \zeta(y)
$$

$$
= \mathbb{E}_{y \sim p(y)} \langle \hat{\Phi}, \Phi(y) \rangle \zeta(y)
$$

$$
= \mathcal{A}.
$$

(6.14)

To ground this lemma in our problem setting, consider the case where $p = p_{\Theta}$, and $\zeta(y) = \nabla_{\Theta} \log p_{\Theta}(y)$, then the quantity $\mathcal{A} = \mathbb{E}_{y \sim p(y)} \langle \hat{\Phi}, \Phi(y) \rangle \zeta(y)$ is equal to the overall gradient of the MM loss, for a given value of the model parameters $\Theta$ (by the formula (6.4) obtained earlier, and up to a constant factor). We would like to obtain an unbiased stochastic gradient estimator of this gradient, in other words, we want to obtain an unbiased estimator of the quantity $\mathcal{A}$.

By Lemma 6.2.1, $\mathcal{A}$ is equal to the expectation of $\mathcal{B} (y^{(1)}, \ldots, y^{(J)})$, where $y^{(1)}, \ldots, y^{(J)}$ are drawn i.i.d from distribution $p$. In other words, if we sample one set of $J$ samples from $p$, and compute $\mathcal{B} (y^{(1)}, \ldots, y^{(J)})$, where $y^{(1)}, \ldots, y^{(J)}$ on this set, then we obtain an unbiased estimate of $\mathcal{A}$. As a result, we obtain an unbiased estimate of the gradient of the overall MM loss, which is exactly what we need.

In principle, therefore, we need to first sample $y^{(1)}, \ldots, y^{(J)}$, and to compute

$$
\mathcal{B} (y^{(1)}, \ldots, y^{(J)})
$$

$$
\equiv \frac{1}{J} \left[ \langle \hat{\Phi}^{(-1)}, \Phi(y^{(1)}) \rangle \nabla_{\Theta} \log p_{\Theta}(y^{(1)}) + \ldots + \langle \hat{\Phi}^{(-J)}, \Phi(y^{(J)}) \rangle \nabla_{\Theta} \log p_{\Theta}(y^{(J)}) \right].
$$

and then use this quantity as our stochastic gradient. In practice, what we do is to first sample $y^{(1)}, \ldots, y^{(J)}$, and then use the components of the sum:

$$
\langle \hat{\Phi}^{(-j)}, \Phi(y^{(j)}) \rangle \nabla_{\Theta} \log p_{\Theta}(y^{(j)})
$$

as our individual stochastic gradients. Note that this computation differs from the original one by a constant factor $\frac{1}{J}$, which can be accounted for by manipulating the learning rate.

### 6.2.7 Training with the Moment Matching Technique

Recall the goal of our technique is to preserve certain aspects of generated target sequences according to prior knowledge. In principle, the technique does not teach the model how to generate a proper target sequence based on the given source sequence (Ranzato et al., 2015).
Algorithm 6: General Algorithm for Training with Moment Matching Technique

1: **Input**: a pre-trained model $\Theta$; parallel training data $\mathcal{D}$; $\lambda$ is balancing factor in interpolation training mode if used; a maximum number of steps $M$.

2: **for** step = 1, …, M **do**

3: Select a batch of size $N$ source and target sequences in $X$ and $Y$ in $\mathcal{D}$.

4: **if** MM mode is required **then**

5: Sample $J$ translations for the batch of source sequences $X$.

6: Compute the MM gradient value $\Gamma_{\Theta,n}^{MM}$ in Equations 6.10, 6.11.

7: **if** alternation mode **then**

8: **if** MM mode **then**

9: Update model parameters using the defined MM gradients $\Gamma_{\Theta,n}^{MM}$ with SGD.

10: **else if** CE mode **then**

11: Update model parameters using standard CE based gradients $\Gamma_{\Theta,n}^{CE}$ with SGD.

12: **else** (interpolation mode)

13: Compute the standard CE based gradients $\Gamma_{\Theta,n}^{CE} \equiv \mathbb{E}_{x \sim X, y \sim Y} [\nabla_\Theta \log p_\Theta(y|x)]$.

14: Update model parameters according to $\Gamma_{\Theta,n}^{interpolation} \equiv \Gamma_{\Theta,n}^{CE} + \lambda \Gamma_{\Theta,n}^{MM}$.

15: After some steps, save model parameters using best score $\mathcal{J}_{MM}^{dev}$ in Equation 6.15.

16: **return** newly-trained model $\Theta_{new}$.

For that reason, it has to be used along with standard CE training of seq2seq model. In order to train the seq2seq model with the proposed technique, we suggest to use one of two training modes: alternation and interpolation. For the alternation mode, the seq2seq model is trained alternatively using both CE loss and moment matching loss. More specifically, the seq2seq model is initially trained with CE loss for some iterations, then switches to using moment matching loss; and vice versa. For the interpolation mode, the model will be trained with the interpolated objective using two losses with an additional hyper-parameter balancing them. In summary, the general technique can be described as in Algorithm 6.

After some iterations of the algorithm, we can approximate $\mathcal{L}_{MM}$ over the development data (or sampled training data) through:

$$\mathcal{J}_{MM}^{dev}(\Theta) \approx \frac{1}{N} \sum_{n=1}^{N} \| \hat{\Phi}_n^{approx}(\Theta) - \tilde{\Phi}_n \|_2^2. \quad (6.15)$$

We expect $\mathcal{L}_{MM}$ to decrease over iterations, potentially improving the explicit evaluation measure(s), e.g., BLEU (Papineni et al., 2002) in NMT.
6.2 Proposed Model

6.2.8 Connections to Previous Work

**Maximum Mean Discrepancies (MMD).** The method of moment matching draws a close connection to Maximum Mean Discrepancies. The MMD is a way to measuring discrepancy between two distributions (the empirical distribution and the model distribution) which is a generalisation of the distance we are using: using arbitrary kernels instead of the simple linear kernel (implicitly in our MM formulation). The use of such more general kernels could potentially be useful in the long term to refine our MM technique. Note that MMD has been successfully applied to computer vision, e.g., an alternative to learning generative adversarial network (Li et al., 2017, 2015b).

**Adversarial Training.** Also note that in our proposed technique, exploiting a feature function bears some analogy to stipulating an *a priori* discriminator in an adversarial training regime such as in GANs (Goodfellow et al., 2014; Li et al., 2017).

**The Method of Moments** Recently, Ravuri et al. (2018) proposed a similar idea of using moment matching technique as additional loss. A key difference is that they define feature function parameterised by some parameters and let them learn along with model parameters. They have applied the method of moments to situations in which maximum likelihood estimation (MLE) is not applicable, but where MM can find the correct model distribution on its own. Hence the focus on having (and learning) a large number of features, because only many features will allow to approximate the actual distribution. In our case, we are not relying on MM to model the target distribution on its own. Doing so with a small number of features would be risky (e.g, thinking of the length ratio feature: it would only guarantee that the translation has a correct length, irrespective of the lexical content). We are using MM to complement MLE, in such a way that task-related important features are attended to even if that means reducing the likelihood of the training set. On the other hand, MM can be regarded as a dual interpretation of MLE as satisfying linear constraints represented by feature expectations over the training data. Also, one can see our use of MM as a form of regularisation technique for complementing MLE training and this is an important aspect of our proposed MM approach.
6.3 Experiments

6.3.1 Prior Features for NMT

In order to validate the proposed technique, we re-applied two prior features used for training NMT as in Zhang et al. (2017), including source and target length ratio and lexical bilingual features. Zhang et al. (2017) showed in their experiments that these two are the most effective features for improving NMT systems.

The first feature is straightforward, just about measuring the ratio between source and target length. This feature aims at forcing the model to produce translations with consistent length ratio between source and target sentences, in such a way that too short or too long translations will be avoided.

Given the respective source and target sequences $x$ and $y$, we define this source and target length ratio feature function $\Phi_{\text{len\_ratio}}$ as follows:

$$
\Phi_{\text{len\_ratio}} = \begin{cases} 
\beta \frac{\|x\|}{\|y\|} & \text{if } \beta \times |x| < |y| \\
\frac{|y|}{\beta \times |x|} & \text{otherwise}
\end{cases}, \quad (6.16)
$$

where $\beta$ is additional hyper-parameter, normally set empirically based on prior knowledge about source and target languages. In this case, the feature function is a real value.

The second feature we used is based on a word-to-word lexical translation dictionary produced by an off-the-shelf SMT system (e.g., Moses). The goal of this feature is to ask the model to take external lexical translations into consideration. This feature will be potentially useful in cases such as: translation for rare words, and in low resource setting in which parallel data can be scarce. Following Zhang et al. (2017), we defined sparse feature functions

$$
\Phi_{bd} = \left[ \phi_{(w_{x1}, w_{y1})}, \ldots, \phi_{(w_{x_i}, w_{y_i})}, \ldots, \phi_{(w_{x_{|D_{\text{lex}}|}}, w_{y_{|D_{\text{lex}}|}})} \right],
$$

where:

$$
\phi_{(w_x, w_y)} = \begin{cases} 
1 & \text{if } w_x \in x \land w_y \in y \\
0 & \text{otherwise}
\end{cases},
$$

and where $D_{\text{lex}}$ is a lexical translation dictionary produced by Moses.

---

3 https://github.com/moses-smt/mosesdecoder

4 NMT has been empirically found to be less robust in such a setting than SMT.
6.3 Experiments

<table>
<thead>
<tr>
<th>BLEU</th>
<th>MM Loss</th>
</tr>
</thead>
<tbody>
<tr>
<td>tensor2tensor (Vaswani et al., 2017)</td>
<td>27.69</td>
</tr>
<tr>
<td>base (our reimplementation - Transformer-DyNet)</td>
<td>28.53</td>
</tr>
<tr>
<td>base+mm</td>
<td><strong>29.17</strong>†</td>
</tr>
</tbody>
</table>

Table 6.2 Evaluation scores for training moment matching with length ratio between source and target sequences; **bold**: statistically significantly better than the baselines, †: best performance on dataset.

<table>
<thead>
<tr>
<th>BLEU</th>
<th>MM Loss</th>
</tr>
</thead>
<tbody>
<tr>
<td>tensor2tensor (Vaswani et al., 2017)</td>
<td>27.69</td>
</tr>
<tr>
<td>base (our reimplementation - Transformer-DyNet)</td>
<td>28.53</td>
</tr>
<tr>
<td>base+mm</td>
<td><strong>29.11</strong>†</td>
</tr>
</tbody>
</table>

Table 6.3 Evaluation scores for training moment matching with bilingual lexical dictionary; **bold**: statistically significantly better than the baselines, †: best performance on dataset.

