# Domain Adaptation for Statistical Machine Translation and Neural Machine Translation

Jian Zhang

B.Sc. in Applied Computational Linguistics

A dissertation submitted in fulfilment of the requirements for the award of

Doctor of Philosophy (Ph.D.)

at the


School of Computing

Dublin City University

Supervised by

Prof. Qun Liu and Prof. Andy Way

April 2017

## Declaration

I hereby certify that this material, which I now submit for assessment on the programme of study leading to the award of Ph.D is entirely my own work, and that I have exercised reasonable care to ensure that the work is original, and does not to the best of my knowledge breach any law of copyright, and has not been taken from the work of others save and to the extent that such work has been cited and acknowledged within the text of my work.

Jian Zhang

Student ID: 56101775

Date: 01 April 2017

## Acknowledgements

It has been a long journey and finally, it is the time to honour the people who helped me along the way to achieve this doctoral degree.

First and foremost, I would like to express my sincere gratitude to my supervisors Prof. Qun Liu and Prof. Andy Way. Doing a Ph.D. is one of the toughest tests for anyone in academic life. Their patience, support, courage and immense knowledge guided me through these years. They make me feel that I am the luckiest person on the planet. I could not imagine having better supervisors to work with, and this work would not have been achievable without their guidance. I would also like to thank my examiners: Prof. Marcello Federico and Prof. Gareth Jones, for their thoughtful and detailed comments in this thesis, and Dr. Monica Ward for chairing my viva.

A heartfelt thanks to Dr. Liangyou Li and Dr. Xiaofeng Wu for making me believe that science doesn't always work at the first attempt but endure a long calmly wait for the breakthrough. I am also indebted to Dr. Jinhua Du, Dr. Yvette Graham, Dr. Teresa Lynn, Dr. Joss Moorkens, Dr. Antonio Toral, Dr. Xiaojun Zhang and Dr. Ergun Biçici for sacrifice their personal time to give me generous advice through out my research. I would like to thank Chris Hokamp and Piyush Arora for organizing a study group in which I discovered my interest for deep learning, and Dr. Joachim Wagner and the Irish Centre for High-End Computing for setting up the best experiment environment. I must also thank Iacer Galixto, Peyman Passban, Longyue Wang, who exchanges of knowledge and skills in the lab, and the management team for providing much-needed assistance with administrative tasks. I happen to have the honour to spend six months in the top academia, the Institute of Computing Technology at Chinese Academy of Sciences, where I had the chance to collaborate with some fantastic researchers. More specifically, I would like to thank Dr. Wenbin Jiang, Dr. Jia Xu, Dr. Mingxuan Wang and Xiang Li for their team work.

I also have a deep appreciation to Dr. John Tinsley, Dr. Alexandru Ceauşu and Dr. Páraic Sheridan from the PLuTO project. It was under their tutelage that I became interested in machine translation, especially on this research topic.

I wish to thank my friends Nicola, Catherine, Steven and Peter, for helping me get through the difficult times and caring they provided.

Last but not the least, I would like to express my deepest gratitude to my family for their support and encouragement not only throughout this journey but also in my life.

# Contents

# List of Figures

# List of Tables

# Abbreviations

**BLEU** Bilingual Evaluation Understudy. viii, ix, 18, 19, 30, 37, 42, 81, 84–86, 103–106, 115, 124

**CBMT** Corpus-based machine translation. 1

**CBOW** Continuous Bag of Words. 22

**CVC** Context Vector Concatenation. 70, 71, 77

**CVS** Context Vector Sum. 70, 71, 77

**EM** Expectation-Maximization. 13, 37

**EOS** end-of-sentence. 30, 92, 103, 112

**GCVA** Gated Context Vector Adaptation. 72, 77

**GD** General-Domain. 4–7, 34–40, 43–47, 49–52, 54, 55, 57–63, 66–68, 70–77, 79, 85, 87, 88, 117, 118, 120

**GDA** Gated Domain Adaptation. 72, 77, 79–81, 84–88, 119–122

**GRU** Gated Recurrent Unit. 24–26, 30, 33, 75, 88, 98, 100, 103, 122

**GWVA** Gated Word Vector Adaptation. 71, 72, 77

**HTMM** Hidden Topic Markov Model. 4, 7, 39, 91, 93, 94, 103, 105, 106, 115, 119

**ID** In-Domain. 4–6, 8, 34–40, 43–47, 49–52, 54, 55, 59–63, 66–68, 70, 72, 74–77, 79, 87, 88, 117, 118, 120, 121

**IWSLT** International Workshop on Spoken Language Translation. 51–55, 59–62

**LDA** Latent Dirichlet Allocation. 4, 7, 39, 91–94, 103–106, 113, 114, 119

**LM** Language Model. ix, 6, 8, 10, 11, 16, 17, 19, 23, 25, 27, 28, 33–35, 39, 40, 42, 49, 50, 52, 65–67, 69, 74–77, 80–88, 103, 118–120, 123, 124

**LSTM** Long Short-Term Memory. 24, 122

**MT** Machine Translation. viii, 1–3, 5–8, 10, 18, 34, 38, 42, 64, 65, 85, 87, 88, 94, 102, 103, 114, 116–120, 124

**NIST** National Institute of Standards and Technology. 85, 86, 102, 105, 106, 110, 113

**NLP** Natural Language Processing. 1, 5, 10, 21, 66, 84, 118

**NMT** Neural Machine Translation. viii, ix, 1, 2, 7, 8, 10, 19, 28, 30, 31, 33, 34, 40, 42, 65, 74, 84–86, 88–92, 94, 95, 97, 98, 100–106, 108, 110, 112–115, 117–120, 122–124

**PBSMT** Phrase-based Statistical Machine Translation. 10, 13, 15, 19, 42, 45, 52, 54, 81, 85, 86, 117, 123

**RNN** Recurrent Neural Network. 5–8, 10, 19, 23–25, 27–29, 32, 33, 67, 69–71, 76, 77, 87, 98, 100, 103, 114, 117, 118, 120, 124

**SGD** Stochastic Gradient Descent. 75, 85

**SMT** Statistical Machine Translation. 1, 2, 5–7, 11–13, 16–19, 34–36, 38, 39, 42, 43, 46, 49, 50, 52, 54, 63, 65, 66, 81, 84, 88, 103, 117, 119, 120

**SVM** Support Vector Machine. 47, 48

**TED** Technology, Entertainment, Design. 51, 59, 62

**UNK** Unknown. 7, 8, 27, 29, 75, 80, 85, 87, 92, 103, 106, 110–113, 119, 121

**WCVC** Weighted Context Vector Concatenation. 70, 71, 77

**WCVS** Weighted Context Vector Sum. 71, 77

**WMT** Workshop on Statistical Machine Translation. 51–54

**WVC** Word Vector Concatenation. 68, 71, 76, 77

**WVS** Word Vector Sum. 69, 71, 76

**WWVC** Weighted Word Vector Concatenation. 68, 71, 77

**WWVS** Weighted Word Vector Sum. 69, 71, 77

# Domain Adaptation for Statistical Machine Translation and Neural Machine Translation

Jian Zhang

## Abstract

Both Statistical Machine Translation and Neural Machine Translation (NMT) are data-dependent learning approaches to Machine Translation (MT). The prerequisite is a large volume of training data in order to generate good statistical models. However, even if a large volume of training corpora are available for MT, finding training data which are similar to the specific domains is still difficult. The MT models trained using the limited specific domain data cannot have sufficient coverage on the linguistic phenomena in that domain, which makes this a very challenging task. Because word meanings, genres or topics differ between domains, using additional data from other domains can increase the dissimilarities between the training and testing data, and result in reduced translation quality. Such a challenge is defined as the 'domain adaptation' challenge in the literature. In this thesis, we investigate domain adaptation in two different scenarios, namely a *domain-awareness* scenario and a *domain-unawareness* scenario.

In a *domain-awareness* scenario, the domain information is given explicitly in the training data. We are interested in developing domain-adaptation techniques which transfer knowledge gained from other domains to a desired domain. In the approach proposed here probabilistic values indicating the domain-likeness features for words are estimated by the context rather than by the words themselves. We then apply those features to the combined translation models in an MT system. We empirically show that translation quality can be significantly improved, i.e. absolute 0.36 (1.3% relative) and 0.82 (2.66% relative) Bilingual Evaluation Understudy (BLEU) scores in two experiments, compared with previous related work.

We then turn our interest to the recently proposed neural network training (Cho et al. 2014, Sutskever et al. 2014). We describe a domain-adaptation approach which can exploit large pre-trained word vector models. We evaluate our approach on both Language Model (LM) and NMT tasks to demonstrate its efficiency, effectiveness and flexibility in a *domain-awareness* scenario. We observe that the proposed approach can reduce the perplexity by 7.4 points compared to the baseline neural network LM. In the NMT experiment, we achieve 0.82 (absolute, 2.3% relative) and 0.42 (absolute, 1.3% relative) improvements in BLEU score on two test sets compared with the NMT model without adaptation.

In a *domain-unawareness* scenario, the domain information is not given explicitly in the training data. The training data is heterogeneous, e.g. originating from tens or even hundreds of different resources without well-defined domain labels. We overcome such a challenge by deriving the topic information from the training corpora using well-estimated topic modelling algorithms. In this scenario, we pay particular attention to the most recent NMT framework. We are concerned with improving the overall translation quality. Experimentally, we show that our model can improve 1.15 (absolute, 3.3% relative) and 1.67 (absolute, 5.4% relative) in BLEU score in contrast with the NMT baseline model.

# Chapter 1

# Introduction

Corpus-based machine translation (CBMT), e.g. Statistical Machine Translation (SMT) or Neural Machine Translation (NMT), is an active research topic in Natural Language Processing (NLP) with attention from many researchers. The training phase is a data-dependent process. The translation quality of a CBMT system is strongly influenced by the quantity and the quality of the training data (Bertoldi and Federico 2009, Haddow and Koehn 2012, Wang et al. 2012, Luong and Manning 2015). The quality factor means that the training data needs to be clean and drawn from the same domain as the testing data. The quantity factor requires that the size of the training data should be large enough to cover the linguistic phenomena in the desired domain. Therefore, a prerequisite for a Machine Translation (MT) system, e.g. a CBMT system, is to collect as much high-quality training data as possible to achieve good performance.

Despite the increasing amounts of data available from the web, they are nevertheless restricted to a limited number of domains and language pairs. Furthermore, human language is a complex system, so it is practically impossible to collect complete knowledge for any language in any domain. Therefore, it is frequently necessary to supplement the scarce training data of the desired domain with some additional data from other domains (Axelrod et al. 2011, Haddow and Koehn 2012).

However, one challenge arises: when there are dissimilarities between the training and

| | |
|---|---|
| ***News*** | Newspapers everywhere carried stories predicting that computer systems would crash on January 1, 2000, causing much of the world to shut down. |
| ***Europarl*** | I propose that we vote on the request of the Group of the Party of European Socialists that the Commission statement on its strategic objectives should be reinstated. |

Figure 1.1: These examples show that domains can be distinguished according to the provenances of the data, where the sentence in the *News* data is more informal than the sentence in the *Europarl* data. These sentences are selected from the News corpus and the Europarl corpus (Koehn 2005).

testing domains, the performance of a MT system decreases. Such a challenge is often referred as 'domain adaptation' in the literature. In this thesis, we report our research on domain adaptation for two MT paradigms: SMT and NMT. The aim of this study is to investigate new domain-adaptation approaches to improve the translation quality.

## 1.1 What are Domains?

In most previous work, the term "domains" refers to the "provenance" of the training data, e.g. Foster and Kuhn (2007), Moore and Lewis (2010), Bisazza et al. (2011), Haddow and Koehn (2012), Sennrich (2012). Essentially, this means that the data in one corpus may be in a different domain than the data in another corpus. Chen et al. (2013) state that "the best translation practice differs widely across genres, topics, and dialects", and define a combination of all these factors to represent domains. Domains can also be interpreted as the difference of words and grammars between corpora (Pecina et al. 2012). Hasler (2015) defines domains as the thematic content in the training data, which might be described as the topics contained in the data.

In this thesis, we follow previous work (Bisazza et al. 2011, Chen et al. 2013) and note that domains depend on several factors, e.g. provenance, genres, topics, dialects or styles, and even the combination of all those factors.

Figure 1.1 and Figure 1.2 demonstrate examples of using *provenance* or *topic* as domains to distinguish domains, respectively. In Figure 1.1, the example sentences have dif-

***Finance*** The consumer price index, a main gauge of inflation, rose 2.7 percent in February, the National Bureau of Statistics announced Thursday.

***Political*** High number reflects the novelty of the policy and program responsibilities and the requirement to enhance public confidence in the government's capacity to respond to national security and terrorism threats.

Figure 1.2: These examples show that domains can be distinguished according to the topics of the data. The sentence in the *Finance* domain describes the study of financial investments, whereas the sentence in the *Political* domain describes the international affairs. The examples are selected from the Common Crawl corpus which are crawled from web pages.

ferent origins, e.g. the News corpus and the Europarl corpus (Koehn 2005). In Figure 1.2, the example sentences are selected from the same corpus – the Common Crawl corpus.[1] However, two sentences are different in topics, e.g. the *Finance* topic or the *Political* topic.

## 1.2 Research Hypothesis

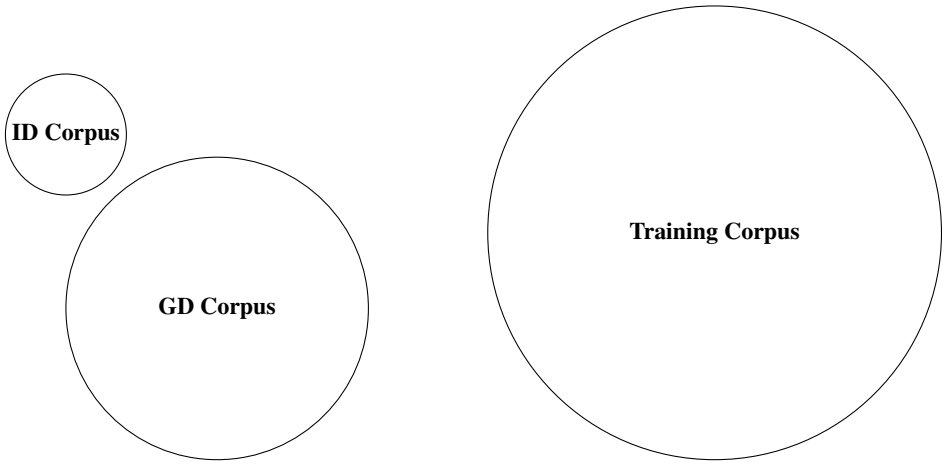
The training data of a MT system could be collected from various resources, e.g. from a small or an extensive domain, with or without well-defined domain labels, or are separated into many corpora or mixed as a single large corpus. Training data in the desired domain might even be unavailable. Motivated by the above challenges, we assume the following two scenarios in this thesis:

***Domain-awareness:*** The domain information is given explicitly in the training data.

***Domain-unawareness:*** The domain information is not given explicitly in the training data.

In a *domain-awareness* scenario, we are interested in developing domain-adaptation techniques which transfer knowledge gained from the other domains to a desired domain. Furthermore, we assume that a small amount of training data in the desired domain is available, namely the In-Domain (ID) training data. We also assume that the training data from

[1] http://commoncrawl.org/




(a) A *domain-awareness* scenario (b) A *domain-unawareness* scenario

Figure 1.3: A *domain-awareness* scenario vs. a *domain-unawareness* scenario. In a *domain-awareness* scenario, the domain information is given explicitly in the training data. Furthermore, we assume that a small amount of training data (the ID corpus) which is close to the desired domain is available and the training data from the other domains (the GD corpus) is large in size. In a *domain-unawareness* scenario, the domain information is not given explicitly in the training data. They may originate in tens or even hundreds of different resources without well-defined domain labels. Therefore, we concatenate all the training data and rely on some topic-learning algorithms to discover the domains.

the other domains – the General-Domain (GD) training data – are large in size, as seen in Figure 1.3a. In this scenario, we are concerned with making better use of the large GD training data to further improve the translation quality in ID.

In a *domain-awareness* scenario, we are aware of which parts of the training data are the ID and which parts are the GD training data. However, this is not always true in practice (Hasler 2015). For example, the training data may come from tens or even hundreds of different resources without well-defined domain labels to distinguish them. Therefore, we focus on a different challenge in a *domain-unawareness* scenario; the domain information is not given explicitly in the training data. However, we can treat the domains as the *latent* variables in the training data and rely on well-established topic-learning algorithms, e.g. Latent Dirichlet Allocation (LDA) (Blei et al. 2003) or Hidden Topic Markov Model (HTMM) (Gruber et al. 2007), to discover the domain information. In this scenario, we are concerned with making a better lexical choice and improving the overall translation quality. Figure 1.3b is an illustration of the training data in a *domain-unawareness* scenario where

we assume that all data from different resources are concatenated into a single corpus.

## 1.3 Motivations and Research Questions

Word and phrase meanings are implied by the context rather than by the words themselves (Banchs 2014). One way to establish the domain for a word or a phrase is to determine the domain of the sentence from which they emanate. For example, if a sentence has a higher probability of being in a domain, the words or phrases within that sentence are also more likely to be in the same domain, and vice versa. Given such observation, our hypothesis is that the context (either words around or even the full sentence) of words can be used as an important feature for domain-adaptation challenge. Accordingly, our first research question is:

> **RQ1** *In a domain-awareness scenario, how can we further improve the current domain adaptation method of an SMT by availing of the domain-likeness of the context in which a word or a phrase appears?*

To answer **RQ1**, we describe a domain-likeness model that can be used to estimate probabilities of bilingual phrase pairs are in ID or GD. Furthermore, we apply estimated probabilities to an SMT system to demonstrate the translation quality improvements over the current domain-adaptation approach (Bisazza et al. 2011), which uses a binary type feature indicating the provenance of the phrase translations.[2] Our probability estimation can be interpreted as the distance from ID to GD, i.e. phrase pairs with lower probability values indicate that they are close to ID; phrase pairs with higher probability values indicate that they are far away from ID and close to GD. The domain-likeness model is trained with features inspired by a commonly used data selection approach (Axelrod et al. 2011).

Because MT technologies have changed rapidly in recent years, our attention is always on the state-of-the-art methods. We move our focus from SMT to the more recently proposed neural network training after **RQ1**.

[2]We will provide a review for current domain-adaptation approaches in Section 2.3.

There are several aspects which motivated us to study domain-adaptation in neural network training. Firstly, recent studies have shown remarkable results in applying neural networks in NLP, especially using Recurrent Neural Network (RNN). For example, Mikolov et al. (2010) report that the perplexity was significantly reduced when using RNN for Language Model (LM) training compared to the traditional *n*-gram LM training. In the field of MT, significant improvements have also been observed (Bahdanau et al. 2015, Luong et al. 2015b, He et al. 2016, Tu et al. 2016) when utilizing neural network training in translation. This success has strongly motivated us to study this approach in the domain-adaptation challenge. Secondly, there is very little work to be found in the MT literature to address the domain-adaptation challenge related to neural network training. Most previous work (Luong et al. 2015b, He et al. 2016, Tu et al. 2016) aims to increase general model performance regardless of domain. Moreover, previous domain-adaptation techniques based on *n*-gram LM and SMT (Foster and Kuhn 2007, Bisazza et al. 2011, Hasler et al. 2012, Zhang et al. 2014b) are not feasible to be transferred directly to neural network framework as the learning algorithms used between the two approaches are different. Thus, there is a need to investigate new approaches under the neural network framework. Finally, neural network training is still a data-driven process, so we expect that the domain-adaptation challenge to be relevant here too.

One of the important building block in a neural network is the word embedding layer, which is used to represent words in the word vectors. Such representations are known to be better at generalization than plain text format (Mikolov et al. 2013b) because the neural network is able to learn which words are semantically close and then switch one to a neighbouring one. A word vector layer (word vector model) can also be pre-trained with a large amount of data (Mikolov et al. 2013a, Pennington et al. 2014) and used to initialize the embedding layer in a neural network in the situation when the relevant training data is limited. Our hypothesis is that the pre-trained and the task-specific-trained word vector models are complementary with each other in a neural network training. The pre-trained word vector models can be applied to overcome the challenge that ID training data is too

small in domain adaptation. Accordingly, our second research question is:

> **RQ2** *Whether the vector model trained using GD data can be used in domain adaptation in a domain-awareness scenario?*

To address **RQ2**, we propose a novel domain-adaptation mechanism in neural network training. Instead of learning and adapting the neural network on millions of training sentences – which can be very time-consuming or even infeasible in some cases – we design a domain-adaptation gating mechanism which can be used in RNN and quickly learn the GD knowledge directly from the pre-trained word vector models with little speed overhead. We make a comparison between several adaptation techniques. Furthermore, we also apply the proposed approach into an NMT system to demonstrate its effectiveness.

While **RQ1** and **RQ2** are based on a *domain-awareness* scenario, we then switch our attention to the *domain-unawareness* scenario. As a matter of fact, a *domain-unawareness* scenario is common in MT training, where a huge amount of training data is often collected regardless of domain. There are no well-defined domain labels. All data are mixed as a single corpus; some sentences can be very close to the testing domain and most not.

When the domain information is not explicitly given, such as in a *domain-unawareness* scenario, one approach is to use the latent domain information captured by the well-established topic-learning algorithms, e.g. LDA or HTMM, to guide the translation process (Hasler et al. 2012, Zhang et al. 2014b) in the SMT framework. However, it is unclear whether and how the similar method can be also applied on the NMT models. Accordingly, our **RQ3** is as follows:

> **RQ3** *How word topic distributions can be used to improve translation quality for NMT models in a domain-unawareness scenario?*

One observation we obtain from MT training data is that some of the words within the same sentence often belong to the same (or similar) topic. The similar "topic consistent" behaviour is also observed by Su et al. (2015). Intuitively, if we can guide the translation

process by maintaining the same topic in the translations, better translations can be produced. We propose to achieve this by incorporating word topic information in source and target sentences in an NMT system. Furthermore, we will find that applying topic information in an NMT system can not only improvement the translation quality, but also lower the number of Unknown (UNK) tokens appearing in the translations.

## 1.4 Outline

In this thesis, we address the domain-adaptation challenge in two scenarios: the *domain-awareness* and *domain-unawareness* scenarios. Overall, the goal is to improve the MT quality by using domain-adaptation techniques. This thesis comprises six chapters including the current introductory chapter.

In Chapter 2, we provide background information about MT models and algorithms. However, we do not try to exhaustively cover all aspects in the field, but to only focus on the work related to this thesis. We will also review related work on domain adaptation for MT.

In Chapter 3, we study the translation model combination approach of Bisazza et al. (2011), where a binary type of provenance feature is used when the models are combined. In our work, we propose a more fine-grained translation model combination approach. The used feature is estimated by a domain-likeness model. We show that our approach can significantly improve translation quality over the previous approach. In our analysis, we also provide phrase pair distributions and examples in the combined translation model.

In Chapter 4, we move our attention to neural network training approaches, particularly on the neural LM (Mikolov et al. 2010) and NMT models (Bahdanau et al. 2015). We study the possibility of adapting large pre-trained word vector models (Collobert et al. 2011, Mikolov et al. 2013a, Pennington et al. 2014) into ID LM training. We propose several adaptation mechanisms. Our work has the advantages of (i) very little computation overhead in the neural network training framework, (ii) benefiting the lower-frequency

words in the training data, and (iii) the flexibility of being used in any sequential network applications when an RNN is used. We present our experimental results on neural LM and NMT to show the efficiency of the proposed approach.

In Chapter 5, our focus is on the NMT models. We first present the "topic consistent" (Su et al. 2015) observation where some of the words within the same sentence often belong to the same or similar topic. Then we propose our topic-based NMT models that are built with the incorporation of word topic information learned from the training data. In our analysis, we show that our models can produce better translations and a lower number of UNK tokens.

We conclude in Chapter 6 with a summary of our work and contributions of the thesis. Finally, we present avenues for future work.

In summary, all of our proposed domain-adaptation approaches and experiments are presented in Chapter 3, 4 and 5, which are related to **RQ1**, **RQ2** and **RQ3**, respectively. **RQ1** and **RQ2** are studied in a *domain-awareness* scenario, and **RQ3** is studied in a *domain-unawareness* scenario.

## 1.5 Related Publications

The published papers which are related with this thesis are as follows:

1. Jian Zhang, Liangyou Li, Andy Way, Qun Liu. (2016). Topic-Informed Neural Machine Translation. In *Proceedings of the 26th International Conference on Computational Linguistics*, pages 1807–1817, Osaka, Japan, December 11-17 2016.

2. Jian Zhang, Xiaofeng Wu, Andy Way, Qun Liu. (2016). Fast Gated Neural Domain Adaptation: Language Model as a Case Study. In *Proceedings of the 26th International Conference on Computational Linguistics*, pages 1386–1397, Osaka, Japan, December 11-17 2016.

3. Jian Zhang, Liangyou Li, Andy Way, Qun Liu. (2014). A Probabilistic Feature-Based Fill-up for SMT. In *Proceedings of the 11th Conference of the Association*

*for Machine Translation in the Americas, (Vol. 1: MT Researchers Track)*, pages 96–109, Vancouver, BC, October 22-26 2014.