6.3.2 Datasets and Baseline

We proceed to validate the proposed technique with small-scale experiments. We used the IWSLT’15 dataset, translating from English to Vietnamese. This dataset is relatively small, containing approximately 133K sentences for training, 1.5K for development, and 1.3K for testing. We re-implemented the transformer architecture (Vaswani et al., 2017) for training our NMT model with hyper-parameters: 4 encoder and 4 decoder layers; hidden dimension 512 and dropout probability 0.1 throughout the network. For the sampling process, we generated 5 samples for each moment matching training step. We used interpolation training mode with a balancing hyper-parameter of 0.5. We observed that changing this hyper-parameter only slightly affects the overall result. For the feature with length ratio between source and target sequences, we used the length factor $\beta = 1$. For the feature with bilingual lexical dictionary, we extracted it by Moses’s training scripts. In this dictionary, we filtered out the bad entries based on word alignment probabilities produced by Moses, e.g., using a threshold less than 0.5 following Zhang et al. (2017).
Algorithm 7 Adapted Algorithm for Training with Moment Matching Technique for Unsupervised Domain Adaptation

1: **Input**: a pre-trained model $\Theta$; in-domain monolingual source data $\mathcal{D}_s$ and target data $\mathcal{D}_t$; a maximum number of steps $M$.
2: Optional: pre-train in-domain monolingual LM models $\Theta_{LM_{src}}$ and $\Theta_{LM_{trg}}$
3: for $\text{step} = 1, \ldots, M$ do
4: Select a batch of size $N$ source sequences in $\mathcal{D}_s$.
5: Sample $S$ translations for each in a selected batch.
6: Observe MM score for each of samples: $\langle \hat{\Phi}_n, \Phi(y|x_n) - \bar{\Phi}_n \rangle$.
7: Update model parameters using MM gradient $\Gamma_{\Theta,n}$ in Equation 6.17.
8: return newly-trained model $\Theta_{\text{adapted}}$

6.3.3 Results and Analysis

Our results can be found in Table 6.2 and 6.3. As can be seen from the tables, as long as the model attempted to reduce the moment matching loss, the BLEU scores (Papineni et al., 2002) improved statistically significantly with $p < 0.005$ (Koehn, 2004). This was consistently shown in both experiments as an encouraging validation of our proposed training technique with moment matching.

6.4 Moment Matching for Unsupervised Domain Adaptation

Here, we relate to the context of unsupervised domain adaptation problem for NMT as shown in Section 5.2 and further envision an application of our proposed moment matching technique.

Our core idea is to add an auxiliary loss to bridging the distribution gap between general-domain and in-domain, hence reducing the domain discrepancy. Inspired by the moment matching as discussed earlier, we propose the moment matching loss and its gradient for unsupervised domain adaptation, formulated as:

$$J_{MM-\text{ad}}(\Theta) = \frac{1}{N} \sum_{n=1}^{N} \left\| \hat{\Phi}_n(\Theta) - \bar{\Phi}_n \right\|_2^2$$

$$\Gamma_{\Theta,n} = 2\mathbb{E}_{y \sim p_{\Theta}(\cdot|x_n)} \left[ \langle \hat{\Phi}_n(\Theta) - \bar{\Phi}_n, \Phi(y|x_n) - \bar{\Phi}_n \rangle \nabla_{\Theta} \log p_{\Theta}(y|x_n) \right].$$

(6.17)

---

5 in our open source toolkit: https://github.com/duyvuleo/Transformer-DyNet
Here, the (pre-trained) NMT system will have access over the in-domain monolingual data, e.g., $N$ examples in in-domain source language. As can be seen from Equation 6.17, similar to the original moment matching loss, we need to compute model average $\hat{\Phi}_n(\Theta)$ and empirical average $\overline{\Phi}_n$ estimates. One difficulty is that the model average estimate $\hat{\Phi}_n(\Theta)$ has to provide a plausible score telling the model whether its samples are sufficiently good in terms of in-domain relevancy. Another difficulty is that we cannot directly observe $\overline{\Phi}_n$ due to lacking of in-domain parallel data. We need an approximate way to impute this.

We address the model average estimate by leveraging the simple cross-entropy formulation for target monolingual language model $L_{M_{in}}^{trg}$. The intuition is that a sample is relevant to in-domain if its cross-entropy score is sufficiently low, formulated as:

$$\Phi(y|x) \equiv -\frac{1}{|y|} \sum_{i=1}^{|y|} \log p_{L_{M_{in}}^{trg}}(y_i|y_{<i}).$$

As shown earlier, we will use an unbiased estimate of empirical average $\hat{\Phi}_{n,y_j}(\Theta)$, it leads to:

$$\hat{\Phi}_{n,y_j}(\Theta) \equiv \frac{1}{J-1} \sum_{y_j \neq y_j} \Phi(y_j|x_n).$$

Here, also note that we are ignoring the existence of source sequence $x$ for simplicity. We can further take $x$ into consideration by adapting Axelrod’s method in old-style domain adaptation (Axelrod et al., 2011). We need to do some experiments to figure the best suitable method.

More challenging problem is the estimate of the empirical average $\overline{\Phi}_n$. There are possible proxies for imputing it. The first proxy is by first sampling some real samples from in-domain monolingual data (in target language), then taking the mean of their cross-entropy scores. Again, we have to think a better way to leverage the source sequence $x$ as well, e.g.,

$$\Phi_x \equiv \frac{1}{S'} \sum_{s} -\frac{1}{|y_s|} \sum_{i=1}^{|y_i|} \log p_{L_{M_{in}}^{trg}}(y_{s,i}|y_{s,<i}).$$

The second possible proxy is to use a regressor $g_r(x)$ (can be a simple neural network or linear regression model) to predict a real-value cross-entropy scores of target sequence $y$ given a source sequence $x$, e.g,

$$\hat{\Phi}_x \equiv g_r(x).$$

To be able to train this regressor, we need an existing very small in-domain parallel data, perhaps around 1000 sentences - which is feasible to obtain.
We suggested an adapted version of moment matching training for unsupervised domain adaptation which is presented in Algorithm 7. One of the key factors of this algorithm is to develop an appropriate proxy that effectively allows two distributions, one in a general domain and another one in an expected target domain, to be as close as possible within the proposed moment matching framework. We leave experimentation and refinement of the method to future work.

6.5 Summary

We have shown some nice mathematical properties of the proposed moment matching training technique (in particular, unbiasedness) and believe it is promising. Our initial experiments indicate its potential for improving existing NMT systems using simple prior features. Future work may include exploiting more advanced features for improving NMT and evaluating our proposed technique on larger-scale datasets. Further, we would like to investigate our suggestion of applying the proposed moment matching technique for unsupervised domain adaptation as discussed in §6.4.
Part 2

Decoding
Chapter 7

A Novel Decoding Framework with Relaxed Continuous Optimisation

Contents

7.1 Introduction and Motivation .................................................. 154
  7.1.1 Related Work ......................................................... 156
7.2 Problem Formulation .......................................................... 157
  7.2.1 Decoding as Continuous Optimisation ............................... 158
7.3 Algorithms for Relaxed Continuous Optimisation Framework ....... 159
  7.3.1 Exponentiated Gradient (EG) ....................................... 160
  7.3.2 Stochastic Gradient Descent (SGD) ................................ 161
7.4 Decoding in Extended NMT .................................................. 162
7.5 Experiments ......................................................................... 163
  7.5.1 Datasets ..................................................................... 163
  7.5.2 Baselines and Setup ..................................................... 164
  7.5.3 Evaluation Metrics ...................................................... 164
7.6 Results and Analysis ............................................................ 165
  7.6.1 Initialisation and Step Size .......................................... 165
  7.6.2 Continuous vs Discrete Costs ....................................... 167
  7.6.3 EG vs SGD .................................................................. 168
  7.6.4 EG vs Reranking ......................................................... 169
  7.6.5 Computational Efficiency ............................................. 169
We now turn to the second part of this thesis, the Decoding part, of which the focus is on improving the existing decoding, a.k.a. inference, algorithms in the context of neural translation models. Our main goal is to address one of the drawbacks of existing approximate decoding methods, which is the limited ability to incorporate global factors (e.g., bidirectional, bilingual) due to intractable search space. Here, we propose a novel decoding approach for neural machine translation (NMT) based on relaxed continuous optimisation (§7.1). We reformulate conventional decoding, a discrete optimisation problem, as a continuous problem (§7.2). As a result, decoding can make use of efficient gradient-based techniques such as Stochastic Gradient Descent (SGD) and Exponentiated Gradient (EG) methods, which are mainly used to train models in the literature (§7.3). Our powerful decoding framework allows for more accurate decoding in standard neural machine translation models. Furthermore, it enables decoding in intractable models such as intersection of several different NMT models, including left-to-right and right-to-left models as well as source-to-target and target-to-source NMT models (§7.4). Our empirical results (§7.5, §7.6) show that our decoding framework is effective, and can lead to substantial improvements in translation quality, especially in situations where greedy search and beam search are not feasible. Finally, we show how the technique is highly competitive with, and complementary to, reranking.

Parts of this chapter were published in:


7.1 Introduction and Motivation

Sequence to sequence learning with neural networks (Graves, 2013; Lipton, 2015; Sutskever et al., 2014) is typically associated with two phases: training and decoding (a.k.a. inference). Model parameters are learned by optimising the training objective so that the model
can produce good translations when decoding unseen sentences. The majority of research has focused on the training paradigm or network architecture; however, effective means of decoding have been under-investigated. Conventional heuristic-based approaches for approximate decoding, include greedy, beam, and stochastic search. Greedy and beam search have been empirically proved to be adequate for many sequence to sequence tasks, and are the standard methods for NMT decoding.

However, these decoding approaches have several drawbacks. Firstly, although auto-regressive NMT models use a left-to-right generation which would appear to facilitate efficient search, the models themselves use a recurrent architecture, and accordingly are non-Markov. This prevents dynamic programming for exact solutions, and moreover, limits the potential to incorporate additional global features or constraints. Global factors can be highly useful in producing better and more diverse translations. Secondly, the sequential decoding of symbols in the target sequence, the inter-dependencies among the target symbols are not fully exploited. For example, when decoding the words of the target sentence in a left-to-right manner, the right context is not exploited, leading potentially to inferior performance (see (Watanabe and Sumita, 2002) who applied this idea in traditional statistical MT). A natural way to capture this is to intersect left-to-right and right-to-left models; however, the resulting model has no natural generation order, and thus, standard decoding methods are unsuitable.

In neural MT literature, existing works have addressed this issue via re-ranking approaches. For example, Birch (2016) utilised right-to-left model to re-rank the n-best translation list produced by left-to-right model, resulting in translation diversity improvement for German-to-English translation. Similarly, the n-best translation list produced by beam search can be diversified by integrating bidirectional (left-to-right and right-to-left) scores in re-ranking (Li and Jurafsky, 2016a), or by adjusting the beam diversity with reinforcement learning (Li et al., 2016). Another approach by Cho (2016) proposed the Noisy Parallel Approximate Decoding (NPAD) strategy for handling the uncertainty dynamics in the conditional recurrent language model via injecting noises in the hidden transition function. By running multiple parallel decoding processes, NPAD can produce better and more diversified translation results compared to the greedy and beam search methods.

We introduce a novel decoding framework (§ 7.2) that relaxes this discrete optimisation problem into a continuous optimisation problem. This is akin to linear programming relaxation approach for approximate decoding in graphical models with discrete random variables, where the exact decoding is NP-hard (Belanger and McCallum, 2016; Sontag, 2010). The resulting continuous optimisation problem is challenging due to the non-linearity and non-convexity of the relaxed decoding objective. We make use of stochastic gradient descent
(SGD) and exponentiated gradient (EG) algorithms for decoding based on our relaxation approach.\footnote{Both methods are mainly used for \textit{training} in prior work.} Our decoding framework is powerful and flexible, as it enables us to decode with global constraints involving intersection of multiple NMT models (§7.4). We present experimental results on Chinese-English and German-English translation tasks, confirming the effectiveness of our relaxed optimisation method for decoding (§7.5).

\subsection*{7.1.1 Related Work}

Decoding (inference) for neural models is an important task; however, there is limited research in this space perhaps due to the challenging nature of this task, with only a few work exploring some extensions to improve upon them. The most widely-used decoding methods include sampling (Cho, 2016), greedy and beam search (Bahdanau et al., 2015; Sutskever et al., 2014, \textit{inter alia}), and reranking (Birch, 2016; Li and Jurafsky, 2016a).

Cho (2016) proposed to perturb the neural model by injecting noise(s) in the hidden transition function of the conditional recurrent neural language model during greedy or beam search, and execute multiple parallel decoding runs. This strategy can improve over greedy and beam search; however, it is not clear how, when and where noise should be injected to be beneficial. Recently, Wiseman and Rush (2016) proposed beam search optimisation while \textit{training} neural models, where the model parameters are updated in case the gold standard falls outside of the beam. This exposes the model to its past incorrect predicted labels, hence making the training more robust. This is orthogonal to our approach where we focus on the decoding problem with a pre-trained model.

Reranking has also been proposed as a means of global model combination: Birch (2016) and Li and Jurafsky (2016a) re-ranked the left-to-right decoded translations based on the scores of a right-to-left model, learning to more diverse translations. Related, Li et al. (2016) proposed an algorithm of learning to adjust the beam diversity based on reinforcement learning.

Perhaps most relevant is Snelleman (2016) who performed concurrently to this work, who also proposed an decoding method for NMT using linear relaxation. Snelleman (2016)’s method was similar to our SGD approach, however he did not manage to outperform beam search baselines with an encoder-decoder. In contrast we go much further, proposing the EG algorithm, which we show works much more effectively than SGD, and demonstrate how this can be applied to decoding in an attentional encoder-decoder. Moreover, we demonstrate the utility of related optimisation for decoding over global ensembles of models, resulting in consistent improvements in search error and end translation quality.
Recently, relaxation techniques have been applied to deep models for training and decoding in text classification (Belanger and McCallum, 2016; Belanger et al., 2017), and fully differentiable training of sequence-to-sequence models with scheduled-sampling (Goyal et al., 2017b). Our work has applied the relaxation technique specifically for decoding in NMT models.

### 7.2 Problem Formulation

In neural machine translation (NMT), the probability of the target sentence \( y \) given a source sentence \( x \) is written as:

\[
P_{\Theta}(y|x) = \sum_{i=1}^{\|y\|} \log P_{\Theta}(y_i|y_{<i}, x)
\]

(7.1)

\[y_i|y_{<i}, x \sim \text{softmax}(f(\Theta, y_{<i}, x))\]

where \( f \) is a non-linear function of the previously generated sequence of words \( y_{<i} \), the source sentence \( x \), and the model parameters \( \Theta \). In this paper, we realise \( f \) as follows:

\[
f(\Theta, y_{<i}, x) = W_w \cdot \text{MLP}(c_i, E_T^{y_{i-1}}, g_i) + b_v
\]

\[g_i = \text{RNN}^{\phi}_{\text{dec}}(c_i, E_T^{y_{i-1}}, g_{i-1})\]

where MLP is a single hidden layer neural network with tanh activation function, and \( E_T^{y_{i-1}} \) is the embedding of the target word \( y_{i-1} \) in the embedding matrix \( E_T \in \mathbb{R}^{n_e \times |V_T|} \) of the target language vocabulary \( V_T \), and \( n_e \) is the embedding dimension. The state \( g_i \) of the decoder RNN is a function of \( y_{i-1} \), its previous state \( g_{i-1} \), and the context \( c_i = \sum_{j=1}^{\|x\|} \alpha_{ij} h_j \) summarises parts of the source sentence which are attended to, where

\[
\alpha_j = \text{softmax}(e_j) \quad e_{ij} = \text{MLP}(g_{i-1}, h_j)
\]

\[h_j = \text{biRNN}^{\phi}_{\text{enc}}(E_S^{x_j}, \overline{h}_{j-1}, \overline{h}_{j+1})\]

In above, \( \overline{h}_i \) and \( \overline{h}_i \) are the states of the left-to-right and right-to-left RNNs encoding the source sentence, and \( E_S^{x_j} \) is the embedding of the source word \( x_j \) in the embedding matrix \( E_S \in \mathbb{R}^{n'_e \times |V_S|} \) of the source language vocabulary \( V_S \) and \( n'_e \) is the embedding dimension.
Given a bilingual corpus $\mathcal{D}$, the model parameters are learned by maximizing the conditional log-likelihood,

$$\Theta^* = \arg\max_{\Theta} \sum_{(x,y) \in \mathcal{D}} \log P_{\Theta}(y \mid x).$$  \hspace{1cm} (7.2)

The model parameters $\Theta$ include the weight matrix $W_o \in \mathbb{R}^{|V_T| \times H}$ and the bias $b_o \in \mathbb{R}^{|V_T|}$ – with $H$ denoting the hidden dimension size – as well as the RNN encoder $\text{biRNN}^\theta_{\text{enc}}$ / decoder $\text{RNN}^\phi_{\text{dec}}$ parameters, word embedding matrices, and the parameters of the attention mechanism. The model is trained end-to-end by optimising the training objective using stochastic gradient descent (SGD) or its variants. In this paper, we focus on the decoding problem, which we turn to in the next section.

### 7.2.1 Decoding as Continuous Optimisation

In decoding, we are interested in finding the highest probability translation for a given source sentence:

$$\min_y - P_{\Theta}(y \mid x) \text{ s.t. } y \in \mathcal{Y}_x$$  \hspace{1cm} (7.3)

where $\mathcal{Y}_x$ is the space of possible translations for the source sentence $x$. In general, searching $\mathcal{Y}_x$ to find the highest probability translation is intractable due to the recurrent nature of Equation 7.3 which prevents dynamic programming for efficient search. This is problematic, as the space of translations is exponentially large with respect to the output length $|y|$.

We now formulate this discrete optimisation problem as a continuous one, and then use standard algorithms for continuous optimisation for decoding. Let us assume that the maximum length of a possible translation for a source sentence is known and denote it as $\ell^*$. The best translation for a given source sentence solves the following optimisation problem:

$$y^* = \arg\min_{y_1,\ldots,y_\ell} \sum_{i=1}^\ell - \log P_{\Theta}(y_i \mid y_{<i}, x)$$  \hspace{1cm} (7.4)

s.t. $\forall i \in \{1 \ldots \ell\} : y_i \in V_T.$

where we allow the translation to be padded with sentinel symbols to the right, which are ignored in computing the model probability. Equivalently, we can rewrite the above discrete
optimisation problem as follows:

\[
\arg\min_{\tilde{y}_1, \ldots, \tilde{y}_\ell} - \sum_{i=1}^{\ell} \tilde{y}_i \cdot \log \text{softmax} \left( f \left( \Theta, \tilde{y}_{<i}, x \right) \right)
\]

s.t. \quad \forall i \in \{1 \ldots \ell\} : \tilde{y}_i \in \mathbb{R}^{|V_T|}

(7.5)

where \( \tilde{y}_i \) are vectors using the one-hot representation of the target words \( \mathbb{R}^{|V_T|} \).

We now convert the optimisation problem (7.5) to a continuous one by dropping the integrality constraints \( \tilde{y}_i \in \mathbb{R}^{|V_T|} \) and require the variables to take values from the probability simplex:

\[
\arg\min_{\hat{y}_1, \ldots, \hat{y}_\ell} - \sum_{i=1}^{\ell} \hat{y}_i \cdot \log \text{softmax} \left( f \left( \Theta, \hat{y}_{<i}, x \right) \right)
\]

s.t. \quad \forall i \in \{1 \ldots \ell\} : \hat{y}_i \in \Delta_{|V_T|}

where \( \Delta_{|V_T|} \) is the \( |V_T| \)-dimensional probability simplex, i.e., \( \{ \hat{y}_i \in [0, 1]^{\mathbb{R}^{|V_T|}} : \|\hat{y}_i\|_1 = 1 \} \).

In practice, this amounts to replacing \( E_{\tilde{y}_i} \) with the expected embedding of target language words \( E_{\hat{y}_i} \) under the distribution \( \hat{y}_i \).

After solving the above constrained continuous optimisation problem, there is no guarantee that the resulting solution \( \{ \hat{y}_i^* \}_{i=1}^\ell \) will comprise one-hot vectors, i.e., target language words. Instead it can find fractional solutions, that require ‘rounding’ in order to resolve them to lexical items. To solve this problem, we take the \( \arg \max \),\(^2\) i.e., take the highest scoring word for each position \( \hat{y}_i^* \). We leave exploration of more elaborate projection techniques to the future work.

In the context of graphical models, the above relaxation technique gives rise to linear programming for approximate decoding (Belanger and McCallum, 2016; Sontag, 2010). However, our decoding problem is much harder due to the non-linearity and non-convexity of the objective function operating on high dimensional space for deep models. We now turn our attention to optimisation algorithms to effectively solve the decoding optimisation problem.

### 7.3 Algorithms for Relaxed Continuous Optimisation Framework

Here, we present two different algorithms for solving the above decoding continuous optimisation problem, including:

\(^2\)Ties are broken arbitrarily.
Algorithm 8 The EG Algorithm for Decoding by Optimisation

1: For all $i$ initialise $\hat{y}_i^0 \in \Delta_{|V_T|}$
2: for $t = 1, \ldots, \text{MaxIter}$ do
3: For all $i, w$ : calculate $\nabla_{t-1}^i, w = \frac{\partial Q(\hat{y}_1^{t-1}, \ldots, \hat{y}_\ell^{t-1})}{\partial \hat{y}_i^t(w)}$ ▷ using back-propagation
4: For all $i, w$ : update $\hat{y}_i^t(w) \propto \hat{y}_i^{t-1}(w) \cdot \exp \left( -\eta \nabla_{t-1}^i, w \right)$ ▷ $\eta$ is the step size
5: return $\arg \min_t Q(\hat{y}_1^t, \ldots, \hat{y}_\ell^t)$

7.3.1 Exponentiated Gradient (EG)

Exponentiated gradient (Kivinen and Warmuth, 1997) is an elegant algorithm for solving optimisation problems involving simplex constraints. Recall our constrained optimisation problem:

$$\arg \min_{\hat{y}_1, \ldots, \hat{y}_\ell} Q(\hat{y}_1, \ldots, \hat{y}_\ell)$$

s.t. $\forall i \in \{1 \ldots \ell\} : \hat{y}_i \in \Delta_{|V_T|}$ \hspace{1cm} (7.6)

where $Q(\hat{y}_1, \ldots, \hat{y}_\ell)$ is defined as

$$- \sum_{i=1}^\ell \hat{y}_i \cdot \log \text{softmax} \left( f(\Theta, \hat{y}_{<i}, x) \right).$$ \hspace{1cm} (7.7)

EG is an iterative algorithm, which updates each distribution $\hat{y}_i^t$ in the current time-step $t$ based on the distributions of the previous time-step as follows:

$$\forall w \in V_T : \hat{y}_i^t(w) = \frac{1}{Z'_i} \hat{y}_i^{t-1}(w) \exp \left( -\eta \nabla_{i,w}^{t-1} \right)$$ \hspace{1cm} (7.8)

where $\eta$ is the step size, $\nabla_{i,w}^{t-1} = \frac{\partial Q(\hat{y}_1^{t-1}, \ldots, \hat{y}_\ell^{t-1})}{\partial \hat{y}_i^t(w)}$ and $Z'_i$ is the normalisation constant

$$Z'_i = \sum_{w \in V_T} \hat{y}_i^{t-1}(w) \exp \left( -\eta \nabla_{i,w}^{t-1} \right).$$

The partial derivatives $\nabla_{i,w}^{t-1}$ are calculated using the back propagation algorithm treating $\{\hat{y}_i\}_{i=1}^\ell$ as parameters and the original parameters of the model $\Theta$ as constants. Adapting EG to our decoding problem leads to Algorithm 8.
It can be shown that the EG algorithm is a gradient descent algorithm for minimising the following objective function subject to the simplex constraints:

\[
Q(\hat{y}_1, \ldots, \hat{y}_\ell) - \gamma \sum_{i=1}^{\ell} \text{Entropy}(\hat{y}_i)
\]

\[
= Q(\hat{y}_1, \ldots, \hat{y}_\ell) - \gamma \sum_{i=1}^{\ell} \sum_{w \in \mathcal{V}} \hat{y}_i(w) \log \frac{1}{\hat{y}_i(w)}. \quad (7.9)
\]

In other words, the algorithm looks for the maximum entropy solution which also maximises the log likelihood under the model. There are intriguing parallels with the maximum entropy formulation of log-linear models (Berger et al., 1996) as well as minimising entropy in semi-supervised learning (Grandvalet and Bengio, 2004). In our setting, the entropy term acts as a prior which discourages overly-confident estimates in the absence of sufficient evidence.