# Chapter 2

# Background

In this chapter, we provide basic information about Machine Translation (MT) models and algorithms. MT is an active research field in Natural Language Processing (NLP) and many models and approaches have been intensively studied in the literature. Furthermore, the technologies used in MT have also changed rapidly in recent years. Therefore, we do not try to exhaustively cover all aspects in the field, but only focus on work which is related to this thesis.

This chapter is organized as follows: we first introduce the framework of Phrase-based Statistical Machine Translation (PBSMT), including the models and evaluation metrics in Section 2.1. We then give background information about Neural Machine Translation (NMT) in Section 2.2, including word vector models, Recurrent Neural Network (RNN), RNN Language Model (LM) and NMT models. In Section 2.3, we present related domain-adaptation work in this thesis. Section 2.4 lists the tools used in this thesis. Finally, we summarize the content of this chapter in Section 2.5.

For notational convenience, we use the following notations through this chapter. Assume a sentence pair $F$ and $E$, where $F$ is in the source language, and $E$ is in the target language. $F = \{f_1, f_2, \ldots, f_{l-1}, f_l\}$ and $E = \{e_1, e_2, \ldots, e_{q-1}, e_q\}$, where $l$ and $q$ represent the sentence lengths, and $f$ and $e$ denote the words in the sentences for $F$ and $E$, respectively. We also use $i$ ($1 \leqslant i \leqslant l$) and $j$ ($1 \leqslant j \leqslant q$) to represent the word positions in


Figure 2.1: Noisy channel model

$F$ and $E$, respectively.

## 2.1 Phrase-based Statistical Machine Translation

Statistical Machine Translation (SMT) has received the most research attention since Brown et al. (1990, 1993). The training of an SMT system is a data-driven process, where large amounts of training data are required in order to sufficiently cover the linguistic phenomena for the desired language pair. The training data requires to be *parallel*, where each sentence in the target language is the translation of the corresponding sentence in the source language. An SMT system consists of several models: a translation model is used to translate text from a source language to a target language; an reordering model decides in which order of the translates are produced and a LM is used to evaluate the fluency for the translations. These models are integrated using the log-linear framework (Och and Ney 2002) as feature functions to optimize the model weights.

In the rest of Section 2.1, we formally define the noisy channel model, log-linear framework, translation model, reordering model and LM.

### 2.1.1 The Noisy Channel and Log-linear Framework

Early SMT was based on the classical *noisy channel* model used in speech recognition, as seen in Figure 2.1. In the noisy channel model, a distorted message is observed by the *receiver* and we want to recover the original message sent by the *sender*. In this formulation, the translation problem is regarded as the decoding of the target sentence $E$ given the source sentence $F$ (as seen in Figure 2.1). Based on Bayes decision theory, we can formulate SMT as in Equation (2.1) (Brown et al. 1990, 1993):

$$\begin{aligned}\hat{E} &= \arg\max_{E} P(E|F) \\ &= \arg\max_{E} \frac{P(F|E)P(E)}{P(F)} \\ &= \arg\max_{E} P(F|E)P(E)\end{aligned} \tag{2.1}$$

where $\hat{E}$ denotes the translation output which has the highest translation probability. In a nutshell, we need to find $\hat{E}$ given $F$. In Equation (2.1), the translation problem is factored into $P(F|E)$ and $P(E)$. $P(F|E)$ and $P(E)$ represent the inverse translation probability and language model probability, respectively. The denominator $P(F)$ in Equation (2.1) is ignored since it remains constant for a given source sentence $F$. The advantage of this decomposition is that we can learn separate probabilities in order to compute $\hat{E}$.

The log-linear framework (Och and Ney 2002) is a generalization of the noisy channel approach to formulate SMT, as presented in Equation (2.2):

$$\begin{aligned}\hat{E} &= \arg\max_{E} P(E|F) \\ &= \arg\max_{E}\{\frac{\exp\sum_{m=1}^{M}\lambda_m h_m(E,F)}{\sum_{E'\in E}\exp\sum_{m=1}^{M}\lambda_m h_m(E',F)}\} \\ &= \arg\max_{E}\{\exp\sum_{m=1}^{M}\lambda_m h_m(E,F)\}\end{aligned} \tag{2.2}$$

where $M$ indicates the total number of features, $h_m(E,F)$ indicate feature functions on $F$ and $E$, and $\lambda_m$ are the corresponding optimal weights, which are learned from a small set of parallel sentences. Such a process is often called *tuning*, and the small set of parallel sentences is called the *tuning* (or *development*) set. The denominator in Equation (2.2) is ignored since it is a constant denoting the sum of the probabilities of all possible translations. The log-linear framework has the advantages of including additional feature functions which can usually improve the translation quality and be optimized by only tuning

the feature weights.

### 2.1.2 Translation Model

In Brown et al. (1990, 1993), words are used as the fundamental translation units, namely word-based models. Word-based models introduce the concept of a word-alignment model which maps words in a sentence pair with translation probabilities (word-translation probabilities). A word-alignment model treats word-alignments as hidden variables and can be learned iteratively from a bilingual corpus using the Expectation-Maximization (EM) algorithm (Dempster et al. 1977). The translation probability of a target sentence is composed of the product of word-translation probabilities which are leaned from a bilingual corpus. However, words are not the best translation units because one source word can be translated into multiple words in the target language, and vice versa, there is no local context used in translation. Therefore, the word-based SMT model is not widely used nowadays. The state-of-the-art SMT (Koehn et al. 2003) uses phrases as translation units. A *phrase* is a sequence of words with not necessarily linguistically motivated. Using phrases instead of words has several advantages: we can overcome the obstacles in word-based models and translate multiple words from a source language to a target language as a single unit.

The PBSMT proposed in Koehn et al. (2003) consists of inverse phrase translation probability, inverse lexical translation probability, direct phrase translation probability and direct lexical translation probability. The *bilingual phrase pairs*, i.e. the target phrase is the translation of the source phrase, are firstly extracted based on word alignments using alignment tools, such as GIZA++ (Och and Ney 2003). Because single directional word alignments only allow many-to-one mappings, symmetrized word alignments (by training word alignments in both directions) are often used to obtain many-to-many alignments. The extraction process follows heuristics (i) no word can be aligned to the outside of the extracted phrase pairs, (ii) no phrase can be extracted if only unaligned words can be found.

As seen in Figure 2.2, we cannot extract phrase pair ($f_1$ $f_2$ , $e_1$ $e_2$) since $f_2$ is also aligned with $e_3$ and $e_4$ outside of the extracted phrase pair. As another example, $f_5$ or $f_6$

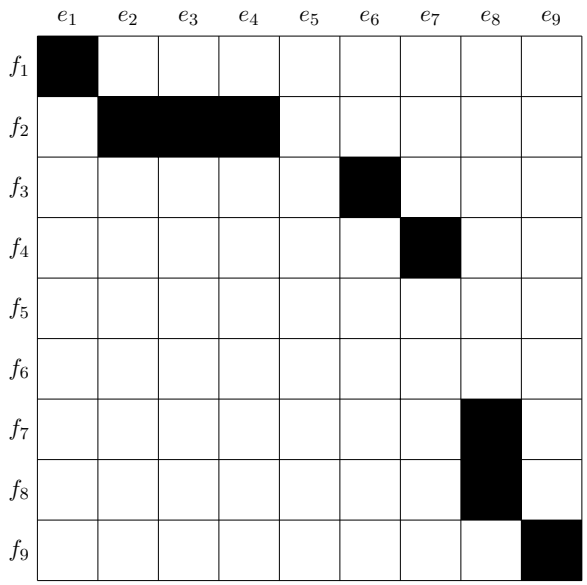



Figure 2.2: An example of phrase pair extraction using "consistent with a word alignment" (Koehn et al. 2003) algorithm. The black colored cells indicate the word alignment, e.g. word $f_1$ is aligned with word $e_1$. We cannot extract phrase pair ($f_1$ $f_2$ , $e_1$ $e_2$) since $f_2$ is also aligned with $e_3$ and $e_4$ outside of the extracted phrase pair. As another example, $f_5$ or $f_6$ cannot be extracted individually since they are not aligned with any $e$.

cannot be extracted individually since they are not aligned with any $e$. Such an extraction algorithm is defined as "consistent with a word alignment" (Koehn et al. 2003).

Given a collection of phrase pairs, the direct phrase translation probability can be estimated using relative frequency as follows:

$$p(\bar{f}|\bar{e}) = \frac{count(\bar{f}, \bar{e})}{\sum_{\bar{f}} count(\bar{f}, \bar{e})} \tag{2.3}$$

where $\bar{f}$ and $\bar{e}$ are the source and target phrase pairs, respectively. The length of the extracted phrase pairs can be different. In practice, we set the maximum length to 7 (Koehn et al. 2003).

However, we always observe lower frequency for longer phrases, the translation probabilities computed from lower frequency phrase pairs are unreliable and cannot truly represent the actual translation occurrence. Thus, the lexical translation probability feature is introduced, which is estimated as in Equation (2.4):

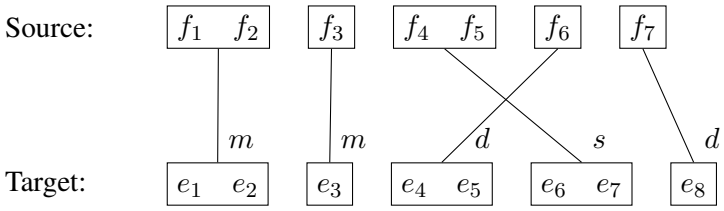



Figure 2.3: Possible orientations in reordering model, where $m$ is the monotone orientation, $s$ is the swap orientation and $d$ is the discontinuous orientation. Given a current phrase pair with respect to the previous target phrase, the monotone orientation predicts if the current source phrase is located immediately to the right of the previous source; the swap orientation predicts if the current source phrase is located immediately to the left of the previous source and the discontinuous orientation predicts if the current source phrase is located anywhere else (not monotone or swap).

$$p(\bar{f}|\bar{e}, a) = \prod_{i=0}^{l} \frac{1}{|\{j|(i,j) \in a\}|} \sum_{\forall(i,j)\in a} w(f_i|e_j) \tag{2.4}$$

where $a$ is the word alignment; $l$ is the length of source phrase $\bar{f}$; $i$ and $j$ indicate the word position in $\bar{f}$ and $\bar{e}$, respectively. $w(f_i|e_j)$ is the lexical weights as in Equation (2.5):

$$w(f_i|e_j) = \frac{count(f_i, e_j)}{\sum_{f'} count(f', e_j)} \tag{2.5}$$

where $f'$ indicates all source words aligned with $e_j$.

The inverse phrase translation probability and inverse lexical translation probability can also be computed accordingly.

### 2.1.3 Lexicalised Reordering Model

Different reordering models have been proposed for PBSMT in the literature (Koehn et al. 2005, Xiong et al. 2006, Galley and Manning 2008, Bisazza and Federico 2013). We will focus the most widely used one. Motivated by Tillmann (2004), the lexicalized reordering model (Koehn et al. 2005, Galley and Manning 2008) estimates three types of orientations – monotone, swap and discontinuous – of a phrase pair based on previous adjacent target phrase, as illustrated in Figure 2.3. Given a current phrase pair with respect to the previ-

ous target phrase, the monotone orientation predicts if the current source phrase is located immediately to the right of the previous source; the swap orientation predicts if the current source phrase is located immediately to the left of the previous source and the discontinuous orientation predicts if the current source phrase is located anywhere else (not monotone or swap). The reordering probabilities are computed as follows:

$$p_o(orientation|\bar{f},\bar{e}) = \frac{count(orientation,\bar{f},\bar{e})}{\sum_o count(o,\bar{f},\bar{e})} \tag{2.6}$$

where $\bar{f}$ and $\bar{e}$ are the source and target phrase pairs, respectively. The reordering can also be in two directions, e.g. backward and forward directions, and computed accordingly.

### 2.1.4 *n*-gram Language Model

The *n*-gram LM is an essential component in SMT. It is used to evaluate the fluency of the translations in the target language. It estimates the likelihood of a word appearing next in a sequence of target words. According to the chain rule, an *n*-gram LM can be denoted (using target sentence $E$ as an example) as in Equation (2.7):

$$p(e_0,e_1,\ldots,e_{j-1},e_j) = p(e_0)p(e_1|e_0)\ldots p(e_j|e_0,e_1,\ldots,e_{j-1}) \tag{2.7}$$

However, it is impossible to compute such statistics in real life, as we could never observe all possible sequences in a language. Furthermore, due to the fact that computation costs are high and data becomes scarce for longer sentences, we only consider a limited $n-1$ number of historical words according to the Markov assumption. For example, a bigram language model is computed as in Equation 2.8:

$$p(e_0,e_1,\ldots,e_{j-1},e_j) = p(e_0)p(e_1|e_0)\ldots p(e_j|e_{j-1}) \tag{2.8}$$

The *n*-gram probabilities are estimated by counting relative frequencies, as in Equation

(2.9),

$$p(e_j|e_{j-n},\ldots,e_{j-1}) = \frac{count(e_{j-n},\ldots,e_{j-1})}{\sum_e count(e_{j-n},\ldots,e_j)} \tag{2.9}$$

In order to avoid a zero probability prediction, smoothing methods should be applied, such as the add-one smoothing or Kneser-Ney smoothing (Chen and Goodman 1996).

As *n*-gram LMs measure the probability of how likely words are appearing next in a sentence, a 'good' *n*-gram LM should assign a higher probability to an observed text than a 'bad' LM. To evaluate the performance of an *n*-gram LM, we use perplexity, as in Equation (2.10):

$$perplexity = 2^{H(test,LM)} \tag{2.10}$$

where $H(test, LM)$ is the *cross-entropy* value modelling two distributions: $test$ is the test data indicating a (true) distribution and $LM$ is the LM distribution. The *cross-entropy* is defined as in Equation (2.11):

$$H(test, LM) = -\frac{1}{|test|}\log p(test|LM) \tag{2.11}$$

which indicates as the average negative log-likelihood per word. $|test|$ is the total number of words in $test$.

A lower perplexity value indicates a better *n*-gram LM. In the domain adaptation literature (Moore and Lewis 2010, Axelrod et al. 2011, Duh et al. 2013), the perplexity value of a sentence given by an LM trained with domain data can be interpreted as the closeness of the sentence to that domain. The lower the perplexity is, the more likely the sentence is in that domain.

In SMT trainings, the LM can be built solely from the target side of the parallel data. In practice, much larger amounts of monolingual data are used to supplement the target-language data in the parallel corpus.

### 2.1.5 Machine Translation Evaluation Metrics

To evaluate SMT translation quality, we use automatic evaluation metrics. Compared to human evaluation, automatic evaluation metrics are faster and more consistent. Many automatic evaluation metrics have been proposed in the field, e.g. Sentence Error Rate (SER), Word Error Rate (WER) (Stolcke et al. 1997), Bilingual Evaluation Understudy (BLEU) (Papineni et al. 2002), METEOR (Banerjee and Lavie 2005) and Translation Edit Rate (TER) (Snover et al. 2006). In this thesis, we choose to use BLEU to estimate the SMT translation quality as it is the most commonly used one in MT.

BLEU is a reference-based MT evaluation metric, so reference translations are essential when computing the evaluation scores. It is language-independent. The output of BLEU is a score between 0 and 100% indicating the similarity between the MT outputs and the reference translations. BLEU is computed over the entire test set. The higher the scores are, the better the translations are. BLEU scores are computed based on a modified *n*-gram precision, as in Equation (2.12):

$$BLEU = BP * \exp \sum_{n=1}^{N} \frac{1}{N} \log(\frac{|m_n \cap m_r|}{|m_n|}) \tag{2.12}$$

where $n$ represents the order of the *n*-grams compared between the translations and references. Typically, n is from 1 to 4. $m_n$ and $m_r$ indicate the *n*-grams occurring in the MT outputs and the corresponding references, respectively. $|m_n \cap m_r|$ is the number of *n*-grams occurring in both translations and references. In the case of multiple occurrences *n*-grams, we clip $|m_n \cap m_r|$ to the maximum number of times that an *n*-gram occurs in the reference. The motivation is that MT systems can overgenerate improbable translations and "a reference word should be considered exhausted after a matching candidate word is identified" (Papineni et al. 2002). A high BLEU score candidate translation should also match the reference translations in length, therefore, $BP$ is introduced. $BP$ is the brevity penalty to penalize shorter translations than the references, which is computed as in Equation (2.13):

$$BP = \exp^{max(1-\frac{length(r)}{length(n)},0)} \tag{2.13}$$

where $n$ and $r$ indicate the translation output and reference translation, respectively.

### 2.1.6 Summary

In summary, Figure 2.4 describes the process of PBSMT training. Given a parallel training corpus, words within the corpus are first aligned and phrase pairs are extracted using the word-aligned parallel training corpus. We can then learn the translation model and the lexicalised reordering model as described in Section 2.1.2 and 2.1.3, respectively. Using the target training data (or concatenating with some extra monolingual data in the target language if available), we can also learn an *n*-gram LM as described in Section 2.1.4. After this, the models are optimized under the log-linear framework as described in Section 2.1.1 to maximize the performance using a small tuning set. Translation performance is measured with an evaluation metric, such as BLEU. With the optimized weight parameters of the features in the models, we can now translate and evaluate the test set to output the evaluation scores which indicate the performance of the PBSMT system.

## 2.2 Neural Machine Translation

As presented in Section 2.1, PBSMT consists of a translation model, a reordering model and an LM, which are linearly integrated using the log-linear framework. NMT (Sutskever et al. 2014, Cho et al. 2014, Bahdanau et al. 2015, Luong et al. 2015b), being a new approach, employs an individual large neural network to model the entire translation process. Tu et al. (2016) state the advantages of NMT over SMT are as follows:

- NMT uses distributed word representations during training,
- Explicit feature design is not required to capture translation regularities in NMT,

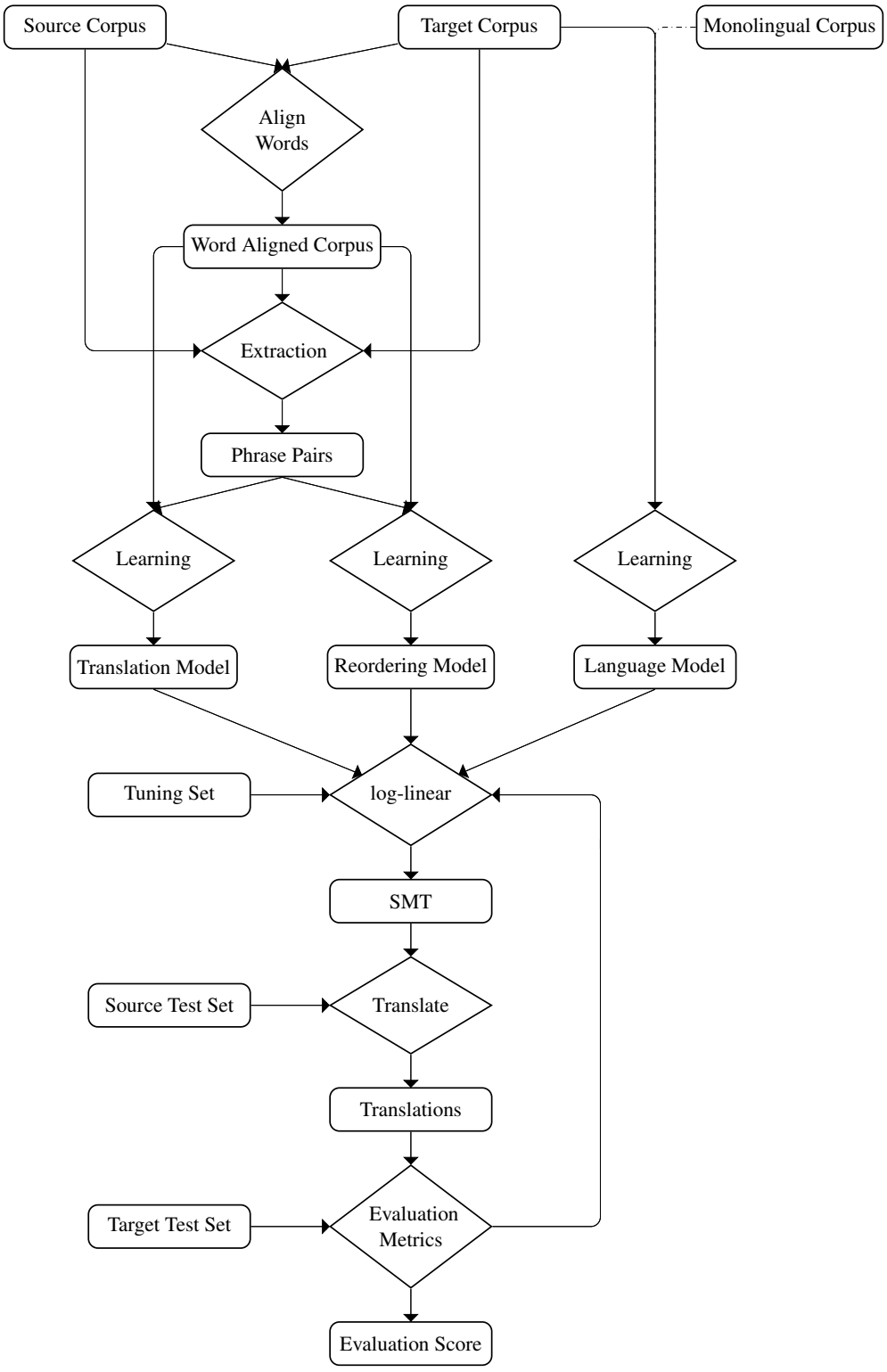



Figure 2.4: This diagram shows the training steps of a PBSMT system.

| | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
|---|---|---|---|---|---|---|---|---|
| $w_1$ | -0.326 | 0.326 | 0.036 | 0.708 | -0.535 | 1.709 | -0.981 | 0.496 |
| $w_2$ | 0.941 | -0.284 | -0.337 | 0.349 | 0.761 | -0.039 | -0.662 | 1.438 |
| $w_3$ | -0.355 | 0.292 | 0.989 | -0.834 | 0.769 | 0.740 | 0.152 | -0.420 |
| $w_4$ | 0.934 | -0.552 | 0.418 | 0.555 | -1.894 | -0.273 | -0.711 | 0.092 |
| $w_5$ | 1.696 | 0.994 | 0.960 | 0.039 | -0.631 | 0.783 | 0.538 | -0.822 |
| $w_6$ | 0.552 | -0.807 | 0.066 | -0.862 | 0.473 | -0.240 | -0.974 | -0.975 |
| $w_7$ | -0.211 | -1.714 | 0.557 | -0.740 | -0.907 | -0.830 | 0.517 | 0.749 |
| $w_8$ | -0.734 | -0.040 | 1.509 | -0.795 | 0.006 | 0.512 | -0.970 | -0.028 |
| $w_9$ | -0.311 | 0.447 | -0.825 | -1.027 | 1.335 | 0.696 | 0.621 | 0.546 |
| $w_{10}$ | -0.809 | -1.078 | 0.847 | -0.446 | -0.789 | -0.577 | 0.472 | -0.734 |

Figure 2.5: A word vector model is a $v \times n$ matrix, where $v$ is the size of vocabulary and $n$ is size of the word vector. In this example, the size of the vocabulary is 10 ($w_1 \dots w_{10}$), each word is represented in 8 dimensions (size of the word vector).

- NMT is based on RNNs, which are better at capturing long-distance reordering than SMT.

In this section, we review word vector models, RNNs, RNN LM, the encoder-decoder NMT framework (Cho et al. 2014, Sutskever et al. 2014), the state-of-the-art attention-based NMT (Bahdanau et al. 2015) and bidirectional RNNs.

### 2.2.1 Word Vector Models

Word vectors (Distributed word representations) are important building blocks in neural networks. In NLP, word vectors have the advantage that similar words are represented closely in the vector space.

A word vector model is a $v \times n$ matrix which can map a word in a vocabulary to a real-value word vector, where $v$ is the size of vocabulary and $n$ is size of the word vector. Figure 2.5 is an illustration of a word vector model. While much work has been introduced (Hinton et al. 1986, Rumelhart et al. 1986, Mikolov et al. 2013a, Pennington et al. 2014) for word vector models, in this thesis we focus on the approach proposed by Mikolov et al. (2013a). In Mikolov et al. (2013a), labelled data is not required for the word vector model training. It uses context words as features to predcit the current word. A recent study (Mikolov et al. 2013c) showed that distributed word representations can capture linguistic regularities and

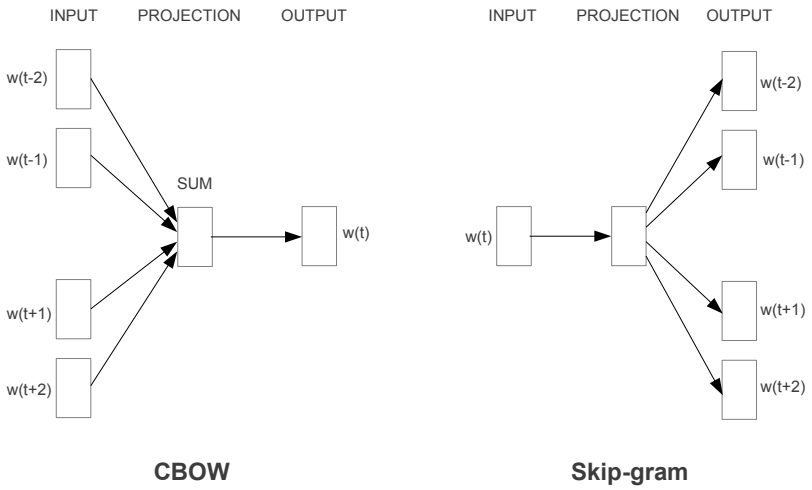



Figure 2.6: Word vector model architectures proposed in Mikolov et al. (2013a).

similarities in the training corpus. For example, given the word vectors of words 'king', 'man' and 'woman', we can apply vector operations on them, such as:

*king - man + woman*

The result word vector is close to the word representation of 'queen'.[1]

Mikolov et al. (2013a) proposed two different architectures for distributed word representation training, where the Continuous Bag of Words (CBOW) architecture predicts the current word based on the context words, and the Skip-gram predicts surrounding words given the current word. Intuitively, the CBOW architecture reverses the training of the Skip-gram, as seen in Figure 2.6.[2] For example, assume the following sentence is the training data for the distributed word representation training:

*The cat is **sitting** on the mat .*

The CBOW architecture models the conditional probability $p(sitting|cat, is, on, the)$ (if the current word is $sitting$ and we use forward and backward context window of 2); the

[1]This example is taken from Mikolov et al. (2013c)
[2]This figure is taken from Mikolov et al. (2013a).