### 7.3.2 Stochastic Gradient Descent (SGD)

To be able to apply SGD to our optimisation problem, we need to make sure that the simplex constraints are enforced. One way to achieve this is by reparameterising using the softmax transformation, i.e. \( \hat{y}_i = \text{softmax}(\tilde{r}_i) \). The resulting unconstrained optimisation problem, now over \( \tilde{r}_i \), becomes

\[
\arg\min_{\tilde{r}_1, \ldots, \tilde{r}_\ell} - \sum_{i=1}^{\ell} \text{softmax}(\tilde{r}_i) \cdot \log \text{softmax}(f(\Theta, \hat{y}_{<i}, x))
\]

(7.10)

where \( E^{y_i}_{\tilde{r}} \) is replaced with the expected embedding of the target words under the distribution resulted from the \( E_{\text{softmax}(\tilde{r})}[E^{w}_{\tilde{r}}] \) in the model.

To apply SGD updates, we need the gradient of the objective function with respect to the new variables \( \tilde{r}_i \) which can be derived with the back-propagation algorithm based on the chain rule:

\[
\frac{\partial Q}{\partial \tilde{r}_i(w)} = \sum_{w' \in \mathcal{V}_T} \frac{\partial Q(\ldots)}{\partial \hat{y}_{i}(w')} \frac{\partial \hat{y}_{i}(w')}{\partial \tilde{r}_i(w)}
\]

The resulting SGD algorithm is summarised in Algorithm 9.
Algorithm 9 The SGD Algorithm for Decoding by Optimisation

1: For all \( i \) initialise \( \hat{r}_i^0 \)
2: for \( t = 1, \ldots, \text{MaxIter} \) do
3:   For all \( i, w \) : calculate \( \nabla_{i,w}^{-1} = \sum_{w' \in V} \frac{\partial Q(\hat{y}_t^{-1}, \ldots, \hat{y}_t^{\ell})}{\partial \hat{y}_i} \frac{\partial \hat{y}_i}{\partial \hat{r}_i}(w') \) \( \triangleright \) using backpropagation
4:   For all \( i, w \) : update \( \hat{r}_i^t(w) = \hat{r}_i^{t-1}(w) - \eta \nabla_{i,w}^{t-1} \) \( \triangleright \eta \) is the step size
5: return \( \arg \min_t Q(\text{softmax}(\hat{r}_i^1), \ldots, \text{softmax}(\hat{r}_i^\ell)) \)

7.4 Decoding in Extended NMT

Our decoding framework allows us to effectively and flexibly add additional global factors over the output symbols during decoding. This enables decoding for richer global models, for which there is no effective means of greedy decoding or beam search. We outline several such models, and their corresponding relaxed objective functions for optimisation-based decoding.

Bidirectional Ensemble. Standard NMT generates the translation in a left-to-right manner, conditioning each target word on its left context. However, the joint probability of the translation can be decomposed in a myriad of different orders; one compelling alternative would be to condition each target word on its right context, i.e., generating the target sentence from right-to-left. We would not expect a right-to-left model to outperform a left-to-right, however, as the left-to-right ordering reflects the natural temporal order of spoken language. However, the right-to-left model is likely to provide a complementary signal in translation, as it will be bringing different biases and making largely independent prediction errors to those of the left-to-right model. For this reason, we propose to use both models, and seek to find translations that have high probabilities according to both models (this mirrors work on bidirectional decoding in classical statistical machine translation by Watanabe and Sumita (2002)). Decoding under the ensemble of these models leads to an intractable search problem, not well suited to traditional greedy or beam search algorithms, which require a fixed generation order of the target words. This ensemble decoding problem can be formulated simply in our linear relaxation approach, using the following objective function:

\[
\mathcal{C}_{\text{bidir}} = -\alpha \log P_{\Theta_-}(y | x) - (1 - \alpha) \log P_{\Theta_+}(y | x);
\]

where \( \alpha \) is an interpolation hyper-parameter, which we set to 0.5; \( \Theta_- \) and \( \Theta_+ \) are the pre-trained left-to-right and right-to-left models, respectively. This bidirectional agreement may
also lead to improvement in translation diversity, as shown in Li and Jurafsky (2016a) in a re-ranking evaluation.

**Bilingual Ensemble.** Another source of complementary information is in terms of the translation direction, that is forward translation from the source to the target language, and reverse translation in the target to source direction. Decoding must find a translation which scores well under both the forward and reverse translation models. This is inspired by the direct and reverse feature functions commonly used in classical discriminative SMT (Och and Ney, 2002) which have been shown to offer some complementary benefits (although see (Lopez and Resnik, 2006)). More specifically, we decode for the best translation in the intersection of the source-to-target and target-to-source models by minimizing the following objective function:

\[
\mathcal{C}_{\text{biling}} = -\alpha \log P_{\Theta_{s\rightarrow t}}(y \mid x) - (1 - \alpha) \log P_{\Theta_{s\leftarrow t}}(x \mid y);
\]

where \(\alpha\) is an interpolation hyper-parameter to be fine-tuned; and \(\Theta_{s\rightarrow t}\) and \(\Theta_{s\leftarrow t}\) are the pre-trained source-to-target and target-to-source models, respectively. Decoding for the best translation under the above objective function leads to an intractable search problem, as the reverse model is global over the target language, meaning there is no obvious means of search with a greedy algorithm or similar.

**Discussion.** There are two important considerations on how best to initialise the relaxed optimisation in the above settings, and how best to choose the step size. As the relaxed optimisation problem is, in general, non-convex, finding a plausible initialisation is likely to be important for avoiding local optima. Furthermore, a proper step size is a key in the success of the EG-based and SGD-based optimisation algorithms, and there is no obvious method how to best choose its value. We may also adaptively change the step size using (scheduled) annealing or via the line search. We return to this considerations in the experimental evaluation.

### 7.5 Experiments

#### 7.5.1 Datasets

We conducted our experiments on datasets with different scales (small, medium, large), translating between Chinese→English using the BTEC corpus, and German→English using
Table 7.1 Statistics of the training and evaluation sets; token and types are presented for both source/target languages.

<table>
<thead>
<tr>
<th></th>
<th># tokens</th>
<th># types</th>
<th># sents</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>BTEC zh→en</td>
<td></td>
<td></td>
</tr>
<tr>
<td>train</td>
<td>422k / 454k</td>
<td>3k / 3k</td>
<td>44,016</td>
</tr>
<tr>
<td>dev</td>
<td>10k / 10k</td>
<td>1k / 1k</td>
<td>1,006</td>
</tr>
<tr>
<td>test</td>
<td>5k / 5k</td>
<td>1k / 1k</td>
<td>506</td>
</tr>
<tr>
<td></td>
<td>TED Talks de→en</td>
<td></td>
<td></td>
</tr>
<tr>
<td>train</td>
<td>4m / 4m</td>
<td>26k / 19k</td>
<td>194,181</td>
</tr>
<tr>
<td>dev-test2010</td>
<td>33k / 35k</td>
<td>4k / 3k</td>
<td>1,565</td>
</tr>
<tr>
<td>test2014</td>
<td>26k / 27k</td>
<td>4k / 3k</td>
<td>1,305</td>
</tr>
<tr>
<td></td>
<td>WMT 2016 de→en</td>
<td></td>
<td></td>
</tr>
<tr>
<td>train</td>
<td>107m / 108m</td>
<td>90k / 78k</td>
<td>4m</td>
</tr>
<tr>
<td>dev-test2013&amp;14</td>
<td>154k / 152k</td>
<td>20k / 13k</td>
<td>6003</td>
</tr>
<tr>
<td>test2015</td>
<td>54k / 54k</td>
<td>10k / 8k</td>
<td>2169</td>
</tr>
</tbody>
</table>

the IWSLT 2015 TED Talks (Cettolo et al., 2014) and WMT 2016\(^3\) corpora. The statistics of the datasets can be found in Table 7.1.

### 7.5.2 Baselines and Setup

We implemented our continuous-optimisation based decoding method on top of the Mantidae toolkit\(^4\) (Cohn et al., 2016), and using the dynet\(^5\) deep learning library (Neubig et al., 2017). All neural network models were configured with 512 input embedding and hidden layer dimensions, and 256 alignment dimension, with one and two hidden layers in the source and target, respectively. We used a LSTM recurrent structure (Hochreiter and Schmidhuber, 1997) for both source and target RNN sequences. For the vocabulary, we use word frequency cut-off of 5, and words rarer than this were mapped to a sentinel. For the large-scale WMT dataset, we applied byte-pair encoding (BPE) method (Sennrich et al., 2016d) to better handle unknown words.\(^6\) For training our neural models, we use early stopping based on development perplexity, which usually occurs after 5-8 epochs.

### 7.5.3 Evaluation Metrics

We evaluated in terms of search error, measured using the model score of the inferred solution (either continuous or discrete), as well as measuring the end translation quality with

---

\(^3\)http://www.statmt.org/wmt16/translation-task.html

\(^4\)https://github.com/duyvuleo/Mantidae

\(^5\)https://github.com/clab/dynet

\(^6\)With BPE, the out of vocabulary rates on heldout data are < 1%. Ideally, BPE would be applied in both small- and large- scale cases.
case-insensitive BLEU (Papineni et al., 2002). We used the continuous cost which measures \[-\frac{1}{|y|} \log P_{\Theta}(\hat{y} | x)\] under the model \(\Theta\); and the discrete model score has the same formulation, albeit using the discrete rounded solution \(y\) (see §7.2). Note the cost can be used as a tool for selecting the best decoding solution, as well as assessing convergence, as we illustrate below.

7.6 Results and Analysis

7.6.1 Initialisation and Step Size

As our relaxed optimisation problems are non-convex, local optima are likely to be a problem. We test this empirically, focusing on the effect that initialisation and step size, \(\eta\), have on the decoding quality.

For plausible initialisation states, we evaluate different strategies: uniform in which the relaxed variables \(\hat{y}\) are initialised to \(\frac{1}{|V_T|}\); and greedy or beam whereby \(\hat{y}\) are initialised based on an already good solution produced by a baseline decoder with greedy (gdec) or beam (bdec). Instead of using the Viterbi outputs as a one-hot representation, we initialise to the probability prediction vectors,\(^7\) which serves to limit attraction of the initialisation condition, which is likely to be a local (but not global) optima.