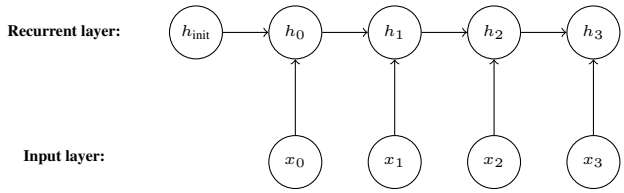



Figure 2.7: A simple unfold RNN which maintains a context vector covering previous sequential information. For example, $h_1$ is computed using $w_1$ and $h_0$. Later, $h_1$ is involved in the computation of $h_2$. $h_{\text{init}}$ is the initial state (a vector of zeros or random numbers) of the network. The context vector is also refereed as the *hidden state* of an RNN and $x_t$ is the input at *time step* $t$.

Skip-gram architecture models the conditional probability $p(cat, is, on, the|sitting)$ (if the current word is $sitting$ and in the 0-skip-4-gram setting). In the Skip-gram architecture, the output is not limited to the immediate context of the input word, we can train the model by skipping a number of words in its context, hence the name of this architecture is called Skip-gram.

Apart from the aforementioned word vector model training methods, word vector models can also be trained together with other tasks (Collobert et al. 2011). As an example, the word vectors used later in Section 2.2.3 involve specific word-vector training for an LM. At the beginning of training, we randomly initialize the word vector for each word in the training vocabulary. Then the word vectors are updated using the errors learned by predicting the current word given its previous words in a sentence.

The dimensions of word vector representation can be different. In practice, a size between 300 to 600 is an efficient setting for most tasks.

### 2.2.2 Recurrent Neural Network

RNNs build neural networks on sequential inputs and assume that the hidden states within the network are dependent, which is true in many sequence prediction tasks. The hidden states can be thought of a 'memory' to maintain the previous history.

A simple RNN, as seen in Figure 2.7, consists of two layers: an input layer and a recurrent layer. The recurrent layer maintains a context vector covering previous sequential information. Each context vector $h_t$ in an RNN is computed by the current input $x_t$ and

previous context vector $h_{t-1}$. The context vector is also refereed as the *hidden state* of an RNN and $x_t$ is the input at *time step* $t$. A non-linear function is then applied to the current context vector $h_t$. A simple RNN can be formalized as in Equation (2.14):

$$h_t = sigmoid(Wx_t + Uh_{t-1}) \tag{2.14}$$

where $W$ and $U$ are the corresponding weight parameters. $sigmoid$ is a non-linear function defined as in Equation (2.15):

$$sigmoid(x) = \frac{1}{1 + e^{-x}} \tag{2.15}$$

where e is Napier's constant.

However, it is known that simple RNNs suffer from the vanishing gradient problem (Bengio et al. 1994), where for long sequence inputs, the early contexts are often forgotten and overwritten by the later contexts. The Long Short-Term Memory (LSTM) (Hochreiter and Schmidhuber 1997) or the more recently introduced Gated Recurrent Unit (GRU) (Chung et al. 2014) use gates to control the information flow from previous words, which are better at capturing long-term dependencies than simple RNNs, and are thus often chosen in practice.

The GRU, as illustrated in Figure 2.8, consists of an update gate and a reset gate, as in Equation (2.16):

$$\begin{aligned} u_t &= sigmoid(W_u x_t + U_u h_{t-1}) \\ r_t &= sigmoid(W_r x_t + U_r h_{t-1}) \\ \tilde{h}_t &= tanh(W x_t + (r_t \odot h_{t-1})) \\ h_t &= (1 - u_t) \odot \tilde{h}_t + u_t \odot h_{t-1} \end{aligned} \tag{2.16}$$

where $u_t$ is the update gate and $r_t$ is the reset gate. $\tilde{h}_t$ is the candidate activation (Chung

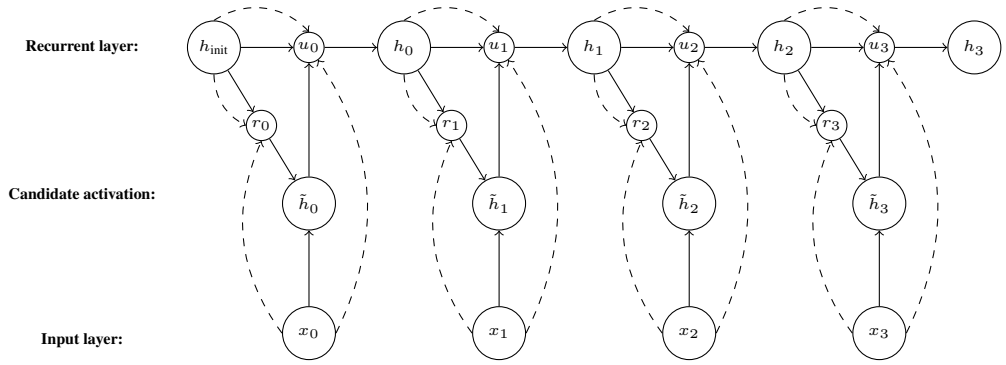



Figure 2.8: Illustration of an GRU network, which consists of an update gate $u$ and a reset gate $r$. Dashed lines indicate the computations for $u$ and $r$, as seen in Equation (2.16) (the bias values are omitted). $h_{\text{init}}$ is the initial state (a vector of zeros) of the network.

et al. 2014) and $\odot$ is the element-wise multiplication operation. $h_t$ is the linear-interpolated output between the previous hidden state $h_{t-1}$ and the candidate activation. Intuitively, the update gate determines the interpolation weights between the previous hidden state $h_{t-1}$ and the candidate activation, and the reset gate determines the information flow from previous hidden states. If the reset gate is set to $1$ and the update gate is set to $0$, the GRU is equivalent to a simple RNN. $W_u$, $U_u$, $W_r$, $U_r$, $W$ and $U$ are the weight parameters, and $b_u$, $b_r$ and $b$ are the bias values of the corresponding gates. $tanh$ is the hyperbolic tangent function.

GRU is a simplified version of LSTM with fewer gates and has been growing increasingly popular. The performance of GRU and LSTM are comparable according to Chung et al. (2014). Thus, we use GRUs in this thesis.

### 2.2.3 Recurrent Neural Network Language Model

The main drawback of the *n*-gram LM is that it uses the Markov assumption due to data sparsity for a large number of historical words, whereas the neural LM has no such assumption. It can capture much longer history by using RNN than *n*-gram LM. In addition, the neural LM is better at generalization for words as distributed word representations are used in training.

The neural LM models the probability of the next word given the previous words. The

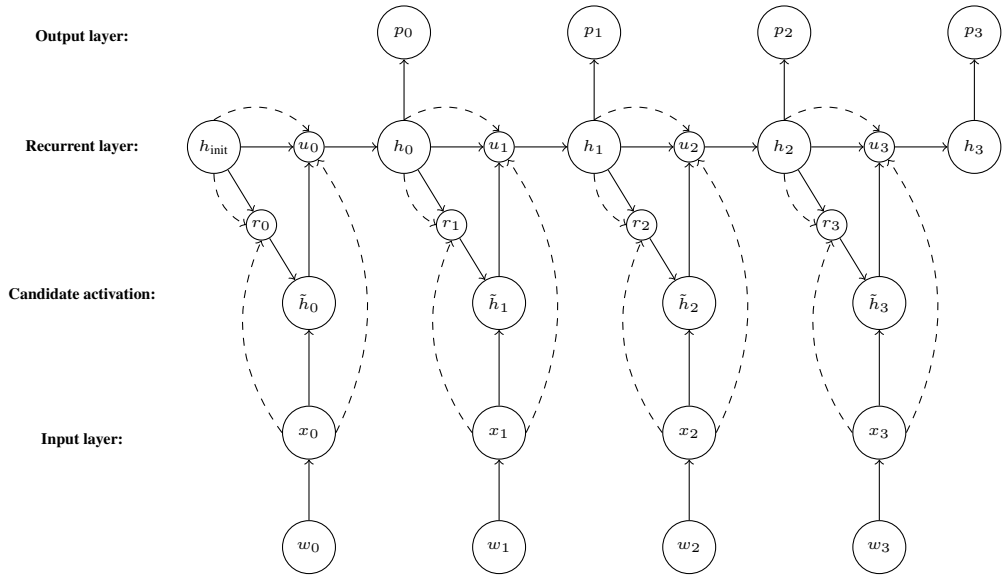



Figure 2.9: This diagram shows an RNNLM. It has an input layer, a recurrent layer and an output layer. The recurrent layer uses a GRU network. $h_{\text{init}}$ is the initial state (a vector of zeros) of the network. For example, if the current input word is $w_1$, we first learn the word vector $x_1$ in the input layer, then compute the context vector $h_1$ in recurrent layer using the GRU network. In the output layer, we compute the probability of the current output $p_1$ using a *softmax* function.

simplest RNN LM has an input layer, a recurrent layer and an output layer, as seen in Figure 2.9. The input layer learns word vectors. The recurrent layer can either be a simple RNN or GRU. The output layer operates a *softmax* function to compute probability distributions over all words in the vocabulary. We can formally define a neural LM (assume that the recurrent layer is a single layer GRU network), as in Equation (2.17):

$$
\begin{aligned}
x_t &= M(w_t) \\
u_t &= sigmoid(W_u x_t + U_u h_{t-1} + b_u) \\
r_t &= sigmoid(W_r x_t + U_r h_{t-1} + b_r) \\
\tilde{h}_t &= tanh(W x_t + U(r_t \odot h_{t-1}) + b) \\
h_t &= (1 - u_t) \odot h_{t-1} + u_t \odot \tilde{h}_t \\
p(t) &= softmax(S(h_t))
\end{aligned}
\tag{2.17}
$$

where $w_t$ is the input word, $M$ is the word vector matrix, $x_t$ is the word vector of $w_t$, $h_t$

is the current context vector computed using a GRU network and $S$ is a transform function which can convert $h_t$ into a vector with dimensions equal to the size of the vocabulary. As we defined in the GRU Equation (2.17), $\tilde{h}_t$ is the candidate activation GRU and $\odot$ is the element-wise multiplication operation. $W_u$, $U_u$, $W_r$, $U_r$, $W$ and $U$ are the weight parameters, and $b_u$, $b_r$ and $b$ are the bias values of the corresponding gates.

For initialization, the weights parameters in the neural LM can be initialized with random values. Words are sequentially fed to the model. At the output, each word is assigned with a probability to indicate the likelihood of being the next word. At each training step, we use cross-entropy to compute the error vectors, model weights are updated with the standard back-propagation algorithm (Rumelhart et al. 1988). For example, we can define the cross-entropy error function as in Equation (2.18):

$$C(y, \hat{y}) = -\sum_i y_i \log(\hat{y}_i) \tag{2.18}$$

where $y$ is the predicted probability distribution and $\hat{y}$ is the true distribution. In practice, we can back-propagate errors not only for time $t$, but also even further. For example, a more complicated back-propagation through time algorithm (Rumelhart et al. 1995) is commonly used to back-propagate errors to the previous constant time-steps. The RNN is unfolded into a flat architecture through time for a certain amount of time-steps and the errors are summed up for all unfolded time-steps. Then gradients of the error are computed and model parameters are updated.

The training of a neural network LM normally runs many epochs, where each epoch loops through all the training data. The model is considered to have converged when no significant improvements are observed based on the log-likelihood on the evaluation data. The perplexity evaluation for *n*-gram LMs is also used for neural LM.

As a pre-processing step in neural LM training, since the *softmax* function in Equation (2.17) needs to distribute probability distributions over all words (typically hundreds of thousands in size), it is a time-consuming operation, we also need to map the lower

frequency words into the Unknown (UNK) token.

After a neural network LM is trained, the input layer can also be thought as a task-specific word-vector model (Collobert et al. 2011). Therefore, word vector models can be pre-trained as we described in Section 2.2.1 or in a task-specific training as the input layer of a neural LM.

Neural network LMs are not restricted to only using RNNs. Early research (Bengio et al. 2003) uses a feed-forward neural network, whereas Wang et al. (2015) use a convolutional neural network. Different data representations have also been explored in recent research, e.g. characters are used for training instead of words (Kim et al. 2016).

### 2.2.4 Encoder-Decoder Framework

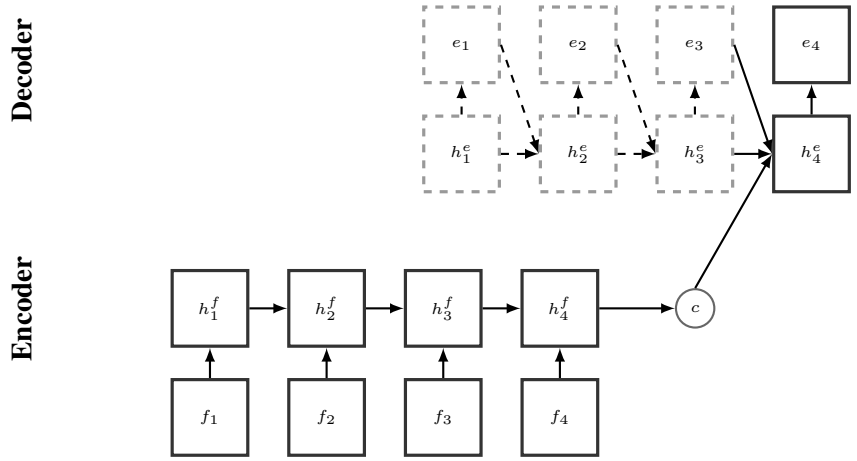
In a nutshell, the fundamental job of the encoder-decoder framework (Cho et al. 2014, Sutskever et al. 2014) in NMT is to probabilistically decode a target sequence given the encoded source sequence, where the two sequences can be of different lengths.

Figure 2.10 illustrates translation process for a source sentence with 4 input words $\{f_1, \ldots, f_4\}$, where $h^f$ indicates the source context vectors at each time step of the source input; $c$ is a fixed-size vector representing the source sentence ($c = h_4^f$, which is the last time step of the encoder RNN); $h^e$ indicates the target context vectors at each time step of the target output and $e$ represents the translated words. Thus, the current output $h_4^e$ is conditioned on $c$, $h_3^e$ and $e_3$. Then, $h_4^e$ can be used to predict $e_4$.

The encoder-decoder framework formulates the translation problem as Equation (2.19):

$$p(E|F) = \prod_{n=0}^{j} p(e_n|e_{0:n-1}, F) \tag{2.19}$$

This can be interpreted as the translation probability of a target sentence $E$ given a source sentence $F$ is computed by multiplying the translation probabilities of each target word; and the translation probability of of each target word, e.g. $e_n$, is computed as the conditional probability of given source sentence $F$ and previous target translations $e_{0:n-1}$.




***Notations:***

$f_1$,...,$f_4$: Source words

$h_1^f$,...,$h_4^f$: Source context vectors

$c$: Source context vector (last time step)

$e_1$,...,$e_4$: Target words

$h_1^e$,..., $h_4^e$: Target context vectors

Figure 2.10: The graphical illustration of the encoder-decoder framework. The source sentence has 4 input words $\{f_1, \ldots, f_4\}$ and the current predicting word is $e_4$ in the target. $h^f$ indicates the source context vectors at each time step of the source input; $c$ is a fixed-size vector representing the source sentence ($c = h_4^f$, which is the last time step of the encoder RNN); $h^e$ indicates the target context vectors at each time step of the target output and $e$ represents the translated words. Thus, the current output $h_4^e$ is conditioned on $c$, $h_3^e$ and $e_3$. Then, $h_4^e$ can be used to predict $e_4$.

The conditional probability is given by the decoder, which uses the *softmax* function outputting the probability distribution of all words $e$ in the target language, as in Equation (2.20):

$$p(e|e_{1:j-1}, F) = softmax(S(t_{j-1}, h_j, c)) \tag{2.20}$$

where $c$ is the source context vector computed by the encoder, $t_{j-1}$ is the word vector of target word $j-1$, $h_j$ is the target context vector for time $j$ and *softmax* is a function defined as in Equation (2.21):

$$softmax(x_t) = \frac{\mathrm{e}^{x_t}}{\sum_v \mathrm{e}^{x_v}} \tag{2.21}$$

where $v$ is the target vocabulary size, e is Napier's constant and $x_t$ is the input of the *softmax* function. It is known that the *softmax* function is inefficient because the probability

distribution is on all words in the target vocabulary, and such an operation is required at each training step. We often reduce the target vocabulary size by replacing lower frequency words to a special token: the UNK token.

In Equation (2.20), $S$ is a function that can transform the inputs into a vector of size $v$, and $h_j$ is defined as in Equation (2.22):

$$h_j = g(t_{j-1}, h_{j-1}) \tag{2.22}$$

where $g$ is a GRU network. Thus, we use the source input sentence and previously translated words to make predictions for the next word.

The NMT model can be trained with the mini-batch Stochastic Gradient Descent (Robbins and Monro 1951) algorithm together with Adadelta (Zeiler 2012), and is validated based on cross-entropy error. During training, we save the trained model based on the number of model parameters updates. For example, the trained can be saved at every 1,000 updates. We can then compute the BLEU scores of each saved model using development data. The best-performing model is the final trained NMT model.

During training, a special token – end-of-sentence (EOS) – is used to append at the end of training sentences. In the decoding phase, then translation process stops if the current output word is the EOS token.

### 2.2.5 Attention-based Neural Machine Translation

The encoder-decoder framework uses a fixed-size vector to represent the whole source input. Although GRU networks are known to be better at capturing long-range dependencies, experimental results (Bahdanau et al. 2015) show that translation quality decreases for long input sentences. Accordingly, Bahdanau et al. (2015) use an *attention model* to learn dynamic *soft-alignment* during the network training. With the attentional model, source information can be spread across the source context vector, and the decoder can selectively pay attention to different parts of the source context during decoding.

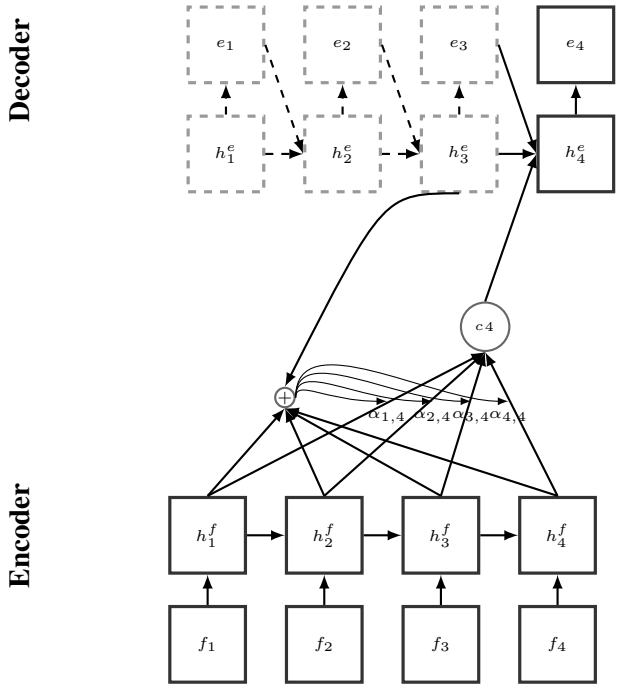



Figure 2.11: The graphical illustration of the attention-based NMT, where $h^f$ indicates the source context vectors and $h^e$ indicates the target context vectors. Suppose there are 4 source input words $\{f_1, \ldots, f_4\}$ and the current predicting word is $e_4$ in the target. The encoder reads the source input words and produces the source context vectors for each source input word. Next, the attention model $\oplus$ computes weights ($\alpha_{1,4}$, $\alpha_{2,4}$, $\alpha_{3,4}$ and $\alpha_{4,4}$) for each $h^f$ and outputs a weighted sum of $h^f$ – a distinct source context vector $c_4$. Then, the distinct source context vector $c_4$, previous translation $e_3$ and previous target context vector $h_3^e$ are used to obtain the current target context vector $h_4^e$, which is used to output translation probability for all target words.

Figure 2.11 is a graphical illustration of the attention-based NMT model, where $h^f$ indicates the source context vectors and $h^e$ indicates the target context vectors.[3] Suppose there are 4 source input words $\{f_1, \ldots, f_4\}$ and the currently predicted word is $e_4$ in the target. The encoder reads the source input words and produces the source context vectors for each source input word. Next, the attention model computes weights ($\alpha_{1,4}$, $\alpha_{2,4}$, $\alpha_{3,4}$ and $\alpha_{4,4}$) for each $h^f$ and outputs a weighted sum of $h^f$ – a distinct source context vector. Then the distinct source context vector $c_4$, previous translation $e_3$ and previous target context vector $h_3^e$ are combined as the current target context vector $h_4^e$, which is used to output translation probability for all target words.

We now formally define the attention-based NMT (Bahdanau et al. 2015). The align-

[3]Gates in the GRU networks are eliminated.

ment model that scores the alignment at position $i$ and $j$ in $F$ and $E$ respectively, is computed as in Equation (2.23):

$$e_{ij} = v^T a(h_{j-1}, h_i) \tag{2.23}$$

where $h_{j-1}$ is the target hidden state of $E$ and $h_i$ is the source context vector at time $i$ in $F$ computed by the encoder RNN, and *a* is a non-linear function, such as the *tanh* function. $v \in \mathbb{R}^n$ is a weight matrix.

Thus, a distinct source context vector $c_j$ can be computed for each word in $E$, and the source context vector $c$ is rewritten as in Equation (2.24):

$$c_j = \sum_{i=1}^{m} \alpha_{ij} h_i \tag{2.24}$$

where $\alpha_{ij}$ is a normalized weight for each context vector of source input in $\{0 \dots i\}$, computed as in Equation (2.25)

$$\alpha_{ij} = \frac{\exp(e_{ij})}{\sum_{i=1}^{m} \exp(e_{ij})} \tag{2.25}$$

Thus, we can update Equation (2.22) with $c_j$, such as in Equation (2.26):

$$h_j = g(t_{j-1}, h_{j-1}, c_j) \tag{2.26}$$

And Equation (2.20) is also updated according such as in Equation (2.27)

$$p(e|e_{1:j-1}, F) = softmax(S(t_{j-1}, h_j, c_j)) \tag{2.27}$$

### 2.2.6 Bidirectional Recurrent Neural Network

During the encoding phase, words can also be fed into the decoder in both directions, using what is known as a bidirectional RNN (Schuster and Paliwal 1997). The intuition behind

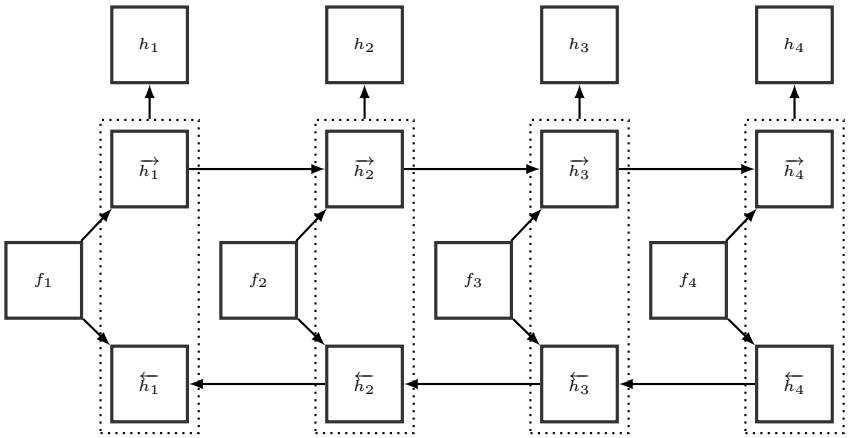



Figure 2.12: The graphical illustration of the bidirectional RNN.

using such a model is to include both *positive* and *negative* time steps of the source input during encoding.