Figure 7.1 illustrates the effect of initialisation on the EG algorithm, in terms of search error (left and middle) and translation quality (right), as we vary the number of iterations of decoding. There is clear evidence of non-convexity: all initialisation methods can be seen to converge using all three measures, however they arrive at highly different solutions. Uniform initialisation is clearly not a viable approach, while greedy and beam initialisation both yield much better results. The best initialisation, beam, outperforms both greedy and beam decoding in terms of BLEU.

Note that the EG algorithm has fairly slow convergence, requiring at least 100 iterations, irrespective of the initialisation. To overcome this, we use momentum (Qian, 1999) to accelerate the convergence by modifying the term \(\nabla_{i,w}^t\) in Algorithm 8 with a weighted moving average of past gradients:

\[
\nabla_{i,w}^{t-1} = \gamma \nabla_{i,w}^{t-2} + \eta \frac{\partial Q(\hat{y}_i^{t-1}, \ldots, \hat{y}_F^{t-1})}{\partial \hat{y}_i(w)}
\]

where we set the momentum term \(\gamma = 0.9\). The EG with momentum (EG-MOM) converges after fewer iterations (about 35), and results in marginally better BLEU scores. The momen-

\(^7\)Here, EG uses softmax normalisation whereas SGD uses the pre-softmax vector.
Fig. 7.1 Analysis on effects of initialisation states (uniform vs. greedy vs. beam), step size annealing, momentum mechanisms from BTEC zh→en translation. 

EG-400: EG algorithm with step size \( \eta = 400 \) (otherwise \( \eta = 50 \)); EG-MOM: EG algorithm with momentum.

EG-400: EG algorithm with step size \( \eta = 400 \) (otherwise \( \eta = 50 \)); EG-MOM: EG algorithm with momentum.
7.6 Results and Analysis

tum technique is usually used for SGD involving additive updates; it is interesting to see it also works in EG with multiplicative updates.

The step size, $\eta$, is another important hyper-parameter for gradient based search. We tune the step size using line search over $[10, 400]$ over the development set. Figure 7.1 illustrates the effect of changing step size from 50 to 400 (compare EG and $\text{EG-400}$ with uniform), which results in a marked difference of about 10 BLEU points, underlining the importance of tuning this value. We found that EG with momentum had less of a reliance on step size, with optimal values in $[10, 50]$; we use this setting hereafter.

Fig. 7.2 Comparing discrete vs continuous costs from BTEC zh→en translation, using the EG algorithm with momentum, $\eta = 50$. Each point corresponds to a sentence.

7.6.2 Continuous vs Discrete Costs

Another important question is whether the assumption behind continuous relaxation is valid, i.e., if we optimise a continuous cost to solve a discrete problem, do we improve the discrete output? Although the continuous cost diminishes with decoding iterations (Figure 7.1 left) and appears to converge, it is not clear whether this corresponds to a better discrete output (note that the discrete cost and BLEU scores do show improvements Figure 7.1 centre and right). Figure 7.2 illustrates the relation between the two cost measures, showing that in
most cases the discrete and continuous costs are identical. Linear relaxation fails only for a handful of cases, where the nearest discrete solution is significantly worse than it would appear using the continuous cost.

7.6.3 EG vs SGD

Both the EG and SGD algorithms are iterative methods for solving the relaxed optimisation problem with simplex constraints. We measure empirically their difference in terms of quality of decoding and speed of convergence, as illustrated in Figure 7.3. Observe that SGD requires 150 iterations for convergence, whereas EG requires many fewer (50). This concurs with previous work on learning structured prediction models with EG (Globerson et al., 2007). Further, the EG algorithm consistently produces better results in terms of both model cost and BLEU.

Fig. 7.3 Analysis on convergence and performance comparing SOFTMAX and EG algorithms from BTEC zh→en translation. Both algorithms use momentum and step size 50.
### 7.6 Results and Analysis

#### Table 7.2

<table>
<thead>
<tr>
<th>Method</th>
<th>BLEU</th>
<th>AvgLen</th>
</tr>
</thead>
<tbody>
<tr>
<td>bdec_left-to-right</td>
<td>26.69</td>
<td>20.73</td>
</tr>
<tr>
<td>filtered rerank</td>
<td>26.84</td>
<td>20.66</td>
</tr>
<tr>
<td>EGdec w/ beam init</td>
<td>27.34</td>
<td>20.73</td>
</tr>
<tr>
<td>full rerank</td>
<td>27.34</td>
<td>21.76</td>
</tr>
<tr>
<td>EGdec w/ rerank init</td>
<td>27.78</td>
<td>21.70</td>
</tr>
</tbody>
</table>

Table 7.2 The BLEU evaluation results with EG algorithm against 100-best reranking on WMT evaluation dataset.

#### 7.6.4 EG vs Reranking

Reranking is an alternative method for integrating global factors into the existing NMT systems. We compare our EG decoding algorithm against the reranking approach with bidirectional factor where the $k$-best outputs of a left-to-right decoder is re-scored with the forced decoder operating in a right-to-left fashion. The results are shown in Table 7.2. Our EG algorithm initialised with the reranked output achieves the best BLEU score. We also compare reranking with EG algorithm initialised with the beam decoder, where for direct comparison we filter out sentences with length greater than that of the beam output in the $k$-best lists. These results show that the EG algorithm is capable of effectively exploiting the search space.

Beyond achieving similar or better translations to re-ranking, note that EG is simpler in implementation, as it does not require $k$-best lists, weight tuning and so forth. Instead this is replaced with iterative gradient descent. The run-time of the two methods are comparable, when reranking uses modest $k$, however EG can be considerably faster when $k$ is large, as is typically done to extract the full benefit from re-ranking. This performance difference is a consequence of GPU acceleration of the dense vector operations in EG decoding.

#### 7.6.5 Computational Efficiency

We also quantify the computational efficiency of the proposed decoding approach. Benchmarking on a GPU Titan X for decoding BTEC zh$\rightarrow$en, the average time per sentence is 0.02 secs for greedy, 0.07s for beam=5, 0.11s for beam=10, and 3.1s for relaxed EG decoding, which uses an average of 35 EG iterations. The majority of time in the EG algorithm is in the forward and backward passes, taking 30% and 67% of the time, respectively. Our implementation was not optimised thoroughly, and it is likely that it could be made significantly faster, which we defer to future research.
Table 7.3 The BLEU evaluation results across evaluation datasets for EG algorithm variants against the baselines; **bold:** statistically significantly better than the best greedy or beam baseline, †: best performance on dataset.

### 7.6.6 Main Results

Table 7.3 shows our experimental results across all datasets, evaluating the EG algorithm and its variants. For the EG algorithm with greedy initialisation (top), we see small but consistent improvements in terms of BLEU. Beam initialisation led to overall higher BLEU scores, and again demonstrating a similar pattern of improvements, albeit of a lower magnitude, over the initialisation values.

Next we evaluate the capability of our decoding method with extended NMT models, where approximate algorithms such as greedy or beam search are infeasible. With the **bidirectional** ensemble, we obtained the statistically significant BLEU score improvements compared to the unidirectional models, for either greedy or beam initialisation. This is interesting in the sense that the unidirectional right-to-left model always performs worse than the left-to-right model. However, our method with bidirectional ensemble is capable of combining their strengths in a unified setting. For the **bilingual** ensemble, we see similar effects, with better BLEU score improvements in most cases, albeit of a lower magnitude, over the bidirectional one. This is likely to be due to a disparity with the training condition for the models, which were learned independently of one another. See also translation examples produced by our proposed approaches as shown in Figure 7.4. Further examples can be found in Appendix §A.
Towards Constrained Decoding for Neural Sequence Models

We have presented a principled framework based on the idea of relaxed continuous optimisation for decoding of NMT models. One can envision a possible extension of the above framework for a constrained decoding problem. Constrained decoding comes from the possibility that there is no guarantee that neural seq2seq models can decode or infer accurate
outputs that satisfy prior "sequential" constraints. This often happens when using neural seq2seq models for solving complex structured prediction tasks (in addition to NMT) such as: word ordering, dependency and constituency parsing. Here, we attempt to demonstrate the applicability of our proposed relaxed continuous optimisation to a new class of problem, i.e., structured prediction with constraints. Unfortunately, our initial results on this idea were inconclusive (thus, not reported here), however we believe the method has potential, and leave evaluation and further development for future work.

7.7.1 Motivating Examples on Syntactic Parsing

Linearisation. In order to apply the seq2seq model for syntactic parsing, one may need the output sequence to be presented in a linearised sequence of tokens. Here, an input sequence is simply a text in a language (e.g., English, Chinese); whereas an output sequence comprises a sequence of tokens from which we can recover a valid parse tree. This is called the "linearisation" process.

We illustrate the linearisation process in Table 7.4 for both constituency parsing (Vinyals et al., 2015a) and transition-based dependency parsing (Dyer et al., 2015). Also note that the example sentence “UNESCO is now holding its biennial meetings in Paris to devise its next projects” was extracted from the PTB dataset.\(^8\) The “linearised” output sequence includes tokens that represent the semantics of the original parse tree, and they must guarantee that one can form a valid parse tree based on that sequence. Otherwise, that linearised sequence is not acceptable.

Analysis of Parse Errors. We already know that the standard seq2seq model is often trained using the cross-entropy objective function based on the generation of each token in the target sequence (here, the linearised sequence). It is unlikely that the seq2seq model understands the semantics, e.g., the prior constraints of the linearised sequence. For example, in the case of constituency parsing, the number of tokens for opening and closing brackets may not be equal; and in the case of dependency parsing, the “SHIFT” tokens may not be sufficient to perform the transition recovering steps.

In order to validate our hypothesis, we conducted an experiment with dependency parsing with the neural seq2seq model. We performed the linearisation process for all dependency parse trees in the PTB WSJ dataset. We used the sections from 2 to 21 for training, section 22 for development and section 23 for evaluation. For neural seq2seq models, we employed

\(^8\)https://catalog.ldc.upenn.edu/ldc99t42
Input Sequence
UNESCO is now holding its biennial meetings in Paris to devise its next projects.

Constituency Parse Tree

```
S
   NP
   | NNP
   | VBZ
   | ADVP
   VP
   | NP
   | VB
   | ADVP
   | VP
   | NB
     | JJ
   | NNS
   | IN
   | NNP
   | VP
   | TO
   | VB
   | NP
   |devise
   | PRPS
   | JJ
   | NNS
   | its
   | next
   | projects
```

“Linearised” Output Sequence
BOS (ROOT (S (NP NNP )NP (VP VBZ (ADVP RB )ADVP (VP VBG (NP PRP$ JJ NNS )NP (PP IN (NP NNP )NP )PP (SBAR (S (VP TO (VP VB (NP PRP$ JJ NNS )NP )VP )VP )S )SBAR )VP )VP . )S )ROOT EOS

Dependency Parse Tree

```
SUBJ
   | ROOT
   | AUX
   | ADVMOD
   | POSS
   | ADVP
   | PP
   | DOBJ
   | AUX
   | PREP
   | DOBJ
   | SUBJ
   | PUNCT
```

“Linearised” Output Sequence
BOS SHIFT SHIFT SHIFT SHIFT L_ADVMOD L_AUX L_NSUBJ SHIFT SHIFT SHIFT L_AMOD L_POSS R_DOBJ SHIFT SHIFT R_POBJ R_PREP SHIFT SHIFT L_AUX SHIFT SHIFT SHIFT L_AMOD L_POSS R_DOBJ R_XCOMP SHIFT R_PUNCT EOS

Table 7.4 Illustrating examples of linearisation for constituency and transition-based dependency parsing.
### Table 7.5 Evaluation results of our baseline seq2seq models on the dependency parsing task.