Figure 2.8 is a graphical illustration of the bidirectional RNN, which consists of forward ($\overrightarrow{h_1}$,...,$\overrightarrow{h_4}$ are the context vectors) and backward ($\overleftarrow{h_1}$,...,$\overleftarrow{h_4}$ are the context vectors) RNNs, where $\{f_1, \ldots, f_4\}$ are the input sequences. The outputs are the concatenations of the context vectors at corresponding time steps, such as $h_1 = [\overrightarrow{h_1}, \overleftarrow{h_1}]$. A bidirectional RNN used in NMT "contains the summaries of both the preceding words and the following words" (Bahdanau et al. 2015) for source inputs. Because an RNN can represent recent inputs better, the outputs of a bidirectional RNN are focused on the context words on both sides (*positive* and *negative* time steps) of the current word. Sutskever et al. (2014) also claim that it is "extremely valuable" and can "greatly boost the performance" by using the reversed source sentences information in NMT models.

### 2.2.7 Summary

Over the last few years, neural network training has attracted much interest and demonstrated promising results. For example, NMT systems achieved many state-of-the-art results on a number of language pairs (Cettolo et al. 2015, 2016, Bojar et al. 2016). In this subsection, we first reviewed the important building blocks in neural network training – the word vector models. We then introduced the RNN, particularly the GRU network, which

has been used in many sequence prediction tasks. Next, we provided information on two applications which are related with this thesis – RNN LMs and NMT. We also discussed attention-based NMT using bidirectional RNN.

## 2.3 Domain Adaptation for MT

The focus of this thesis is applying domain adaptation in MT systems. In this subsection, we review some of selected works for in-depth discussion. We first discuss methods that make better use of the training data which can be applied in both SMT and NMT systems, i.e. the data selection approaches. We then review methods of domain adaptation in SMT, such as SMT model combination and SMT topic-based domain adaptation. Next, we study domain-adaptation approaches for neural LM and NMT. At the end of this subsection, we provide our summary for this subsection.

### 2.3.1 Data Selection

Domain adaptation via data selection focuses on making efficient use of General-Domain (GD) training data in order to improve the translation quality of MT systems. The data selection approaches are applicable to the *domain-awareness* scenario, where there is a clear boundary between the In-Domain (ID) and GD training data. The preliminary setting for using data selection in domain adaptation is that there is a limitation on the size of ID training data, while large amounts of GD training data are normally thought easier to obtain. The aim of data selection is to select some sentences which are similar to the ID training data from a large amount of GD training data. It has the advantage that it can potentially remove some noisy (e.g. incorrectly aligned) data (Haddow and Koehn 2012). The data selection approaches can be used in $n$-gram LM, SMT, neural LM or NMT tasks.

Lü et al. (2007) use information retrieval techniques in a transductive learning framework to increase the count of important ID training instances, which results in phrase-pair weights being favourable to the development set. Biçici and Yuret (2011) employ a fea-

ture decay algorithm which can be used in both active learning and transductive learning settings. The decay algorithm is used to increase the variety of the training data by devaluing features that have already been seen from a training set. In recent studies, a cross-entropy difference method has seen increasing interest for the problem of SMT data selection (Moore and Lewis 2010, Axelrod et al. 2011). The training data set is ranked using cross-entropy difference from LMs trained on ID or GD sentences. A threshold is then set to select the pseudo-ID sentences. The intuition is to find sentences as close to the target domain and as far from the average of the GD as possible. Later, Mansour et al. (2011, p. 2) argue that "An LM does not capture the connections between the source and target words, and scores the sentences independently", and linearly interpolate IBM model 1 (Brown et al. 1990, 1993) into the cross-entropy difference framework. The translation performance is improved on both Arabic-to-English and English-to-French translation tasks compared with the standalone cross-entropy difference approach. Toral (2013) also makes use of linguistic information, such as lemmas, named entity categories and part-of-speech tags, when computing the cross-entropy difference. Another extension of Axelrod et al. (2011) is Banerjee et al. (2012) who propose an approach to perform batch selection with the objective of maximising the SMT performance. Later, Duh et al. (2013) report "the neural language model (Bengio et al. 2003) is a viable alternative, since its continuous vector representation of words is well-suited for modelling sentences with frequent unknown words, providing smooth probability estimates unseen but similar contexts". They then adapt the cross-entropy difference approach using neural LM. Chen and Huang (2016) and Peris et al. (2016) use neural network training to select ID training data where selection problem is treated as a classification task.

However, data selection approaches have several drawbacks. Firstly, data selection is a rather heavy-handed approach. Sentences in the GD data set are either selected and used in SMT training or ignored, despite the fact that those sentences that are not selected might still have a positive contribution to the performance of an SMT performance (Haddow and Koehn 2012). Furthermore, data selection approaches are classified as a ranking task. The

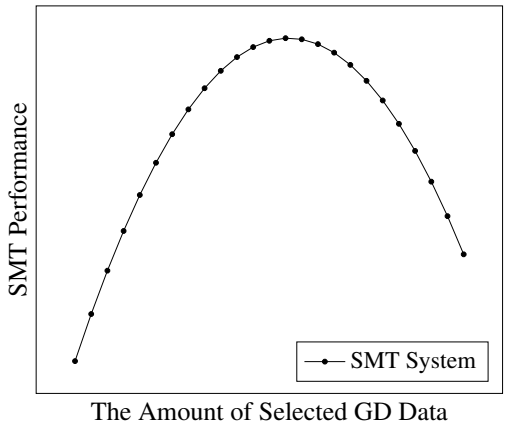



Figure 2.13: Illustration of the schematic relationship between the amount of selected GD data with the corresponding SMT performance.

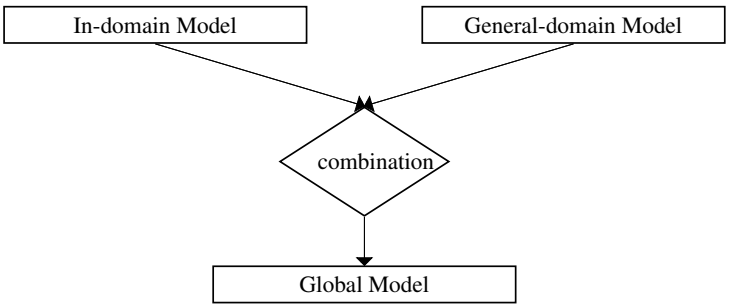



Figure 2.14: SMT Model combination paradigm

GD training data is ranked according to some distance measure, e.g. cosine similarity or cross-entropy difference, compared to the ID training data. A threshold of the selected portion of the GD data needs to be determined empirically. Figure 2.13 illustrates the schematic relationship between the amount of selected GD data with the corresponding SMT performance. In general, we observe SMT performance improvements by adding more selected data (Kirchhoff and Bilmes 2014), the performance reaches a peak point and then begins to decrease. To learn the selected portion of data to reach such a peak point requires us to train and test on many SMT systems, which is a very time-consuming process.

### 2.3.2 Domain Adaptation for SMT

SMT model combination approaches are used when multiple sub-models are available, e.g. each sub-model is trained on its own domain. After the combination operation, a single global model is produced. Figure 2.14 illustrates the SMT model combination paradigm in the case where ID and GD are both available. The combined global model is then used for decoding. Because the model combination technique needs to know the domains of each sub-model, it falls into our *domain-awareness* scenario.

Linear combination (Foster and Kuhn 2007) linearly combines sub-models into one global model, as defined in Equation (2.28):

$$global\_model = \sum_{c} \lambda_c [sub\_model_c] \tag{2.28}$$

where $c$ indicates a domain. Thus, each model of a domain has its corresponding weight $\lambda$ and $\sum_c \lambda_c = 1$. The weights are learned according to some distance metrics, e.g. *tf/idf*, perplexity, Latent Semantic Analysis or the EM algorithm, in Foster and Kuhn (2007). In the experiments of Foster and Kuhn (2007), using the EM algorithm to learn $\lambda$ slightly outperforms other distance metrics in terms of BLEU score. Bisazza et al. (2011, p. 3) also note that "there is not a consensus on the best technique to optimize the mixture weights" in the linear combination.

One drawback of the linear combination approach is that the phrase pairs in the ID translation model can be penalized when learning $\lambda$ in Equation (2.28). For example, the EM algorithm will assign higher probabilities to the phrase pairs in the GD translation model if the GD translation model already has good coverage of the desired domain. Consequently, correct phrase pairs in the ID translation model are ignored during decoding and the translation performance decreases (Chen et al. 2014). To overcome such an issue, Foster et al. (2013) propose to balance the $\lambda$ learning process by randomly sampling an equal number of instances between ID and GD translation models.

As we mentioned in Section 2.1.1, the log-linear framework has the ability to include

more feature functions. Accordingly, the sub-models of each domain can be interpolated into the log-linear framework. Such a combination technique can be denoted as in Equation (2.29):

$$global\_model = \exp \sum_{c} \lambda_c [sub\_model_c] \tag{2.29}$$

Therefore, the weights of log-linear combination are optimized directly using the standard tuning procedure. However, log-linear combination has a serious drawback. During translation, either the translation hypotheses must be found in all sub-models or a smooth value must be given, in order to avoid a zero probability for the missing hypotheses. Otherwise, useful information contained in the small sub-models might be discarded (Foster et al. 2013, Chen et al. 2014). As an alternative, Koehn and Schroeder (2007) investigate an idea of using separate models as alternative decoding paths, where a phrase pair is scored by each model individually and each model has its own set of optimized weights.

Another well-established SMT model combination approach is called model fill-up (Nakov 2008). In the fill-up approach, models which are trained on the ID training data are unchanged in the global model. However, models trained using the GD data are '*filled*' into the global model. Only phrase pairs in the GD model that are not appeared in the ID model are added into the global model, as in Equation (2.30):

$$global\_model = sub\_model_c \cup \{sub\_model_{\hat{c}} - sub\_model_c\} \tag{2.30}$$

where $sub\_model_c$ and $sub\_model_{\hat{c}}$ indicate the ID and GD models, respectively. The *relative complement* of $sub\_model_{\hat{c}}$ in $sub\_model_c$ is represented as $sub\_model_{\hat{c}} - sub\_model_c\}$. Furthermore, a new feature value (1 or 0.5) is allocated to each phrase pair in the combined model to indicate its provenance. Bisazza et al. (2011) modify the feature value of Nakov (2008) by using an additional feature, such as 1 (= $exp(0)$) and 2.718 (= $exp(1)$), to define the provenance of each phrase pairs in the translation models. The attractive properties of the fill-up approach can be seen from experimental results which demonstrate comparable translation performance with linear and log-linear combination approaches. It can also in-

crease the efficiency of the tuning procedure (Bisazza et al. 2011). For example, the tuning of the fill-up approach converges much faster than the log-linear combination approach as there are more parameters to tune in the latter case.

The model combination technique is one of the most popular domain-adaptation approaches in MT. Using model combination, we introduce additional knowledge from other domains to overcome the out-of-vocabulary words problem when using only the insufficient ID data.

The model combination technique is applicable to the *domain-awareness* scenario, where there is a clear boundary between the ID and GD training data. However, in a *domain-unawareness* scenario, the domain information is not given explicitly in the training data. One approach is to use the topic information as domains to perform adaptation, namely topic-based domain adaptation.

In topic-based domain adaptation, the domains in the training data are introduced implicitly by some topic learning algorithms. It is not a hard-handed – 1 or 0 relationship between domains – approach to represent domains, but to use probabilities instead. The topic-based domain adaptation falls into our *domain-unawareness* scenario.

Su et al. (2012) employ two topic models trained on the monolingual ID data and the source side of the parallel GD data. The translation models are then conditioned on the probabilities mapping between the ID topic distribution to the GD topic space. Eidelman et al. (2012) achieve translation performance improvement by including a lexical weight topic feature into the translation models. The lexical feature is conditioned on the topic distributions learned on the source side of the training sentences using Latent Dirichlet Allocation (LDA) (Blei et al. 2003). Hasler et al. (2012) learn topic features for word and phrase pairs, the features are then added as sparse features into SMT. However, Hidden Topic Markov Model (HTMM) (Gruber et al. 2007) is employed instead of LDA. Xiao et al. (2012) and Zhang et al. (2014b) focus on document translations and propose a topic-similarity model and a topic-sensitivity model for SMT. The topic-similarity model is used to encourage or penalize topic-sensitive rules, and the topic-sensitivity model is applied to

balance the topic-insensitive rules. Su et al. (2015) observe that words are consistent with topics in the target sentences. In their work, a context-aware topic model is integrated into the translation system for better lexical selection.

We will shortly discuss some other domain-adaptation techniques in SMT. A self-enhancement approach is also used to overcome the challenge of insufficient ID training data (Ueffing 2006, Schwenk 2008, Bertoldi and Federico 2009), where an ID SMT system is employed to translate the monolingual data into the target language and the resultant translations are used as additional training data. A more fine-grained investigation on the self-enhancement approach is proposed by Chen et al. (2008), where different approaches are proposed for the translation model, reordering model and LM. Wang et al. (2012) use a classifier at decoding time to classify source sentences into the most favourable domains. Given the classified domain, the decoder can then decode with domain-specific features. Haddow and Koehn (2012) discuss the usefulness of domain adaptation in the phrase pair extraction and translation model training steps. One of the conclusions is that while GD can improve the translation coverage for rare words, it may be harmful for common ID words. This suggests that the translations which contain a lot of ID evidence should be kept. Chen et al. (2013) assign vector-similarity measures to the entries in translation models. The similarities are computed by comparing the vectorized representation of translation model entries extracted from the development set and the training set.

### 2.3.3 Domain Adaptation for NMT

In neural LM, one approach to performing domain adaptation is to use an additional adaptation layer to combine the GD LM into the ID LM (Park et al. 2010, Ter-Sarkisov et al. 2015). However, a LM trained on all GD data is required, which can be time-consuming if the GD data is very large. Curriculum learning (Bengio et al. 2009), which rearranges the training data in a particular order to improve generalization, has also been applied on neural LM domain adaptation by Shi et al. (2013). In Mikolov and Zweig (2012), word predictions are conditioned on the word topic representations. Thus, building multiple topic-specific

language models is avoided.

To date, several domain adaptation techniques for NMT models have been proposed in the literature. Luong and Manning (2015) find that using ID training data to fine-tune the existing GD NMT models can be a very useful domain-adaptation technique. An absolute gain of 3.8 BLEU points improvement can be observed on the International Workshop on Spoken Language Translation task in the English-to-German language pair. Servan et al. (2016) apply a similar idea in a specific Computer Assisted Translation framework. Another difference between Luong et al. (2015b) and Servan et al. (2016) is that Luong et al. (2015b) conduct training over many more iterations than the work in Servan et al. (2016). Freitag and Al-Onaizan (2016) also report the efficiency of fine-tuning on ID data. However, one drawback of the fine-tuning approach is that there is an assumption that there are only two domains.

Inspired by Sennrich et al. (2016), Kobus et al. (2016) annotate domain tags in NMT training. The sentence-domain tag is appended to each source sentence; the word-domain tag is concatenated with each source token. For example, if the training or testing source sentence belongs to the *Medical* domain, a sentence-domain tag **@MED@** is appended to the end of the source sentence during training and testing. The word-domain tag is used in a similar way but appended to each source word. A prerequisite of such an approach is to know the domain of the translating sentences in advance.

### 2.3.4 Summary

For a better illustration, Figure 2.15 presents the overall domain adaptation for the MT work described in Section 2.3. It is worth mentioning that our **RQ1** is related to the SMT model combination section, **RQ2** is related to neural LM and NMT domain adaptation section and **RQ3** is related to the neural NMT domain adaptation section. Finally, our proposed method in Chapter 3 for **RQ1** is inspired by the cross-entropy difference method described in the data selection section.

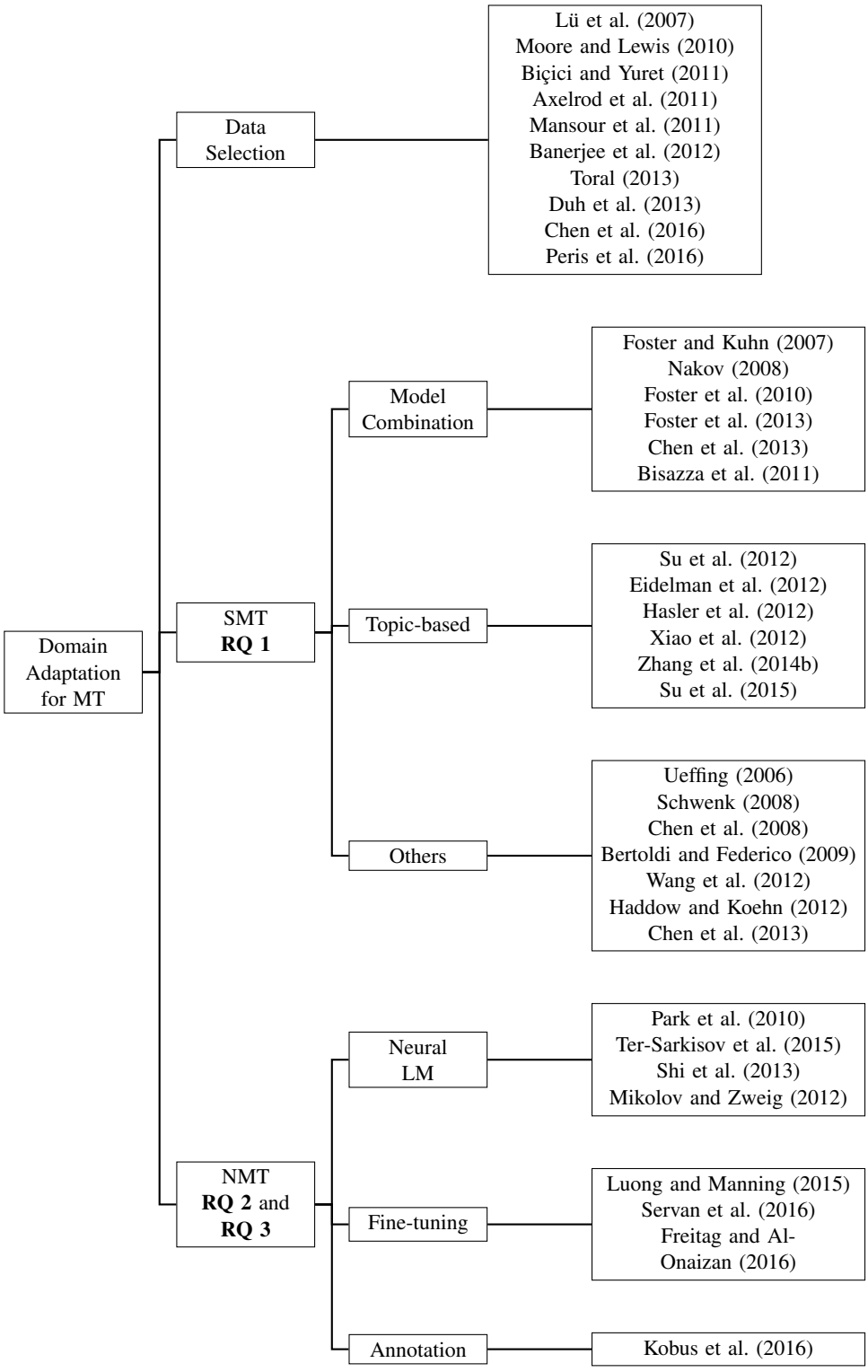



Figure 2.15: Domain adaptation related work

## 2.4 Summary

In this chapter, we provided detailed background information related to this thesis. We firstly reviewed the models and the overall framework in a PBSMT system. We discussed the BLEU evaluation metric. We also gave information about *n*-gram LM. We then studied the most recently proposed work on neural LM and NMT. We also reviewed selected domain-adaptation approaches in the MT literature in the categories of data selection, domain adaptation for SMT and domain adaptation for NMT.

In the next chapter, we will address our first research question:

> **RQ1** *In a domain-awareness scenario, how can we further improve the current domain adaptation method of an SMT by availing of the domain-likeness of the context in which a word or a phrase appears?*

We present a unique translation model combination approach which can be thought of as an extension of previous studies of Bisazza et al. (2011).

# Chapter 3

# Domain Adaptation for SMT by Probabilistic Combination of Models

## 3.1 Introduction

When there is a clear boundary between the domains in the training data, i.e. in the *domain-awareness* scenario, one approach is to train two separate translation models in Statistical Machine Translation (SMT), namely an In-Domain (ID) translation model and a General-Domain (GD) translation model. The two translation models can later be combined into a single global translation model (as seen in Figure 2.14). Such a combination method is often called 'model combination' in SMT (Foster and Kuhn 2007, Bisazza et al. 2011, Foster et al. 2013, Chen et al. 2014). It is an effective technique when the ID training data is too small in size and the vocabulary coverage is low (many untranslated words) in translation outputs because it uses the GD translation model to increase the vocabulary coverage as the GD training data is much larger in size. Furthermore, we can also maintain the topics or styles of the ID data if the model combination approach is used.

One model combination approach, namely the fill-up method of Nakov (2008), uses a feature value to define the origin of each phrase pair in the translation models, i.e. 1 and 0.5 for phrase pairs in the ID and GD phrase table, respectively. Bisazza et al. (2011) extend

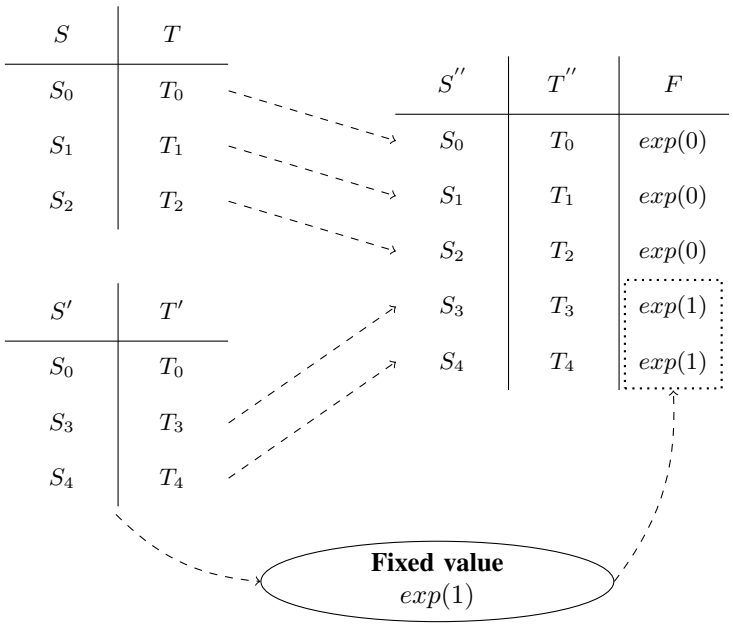



Figure 3.1: The fill-up (Bisazza et al. 2011) model combination approach with a provenance type feature, where ($S$,$T$), ($S'$,$T'$) and ($S''$,$T''$) represent the phrase pairs in the ID, GD and global translation models, respectively. $F$ represents the added feature column in the global translation model. The phrase pairs in the ID translation model are kept in the global translation model with the additional feature value of exp(0); the GD phrase pairs which can be found in the ID translation model are ignored, e.g. ($S_0$,$T_0$). Otherwise, the phrase pairs will be added into the global translation model with an additional feature value of exp(1), e.g. ($S_3$,$T_3$) and ($S_4$,$T_4$).

the original fill-up method with a provenance type feature: values of 2.718 (=$exp(1)$) and 1 (=$exp(0)$) are applied to the phrase pairs in the global translation model. The combination in Bisazza et al. (2011) uses rules as seen in Figure 3.1, where ($S$,$T$), ($S'$,$T'$) and ($S''$,$T''$) represent ID, GD and global translation models, respectively. All phrase pairs in the ID translation model are kept in the global translation model, with the provenance feature value of $exp(0)$. If a GD phrase pair can be found in the ID translation model, it will be ignored in the global translation model. Otherwise, phrase pairs in the GD translation model will be added into the global translation model with the provenance feature value of $exp(1)$.

The fill-up method of Bisazza et al. (2011) uses tuning to learn weights for the global translation model. The optimized weight of the provenance feature indicates a scaling factor of the phrase pairs added from GD translation model into the global translation model. For example, the tuning process outputs a single feature weight, which is then applied to the

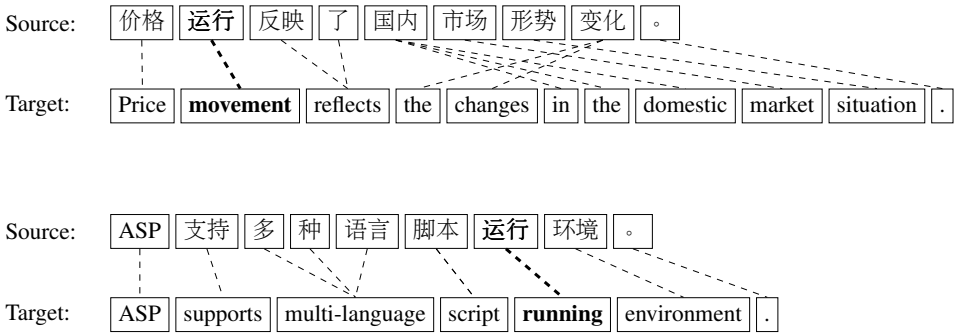



Figure 3.2: An example of phrase pair extraction from parallel sentences. We can extract phrase pairs "(运行, movement)" from the first sentence pair and "(运行, running)" from the second sentence pair.

provenance feature (either 2.718 or 1) during the decoding phase. Bisazza et al. (2011) show that the fill-up method can outperform other translation model combination approaches, i.e. linear or log-linear combinations, with less model weights to optimize, which motivated us to study it.

However, GD translation models are often trained using large corpora which comprise different domains. Some GD data can be more similar/dissimilar to the ID data. Therefore, if only features like 2.718 or 1 are used in the global translation model, the entire phrase pairs in the GD translation model will either be marked as close to or far away from the desired domain. The 'good' or 'bad' phrase pairs in the GD translation model are all used by the same tuned weights.

For example, in Figure 3.2, we can extract phrase pairs "(运行, movement)" and "(运行, running)" from the two Chinese-to-English sentence pairs. We assume these sentence pairs are extracted from the GD corpus. Thus, both phrase pairs will be added into the global translation model with the same additional feature value of $exp(1)$ and compete with each other. However, it is obvious that the phrase pair "(运行, running)" is a better phrase pair if 'Computer Technology' is the target domain, where more attention should be given to it than "(运行, movement)" in the global translation model.

Additionally, the phrase pair extraction step in Phrase-based Statistical Machine Translation (PBSMT) (Koehn et al. 2003) is not linguistically motivated. The contextual infor-

mation, i.e. words around the extracted phrase pairs, is largely ignored. The extraction process simply follows the heuristic that extracted phrase pairs need to be consistent with the corresponding word alignments (Koehn et al. 2003). However, according to the distributional hypothesis, word meanings are implied by the context rather than by the words themselves (Banchs 2014). The contextual information can be a useful indication to measure the closeness of a phrase pair from ID to GD. In this chapter, we seek to answer our first research question:

> **RQ1** *In a domain-awareness scenario, how can we further improve the current domain adaptation method of an SMT by availing of the domain-likeness of the context in which a word or a phrase appears?*

To answer **RQ1**, we propose a fine-grained translation model combination approach, which can be thought as an extension of previous studies (Nakov 2008, Bisazza et al. 2011). Instead of assigning the same feature value of $exp(1)$ to all phrase pairs in a GD translation model, we estimate and assign a probabilistic feature value to each phrase pair in the GD translation model. For example, we assign different feature values to the phrase pairs "(运行, running)" and "(运行, movement)" to give more attention to the one which is closest to ID. The probabilistic feature values are interpreted as the distance of phrase pairs to ID, i.e. phrase pairs with lower probability values indicate that they are close to ID (shorter distance, more likely to be in ID); phrase pairs with higher probability values indicate that they are far away from ID (longer distance, more likely to be in GD). It is worth mentioning that **RQ1** is related to the SMT model combination section (Section 2.3.2).

## 3.2 Our Approach

This section describes a fine-grained translation model combination approach. Figure 3.3 illustrates the proposed method, where ($S$,$T$), ($S'$,$T'$) and ($S''$,$T''$) represent ID, GD and global translation models, respectively. Phrase pairs in the ID translation model are kept in the global translation model with the additional feature value $exp(0)$. GD phrase pairs

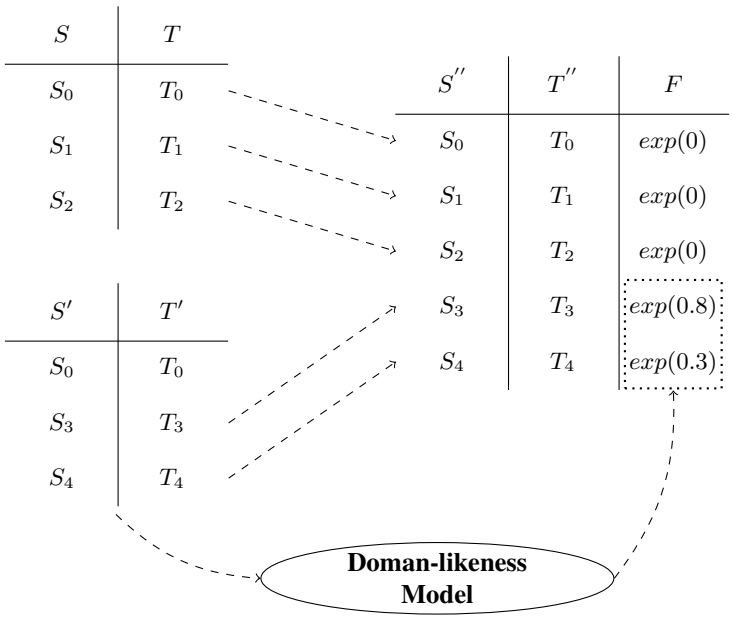



Figure 3.3: The fill-up model combination approach with probabilistic features, where ($S$,$T$), ($S'$,$T'$) and ($S^{''}$,$T^{''}$) represent ID, GD and global translation models, respectively. $F$ represents the feature column. Phrase pairs in the ID translation model are kept in the global translation model with the additional feature value exp(0). GD phrase pairs which can be found in the ID translation mode are ignored, i.e. ($S_0$,$T_0$). Otherwise, phrase pairs will be added into the global translation model with an additional probabilistic feature value, e.g. ($S_3$,$T_3$) and ($S_4$,$T_4$). The probabilistic feature values for GD phrase pairs are computed by a domain-likeness model.

which can be found in the ID translation model are ignored, i.e. ($S_0$,$T_0$). Otherwise, phrase pairs will be added into the global translation model with an additional probabilistic feature value, e.g. ($S_3$,$T_3$) and ($S_4$,$T_4$). The probabilistic feature values for GD phrase pairs are computed by a domain-likeness model.

The probability feature is interpreted as the distance from ID to GD, i.e. phrase pairs with lower probability feature values indicate that they have shorter distance to ID; phrase pairs with higher probability feature values indicate that they have longer distance to ID.

### 3.2.1 Domain-likeness Model

The domain-likeness model, which is trained using a Support Vector Machine (SVM) (Cortes and Vapnik 1995), which is a well-known machine-learning algorithm often applied to classification or regression tasks. In classification, SVM maps a testing instance into a

hyperplane which optimally separates the training data, and then outputs the predicted class label that the testing instance belongs to.

The advantage of choosing a SVM as our learning algorithm is that it supports kernels. In some situations, the features used in training are linearly non-separable for a training algorithm. A kernel function is able to project those features into a high-dimensional space, e.g. by computing the similarities between the features using a similarity function. Thus, the separability of the features can be improved.

The objective function of SVM is defined by Cortes and Vapnik (1995) as in Equation (3.1):

$$\begin{aligned} \min_{w,b,\xi} \quad & \tfrac{1}{2} w^T w + C \sum_{i=1}^{l} \xi_i \\ \text{s.t.} \quad & y_i(w^T \phi(x_i) + b) \geq 1 - \xi_i, \\ & \xi_i \geq 0, y \in \{1, -1\} \end{aligned} \tag{3.1}$$

where $w$ is the weight vector, $C$ is a tunable trade-off parameter indicating a penalty for misclassified decisions, $l$ is the number of training instances, and $\xi_i$ is known as the *slack* variable. $\phi$ is the kernel function (Radial Basis Function), which can be defined as in Equation (3.2):

$$\phi(u, v) = exp\{-\gamma |u - v|^2\} \tag{3.2}$$

where the gamma parameter $\gamma$ is a tunable variable which adjusts the width of the kernel function. SVM can not only predict class labels, but also give a probability estimation for every prediction (Chang and Lin 2011). In our work, we use this predicted probability to indicate the domain-likeness estimation.

The domain-likeness model is trained using the SVM implementation of Dimitriadou et al. (2009). It is an interface to the libsvm (version 2.6) (Chang and Lin 2011) implementation.

### 3.2.2 From Phrase Pairs to Sentence Pairs

Figure 3.3 shows that we need to predict a probability value for a given phrase pair using the domain-likeness model. However, distinguishing domains directly for phrase pairs is a great challenge: phrases are too short in a translation model. The typical phrase pair length (either source-phrase or target-phrase) is 7 tokens in the default translation model in Moses (Koehn et al. 2007), and phrase pairs with 3 tokens in length already have the greatest contribution to translation outputs (Koehn et al. 2003). Previous work shows that distinguishing the domains for phrases can be useful to the translation models by employing a large set of feature to distinguish phrase sense Carpuat and Wu (2007). Giménez and Màrquez (2007) use only source language with a local-context (5 tokens to the left and to the right), and words, parts-of-speech and lemmas features are also used. Neale et al. (2016) use rich semantic ontologies from WordNet (Miller 1995) on a weighted graph representation to perform word sense disambiguation.

In this work, we make the assumption that phrase pairs are in the same domain as the sentence pairs from which they are extracted to overcome the challenge of distinguishing domains directly for phrase pairs. Such an assumption is often applied in SMT data selection algorithms for domain adaptation (Moore and Lewis 2010). In Moore and Lewis (2010), the cross-entropy values are computed at the sentence level. The selected ones are then used for phrase extraction which are added into the translation models. In our case, we first find out the sentence pair from which a given phrase pair is extracted from, then we compute a probability estimation for that sentence pair. Therefore, the contextual information of the phrase pairs is used.

### 3.2.3 Domain-likeness Model Feature Set

Motivated by Moore and Lewis (2010), Axelrod et al. (2011), where cross-entropy values are used as evidence to separate domains, we also use cross-entropy values as our features for the domain-likeness model training.

In Axelrod et al. (2011), sentence pairs are ranked based on cross-entropy difference of

domains, such as in Equation (3.3):

$$\begin{aligned} &Score(s,t|S_{IN},S_{GD},T_{IN},T_{GD}) = \\ &\quad H_{LM(S_{IN})}(s) - H_{LM(S_{GD})}(s) + H_{LM(T_{IN})}(t) - H_{LM(T_{GD})}(t) \end{aligned} \tag{3.3}$$

where $s$ and $t$ are the given sentence pair to rank; and $S_{IN}$, $S_{GD}$, $T_{IN}$ and $T_{GD}$ represent the Language Model (LM) training corpus of source ID, source GD, target ID and target GD, respectively. We train LMs using $S_{IN}$, $S_{GD}$, $T_{IN}$ and $T_{GD}$, then compute $H_{LM(S_{IN})}(s)$, $H_{LM(S_{GD})}(s)$, $H_{LM(T_{IN})}(t)$ and $H_{LM(T_{GD})}(t)$ representing ID LM and GD LM the cross-entropy values for the given $s$ or $t$, accordingly. The sum of the cross-entropy differences, i.e. $H_{LM(S_{IN})}(s) - H_{LM(S_{GD})}(s)$ and $H_{LM(T_{IN})}(t) - H_{LM(T_{GD})}(t)$, are then used for ranking.

Accordingly, we design our feature template into three sets:

- **Domain Features in Source Language**: the domain evidence shown from the source side of the training data. We use the cross-entropy values computed from the ID and GD LM in this feature set, i.e. $H_{LM(S_{IN})}(s)$ and $H_{LM(S_{GD})}(s)$ in Equation (3.3).
- **Domain Features in Target Language**: the domain evidence shown from the target side of the training data. We use the cross-entropy values computed from the ID and GD LM in this feature set, i.e. $H_{LM(T_{IN})}(t)$ and $H_{LM(T_{GD})}(t)$ in Equation (3.3).
- **Domain Distance Features**: a feature set representing the cross-entropy differences, i.e. $H_{LM(S_{IN})}(s) - H_{LM(S_{GD})}(s)$, $H_{LM(T_{IN})}(t) - H_{LM(T_{GD})}(t)$ and $Score(s,t|S_{IN},S_{GD},T_{IN},T_{GD})$ in Equation (3.3).

Following Axelrod et al. (2011), $S_{IN}$ and $T_{IN}$ can be obtained directly from the ID SMT training data; $S_{GD}$ and $T_{GD}$ are randomly selected from the GD SMT training data; tokens in $S_{GD}$ and $T_{GD}$ are treated as unknown tokens unless they appear at least twice in $S_{IN}$ and $T_{IN}$, respectively.

One concern is that a phrase pair in a translation model can be extracted from a number of different training sentence pairs. Accordingly, those training sentence pairs will be

estimated to have different feature values by our domain-likeness model. We define the following three simple heuristics to address this issue:

- $Min$: the feature value uses the minimum domain-likeness estimations from the extracted sentence pairs. The motivation for this is that if a phrase pair is extracted from a sentence pair which has strong evidence to be excluded from GD, such a phrase pair should not be classified as ID.

- $Arithmetic\ Mean$: use the arithmetic mean of all the domain-likeness estimations. There is no bias to any sentence pair since all sentence pairs will be able to contribute to the final feature value.

- $Geometric\ Mean$: use the geometric mean value to describe the central tendency of all domain-likeness estimations.

### 3.2.4 Domain-likeness Model Training

We use the training data of French-to-English language pair from the Workshop on Statistical Machine Translation (WMT) and International Workshop on Spoken Language Translation (IWSLT) translation tasks as the same language pair is also used in Bisazza et al. (2011). In the WMT data, News Commentary and Europarl data (Koehn 2005) are used as the ID and GD corpus, respectively.[1] In the IWSLT data, Technology, Entertainment, Design (TED) and news-commentary-v9 data (Tiedemann 2012) are used as as the ID and GD corpus, respectively. We first perform some standard data-cleaning steps, including tokenization, punctuation normalization, replacement of special characters, lower casing and long sentence removal (>80), resulting in the preprocessed data summarized in Table 3.1. We use scripts provided within Moses (Koehn et al. 2007) for all cleaning steps.

The experimental setup is to assess our approach in both of the following situations: (i) the GD dataset being significant larger than the ID data, as seen in the WMT corpus, and (ii) the two datasets being similar in size, as seen seen in the IWSLT corpus.

[1] http://www.statmt.org/wmt07/shared-task.html

| | Corpus | Sentences | Tokens (French/English) |
|---|---|---|---|
| **WMT** | ID Training (News Commentary) | 42,884 | 1,197k/1,018k |
| | GD Training (Europarl) | 1,257,436 | 37,487k/33,788k |
| | Development (news-dev2007) | 1,064 | 31k/25k |
| | Test (news-test2007) | 2,007 | 58k/49k |
| **IWSLT** | ID Training (TED) | 106,642 | 2,152k/2,016k |
| | GD Training (news-commentary-v9) | 181,274 | 5,580k/4,509k |
| | Development (ted-dev2010) | 934 | 20k/20k |
| | Test (ted-test2010) | 1,664 | 33k/31k |

Table 3.1: Corpus statistics of the French-to-English language pair

| | ***M*** | ***N*** | ***T*** |
|---|---|---|---|
| **WMT** | 40,000 | 2,884 | 1,064 |
| **IWSLT** | 50,000 | 5,000 | 934 |

Table 3.2: Domain-likeness model training data statistics, where *M*, *N* and *T* are the data sizes (in sentences) used for training, tuning and testing, respectively.

| | $C$ | $\gamma$ | **ID Accuracy** |
|---|---|---|---|
| **WMT** | 16 | 0.125 | 81.39% |
| **IWSLT** | 2 | 0.03125 | 85.65% |

Table 3.3: Domain-likeness models tuned parameters $C$ and $\gamma$, where $C$ is the trade-off parameter in Equation (3.1), and $\gamma$ adjusts the width of the kernel function in Equation (3.2). We use the SMT tuning data to test our domain-likeness model, with the accuracy based on ID.

The data statistics of our domain-likeness model for training, tuning and test sets are summarized in Table 3.2. Following Moore and Lewis (2010), we randomly select (from Table 3.1) the equal number (size *M*) of sentences as ID and GD corpus to train the domain-likeness model; *N* training sentences to tune the parameters in Equations (3.1) and (3.2). The test data (size *T* in Table 3.2) for the domain-likeness models are the corresponding tuning data from Table 3.1.

In Moore and Lewis (2010), only 4-gram LMs are used to distinguish between ID and GD. In order to increase the variety of our feature set, we train *n*-gram LMs, where $n = \{2 \ldots 5\}$, as we described in Equation (3.3). We then extract the perplexity features from each *n* setting as the feature template we described in Section 3.2.2. The tuned parameters and model accuracies are presented in Table 3.3.

| | Algorithms | Tuning | Test |
|---|---|---|---|
| **WMT** | Binary Fill-up (Bisazza et al. 2011) | 27.33 | 28.01 |
| | Probabilistic Fill-up ($Min$) | 27.27 | 28.03 |
| | Probabilistic Fill-up ($Arithmetic\ Mean$) | 27.84 | 28.21 |
| | Probabilistic Fill-up ($Geometric\ Mean$) | 28.00 | **28.37**‡ |
| **IWSLT** | Binary Fill-up (Bisazza et al. 2011) | 27.66 | 30.82 |
| | Probabilistic Fill-up ($Min$) | 28.05 | 30.73 |
| | Probabilistic Fill-up ($Arithmetic\ Mean$) | 28.14 | **31.64**‡ |
| | Probabilistic Fill-up ($Geometric\ Mean$) | 28.02 | 31.21‡ |

Table 3.4: BLEU scores of provenance fill-up and probabilistic fill-up. We use ‡ to indicate statistically significant (Koehn 2004) improvements upon the baseline fill-up system. The significance testing uses bootstrapping method (Koehn 2004) at the level p = 0.01 level with 1,000 iterations.

## 3.3 Experiments

### 3.3.1 SMT Experimental Setup

Training data in Table 3.1 are also used for our SMT experiments. The PBSMT systems are trained using Moses (Koehn et al. 2007). The reordering model is not included in our translation system since we are interested only in measuring the system effects coming from translation models. We use the word aligner MGIZA (Gao and Vogel 2008) for word alignments in both translation directions, and then symmetrize the word alignment models. The translation systems are tuned with minimum error rate training (Och 2003) using case-insensitive BLEU (Papineni et al. 2002) as the optimization measure. We use the Moses default LM toolkit KenLM (Heafield 2011) at tuning and decoding time. We set our baseline systems to be the fill-up system (Bisazza et al. 2011), which has been integrated within Moses.

### 3.3.2 Domain Adaptation Results

Table 3.4 reports our experimental results on the corresponding development and test sets. We use ‡ to indicate statistically significant (Koehn 2004) improvements over the baseline fill-up system. The significance testing uses the bootstrapping method (Koehn 2004) at the p = 0.01 level with 1,000 iterations.

The result on the WMT data experiment shows that the probabilistic feature fill-up systems using three heuristics for domain-likeness calculation can improve the translation

performance over the baseline system. The system using the central tendency heuristic ($Geometric\ Mean$) for domain-likeness estimation outperforms the others, obtaining 0.36 absolute BLEU score and 1.3% relative improvements over the baseline system, which is statistically significant.

In the IWSLT experiment, the $Geometric\ Mean$ calculation produces a strong BLEU score, 0.39 absolute (1.3% relative) higher compared to the baseline system. However, the $Arithmetic\ Mean$ calculation achieves the best result in this experiment with a 31.64 BLEU score, which is absolute 0.82 (2.66% relative) BLEU score higher than the baseline system on the test set. Both of the above two systems in the IWSLT experiment qualify as statistically significant improvements over the baseline system at $p = 0.01$ level. The $Min$ system produces a similar result with 0.1 absolute BLEU score difference compared to the baseline system.

### 3.3.3 Comparison with Data Selection

In this experiment, we compare our probabilistic feature fill-up approach with the data selection approach proposed in Axelrod et al. (2011). In general, data selection is one of the standard approaches used in domain adaptation in SMT. However, it requires us to train many SMT systems on different proportions of the selected GD data. We can then evaluate each system in order to determine the one with the best translation performance. In this experiment, we first rank the GD corpus according to the cross-entropy difference defined in Equation (3.3). We then select the top-$p$ proportion of the ranked GD corpus to concatenate to the ID corpus, which is then used to train the PBSMT systems. We employ the same experimental settings described in Section 3.3.1 for this set of experiments except that the word alignments are computed in advance using the combination of all ID and GD data.[2] The tuning and test data sets described in Table 3.1 are also used in order to compare these results with those results described in Table 3.4.

Figures 3.4 and 3.5 illustrate the effects of the selection proportion and BLEU score

[2]Thus, we do not need to run word alignment models for each system, which is time-efficient.

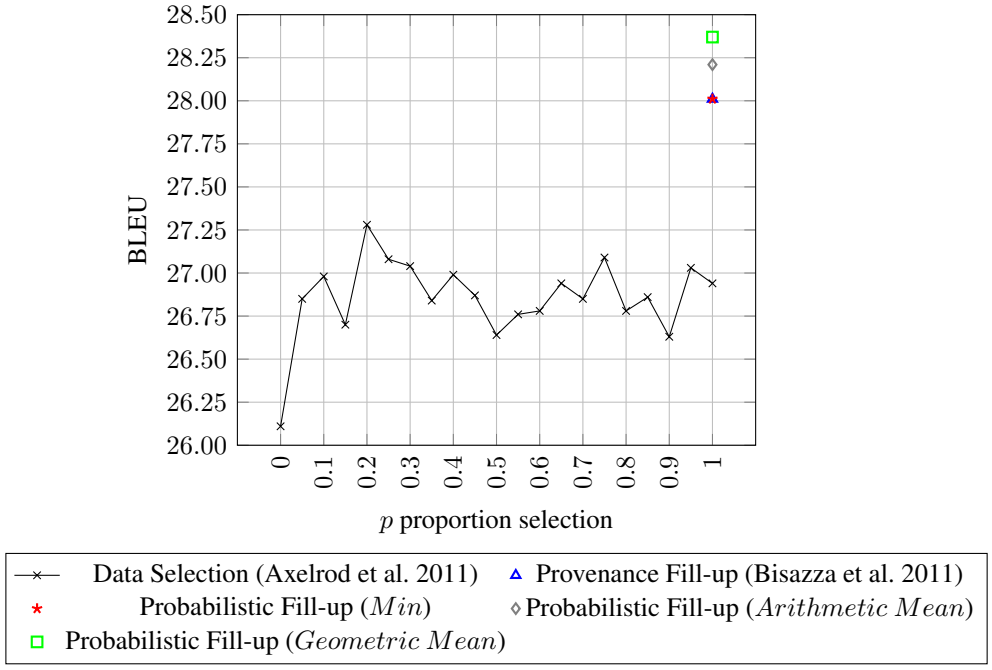



Figure 3.4: BLEU scores with different $p$ proportion of data selection on WMT data set, where the optimal translation system is $p$ = 0.2. All fill-up approaches can outperform the optimal translation system.

on the corresponding test data of the trained systems. As we might expect, additional GD training data can benefit translation performance, with 20% and 65% selections of GD, obtaining 27.28 and 31.73 BLEU scores on the WMT and IWSLT experiments, respectively. In addition, we found that it is harmful to translation quality if a large proportion of GD data is included, i.e. when $p$ >20% and $p$ >65% in the WMT and IWSLT experiments, respectively, as this results in the domain of the training data shifted from ID to GD.

In contrast, the provenance and probabilistic feature fill-up systems can outperform the optimal data selection system in the WMT data set. Thus, we conclude that GD data contribute positively to the translation system, and that data selection for SMT may be considered a somewhat heavy-handed approach. A similar conclusion is also drawn in Haddow and Koehn (2012). When $Geometric\ Mean$ is used to compute the domain-likeness feature value, we can observe an absolute 1.09 (3.98% relative) improvement in BLEU score compared to the optimal data selection system. This improvement is statistically significant at the p = 0.01 level with 1,000 iterations.

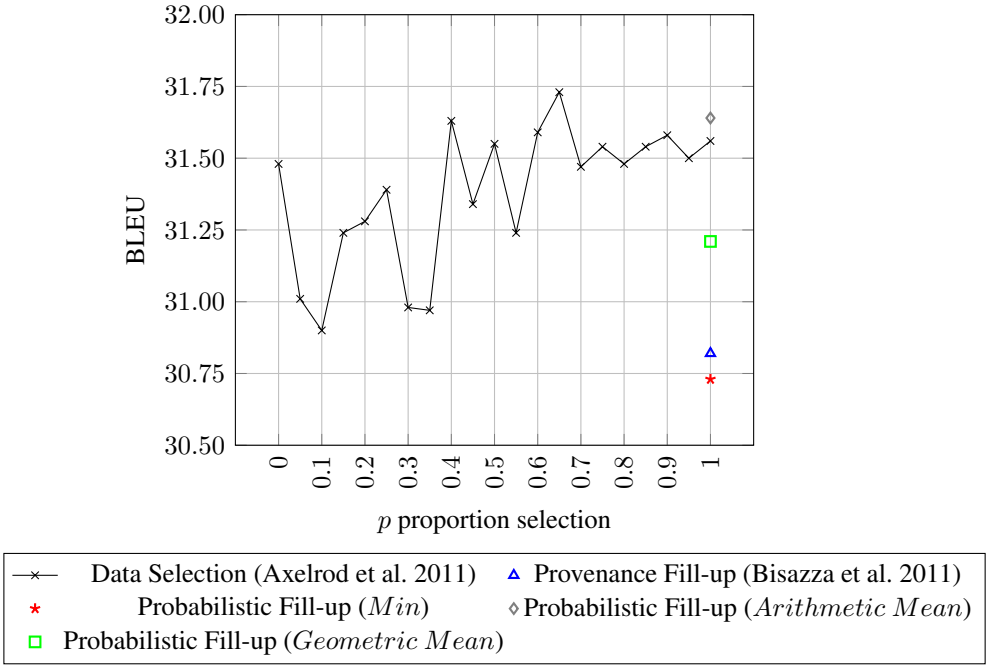



Figure 3.5: BLEU scores with different $p$ proportion of data selection on IWSLT data set, where the optimal translation system is $p$ = 0.65. Fill-up ($ArithmeticMean$) produces a comparable result with the optimial translation system in the data selection experiment.