We denote: Unlabeled attachment Score (UAS), Labeled Attachment Score (LAS), number of errors with less shifts (LS), number of errors with more shifts (MS); and root errors (RO).

<table>
<thead>
<tr>
<th>Setting</th>
<th>UAS</th>
<th>LAS</th>
<th>Parsing Errors</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td>LS</td>
</tr>
<tr>
<td>SOTA (Andor et al., 2016)</td>
<td>94.61</td>
<td>92.79</td>
<td></td>
</tr>
<tr>
<td>seq2seq (base) (Wiseman and Rush, 2016)</td>
<td>88.53</td>
<td>84.16</td>
<td></td>
</tr>
<tr>
<td><strong>our baseline seq2seq model</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><em>base_small</em></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>- greedy</td>
<td>88.98</td>
<td>85.76</td>
<td>23</td>
</tr>
<tr>
<td>- beam 5</td>
<td>89.58</td>
<td>86.61</td>
<td>22</td>
</tr>
<tr>
<td>- beam 10</td>
<td>89.57</td>
<td>86.62</td>
<td>23</td>
</tr>
<tr>
<td><em>base_medium</em></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>- beam 5</td>
<td>89.95</td>
<td>86.81</td>
<td>14</td>
</tr>
<tr>
<td><em>base_large</em></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>- beam 5</td>
<td>88.97</td>
<td>85.94</td>
<td>7</td>
</tr>
</tbody>
</table>

The standard RNN-based attentional models, inspired from the NMT literature. Our seq2seq models have the following configurations:

- **base_small**: bidirectional GRU encoder layer, 2 GRU decoder layers, 512 input/hidden dims, 256 attention dimension, source word frequency cutoff 2, case-sensitive in source, linearised target sequence with SHIFT, LEFT_ARC and RIGHT_ARC tokens with 79 types.

- **base_medium**: similar to *base_small* except for 2 bidirectional GRU encoder layers, 4 GRU decoder layers.

- **base_large**: similar to *base_small* except for 8 bidirectional GRU encoder layers, 8 GRU decoder layers.

We evaluated our baseline seq2seq models using standard measures with UAS/LAS. In addition, in order to study how well the seq2seq model captures the semantics of the linearised target sequence, we propose 3 different parsing errors, including:

- **LS**: to count the number of cases in which the target sequence contains less “SHIFT” tokens than the number of tokens in the input sequence.

- **MS**: to count the number of cases in which the target sequence contains more “SHIFT” tokens than the number of tokens in the input sequence.
• **RO**: to count the number of cases in which the “ROOT” token does not exist or there are multiple “ROOT” tokens in the target sequence.

Table 7.5 gives an intuition on the performance of the seq2seq models on the dependency tasks. Our baseline seq2seq models performed on par with the reported seq2seq model (Wiseman and Rush, 2016). Here, we do not aim to reach the SOTA performance (Andor et al., 2016) on the task; but we would like to study implicit errors caused by the seq2seq model. As can be seen from the table, although all of our baseline seq2seq models provide relatively good results in terms of UAS/LAS scores; however, they contain many parsing errors. Note that these kinds of errors lead to not recover a valid and meaningful dependency parse tree. We attempted different decoding methods (greedy, beam search with different beam sizes) and network sizes; but the parsing errors still exist. It means that the seq2seq model is unable to fully capture the semantics of the linearised target sequence.

Inspired from this observation, we believe that the seq2seq model needs decoding with constraints. We believe that it is considerably hard to obtain with standard decoding methods such as: greedy, beam search, or re-ranking. Here, we investigate whether our proposed decoding method based on relaxed continuous optimisation is able to perform constrained decoding.

### 7.7.2 Formulation of Constrained Decoding

Let’s recall our formulation of relaxed continuous optimisation as follows:

$$
\arg \min_{\{\hat{y}_i\}_{i=1}^{C}} Q\left(\{\hat{y}_i\}_{i=1}^{C}\right) = \arg \min_{\{\hat{y}_i\}_{i=1}^{C}} \sum_{i=1}^{C} \hat{y}_i \cdot \log \text{softmax}\left( f_{\Theta}\left(\hat{y}_{<i}, x\right)\right) 
$$

subject to

$$
\forall i \in \{1 \ldots C\} : \hat{y}_i \in \Delta_{|V_T|}.
$$

where $V_T$ is output vocabulary, $C$ is the expected length of output sequence to be generated from the input sequence $x$; $y_{<i}$ is previously-generated output sequence; $\Delta_{|V_T|}$ is the $|V_T|$-dimensional probability simplex, i.e., $\{\hat{y}_i \in [0,1]^{V_T} : \|\hat{y}_i\|_1 = 1\}$, and $f$ is unnormalised prediction probability from the attentional model with pre-trained parameters $\Theta$.

Adapted from the above formulation, the constrained decoding can be formulated as:

$$
\arg \min_{\{\hat{y}_i\}_{i=1}^{C}} Q\left(\{\hat{y}_i\}_{i=1}^{C}\right) = \arg \min_{\{\hat{y}_i\}_{i=1}^{C}} \sum_{i=1}^{C} \hat{y}_i \cdot \log \text{softmax}\left( f_{\Theta}\left(\hat{y}_{<i}, x\right)\right) 
$$

subject to

$$
\forall i \in \{1 \ldots C\} : \hat{y}_i \in \Delta_{|V_T|}
$$

$$
\{\hat{y}_i\}_{i=1}^{C} \in C,
$$
where $C$ is a task-specific constraint space. Later, we will elaborate the constraint space $C$ according to specific structured prediction tasks using the, e.g., constituency and dependency parsing.

Let’s consider some constraints for the two case studies in syntactic parsing, including dependency and constituency parsing.

### 7.7.2.1 Constraints for Dependency Parsing

We realise the constraint space $C$ in Equation 7.12 with different constraints pertaining to **adequacy** and **validity** of a linearised dependency tree sequence.

The **validity** constraint implies the **projectivity** of a parse tree. Concretely, for ARC-LEFT or ARC-RIGHT actions, given an action at time step $t$, there must be at least two items left on the stack, e.g.,

$$
\sum_{l \in \mathcal{AL}} \hat{y}_{t,l} + \sum_{r \in \mathcal{AR}} \hat{y}_{t,r} \leq \sum_{i=1}^{t-1} \left( \hat{y}_{S,SHIFT} - \sum_{l \in \mathcal{AL}} \hat{y}_{s,l} - \sum_{r \in \mathcal{AR}} \hat{y}_{s,r} \right) - 1, \quad \forall t \in [1, \ell], \quad (7.13)
$$

where $\mathcal{AL}$ and $\mathcal{AR}$ are lists of ARC-LEFT (e.g., L_ADVMOD, L_NSBJ) and ARC-RIGHT (e.g., R_DOBJ, R_PUNCT) symbols in the output sequence, respectively.

The **adequacy** constraint is to guarantee that the number of SHIFTs (in dependency parsing) and non-bracketing tokens in the output sequence must be equal to the number of tokens in the input sequence (denoted as $|x|$), formulated as:

$$
\sum_{t=1}^{\ell} \hat{y}_{t,SHIFT} = |x| \quad (7.14)
$$

$$
\sum_{t=1}^{\ell} \left( \sum_{l \in \mathcal{AL}} \hat{y}_{t,l} + \sum_{r \in \mathcal{AR}} \hat{y}_{t,r} \right) = |x| - 1 \quad (7.15)
$$

### 7.7.2.2 Constraints for Constituency Parsing

In general, it is not guaranteed that the output sequence produced by neural seq2seq model(s) satisfies the properties of a constituency parse tree. So, we have to realise the constraint space $C$ in Equation 7.12 based on the **adequacy** and **validity** constraints of a linearised constituency tree sequence.

The **validity** constraint says that the count of open bracketing tokens (e.g., (NP, (ADVP) must be equal to the count of close bracketing tokens (e.g., )NP, )ADVP). Concretely, we
formulate this property to form the **validity** constraint as follows:

$$\forall i \in [1, \ell) : \sum_{i = 1}^{i} \sum_{j \in L_{open}} \hat{y}_{i,j} \geq \sum_{i = 1}^{i} \sum_{j \in L_{close}} \hat{y}_{i,j}, \text{ and} \quad (7.16)$$

where $L_{open}$ and $L_{close}$ are lists of open (e.g., (NP, (ADVP) and close (e.g., )NP, )ADVP) bracketing tokens in the output sequence, respectively.

Also, the **adequacy** constraint is to guarantee that the number of non-bracketing tokens in the target sequence must be equal to the number of tokens in the input sequence, formulated as:

$$\sum_{i = 1}^{\ell} \sum_{j \in L_{NB}} \hat{y}_{i,j} = |x| \quad (7.17)$$

where $L_{NB}$ is a list of non-bracketing tokens (e.g., NNP, VBZ) in the target sequence.

### 7.7.3 Optimisation For Constrained Decoding

We introduce possible approaches for constrained decoding, including:

#### 7.7.3.1 The KKT Approach

We hope that the above constraints guarantee the model to produce plausible generations that are convertible to a valid constituency or dependency parse trees. Realising these constraints in our proposed relaxed optimisation framework (as in Equation 7.12) gives rise to a constrained optimisation problem. In order to solve it, we propose to use the approach based on Lagrange multipliers and the Karush-Kuhn-Tucker (KKT) conditions; hence, we call it the KKT approach.

Considering the constraints in dependency parsing, we introduce a set of multipliers $\Lambda \in \mathbb{R}$ for each type of linear constraints, e.g., $\lambda' \in \mathbb{R}$ for the equality constraint in Equation 7.14; $\forall i \in [1, \ell'] : \lambda_i' \geq 0$ for the inequality constraint in Equation 7.13; and $\lambda'' \in \mathbb{R}$ for the equality constraint in Equation 7.15.

The main goal is to search for the optimality conditions (or so-called KKT conditions) through optimising the Lagrangian function $L_{c.opt}$ with respect to relaxed variables and mul-
\textbf{Algorithm 10} The Minimax algorithm for constrained optimisation.