However, the optimal data selection system in the IWSLT experiment performs better than all the provenance and probabilistic feature fill-up systems. We think the reason for this is that the GD data may be rather similar (useful) to ID. For example, the variance in translation quality is very small for the systems when $p$ >0.65, with the difference being only between 0.06 and 0.25 BLEU scores. Even the system trained using the concatenated ID and GD training data ($p$ = 1) can already achieve a good result, which is a 0.08 lower BLEU score than the best performing data selection system ($p$ = 0.65).

## 3.4 Analysis

In this section, we first study the distribution of the phrase pairs added into the final merged translation model. We then provide examples of phrase pairs regarding the probabilistic domain-likeness feature.

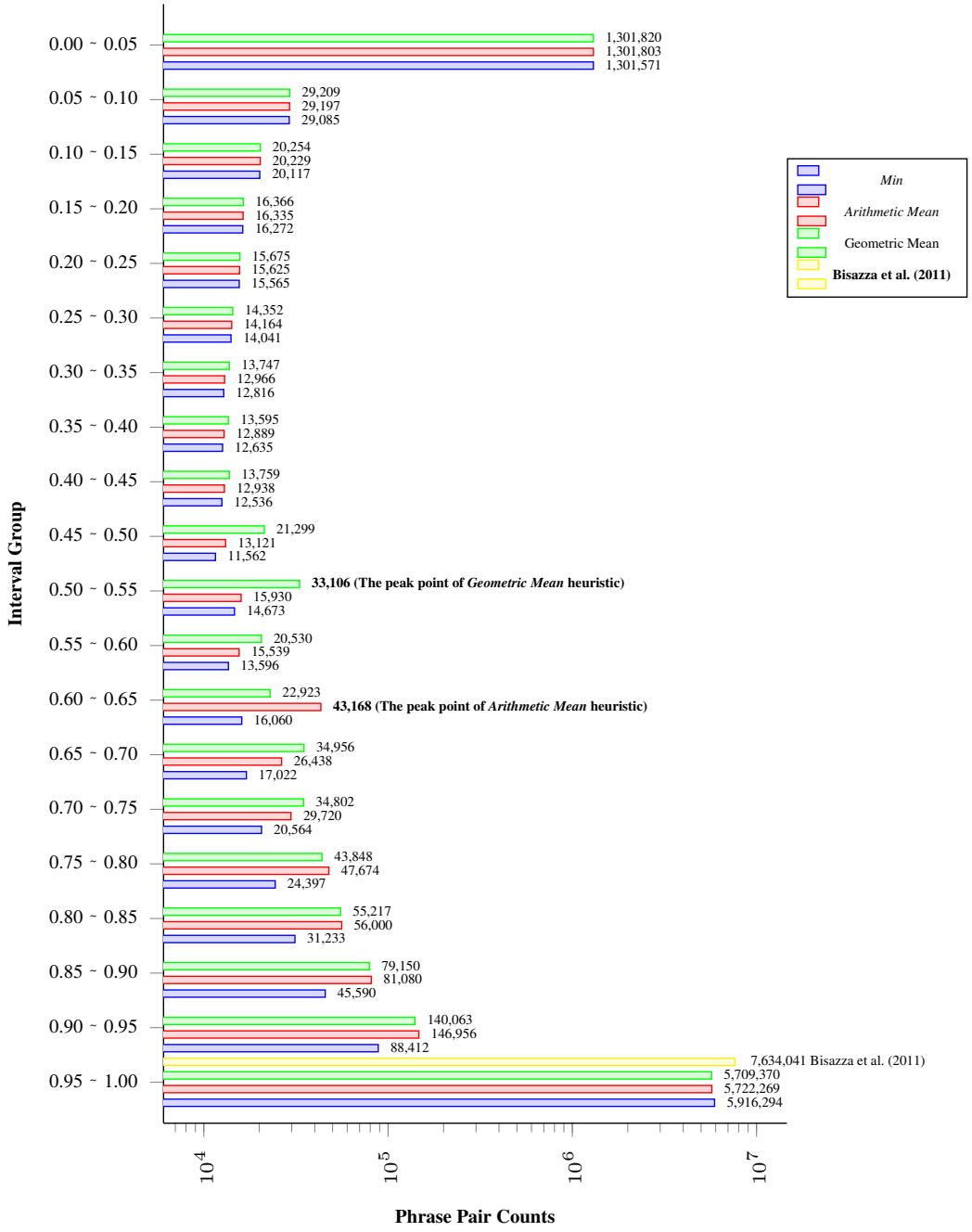



Figure 3.6: Filtered GD translation model phrase pairs counts with intervals of 0.05 according to the domain-likeness feature value in the IWSLT experiment. The phrase pair counts are displayed on a logarithmic scale on the x-axis. The numbers beside each bar indicate the number of phrase pairs which fall into the corresponding interval. For example, there are 17,022, 26,438 and 34,956 of phrase pairs fall into the intervals of 0.65 ~ 0.70 when the Min, Arithmetic Mean and Geometric Mean heuristics are used, respectively. The approach of Bisazza et al. (2011) assigns a fixed value of 1.0 to all GD phrase pairs.

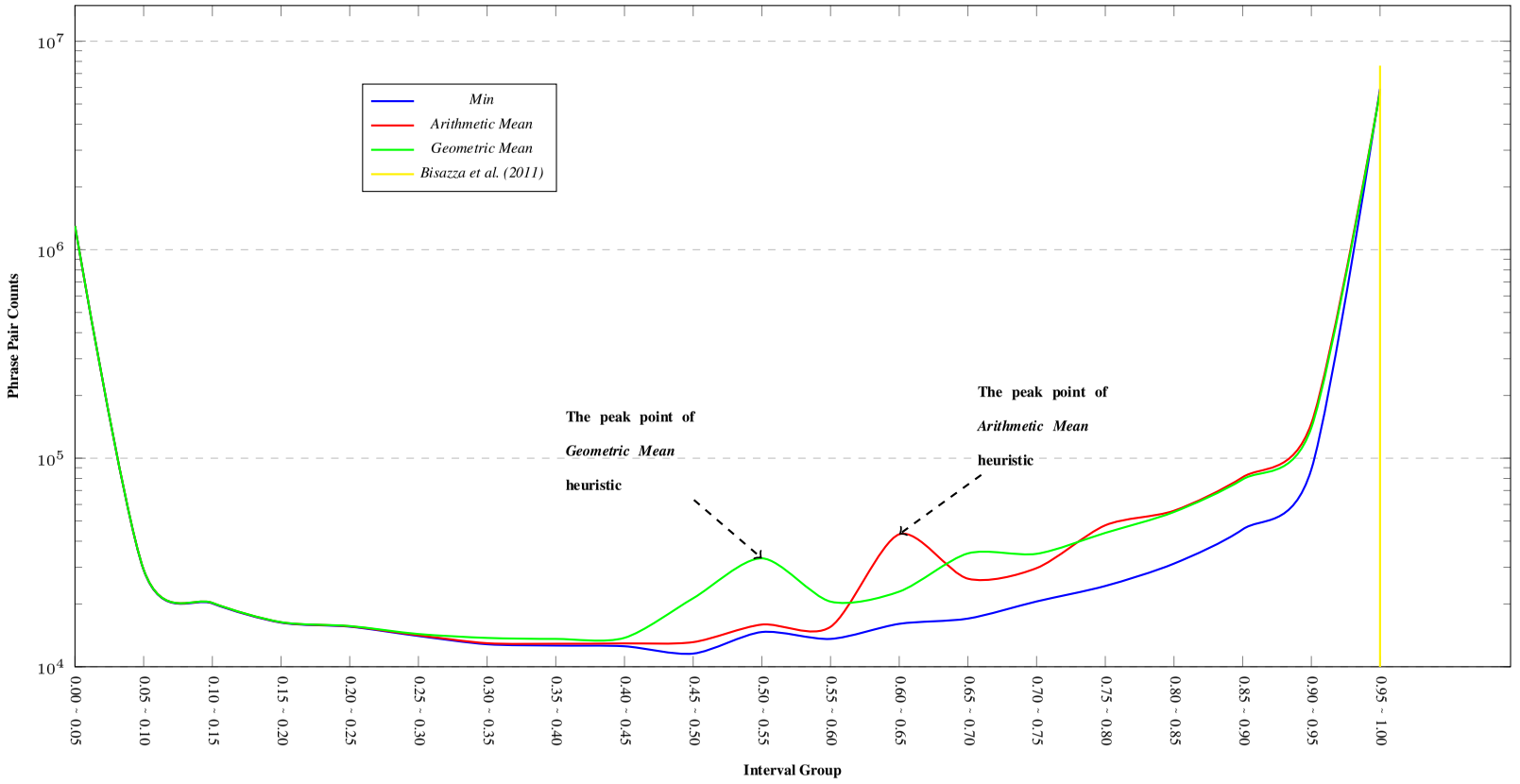



Figure 3.7: Filtered GD translation model phrase pairs counts with intervals of 0.05 according to the domain-likeness feature value in the IWSLT experiment. The phrase pairs counts are displayed on a logarithmic scale on the y-axis.

### 3.4.1 Distributions

The main difference between our approach with the previous fill-up method (Bisazza et al. 2011) is the additional features employed: a probabilistic domain-likeness feature is used in this work, while a provenance feature is applied in previous work. It is easy to establish that the ID part of the produced translation model is identical in both approaches, and that the total number of phrase entries is also the same. Thus, we mainly focus on the phrase pairs of the GD translation model in this section. We take the IWSLT experiment as a case study.

The IWSLT experiment merges translation models trained separately on different domains: 5,790,068 and 12,915,64 phrase pairs can be found in the ID (TED corpus) translation model and the GD (news-commentary-v9 corpus) translation model (filtered using the corresponding test set), respectively. There are 236,779 phrase pairs in the ID translation model which conflict with the phrase pairs in the GD translation model, and so they are neglected. The final global translation model contains 18,468,938 phrase pairs, whereas the standalone translation model using the concatenated ID and GD corpus produces 18,339,548 phrase pairs.

Figure 3.6 demonstrates the distributions of the GD phrase pairs in the merged phrase table in this work and that of Bisazza et al. (2011). For the phrase pairs in this work, we first group the phrase entries in the merged phrase tables with intervals of $0.05$ according to the domain-likeness feature value. Recall that the domain-likeness feature value is interpreted as the distance from ID to GD, i.e. phrase pairs with lower probability values indicate that they are close to ID; phrase pairs with higher probability values indicate that they are far away from ID and close to GD. Therefore, the instances in the 0.00 ~ 0.05 interval are phrase pairs which are predicted to be the closest to ID by our domain-likeness model. All phrase pairs in Bisazza et al. (2011) are in the 0.95 ~ 1.00 interval since a fixed feature value ($exp(1)$) is used.

We can observe in Figure 3.6 that the domain-likeness model predictions fall mostly into the 0.00 ~ 0.05 or 0.95 ~ 1 intervals. We think that this prediction follows the natural

| | |
|---|---|
| ***Source*** | J'imaginais un truc dans le genre de la marche des pingouins, alors j'ai regardé Miguel. |
| ***Reference*** | I was imagining a march of the penguins thing, so I looked at Miguel. |
| **Bisazza et al. (2011)** | I was imagining the works of the penguins, so I looked Miguel. |
| **This work** | I was imagining the march of the penguins, so i looked at Miguel. |

Figure 3.8: The example shows that the domain-likeness probabilistic feature can be helpful in producing better translations. The source sentence has a phrase 'la marche des'. The expected translation is 'a march of' in the reference, and the phrase pair '(la marche des ||| the march of)' can be extracted from the GD corpus. Our approach is able to produce a better translation as the phrase pair '(la marche des ||| the march of)' (in the format of *source* ||| *target*) is determined to be close to ID. The example is selected from the test data in the IWSLT experiment.

composition of the GD data set, so that the composition can be described as consisting of some ID related sentences, some mixed-domain sentences and some GD sentences. All three heuristics create similar numbers of phrase entries for each interval group at the upper bound range: 0.00 ~ 0.45. Later, there is a dramatic increase in the quality of phrase pairs at the interval 0.50 ~ 0.55 for the $Geometric\_Mean$ system. A similar increase also can be found in the $Arithmetic\_Mean$ system at the interval 0.60 ~ 0.65, but the increasing curve is sharper compared with the growth in $Geometric\_Mean$. However, the curve of the $Min$ system is much smoother so that no fluctuation can be observed except a small improvement at the interval 0.50 ~ 0.55. The trend of the $Min$, $Geometric\_Mean$ and $Arithmetic\_Mean$ plots can be observed as in Figure 3.7 where we can see that the $Geometric\_Mean$ domain-likeness model predicts more phrase pairs that are close to ID than the $Min$ and $Arithmetic\_Mean$ domain-likeness models do. In this case, the approach in Bisazza et al. (2011) can be thought as a special case of our approach where all phrase pairs in the GD are assigned the feature value of 1 ($exp(1)$).

| Phrase Pairs | Probability |
|---|---|
| la marche des ||| the march of | 0.001 |
| ne voulons pas ||| aren't willing to do | 0.019 |
| nous ne sommes pas ||| we know we are not | 0.061 |
| laissez moi ||| let me | 0.102 |
| principalement ||| was predominantly | 0.999 |
| politique nucléaire ||| nuclear policy | 0.999 |
| sans précédent ||| undertake unprecedented | 0.999 |
| efficacité ||| effectiveness was limited | 0.999 |

Table 3.5: Phrase pair examples (in the format of *source* ||| *target*) with the domain-likeness probabilistic values, where lower probability values indicating the phrase pair is close to ID.

| Source Phrase | Target Phrase | Probability |
|---|---|---|
| dans le monde entier | around the world | 0.026 |
| | worldwide | 0.509 |
| | global platform | 0.541 |
| en particulier dans | at least in | 0.004 |
| | especially those in | 0.661 |
| | particularly given | 0.937 |

Table 3.6: This table shows that the phrase pair examples of the same source phrases are ranked according to the domain-likeness probabilistic values, where lower probability values indicate the likelihood that the phrase pair is close to ID. For example, the domain-likeness model determines that the target phrase 'around the world' is closer to ID than the target phrase 'global platform' for the same given source phrase 'dans le monde entier'.

### 3.4.2 Examples

In this work, we expect that the phrase pairs with a lower domain-likeness probability (indicating close to ID) will be selected in translations. Figure 3.8 shows a translation example. The source sentence has a phrase 'la marche des'. The expected translation is 'a march of' in the reference, and the phrase pair '(la marche des ||| the march of)' (in the format of source ||| target) can be extracted from the GD corpus. When the translation models are combined, our approach estimates that phrase pair '(la marche des ||| the march of)' is more close to ID, so it is selected in the translation. In contrast, in Bisazza et al. (2011) all phrase pairs extracted in the GD corpus are given the same feature value and compete with each other, which may be the reason that a different translation with the reference is produced. Table 3.5 shows examples of GD phrase pairs with the assigned probabilities in the IWSLT

experiment. For example, we can see that the phrase pair '(la marche des ||| the march of)' has a probability of $0.001$ as if it were extracted from ID.

Table 3.5 shows the phrase pairs that are ranked according to the probabilities estimated by the domain-likeness model. The phrase pairs are selected from the GD translation model in the IWSLT experiment. Recall that we use the TED data as the ID training data, and the news-commentary-v9 corpus as the GD training data in the IWSLT experiment. Both the TED and news-commentary-v9 data are spoken language data. However, we observe that the TED corpus is less formal than the news-commentary-v9 corpus. Therefore, we expect that the domain-likeness model can also make this distinction. For example, a phrase pairs assigned with a lower probability should be more informal than those with higher probabilities.

In Table 3.6, we can observe that the source phrase 'dans le monde entier' can have three different corresponding target phrases, which are ranked as 'around the world' being the closest to the ID and 'global platform' closest to the GD. We think such a ranking is accurate since the phrase 'around the world' is more informal than 'global platform'. A similar example can also be observed for the case of 'en particulier dans' as the source phrase in Table 3.6.

## 3.5 Contribution

In translation model combination, previous fill-up work tries to define a provenance feature value to all phrase pairs extracted from the GD training data. However, we think such a provenance feature may cause potential ID phrase pairs to be treated unfairly.

We extended the previous provenance feature to a domain-likeness probabilistic feature, which represents the domain-likeness of a phrase pair being in GD. Such an approach can distinguish the extracted phrase pairs in the GD phrase model. For example, higher feature values will be assigned to phrase pairs that are predicted to be closest to the GD; lower feature values will be assigned to the phrase pairs that are predicted to be far away to

the ID. This is a "*soft*" approach to combine translation models which can achieve better translation quality compared to previous work.

Furthermore, we confirmed several previous research findings, including that (i) data selection is a heavy-handed approach, (ii) the unselected GD data can still also make good contributions to the translation system, and (iii) it is harmful to translation quality if a large proportion of GD data is concatenated with ID data for SMT training.

As a side note, in the *domain-awareness* scenario, we do not distinguish sentences in the ID training data, such as there is only one boundary between the ID and GD training data and all sentences in the ID training data are equally important to the desired domain. Therefore, the domain-likeness model we proposed in this chapter does not make probability estimations on ID training data. However, it is possible to apply the domain-likeness model also on ID training data if we are not in such a scenario.

## 3.6 Summary

In this chapter, we described an efficient translation model combination approach to address **RQ1** in the *domain-awareness* scenario. We described the rationale behind our probabilistic feature fill-up approach and explained our intuitions regarding the domain-likeness model feature set.

We used the assumption that phrase pairs are in the same domain as the sentence pairs which they are extracted from. Our feature value is predicted using the contextual information of phrase pairs. Furthermore, we use the learned domain-likeness probabilistic feature to balance domain weights of phrase pairs in ID and GD model combination.

We also designed two experimental scenarios, demonstrating that our fill-up approach can significantly improve translation performance in both experiments compared to previous fill-up studies. We made comparisons and illustrated detailed analysis of the three heuristics used for computing the probabilistic feature value. We also compared our approach with the data selection approach and found that our method can outperform or pro-

duce comparable translation results, which can be thought as a substitution of data selection methods.

However, Machine Translation (MT) is an active research area in which technologies are being developed rapidly. With the availabilities of large training corpus and powerful computational resources, neural network training has demonstrated to be the state-of-the-art training algorithm in MT task. In the next chapters, we switch our attention to neural network training in the *domain-awareness* scenario.

# Chapter 4

# Domain Adaptation with Large Pre-trained Word Vector Models

## 4.1 Introduction

It has been over two decades since the conventional Statistical Machine Translation (SMT) (Brown et al. 1993) technique was proposed in the Machine Translation (MT) literature. The aim of MT is clear: to obtain high-quality translations faster and cheaper. SMT has certainly brought us closer to such a goal. However, new exciting and promising methods of how to translate text from one language to another language, or how to model a language, have also been developed rapidly at the same time. In the previous chapter, we investigated the model combination approach for domain adaptation under the traditional statistical training framework. In this chapter, we move our attention to the most recent proposed approach: neural network training. Our focus is on proposing a domain-adaptation approach which can be used in the state-of-the-art neural Language Model (LM) and Neural Machine Translation (NMT) (Bahdanau et al. 2015) models.

Firstly, researchers have been working on using a sequence-to-sequence neural architecture to map source to target languages. Neural network training differs from the statistical approaches: statistical approaches need hand-crafted features, such as the phrase translation

features in SMT or the *n*-gram features in the *n*-gram LM; however, neural approaches have the power to extract features automatically without human intervention. It is achieved by building up connections between nodes in layers and learning weights between them.

Secondly, word vector representations (Mikolov et al. 2013a) are used to represent words for almost all Natural Language Processing (NLP) tasks when neural network training is used. Such representations are known to be better at generalization than plain text format (Mikolov et al. 2013b). For example, assume the following two sentences:

**Sentence 1:** The $cat$ is $sitting$ on the $mat$.

**Sentence 2:** The $dog$ is $sleeping$ on the $rug$.

A neural network is able to learn which words are semantically close and then switch one to a neighbouring one. The network can learn that the word pairs $cat$ and $dog$, $sitting$ and $sleeping$ or $mat$ and $rug$ are much closer to each other in the high-dimensional space.

Thirdly, the two well-known pre-trained word vector models – the Google *word2vec* model (Mikolov et al. 2013a) and the GloVe (Global Vectors for Word Representation) (Pennington et al. 2014) model – have been successful applied in many previous work (Mikolov et al. 2013b, Kim 2014, Zhang et al. 2014a). Both of them are trained with a large amount of data which are either publicly unavailable (in the case of *word2vec*) or only partially available (in the case of GloVe). They are often used in the situation when the relevant data is limited in NLP tasks. For example, the human annotated training data in the text classification task is often too small and difficult to scale. In practice, the pre-trained word vectors are only used to initialize the embedding layer in the network.

Accordingly, our hypothesis is that the pre-trained and the task-specific-trained word vector models are complementary with each other in a neural network training. The pre-trained word vector models can be applied to overcome the challenge that In-Domain (ID) training data is too small. However, the task-specific-trained word vector model cannot be substituted to the pre-trained one directly since they are trained on different domains (the pre-trained word vector models are often trained using very large General-Domain (GD)

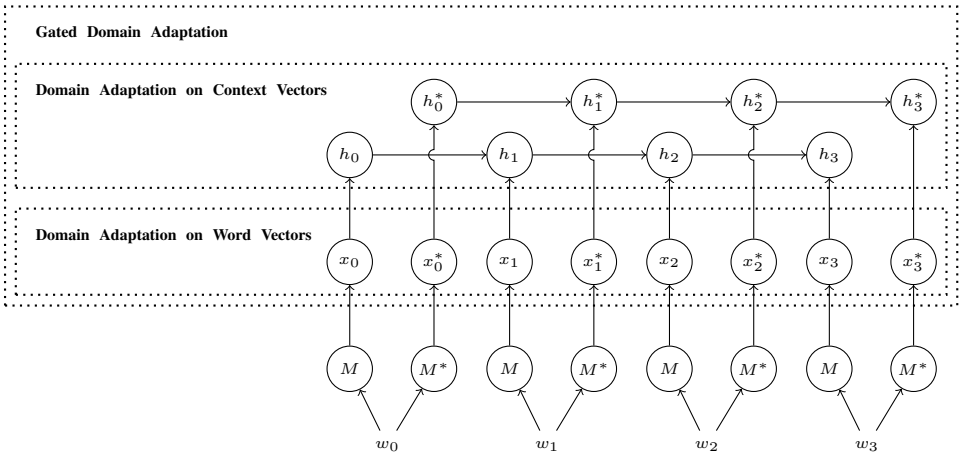



Figure 4.1: A high-level overview of the proposed domain adaptation mechanisms using neural language model as an example. The adaptation processes are on word vectors (Section 4.2.1), context vectors (Section 4.2.2) or using gating mechanisms (Section 4.2.3).

training data). In this chapter, we address our second research question:

> **RQ2** *Whether the vector model trained using GD data can be used in domain adaptation in a domain-awareness scenario?*

We propose to perform domain adaptation from the pre-trained word vector model into the task-specific one. With such an approach, we can make use of huge GD corpora with little speed overhead and also adapt the richer word representations leaned from GD data into ID training.

## 4.2 Our Approach

This section describes several domain-adaptation mechanisms we propose to address **RQ2**. We use Recurrent Neural Network (RNN) LMs as an example to illustrate the proposed approaches.

For notational convenience, the following notations are used in this section:

$w_t$ : input word in time $t$

$M$ : a task-specific word vector model trained using limited ID data

$M^*$ : a pre-trained word vector model trained using huge GD data

$x_t$ : the word vector of $w_t$ obtained from $M$

$x_t^*$ : the word vector of $w_t$ obtained from $M^*$

$\hat{x_t}$ : the domain-adapted word vector

$h_t$ : the context vector of input $x_t$

$h_t^*$ : the context vector of input $x_t^*$

$\hat{h_t}$ : the domain-adapted context vector

Figure 4.1 illustrates the high-level overview of the proposed adaptation approaches in this section. For example, we first propose to perform domain adaptation on word vectors ($x_t$ and $x_t^*$), then we apply the domain adaptation on context vectors ($h_t$ and $h_t^*$). Finally, we propose our gating mechanisms.

### 4.2.1 Adaptation on Word Vectors

We first propose to perform domain adaptation between $x_t$ and $x_t^*$. Given $w_t$, we can obtain $x_t$ and $x_t^*$ word vectors from $M$ and $M^*$, respectively. $x_t$ represents the word meaning in ID and $x_t^*$ represents the word meaning in GD.

We now formally define the operations of domain adaptation on word vectors:

1. Word Vector Concatenation (WVC): we concatenate $x_t$ and $x_t^*$, as in Equation (4.1):

$$\hat{x_t} = [x_t^*, x_t] \tag{4.1}$$

2. Weighted Word Vector Concatenation (WWVC): we can extend the WVC approach by applying a weight matrix $W$ to control the information flow from $M^*$ to $M$, as in Equation (4.2):

$$\hat{x_t} = [Wx_t^*, x_t] \tag{4.2}$$

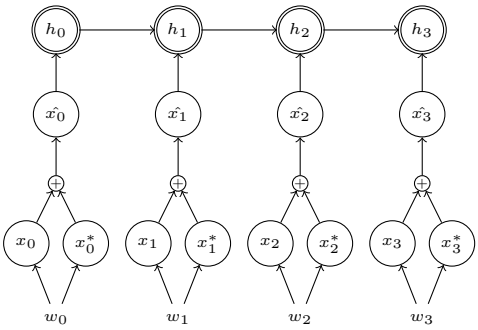



Figure 4.2: RNNLM training with the proposed domain adaptation on word vectors. $\oplus$ indicates the WVC, WWVC, WVS or WWVS domain adaptation operations. For example, we obtain $x_1$ and $x_1^*$ in time step 1. We then perform one of the proposed domain adaptation operations to compute $\hat{x}_1$, which is used for LM training. $h$s are used to map to the size of vocabulary for the *softmax* function.

3. Word Vector Sum (WVS): we sum the vectors of $x_t$ and $x_t^*$. In this approach, the two vectors need to have the same dimensionality, as in Equation (4.3):

$$\hat{x_t} = x_t^* + x_t \tag{4.3}$$

4. Weighted Word Vector Sum (WWVS): a similar weight control can also be applied to the WVS approach, as in Equation (4.4):

$$\hat{x_t} = W x_t^* + x_t \tag{4.4}$$

We then can replace $x_t$ with $\hat{x_t}$ in Equation (2.17) to obtain our proposed LM training with domain adaptation on word vectors. Figure 4.2 is the illustration of this approach. For example, we obtain $x_1$ and $x_1^*$ in time step 1. We then perform one of the proposed domain-adaptation operations to compute $\hat{x}_1$, which is used for LM training.

### 4.2.2 Adaptation on Context Vectors

We can delay the domain-adaptation step until the context information is available during our LM training. The RNN encapsulates the word vector of current word and previous histories, and then produces the current context vector. Intuitively, if we maintain separate

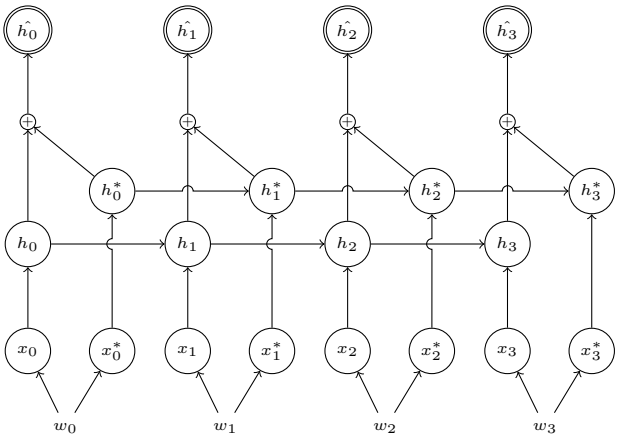



Figure 4.3: RNNLM training with the proposed domain adaptation on context vectors. $\oplus$ indicates the CVC, WCVC, CVS or WCVS domain adaptation operations. $\hat{h}$s are used to map to the size of vocabulary for the *softmax* function.

RNNs, where one uses $x_t$ and another one uses $x_t^*$, there will be two pieces of context information available to us, namely $h_t$ and $h_t^*$. $h_t$ and $h_t^*$ can be thought as the context vectors with the meaning in ID and GD, respectively.

We now formally define the domain-adaptation operations used on context vectors:

1. Context Vector Concatenation (CVC): we can concatenate the two context vectors $h_t$ and $h_t^*$, as in Equation (4.5):

$$\hat{h_t} = [h_t^*, h_t] \tag{4.5}$$

2. Weighted Context Vector Concatenation (WCVC): we can extend the CVC approach by applying a concatenation weight on $h_t^*$ in Equation (4.5). Thus, the network can have some simple control over the amount of the information flowing from GD, as in Equation (4.6):

$$\hat{h_t} = [Wh_t^*, h_t] \tag{4.6}$$

3. Context Vector Sum (CVS): we can also sum $h_t$ and $h_t^*$. We then have the compacted representation of two vectors, as in Equation (4.7):

$$\hat{h_t} = h_t^* + h_t \tag{4.7}$$

4. Weighted Context Vector Sum (WCVS): another approach is to apply a weight matrix to $h_t^*$. Thus the information from GD can be controlled before compacting, as in Equation (4.8):

$$\hat{h_t} = Wh_t^* + h_t \tag{4.8}$$

We then can replace $h_t$ with $\hat{h_t}$ in Equation (2.17) to obtain our proposed LM training with domain adaptation on context vectors. The previous hidden state $h_{t-1}$ in Equation (2.17) is also updated accordingly. Figure 4.3 is the illustration of LM training with domain adaptation on context vectors.

It is worth mentioning that the non-weighted operations (WVC, WVS, CVC and CVS) are the special cases of the corresponding weighted operations (WWVC, WWVS, WCVC and WCVS, respectively) with the weight matrix are all 1s. However, one advantage of non-weighted approaches is that less training parameters are used. More details about training parameters will be described in Section 4.3.

### 4.2.3 Gated Domain Adaptation

However, directly applying adaptation on the word or context vectors may not be efficient since the concatenation or sum operations are too simple. Furthermore, the outputs of the adaptation operations are discarded at each time stamp and not being used at the next time step. For example, $\hat{h_0}$ in Figure 4.3 is only used to predict the probability for step $0$, but not being involved in the computation of step $1$. Ideally, we want to have adaptation operations that can learn from GD at time step $n$, which also can be sequentially used at time step $n+1$, as if the RNN. Thus, we propose various gated domain-adaptation mechanisms.

We now formally define the operations used in gated adaptation approach:

1. Gated Word Vector Adaptation (GWVA): in the GWVA approach, we first design a gate to control the information flow from $w_t^*$, as in Equation (4.9):

$$u_t^{GWVA} = \sigma(W_u x_t + U_u h_{t-1} + W_u^* x_t^* + U_u^* h_{t-1}^* + b_u) \tag{4.9}$$

where $u_t^{GWVA}$ is the update gate on word vectors. It is computed using the known knowledge of $x_t$, $x_t^*$, $h_{t-1}$, and $h_{t-1}^*$. When applying $u_t^{GWVA}$, we use a linear sum to combine $x_t$ and $x_t^*$, as in Equation (4.10):

$$\hat{x_t} = u_t^{GWVA} \odot x_t + (1 - u_t^{GWVA}) \odot x_t^* \tag{4.10}$$

where $\hat{x_t}$ is the domain-adapted word vector. Such an adaptation approach ensures that when the gate $u_t^{GWVA}$ tends to 1, we only use the ID word vector $x_t$, and when $u_t^{GWVA}$ tends to 0, the information from $x_t^*$ is fully cascaded to the $\hat{x_t}$. To use $\hat{x_t}$, we can simply replace it with the original $x_t$ in Equation (2.17).

2. Gated Context Vector Adaptation (GCVA): a similar gating mechanism can also be applied to the context vector, as in Equation (4.11):

$$r_t^{GCVA} = \sigma(W_r x_t + U_r h_{t-1} + W_r^* x_t^* + U_r^* h_{t-1}^* + b_r) \tag{4.11}$$

where $r_t^{GCVA}$ is the update gate on context vectors. We can also use the linear sum operation to combine $h_t$ and $h_t^*$, as in Equation (4.12):

$$\hat{h}_{t-1} = r_t^{GCVA} \odot h_{t-1} + (1 - r_t^{GCVA}) \odot h_{t-1}^* \tag{4.12}$$

We then use the domain-adapted context vector $\hat{h}_{t-1}$ to replace the original context vector $h_{t-1}$ in Equation (2.17).

3. Gated Domain Adaptation (GDA): the GWVA and GCVA operations can be combined. Therefore, we can have full control to the information flowing from GD in training.

Figure 4.4, 4.5 and 4.6 illustrate the LM training with domain adaptation using GWVA, GCVA and GDA, respectively. In Figure 4.6, $h_1$ is computed by $\hat{x_1}$ and $\hat{h_0}$, where $\hat{x_1}$ is the adapted word vector and $\hat{h_0}$ is the adapted context vector. $\hat{x_1}$ is obtained by performing

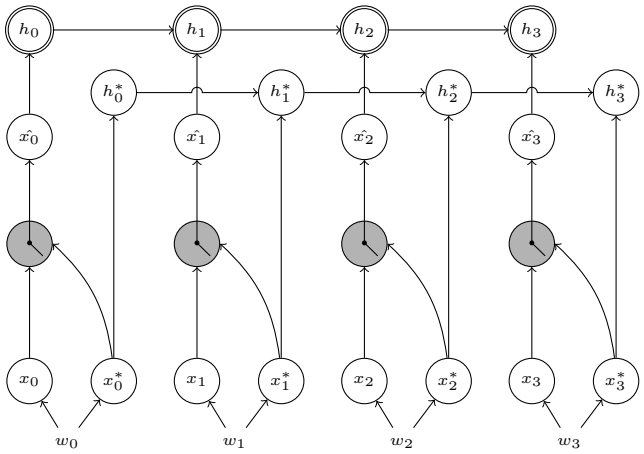



Figure 4.4: RNNLM training with the proposed gated adaptation on word vectors. The shadow nodes indicate the gating operations. For example, $h_1$ is computed by $\hat{x_1}$ and $h_0$. $\hat{x_1}$ is the adapted word vector, where the domain adaptation operation is applied from $x_1$ and $x_1^*$ as seen in Equation (4.10). $h_1^*$, $h_2^*$, $h_3^*$ and $h_4^*$ are used to compute the gate as seen in Equation (4.9). $\hat{h}$s are used to map to the size of vocabulary for the *softmax* function.

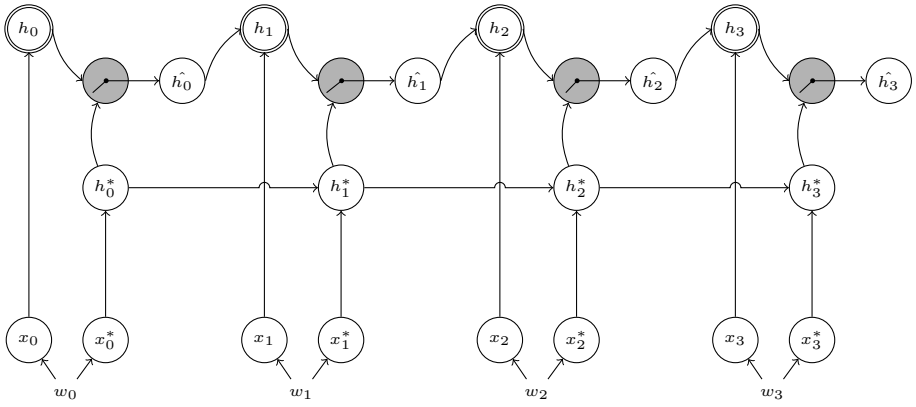



Figure 4.5: RNNLM training with the proposed gated adaptation on context vectors. The shadow nodes indicate the gating operations. For example, $h_1$ is computed by $x_1$ and $\hat{h_0}$, where $\hat{h_0}$ is the adapted context vector and obtained by performing domain adaptation operation on $h_0$ and $h_0^*$ as seen in Equation (4.12). $\hat{h}$s are used to map to the size of vocabulary for *softmax* function.

domain-adaptation operation on $x_1$ and $x_1^*$ as seen in Equation (4.10); $\hat{h_0}$ is obtained by performing domain-adaptation operation on $h_0$ and $h_0^*$ as seen in Equation (4.12).

It is also worth mentioning that the pre-trained GD word vector model is static, which means it is not updated during training. This is because the pre-trained word vector model is obtained from a very large GD data set. The word vectors in such a model are not domain-specific. By keeping it static, we interpret it as a 'knowledge database', and the knowledge

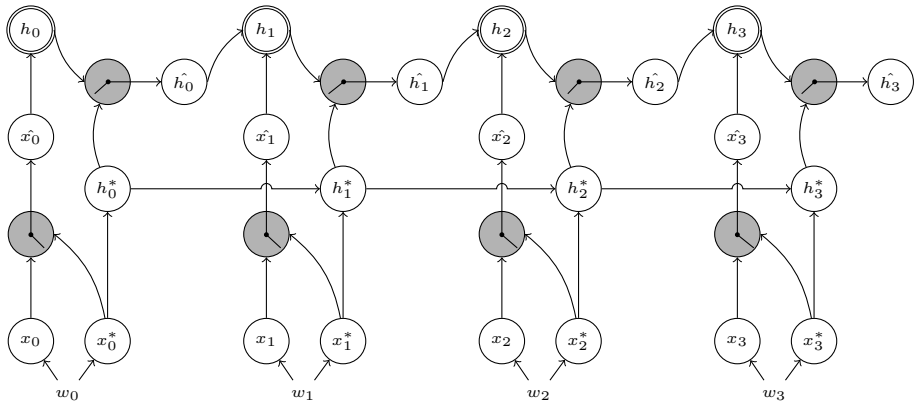



Figure 4.6: RNNLM training with the proposed GDA operation. The shadow nodes indicate the gating operations. For example, $h_1$ is computed by $\hat{x_1}$ and $\hat{h_0}$, where $\hat{x_1}$ is the adapted word vector and $\hat{h_0}$ is the adapted context vector. $\hat{x_1}$ is obtained by performing domain adaptation operation on $x_1$ and $x_1^*$ as seen in Equation (4.10); $\hat{h_0}$ is obtained by performing domain adaptation operation on $h_0$ and $h_0^*$ as seen in Equation (4.12). $\hat{h}$s are used to map to the size of vocabulary for the *softmax* function.

should be consistent. Another practical reason for not updating the pre-trained GD word vector model is that fewer parameters need to be optimized in the network. In general, models with fewer parameters can make training converge faster.

## 4.3 Language Models and SMT Reranking Experiments

To demonstrate that the proposed approach can be successfully applied in different sequence-to-sequence prediction tasks, we use neural LM experiment in this section and NMT experiment in the next section.

### 4.3.1 Experimental Setup

Recall that the setting of the *domain-awareness* scenario is a small amount of ID training data and a significant amount of GD training data. In our first set of experiments, we thus choose to use the widely known Penn Treebank (Marcus et al. 1993) portion of the Wall Street Journal corpus as the ID domain training data,[1][2] and the pre-trained word vector

[1]We download the data from `http://www.fit.vutbr.cz/~imikolov/rnnlm/simple-examples.tgz`

[2]The most widely used data sets for evaluating performance of LMs.

| | Sentences | Tokens |
|---|---|---|
| Training | 42,068 | 887,521 |
| Validation | 3,370 | 70,390 |
| Test | 3,761 | 78,669 |

Table 4.1: Statistics of the Penn Treebank corpus

| | Sentences | Tokens | Vocabulary Size |
|---|---|---|---|
| Training | 181,108 | 4,691k | 72,828 |
| Validation | 3,000 | 64k | 7,761 |
| Test | 3,003 | 71k | 8,187 |

Table 4.2: Statistics of the News corpus

Google *word2vec* (Mikolov et al. 2013a) as GD data.[3] For the Penn Treebank data, the words outside the 10K vocabulary frequency list are mapped to the special Unknown (UNK) token; sections 0-20 are used for training, and sections 21-22 are used for validation. We report the perplexity on data from sections 23-24. The pre-trained word vector Google *word2vec* model is trained on about 100 billion words from different resources. The trained model consists of 3 million words and phrases. The word vectors are 300-dimensional in the *word2vec* model.

In our second set of experiments, the ID data is the news corpus of the target side of the French-to-English News Commentary v10 from the WMT2015 translation task. We use corpus newstest 2013 for validation and newstest 2014 for testing. For this set of data, we map the words outside the 16K vocabulary frequency list to the special UNK token. We also use the pre-trained word vector Google *word2vec* model as our GD data. More detailed data statistics for Penn Treebank and News Corpus are summarized in Table 4.1 and Table 4.2, respectively.

All LMs in our experiments are trained with a single Gated Recurrent Unit (GRU) hidden layer containing 600 hidden units. We uniformly initialize the weight parameters between [-0.1,0.1]. We set the maximum number of training iterations to 25. We set the initial learning rate to be 1, and then apply the learning rate with a decay factor of 0.5 after 14 iterations. The model is optimized using Stochastic Gradient Descent (SGD) with batch

[3] `https://code.google.com/archive/p/word2vec/`

| | Validation Set | Test Set | Parameter Number |
|---|---|---|---|
| **Baselines** | | | |
| Baseline (*n*-gram) | 148.1 | 141.2 | N/A |
| Baseline (*word2vec*) | 121.9 | 117.7 | 1,417k |
| Baseline (RNN LM) | 93.0 | 89.3 | 1,417k |
| **Adaptation on Word Vectors** | | | |
| WVC | 95.1 | 91.4 | 1,063k |
| WWVC | 94.2 | 90.6 | 1,072k |
| WVS | 88.4 | 85.2 | 1,063k |
| WWVS | 104.3 | 100.7 | 1,072k |
| **Adaptation on Context Vectors** | | | |
| CVC | 90.4 | 86.2 | 1,225k |
| WCVC | 88.6 | 85.1 | 1,261k |
| CVS | 88.3 | 84.7 | 1,225k |
| WCVS | 90.3 | 86.7 | 1,261k |
| **Gated Adaptation** | | | |
| GWVA | 91.0 | 87.9 | 1,243k |
| GCVA | 90.3 | 86.8 | 1,330k |
| GDA | 86.2 | **81.9** | 1,387k |

Table 4.3: Language model perplexity on Penn Treebank corpus. This table lists all LM experiments using the proposed domain adaptation mechanisms on Penn Treebank corpus. The GDA approach achieves the best performance on both validation set and test set in perplexity. Parameter number indicates the number of training parameters in the corresponding model.

size of 20. We set the back-propagation through time to 40 time steps[4]. The word vector size in the input layer is set to 600 for the baseline models. For a fair comparison, we use word vector size of 300 for the ID word vector and 300 for the pre-trained word vectors in the adapted LMs.

### 4.3.2 RNNLM Adaptation on Penn Treebank

Table 4.3 lists the perplexity results in the LM experiments on Penn Treebank data. The baseline (*n*-gram) model is a 5-gram LM trained using modified Kneser-Ney smoothing (Chen and Goodman 1996). It results in 148.1 and 141.2 perplexities on the validation and test data set, respectively. The baseline (*word2vec*) model is a neural LM using the pre-trained *word2vec* model as the embedding layer. It can achieve 121.9 and 117.7 perplexities on the validation and test data set, respectively. The baseline LM, which is a standard neural LM trained without domain adaptation, can achieve an 99.0 and 89.2 perplexities on the

[4]The average number of tokens is 21 in Penn Treebank data. However, all sentences are concatenated and then sliced into 40 tokens for model training.

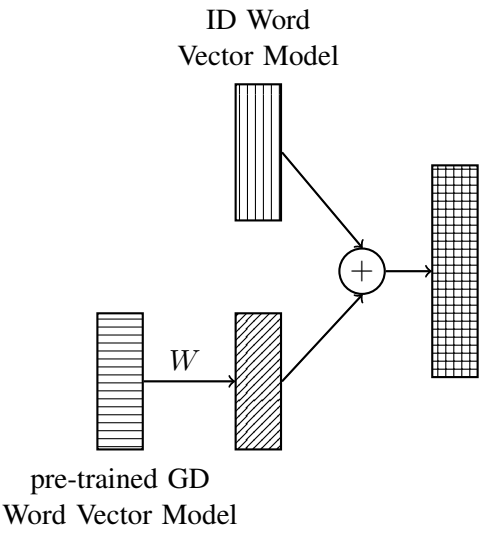
validation and test data set, respectively.

In the adaptation on word vectors experiments, we found that summing up the word vectors (WVS) from ID and GD can outperform the concatenation approach (WVC). By applying a weight control to GD word vectors, the weighted concatenation approach (WWVC) yields a slight improvement, but not in the sum up approach (WWVS).

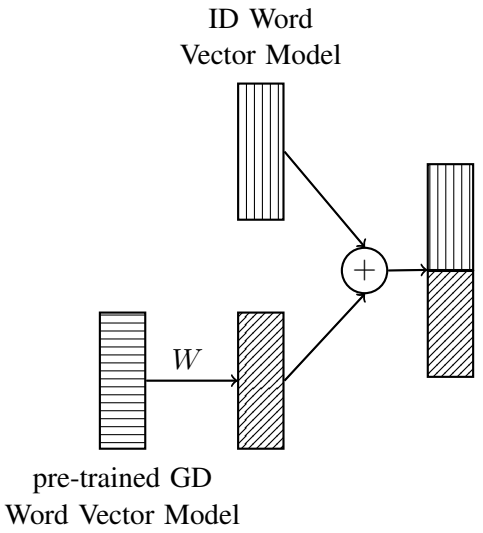
A similar picture can also be seen in the adaptation on the context vectors experiments. Adding up the context vectors (CVS) is more useful than concatenating the context vectors (CVC). In addition, it is better to use the weight vector in the concatenation case. Thus, we can draw the conclusion that information from GD should be compressed (summed) into ID rather than using scattered (concatenated) representations.

However, weighted vectors can be harmful to the sum approaches, e.g. WWVS and WCVS. We think this is because the weight matrix is "*hidden*" behind the sum operation as seen in in Figure 4.7a. Thus the model can be hard to optimize. In contrast, when applying weight matrix in the concatenation cases, the resulted vectors are still separable by domains as seen in in Figure 4.7b. Thus the weights in such adapted models are easier to optimize. However, observing the experimental results, only a small positive impact can be found when applying weights on the concatenation approaches. For example, approximately 1 to 2 perplexity points difference can be found between WVC and WWVC, or CVC and WCVC models. This indicates the approach of using weight matrix for domain adaptation in neural network training is too simple.

In the gated adaptation experiments, the adapted LM from the context vectors can produce a better perplexity result than the one adapts from word vectors, where the GCVA approach can reduce 3.0 perplexity points and the GWVA approach can reduce 1.4 perplexity points compared to the baseline LM. Combining the two gates (GDA) yields the best performed LM model in all proposed adaptation mechanisms. It reduces the perplexity by 7.4 points compared to the baseline RNN LM.




(a) Sum operation. $\oplus$ indicates the sum operation. A new vector is created after the sum operation, which is inseparable in domains.




(b) Concatenation operation. $\oplus$ indicates the concatenation operation. A new vector is concatenated from ID and GD, which is still separable in domains.

Figure 4.7: Sum adaptation vs. concatenation adaptation.

### 4.3.3 Scalability Experiments

To demonstrate the scalability of the GDA adaptation approach, we also train LMs adapting from other freely available word vector models. SENNA (Semantic/syntactic Extraction using a Neural Network Architecture) is the word vector model received after a LM training in Collobert et al. (2011). The training data is obtained from Wikipedia. GloVe (Pennington

| | Validation Set | Test Set |
|---|---|---|
| **Word Embedding Size = 50** | | |
| Baseline | 104.5 | 101.3 |
| SENNA Collobert et al. (2011) | 95.5 | 92.0 |
| GloVe (glove_6b) | 95.3 | **91.3** |
| **Word Embedding Size = 100** | | |
| Baseline | 97.0 | 93.7 |
| GloVe (glove_6b) | 89.5 | **85.3** |
| **Word Embedding Size = 200** | | |
| Baseline | 94.2 | 91.0 |
| GloVe (glove_6b) | 86.6 | **82.8** |
| **Word Embedding Size = 300** | | |
| Baseline | 93.0 | 89.1 |
| GloVe (glove_6b) | 86.4 | 82.6 |
| GloVe (glove_42b) | 87.1 | 82.3 |
| GloVe (glove_840b) | 86.9 | 82.2 |
| Google (*word2vec*) | 86.2 | **81.9** |

Table 4.4: The GDA adaptation on different word vector models. This table presents the experimental results of GDA adaptation using different word vector models on the Penn Treebank corpus. The proposed GDA approach can produce better LM perplexity results in all settings.

et al. 2014) provides several versions of word vector models. The glove_6b model is trained on Wikipedia data and the English Gigaword Fifth Edition corpus;[5] the glove_42b model is trained on the Common Crawl data; and the glove_840b model is trained on the the Common Crawl and additional web data.

Table 4.4 presents the experimental results of GDA adaptation using different word vector models on the Penn Treebank corpus. The word embedding numbers in Table 4.4 indicate the word vector size of the adapting word vector model, e.g. the SENNA model has a word vector size of 50 under Word Embedding Size = 50 setting. For a fair comparison, we also ensure that the baselines in each setting have the same word embedding number as the comparative models. For example, under the Word Embedding Size = 50 setting, we use 100 as the word embedding number for the baseline, and the models adapted from SENNA or GloVe (glove_6b) use 50 as word embedding number for both ID and GD. In Table 4.4, the GDA approach can produce better perplexity results in all settings.

[5] `https://catalog.ldc.upenn.edu/LDC2011T07`

| System | Validation Set | Test Set |
|---|---|---|
| Baseline LM | 94.3 | 109.4 |
| GDA LM (*word2vec*) | 85.6 | **97.4** |

Table 4.5: Language model perplexity on News corpus. Using GDA adaptation approach, we can decrease 12 perplexities on the baseline LM.