1: \textbf{while} not converged \textbf{do} \hfill $\triangleright$ maximum iterations required
2: \hspace{1em} Use EG as in Section 7.3.1 for $\arg\min_{\hat{y}_1, \ldots, \hat{y}_\ell} L(\{\hat{y}_1, \ldots, \hat{y}_\ell\}, \Lambda)$ \hfill $\triangleright$ Min step
3: \hspace{1em} Use SGD for $\arg\max_\Lambda L(\{\hat{y}_1, \ldots, \hat{y}_\ell\}, \Lambda)$ \hfill $\triangleright$ Max step
4: \textbf{return} $\hat{y}_{1}^{\text{best}}, \ldots, \hat{y}_{\ell}^{\text{best}}$

Multipliers, formulated as:

$$L_{\text{copt}} \left( \{\hat{y}_1, \ldots, \hat{y}_\ell\}, \lambda^{'}, \{\lambda^{''}_i\}_{i=1}^\ell, \lambda^{'''} \right)$$

$$= Q(\hat{y}_1, \ldots, \hat{y}_\ell) + \lambda^{'f} (\hat{y}_1, \ldots, \hat{y}_\ell) + \sum_{i=1}^\ell \lambda^{''} g_i (\hat{y}_1, \ldots, \hat{y}_\ell) + \lambda^{'''} h (\hat{y}_1, \ldots, \hat{y}_\ell)$$

s.t.

\begin{enumerate} 
  \item $\forall i \in [1, \ell] : \hat{y}_i \in \Delta_{|V_i|}$
  \item $\forall i \in [1, \ell] : \lambda^{''}_i \geq 0; \lambda^{'}, \lambda^{'''} \in \mathbb{R}$
  \item $f(\hat{y}_1, \ldots, \hat{y}_\ell) = \sum_{i=1}^\ell \hat{y}_{i,\text{SHIFT}} - |x|$
  \item $g_i (\hat{y}_1, \ldots, \hat{y}_\ell) = \sum_{s=1}^{i-1} \left( \hat{y}_{s,\text{SHIFT}} - \sum_{l \in \mathcal{L}_{AL}} \hat{y}_{s,l} - \sum_{r \in \mathcal{L}_{AR}} \hat{y}_{s,r} \right) - 1 - \left( \sum_{l \in \mathcal{L}_{AL}} \hat{y}_{l,j} + \sum_{r \in \mathcal{L}_{AR}} \hat{y}_{l,r} \right)$
  \item $h (\hat{y}_1, \ldots, \hat{y}_\ell) = \sum_{i=1}^\ell \left( \sum_{l \in \mathcal{L}_{AL}} \hat{y}_{l,j} + \sum_{r \in \mathcal{L}_{AR}} \hat{y}_{l,r} \right) - (|x| - 1)$
\end{enumerate}

(7.18)
Similarly, we can formulate the constrained optimisation problem for constituency parsing as follows:

$$
L_{const} \left( \{ \hat{y}_1, \ldots, \hat{y}_{\ell-1} \}, \lambda', \{ \lambda''_i \}_{i=1}^{\ell-1}, \lambda''' \right) 
= Q(\hat{y}_1, \ldots, \hat{y}_{\ell}) + \lambda' f(\hat{y}_1, \ldots, \hat{y}_{\ell}) + \sum_{i=1}^{\ell-1} \lambda''_i g_i(\hat{y}_1, \ldots, \hat{y}_{\ell-1}) + \lambda''' h(\hat{y}_1, \ldots, \hat{y}_{\ell})
$$

s.t.

a) \( \forall i \in [1, \ell] : \hat{y}_i \in \Delta_{|V|} \)

b) \( \forall i \in [1, \ell) : \lambda''_i \geq 0; \lambda', \lambda''' \in \mathbb{R} \)

c) \( f(\hat{y}_1, \ldots, \hat{y}_{\ell}) = \sum_{i=1}^{\ell} \sum_{j \in \mathcal{N}_B} \hat{y}_{i,j} - |x| \)

d) \( g_i(\hat{y}_1, \ldots, \hat{y}_{\ell-1}) = \sum_{i=1}^{\ell} \sum_{j \in \mathcal{N}_open} \hat{y}_{i,j} - \sum_{i=1}^{\ell} \sum_{j \in \mathcal{N}_close} \hat{y}_{i,j} \)

e) \( h(\hat{y}_1, \ldots, \hat{y}_{\ell}) = \sum_{i=1}^{\ell} \sum_{j \in \mathcal{N}_open} \hat{y}_{i,j} - \sum_{i=1}^{\ell} \sum_{j \in \mathcal{N}_close} \hat{y}_{i,j} \)

In the above Lagrangian functions 7.18 and 7.19, the exponentiated gradient algorithm (as presented in Section 7.3.1) can be used to solve the simplex constraints; whereas the other constraints can be optimised via stochastic gradient descent (SGD) algorithm. We propose the Minimax algorithm for solving these Lagrangian functions as shown in Algorithm 10.

### 7.7.3.2 The Projection Approach

We also propose an alternative approach based on the projection onto constraint space, including two steps:

The first step is to employ a gradient-based optimisation (i.e., SGD or its variants) for the non-convex objective function \( Q \) with respect to the relaxed variables \( \{ \hat{y}_1, \ldots, \hat{y}_{\ell} \} \), and the second step is projection, formulated as:

$$
\hat{y}'_i = p \left( \hat{y}'_{i-1} - \lambda'_i \frac{\partial Q(\{ \hat{y}_i \}_{i=1}^{\ell})}{\partial \hat{y}_i} \right) = p \left( \hat{y}'_i \right), \forall i \in [i \ldots \ell],
$$

where \( \hat{y}'_i \) is relaxed variable at position \( i \) at timestep \( t \). The \( p \) function is a projection step to ensure that the solution by the SGD step is feasible, e.g., the simplex and/or prior (linear)
constraints are not violated. We realise the \( p \) function as follows:

\[
p = \arg\min_{\{\hat{y}_i\}_{i=1}^\ell} \frac{1}{2} \sum_{i=1}^\ell \Vert \hat{y}_i^p - \bar{y}_i^t \Vert^2
\]

\[\text{s.t.} \quad \hat{y}_i^p 1 = 1 \quad \text{and} \quad \hat{y}_i^p \geq 0, \forall i = [1 \ldots \ell]
\]

\[
\{\hat{y}_i^p\}_{i=[1 \ldots \ell]} \in C.
\]

This is a quadratic program and the objective function is strictly convex given the assumption that the prior constraints are linear and convex. The constrained optimisation problem in Equation 7.21 can be solved by utilising the available state-of-the-art optimisation solvers such as CVXOPT.\(^9\)

### 7.7.4 Discussion

We have presented our proposals for future work in addressing the open problem of constrained decoding for seq2seq models - which we believe it is an exciting and open research problem. We hope that our initial ideas and formulation presented here can encourage other people in the community working on this problem, particularly the extensive experiments for validation.

### 7.8 Summary

This work presents the first attempt in formulating decoding in NMT as a continuous optimisation problem. The core idea is to drop the integrality (i.e., one-hot vector) constraint from the prediction variables and allow them to have soft assignments within the probability simplex while minimising the loss function produced by the neural model. We have provided two optimisation algorithms – exponentiated gradient (EG) and stochastic gradient descent (SGD) – for optimising the resulting contained optimisation problem, where our findings show the effectiveness of EG compared to SGD. Thanks to our framework, we have been able to decode when intersecting left-to-right and right-to-left as well as source-to-target and target-to-source NMT models. Our results show that our decoding framework is effective and leads to substantial improvements in translations generated from the intersected models, where the typical greedy or beam search algorithms are not applicable.

This work raises several compelling possibilities for future work. First, we want to improve decoding speed, integrating additional constraints such as word coverage and fertility

\(^9\)http://cvxopt.org
into *decoding*,\(^{10}\) and applying our method to other intractable structured prediction problems. Also, we aim to improve the method with faster run-time of decoding, in order to make the method practical for widespread use.

\(^{10}\)These constraints have been used for training in previous work (Cohn et al., 2016).
Chapter 8

Conclusions and Future Work

Contents

8.1 Conclusions .................................................. 183
8.2 Future Work .................................................. 185
8.3 Summary ..................................................... 186

8.1 Conclusions

This thesis presents a comprehensive study on different aspects of using prior and external knowledge for neural sequence models. Neural sequence models have shown to be very successful for a wide range of NLP tasks, particularly in neural language modelling and machine translation. In NLP, traditional machine learning methods (before deep learning era) have proved for usefulness of learning from prior and external knowledge. In this study, we examine whether they have similar effects and benefits for neural sequence models. We have explored different kinds of prior and external knowledge, including side information (or meta-data), linguistics factors, monolingual data, bidirectional and bilingual factors. There were not straightforward ways to incorporate such prior and external knowledge into neural sequence models; therefore, we have explored different approaches and architectures for effective and efficient incorporation of such knowledge, both in the training and decoding phases. We also find that such prior and external knowledge can be complementary in some situations where the training data is limited.

This thesis makes three major contributions. Our first major contribution is to provide empirical evidence that neural sequence models benefit from side information (a.k.a. meta-data). Side information is often ignored or not fully explored in previous machine learning
methods. In our study, we have improved the performance of neural sequence models including a recurrent neural network language model with auxiliary side information (e.g. keywords, title, description, topic headline) of a textual utterance; and a neural machine translation model with auxiliary side information (e.g. topic information, personal trait). Similar in spirit, we also examine the benefits of other kinds of prior and external knowledge including: linguistics factors and monolingual data in improving neural sequence models. Noticeably, we find that incorporating such prior and external knowledge is particularly useful in situations where the training data is limited or unavailable.

The second major contribution is a new training method based on the method of moments in statistics that can be plugged into any neural sequence models. We have successfully derived a novel principled method for exploiting global features of generated sequences in a neural seq2seq model in order to improve the overall quality of the predicted output sequences. Our proposed method is based on minimizing the distance between the expectations (i.e., moments) of global features that encode prior knowledge about the expected sequence in the empirical data and in the generated data. As a result, the new training method provides a moment matching objective that can be used in combination with the standard cross-entropy objective of a neural seq2seq model. Mathematically, we have also proved that our proposed moment matching technique is capable of reducing the number of samples that need to be drawn from the seq2seq model during the training process, while preserving the unbiasedness of the stochastic gradient descent estimate. The proposed MM technique is a step towards solving critical problems that neural seq2seq models have difficulties in controlling certain important global features of the generated output sequences.

The final contribution is a novel decoding framework for neural machine translation based on continuous optimisation for the integration of global factors into the decoding process. We present a principled method for decoding in which we reformulate conventional decoding, a discrete optimisation problem, as a continuous optimisation problem. The resulting continuous optimisation problem allows us to make use of efficient gradient-based techniques, such as stochastic gradient descent and exponentiated gradient methods, widely-used for training neural models in the literature. The decoding framework based on continuous optimisation is powerful, enabling decoding in complete search space, e.g., resulting from the intersection of several different NMT models, including bidirectional (left-to-right and right-to-left) models as well as bilingual (source-to-target and target-to-source) models. As a result, our proposed decoding framework is more effective and leads to substantial improvements in translation quality, particularly in circumstances where greedy search and beam search are not feasible. We also find that how our decoding framework is competitive with, and complementary to re-ranking.
8.2 Future Work

We believe that many findings in this thesis are intriguing and generally applicable to other tasks in NLP. Furthermore, we discuss potential research directions that might further improve our methods presented above.

Here, we put forward some of the ideas:

- **An end-to-end back-translation approach.** Back-translation (Sennrich et al., 2016c) is an effective approach for leveraging monolingual data for improving the existing NMT models. It is further improved with the *iterative* strategy as presented in Chapter §5. The disadvantage of (iterative) back translation is that it requires pipeline processing from practitioners, e.g., selecting appropriate monolingual data, and using decoding algorithms properly (Edunov et al., 2018a). We further propose an end-to-end back translation approach as a future avenue. The main idea here is to use a “soft prediction” instead of using a discrete back-translation produced by a reverse NMT system given a monolingual sentence. Note that soft prediction is the relaxed version of one-hot vector representation, inspired by our proposed relaxation technique in Chapter §7. Thanks to the use of soft prediction, the back translation approach can be trained in an end-to-end manner.

- **Further research with relaxation optimisation framework.** Our study on converting decoding with discrete optimisation to continuous optimisation based on the relaxation technique is powerful. However, we notice that one important concern is its efficiency in practice. Further research has to be done to make it more practical. For example, in order to reduce the extremely high-dimensional optimisation with relaxed variables over the output symbols, we should consider an alternative, for example, using relaxed variables over the word embeddings instead. The advantage of this alternative is that using word embeddings potentially leads to much lower-dimensional optimisation, hence, faster decoding can be achieved.

Also, instead of doing the gradient-based optimisation methods, we might consider using the evolution strategy (ES) (Salimans et al., 2017) and its variants, e.g., CMA-ES (Hansen, 2016; Loshchilov and Hutter, 2016). The deficiency of gradient-based optimisation methods is the back-propagation step, which is expensive in high dimensional spaces. ES, particularly its variants CMA-ES, approximates the back-propagation step via a distributed computation. Furthermore, CMA-ES is good at quickly searching the space globally, after which we could switch to gradient-based methods at the end for better fine tuning.
Another possible extension of the proposed decoding framework is constrained decoding for solving structured prediction problems. We have suggested some possible ideas in §7.7 for constituency and dependency parsing. Further research has to be done to validate these ideas. We also believe this is highly applicable for other problems that often show benefits to using additional constraints, such as summarisation.

- **Extending moment matching framework.** Chapter §6 has shown our proposed moment matching framework. We have provided its theoretical aspects and proofs, particularly with the “unbiasedness” problem. Our initial experiments show its promise. Future avenue might include some directions. First, further empirical experiments with large scale data sets and exploration of more external and prior features for NMT will be interesting. Second, it would be interesting for combining two following ideas, including the back-translation approach (in §5) and the moment matching framework (in §6). The main idea is to use scores from back-translations as feature functions within the moment matching framework. Lastly, we have provided an idea of applying this moment matching framework in §6.4 for unsupervised domain adaptation problem. We believe this would be an interesting research direction in practical applications of NMT.

### 8.3 Summary

This thesis provides further evidence on utilising prior and external knowledge for better understanding and improving neural sequence models both in training and decoding phases; although there are still many interesting research questions remaining to be investigated as discussed §8.2. We hope that this PhD research serves as an useful reference for practitioners in applying neural sequence models in practice.
References


References


References


References


References


Appendix A

Translation Examples
## Translation Examples

<table>
<thead>
<tr>
<th>Source</th>
<th>我确定我昨天给旅馆打电话并且做了预订。</th>
</tr>
</thead>
<tbody>
<tr>
<td>beam dec</td>
<td>i'm sure i called the hotel reservation and i made a reservation.</td>
</tr>
<tr>
<td>EGdec</td>
<td>i'm sure i called the hotel yesterday and i made a reservation.</td>
</tr>
<tr>
<td>Reference</td>
<td>当我到路口时我这边的灯是绿色的。</td>
</tr>
<tr>
<td>beam dec (l2r)</td>
<td>my light was green when i got to the intersection.</td>
</tr>
<tr>
<td>beam dec (r2l)</td>
<td>this was the green UNK i came in my room.</td>
</tr>
<tr>
<td>EGdec</td>
<td>this was the green UNK i came in my room.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Source</th>
<th>请大声点讲。</th>
</tr>
</thead>
<tbody>
<tr>
<td>beam dec (l2r)</td>
<td>please speak a little louder.</td>
</tr>
<tr>
<td>beam dec (r2l)</td>
<td>a little UNK, please.</td>
</tr>
<tr>
<td>EGdec</td>
<td>please speak some more UNK.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Source</th>
<th>我很快会变好吗?</th>
</tr>
</thead>
<tbody>
<tr>
<td>beam dec (l2r)</td>
<td>will i get better soon?</td>
</tr>
<tr>
<td>beam dec (r2l)</td>
<td>will i be ready soon?</td>
</tr>
<tr>
<td>EGdec</td>
<td>will i be well soon?</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Source</th>
<th>驾驶员,能在这停车吗?</th>
</tr>
</thead>
<tbody>
<tr>
<td>beam dec (l2r)</td>
<td>UNK, can i park here?</td>
</tr>
<tr>
<td>beam dec (r2l)</td>
<td>could you stop here, please?</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Source</th>
<th>那些是免税物品。</th>
</tr>
</thead>
<tbody>
<tr>
<td>beam dec (l2r)</td>
<td>those are duty- free items.</td>
</tr>
<tr>
<td>beam dec (r2l)</td>
<td>those are duty- free items.</td>
</tr>
<tr>
<td>EGdec</td>
<td>those are tax-free items.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Source</th>
<th>有对孩子们的折扣吗?</th>
</tr>
</thead>
<tbody>
<tr>
<td>beam dec (l2r)</td>
<td>do you have a discount for that?</td>
</tr>
<tr>
<td>beam dec (r2l)</td>
<td>do you have any reduction for children?</td>
</tr>
<tr>
<td>EGdec</td>
<td>do you have a discount for the?</td>
</tr>
</tbody>
</table>

**BTEC zh→en**
Translation examples for Chapter 6 — A Novel Inference Framework with Relaxed Continuous Optimisation

TED Talks de—en

Source: durchhaltenvermögen ist leidenschafthaft und ausdauernd für sehr langfristige Ziele.
Reference: grit is passion and perseverance for very long-term goals.
EGdec: persistence is passion and resilience for very long-term targets.
+bilingual: resilience is passion and perseverance for very long-term targets.

Source: man muss aber nicht erst eifersüchtig werden, um ehrlich zu sagen, dass es arbeit ist. stimmt’s?
Reference: but you do not have to be jealous to concede that it’s hard work. right?
Beam dec: but you do not need to get to get in order to say that it’s hard work. right?
EGdec: but you do not need to become jealous, in order to say that it’s hard work. right?

Source: wir sind doch alle gute Bürger der sozialen Medien, bei denen die Währung Neid ist. stimmt’s?
Reference: we are all good citizens of social media, where the currency is envy.
EGdec: we are all good citizens of social media, where the currency is envy.
+bilingual: we are all good citizens of social media, where the currency is envy. right?

Source: wir alle wollen alternative Energiesoressen, die preislich mithalten können. aber es existieren keine.
Reference: we want alternative energy sources that can compete on price. none exist.
EGdec: we all want to have the source of energy sources, which are the things that we want.
+bilingual: we all want to have alternative sources of energy, which are the things that we can.

Source: wie lehre ich kindern eine solide Arbeitsmoral?
Reference: how do I teach kids a solid work ethic?
EGdec: how do I teach kids a commercial work?
+bilingual: how do I teach kids a robust ethic?

Source: die rest of us are in between, and by the way, the average person in the street is almost exactly midway.
Reference: we are in between them, and by the way, the average citizen is almost right in the middle.
EGdec: we are in between them, and by the way, the average citizens are almost right in the middle.
+bilingual: we lie in between them, and by the way, the average citizen is almost exactly in the middle.

Source: "ich kann nicht glauben, dass Pixar einen Prinzessinnen-Film gemacht hat."
Reference: "I can’t believe Pixar made a princess movie."
EGdec: "oh, i ca n’t believe that UNK has a UNK."
+bilingual: "oh, i ca n’t believe that pixar has made UNK,"

Reference: this one was taken just weeks after 9/11, and find myself trying to explain what had happened that day in ways a five-year-old could understand.
EGdec: this one was taken just after 9/11, and i had to explain what was happening on the day, so that five UNK could understand it.
+bidirectional: this one was taken just after 9/11, and i had to explain what was happening on the day, so that five-year-old could understand it.

Source: mit sieben Jahren sah ich zum ersten mal eine öffentliche Hinrichtung, aber ich dachte, mein Leben in Nordkorea sei normal.
Reference: when i was seven years old, i saw my first public execution, but i thought my life in north korea was normal.
EGdec: at the age of seven, i first saw a public execution, but i thought my life in north korea was normal.
+bidirectional: at the age of seven, i first saw a public execution, but i thought my life in north korea was normal.
<table>
<thead>
<tr>
<th>Source</th>
<th>die premierminister indiens und japans trafen sich in tokio.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Reference</td>
<td>india and japan prime ministers meet in tokyo</td>
</tr>
<tr>
<td>beam dec</td>
<td>the prime minister of india and japan met in tokyo</td>
</tr>
<tr>
<td>EGdec</td>
<td>the prime ministers of india and japan met in tokyo</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Source</th>
<th>pläne für eine stärkere kerntechnische zusammenarbeit stehen ganz oben auf der tagesordnung.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Reference</td>
<td>high on the agenda are plans for greater nuclear co-operation.</td>
</tr>
<tr>
<td>beam dec</td>
<td>plans for stronger nuclear cooperation are at the top of the agenda.</td>
</tr>
<tr>
<td>EGdec</td>
<td>plans for greater nuclear cooperation are at the top of the agenda.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Source</th>
<th>&quot; so wie gewerkschaften sterben , sterben auch die mittelklassejobs , &quot; sagte ellison , ein demokrat aus minnesota und stellvertretender vorsitzender des progressive caucus im kongress.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Reference</td>
<td>&quot; as go unions , so go middle-class jobs , &quot; says ellison , the minnesota democrat who serves as a congressional progressive caucus co-chair.</td>
</tr>
<tr>
<td>beam dec</td>
<td>&quot; just as unions are dying , the same jobs are dying , &quot; said ellison , a democrat from minnesota and deputy chairman of the progressive caucus.</td>
</tr>
<tr>
<td>EGdec</td>
<td>&quot; just as unions are dying , the same jobs are dying , &quot; said ellison , a democrat from minnesota and deputy chairman of the progressive caucus.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Source</th>
<th>als die frühere ஖rst lady eleanor roosevelt den vorsitz über die internationale menschenrechtskommission innehatte , von der die allgemeine erklärung der menschenrechte entworfen wurde , wie sie 1948 von den vereinten nationen als weltweites abkommen übernommen wurde , fügten roosevelt und die anderen verfasser eine garantie ein , dass &quot; jeder das recht hat , gewerkschaften zum schutze seiner interessen zu bilden oder ihnen beizutreten . &quot;</th>
</tr>
</thead>
<tbody>
<tr>
<td>Reference</td>
<td>when former ஖rst lady eleanor roosevelt chaired the international commission on human rights , which drafted the universal declaration of human rights that would in 1948 be adopted by the united nations as a global covenant , roosevelt and the drafters included a guarantee that &quot; everyone has the right to form and to join trade unions for the protection of his interests . &quot;</td>
</tr>
<tr>
<td>beam dec</td>
<td>when former prime minister eleanor roosevelt held the presidency of the international human rights commission designed by the universal declaration of human rights , adopted by the united nations in 1948 as a global agreement , the united states and the other authors set out a guarantee that &quot; everyone has the right to form or join unions to protect their interests . &quot;</td>
</tr>
<tr>
<td>EGdec</td>
<td>when former prime minister eleanor roosevelt held the presidency of the international human rights commission designed by the universal declaration of human rights , adopted by the united nations in 1948 as a global agreement , the united states and the other authors set out a guarantee that &quot; everyone has the right to form or join unions to protect their interests . &quot;</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Reference</td>
<td>besides the 2007 world championship he also won the champions league in may and the ehf-cup in 2008 ( hsg nordhorn ) and 2010 ( tbv lemgo ) .</td>
</tr>
<tr>
<td>rerank</td>
<td>in addition to the title 2007 in 2007 and the win of the champions league 2014 in 2008 ( hsg nordhorn ) and 2010 ( tbv lemgo ) , he won the ehf cup.</td>
</tr>
<tr>
<td>EGdec</td>
<td>in addition to the title championship in 2007 and the win of the champions league 2014 in 2008 ( hsg nordhorn ) and 2010 ( tbv lemgo ) , he won the ehf cup.</td>
</tr>
</tbody>
</table>