### 4.3.4 RNNLM Adaptation on News Corpus

Given the results of our prior experiments, we focused on the GDA adaptation approach in our second set of experiments, where we firstly explore the GDA method on the much larger News corpus (Table 4.2).

We first report in Table 4.5 the perplexity results of the baseline LM and the GDA LM (adapting *word2vec*) on the News corpus. On the large training corpus, the GDA LM still can yield a better perplexity than the baseline LM which has no adaptation. The difference between the two LMs is 12 perplexity points. Such a result suggests that the performance of the GDA approach is also scalable to a large data set.

Our second experiment investigates whether the GDA method can be helpful on the lower-frequency words. Recall that one of the data prepossessing step in a neural LM training is to map lower-frequency words into a special UNK token. Two reasons can be cited. First, reducing the vocabulary size can be more efficient in terms of training time. The *softmax* function (as seen in Equation (2.17)) needs to output the probability distribution to all words. Using a large vocabulary size means more computation is required in a neural LM training. The second reason is that word embeddings are not accurate for the lower-frequency words. Qu et al. (2015) demonstrated that models performance can be increased by using more training data (increasing the vocabulary frequency) in several sequence labelling tasks. Thirdly, less words in the vocabulary can make model easier to learn. The task of training a neural LM model can be thought as a multi-class classification problem where the predicting labels are the words in the training data and the features are the previous words in a sequence. However, the model can only learn to predict the lower-frequency words much fewer times than the high frequency words during the training phase, which will result us a model can not perform well on those less seen words.

| System | Validation Set | Test Set |
|---|---|---|
| Baseline LM | 207.1 | 266.0 |
| GDA LM (*word2vec*) | 190.1 | **244.8** |

Table 4.6: Language model perplexity on News corpus with all words in the vocabulary are used. Using GDA adaptation approach, we can reduce 21.2 perplexities on the baseline LM. The perplexity decreasing shows that the lower-frequency words can benefit from the proposed GDA approach.

| System | Tuning | Test |
|---|---|---|
| Baseline | 26.18 | 27.14 |
| Baseline + Baseline LM Re-ranking | 26.71 | 27.65 |
| Baseline + GDA LM Re-ranking | 27.17 | **27.96** † |

Table 4.7: BLEU scores for the re-ranking task.

In our proposed GDA approach, we adapt pre-trained word vector models to the task-specific word vector models. Therefore, our approach should be beneficial to the lower-frequency words in the training corpus regarding learning better word vectors. In this experiment, the LMs are trained without mapping lower-frequency words, i.e. we use all words (72,828, as seen in Table 4.2) in the News corpus in training. We set the maximum number of training iterations to 20 and the learning rate starts to decay after 3 iterations. Others parameters are the same as we described in Section 4.3.1. Table 4.6 shows the perplexity results of the baseline LM and the GDA LM. Using GDA adaptation approach, we can decrease 21.2 perplexities on the baseline LM. The perplexity decreasing shows that the lower-frequency words can benefit from the proposed GDA approach.

Our third experiment on the News corpus is to apply the adapted LM to the SMT re-ranking task. To train the Phrase-based Statistical Machine Translation (PBSMT) system, we use the French-to-English News Commentary v10 and Europarl v7 corpus from the WMT2015 translation task. The newstest 2013 and 2014 data sets are used for tuning and testing for the translation system, respectively. The system is trained using the Moses (Koehn et al. 2007) MT framework, with a reordering model (Koehn et al. 2005) and a 5-gram KenLM (Heafield 2011) language model. We use the default parameters in Moses in all experiments.

We then use the LMs in Table 4.5 to re-rank PBSMT systems. Table 4.7 reports the translation evaluation scores in the re-ranking task. Each score is the average score over

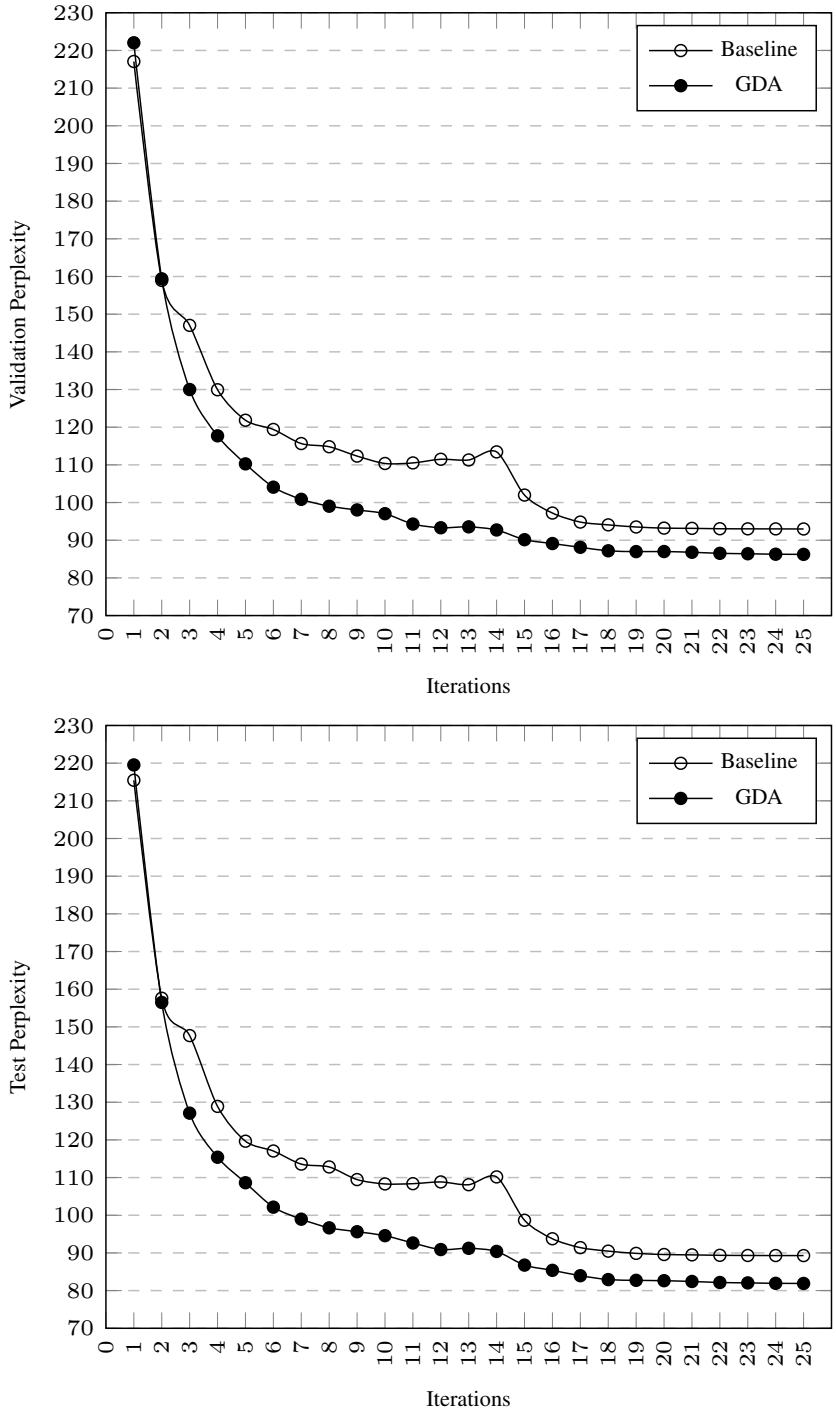



Figure 4.8: Learning curves of the baseline LM and the GDA LM (adapting *word2vec*) on the Penn Treebank corpus.

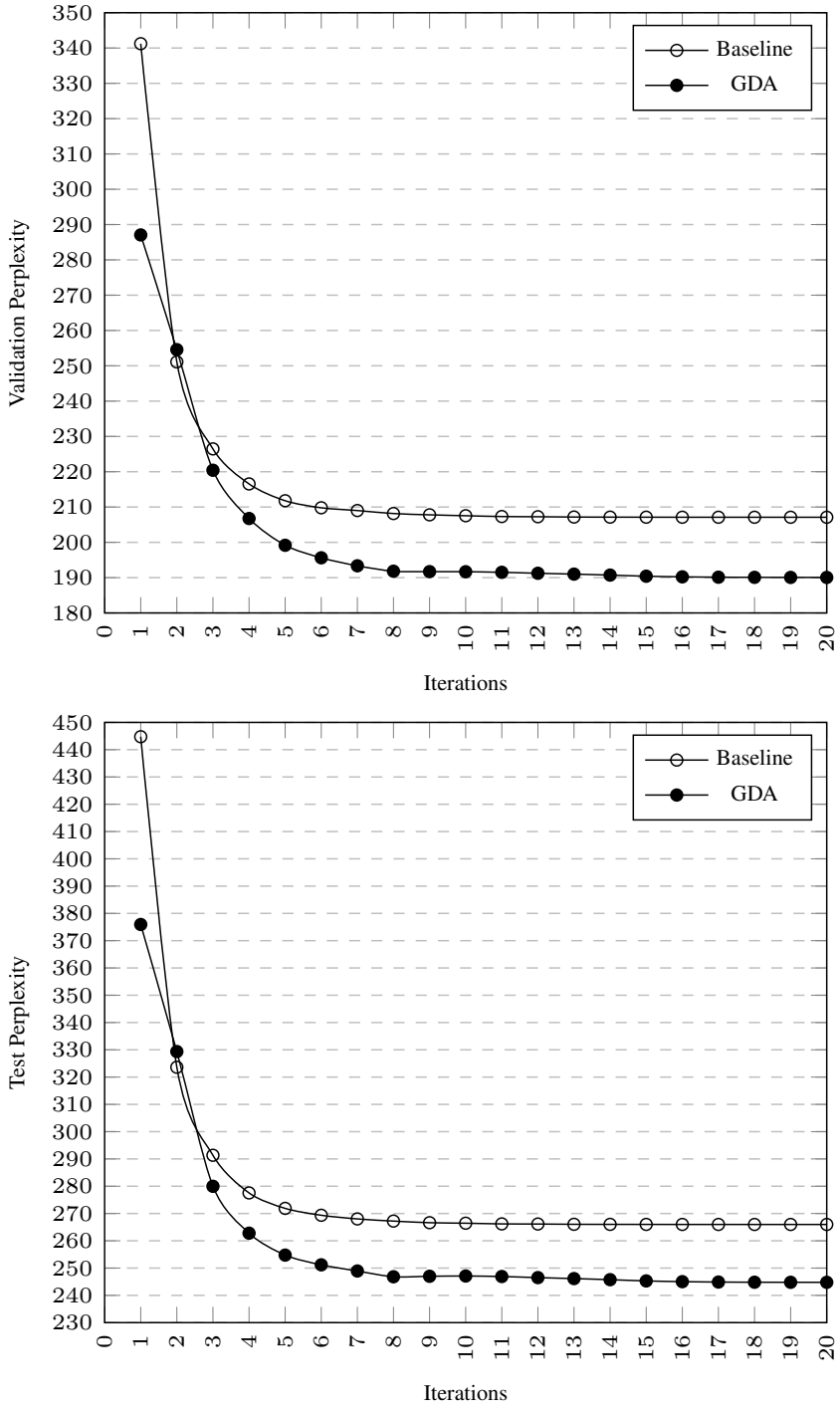



Figure 4.9: Learning curves of the baseline LM and the GDA LM (adapting *word2vec*) on the News corpus with all words in the vocabulary are used.

three runs. On the testing data, the baseline PBSMT system achieves a BLEU score of 27.14. The re-ranked SMT systems using the LM without adaptation gains a 0.51 absolute (1.8% relative) improvement in Bilingual Evaluation Understudy (BLEU) score. Using the adapted LM for re-ranking, we can observe a 0.82 absolute (3% relative) improvement in BLEU score. Comparing the two re-ranked systems, there is a 0.31 absolute (1.1% relative) improvement in BLEU. We use † to indicate statistically significant (Koehn 2004) over the SMT system re-ranked by the baseline LM. The significance testing uses bootstrapping method at the level p = 0.05 level with 1,000 iterations.

### 4.3.5 Language Model Learning Curves

We can also observe the improvement of GDA over baseline model is obtained in the early training iterations. We compare the learning curves between the baseline LM and the GDA LM on validation and test data of the Penn Treebank. As Figure 4.8 shows, the predictions become more certain and accurate after iterations for training both LMs. Already after training iteration 2, the GDA LM starts to outperform the baseline LM in terms of perplexity at every iteration. The plots flatten after 20 iterations, and the learning begins to converge for both the baseline LM and GDA LM. The sharp perplexity decreasing after iteration 14 is the effects of applying learning rate decay. Figure 4.9 demonstrates the learning curves between the baseline LM and the GDA LM on validation and test data of the New corpus when all words in the vocabulary are used in training. The similar conclusion can also be drawn in this plot that the GDA LM starts to outperform the baseline LM in terms of perplexity at every iteration.

Recall that our hypothesis for this work is that the pre-trained and the task-specific-trained word vector models are complementary with each other in a neural network training. In this set experiments, we showed that the proposed approach is an effective method to combine the pre-trained and the task-specific-trained word vector models.

## 4.4 NMT Experiments

### 4.4.1 Experimental Setup

Our proposed GDA mechanism is not only limited to neural LM training. We are interested in exploring its performance on other NLP tasks, especially NMT. Fundamentally, both neural LM and NMT are sequence-to-sequence prediction problems, so our GDA is applicable to both tasks. For example, in NMT, we can perform adaptation on the target context vector $h_{j-1}$ in Equation (2.22) as we used in the adapted LMs.

In this experiment, we report our results on the National Institute of Standards and Technology (NIST) Open Machine Translation evaluation data set in the Chinese-to-English translation direction. Our MT training data are extracted from Linguistic Data Consortium corpora,[6] and NIST 2002 is used as our development set. We use the NIST 2004 and 2005 data as our test sets. The English training data are tokenized and lowercased using scripts in Moses (Koehn et al. 2007) machine translation framework. The Stanford Chinese word segmenter (Tseng et al. 2005) is used to segment the Chinese training data. In NMT, we limit our vocabularies to the 16k most frequent words (Meng et al. 2015), which covers 97.57% and 98.77% of the original words in the source and target training corpora, respectively. Outside the 16k threshold, vocabularies are mapped to the UNK token.

We compare the adapted NMT system with two baselines. The PBSMT baseline is trained using Moses (Koehn et al. 2007), with a reordering model (Koehn et al. 2005) and a 5-gram KenLM (Heafield 2011) LM. We use the default parameters in Moses in all experiments. Our second baseline is an NMT system without adaptation. The same architecture described in Figure 2.11 is used. The word vectors are of size 600 for both source and target words. The hidden layers are of size 1,024. We use beam search during translating, with a beam size of 5. We use a mini batch (with batch size 32) SGD algorithm together with Adadelta (Zeiler 2012) to train our models. All NMT models are trained up to 320,000 updates, and the models are saved at each 1,000 updates. We then choose the final

[6] LDC2002E18, LDC2003E07, LDC2003E14, LDC2004T07, the Hansards portion of LDC2004T08 and LDC2005T06

| System | NIST 2002 | NIST 2004 | NIST 2005 |
|---|---|---|---|
| PBSMT | 33.42 | 32.36 | 30.11 |
| NMT | 34.51 | 35.02 | 31.46 |
| GDA NMT (glove_840b) | 36.07 | **35.99**‡ | 31.73 |
| GDA NMT (*word2vec*) | 35.63 | 35.84‡ | **31.88**‡ |

Table 4.8: BLEU scores for NMT adaptation. We use ‡ to indicate statistically significant (Koehn 2004) improvements upon the NMT baseline model. The significance testing uses bootstrapping method at the p = 0.01 level with 1,000 iterations.

model based on the BLEU score of the development data. The adapted NMT system uses the same settings as our second baseline, except that the target word vector is adapted from the GD word vector model using GDA. We demonstrate the performance of GDA using the two large pre-trained word vector models, glove_840b and *word2vec*.

### 4.4.2 Results

Table 4.8 presents the experiment results when GDA is applied to NMT. We use ‡ to indicate statistically significant (Koehn 2004) improvements over the NMT baseline model. The significance testing uses bootstrapping method at the level p = 0.01 level with 1,000 iterations. PBSMT and NMT achieve 33.42 and 34.51 BLEU scores, respectively, on the NIST 2002 development data, whereas GDA NMT obtains 36.07 and 35.63 BLEU scores, respectively, on the NIST 2002 development data. If we look at the results on the test set, there is no surprise that the baseline NMT can bring 2.66 (absolute, 8.2% relative) and 1.35 (absolute, 4.5% relative) improvements compared to the PBSMT system. When employing GDA in NMT training, we can gain 0.97 (absolute, 2.7% relative) and 0.82 (absolute, 2.3% relative) improvements on the NIST 2004 test data using glove_840b and *word2vec* vector models, respectively. We also can observe BLEU score increases on the NIST 2015 test data, where 0.27 (absolute, 0.9% relative) and 0.42 (absolute, 1.3% relative) increase are gained with glove_840b and *word2vec* vector models, respectively. This set of experiments suggests that the proposed GDA approach can be applied not only to the LM task, but also to the NMT task. Comparing the adaptations from glove_840b and *word2vec*, the two systems provide comparable results, where no significant differences can be observed.

## 4.5 Advantages

As a domain-adaptation approach for neural network applications, the GDA approach has many advantages.

Firstly, GDA is very fast. There is very little speed overhead in the neural network training framework. The reason is that we adapt from a pre-trained word vector model instead of using additional training data. For example, data selection approaches try to select the raw sentences from GD into ID which results the training data are increased in size. However, we think such adaptation approaches are inefficient if a neural network training is used. The additional data we selected using the standard data selection approach can increase both training time and memory consumption in training. To find the optimized selection portion from GD data is also an extremely time-consuming task since many models need to be trained and evaluated. Our approach, however, makes the use of the internal data representation in neural network training. There will be no extra training time brought by the additional training corpus. Furthermore, the process of handling the GD training data is also simplified. In our approach, the GD is given as the pre-trained word vector models. The adapted RNNLM model is still trained using ID training data. The data processing step for a GD training data is not required.

Secondly, our approach can also be beneficial to the lower-frequency words. In the neural network training, we need to group the lower-frequency words into the UNK token for time and performance efficiency reasons. The GDA approach can use the pre-trained word models to result better word vectors for the lower-frequency words in the ID during the training phase.

Thirdly, the GDA approach can be used in any sequential network applications, without the requirement of additional task-oriented data. For example, we do not need additional parallel corpus in MT tasks. The adaptation process only requires a pre-trained word vector model, which can be obtained easily.

## 4.6 Summary

This study makes several contributions to the field. First, we propose to adapt pre-trained word vector models in domain adaptation which makes our work different to previously proposed domain adaptation work. Secondly, we present several adaptation approaches and make comparisons between them. In our experiments, the GDA mechanism based on GRU outperforms the others. We also provide explanations as to why it is so efficient. Thirdly, the GDA mechanism can be used in sequential neural network applications. We adapt the GDA mechanism into the state-of-the-art NMT systems. We observe that the translation quality can be significantly improved. Finally, it is very fast. This approach does not require additional task-oriented data. For example, we only need the pre-trained word vector model from GD.

In this chapter, we described an efficient domain-adaptation approach for **RQ2** in the *domain-awareness* scenario. Our focus was on the most recent neural network training for LM and NMT. We gave detailed definitions for the proposed methods and compared those approaches using different settings. In particular, we demonstrated the efficiency of GDA on both small and large training data using neural LM as a case study. We also examined this technique on various word vector models. When using the trained LM on the SMT re-ranking task, we observed a significant improvement in translation quality in the re-ranking task. We then employed the GDA method on NMT and demonstrated that it can also be successfully used in other sequence-to-sequence prediction tasks.

However, sometimes we do not have a clear distinction between the ID and GD data, but all data are mixed in domains. Word meanings, vocabulary coverage or writing styles can still be different within the training corpus. We call such case as *domain-unawareness* scenario. In fact, such cases are one of the most common situations we encounter in MT. In the next chapter, we switch our attention to the *domain-unawareness* scenario and discuss our last research question:

**RQ3** *How word topic distributions can be used to improve translation qual-*

*ity for NMT models in a domain-unawareness scenario?*

# Chapter 5

# Topic-based Domain Adaptation for Neural Machine Translation

## 5.1 Introduction

In the previous two chapters, we presented our work for **RQ1** and **RQ2** in a *domain-awareness* scenario. The assumption we made in that scenario is that training data of a domain is homogeneous. However, this is not always true in practice (Hasler 2015). The training data may come from tens or even hundreds of different resources without well-defined domain labels. It is possible that some of the data are from a small or an extensive domain, and some can be very close to the desired domain and most not, or the training data of the desired domain might even not be unavailable. We have to use whatever data available, e.g. to concatenate all training data into a large corpus to build the a model. Therefore, the domain information is not given explicitly in the training data. We call this scenario a *domain-unawareness* scenario. In this chapter, we extend the domain-adaptation challenge into a *domain-unawareness* scenario and address our last research question:

> **RQ3** *How word topic distributions can be used to improve translation quality for Neural Machine Translation (NMT) models in a domain-unawareness scenario?*

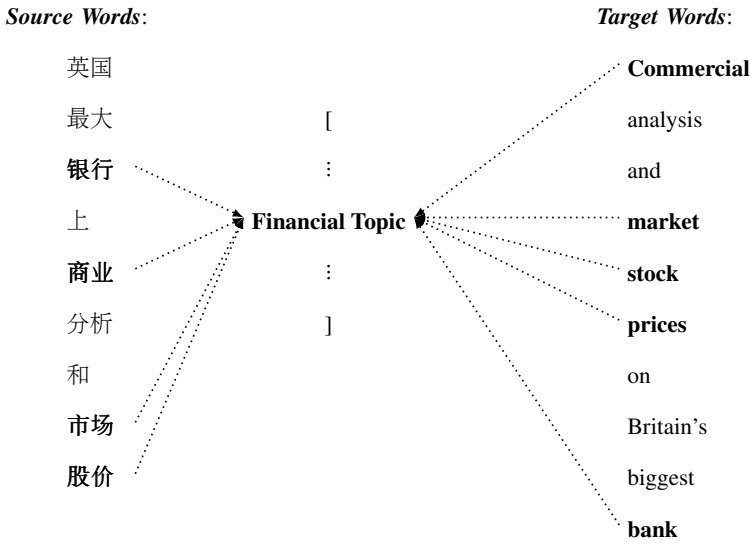



Figure 5.1: This example shows that some words within the same sentence belong to the same (or similar) topic. For example, source words "银行", "商业", "市场" and "股价", and target words "Commercial", "market", "stock", "prices" and "bank" have higher probabilities of being in the *Financial Topic*.

Additionally, we pay particular attention to the start-of-the-art NMT system (Bahdanau et al. 2015).

In a *domain-unawareness* scenario, we confront two challenges: (i) how to discovery domain information without explicit domain labels in training data, and (ii) how can we use the discovered domain information in an NMT system.

Regarding to the first challenge in this scenario, we follow previous topic-based domain-adaptation studies as discussed in Section 2.3.2. to use existing well-established topic modelling tools, e.g. Latent Dirichlet Allocation (LDA) (Blei et al. 2003) or Hidden Topic Markov Model (HTMM) (Gruber et al. 2007), to discover the topic information. We regard topics as domains in this scenario.

Regarding to the second challenge, one observation we obtain from the training data is that words in the sentences often belong to the same (or similar) topic regardless of domain. As seen in Figure 5.1, source words "银行", "商业", "市场" and "股价", and target words "Commercial", "market", "stock", "prices" and "bank" have higher probabilities of being in the *Financial* topic. The similar "topic consistent" behaviour is also observed by Su

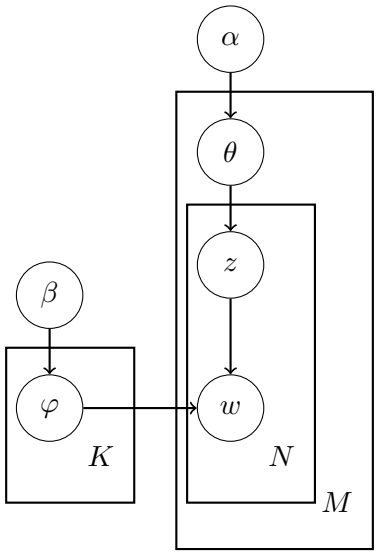



Figure 5.2: The graphical model for smoothed LDA in plate notation (Blei et al. 2003).

et al. (2015). Recall that in NMT, the attention model is used to provide the ability that a decoder can selectively pay attention to different parts of a source sentence to translate. Therefore, intuitively, if we can leverage the source topic information to the attention model, the decoder can also pay attention to the different parts of a source sentence with respect to their topics. Furthermore, as previously translated words can be a strong indication to influence the current translation in an NMT model, that is if we can provide the (target) topic information of previously translated words to an NMT model, then the current translation of being the similar topic as the previous ones should receive higher chances to be selected. In contrast, words which are not favouring to any topics in the target vocabulary, e.g. the Unknown (UNK) and the end-of-sentence (EOS) token, should have less chances to be selected.

## 5.2 Topic Models

LDA (Blei et al. 2003) is a statistical model that tries to discover the hidden topic structures in large document collections. It maps words in the documents to probability distributions over topics. Figure 5.2 shows the graphical model of the LDA generative model in plate no-

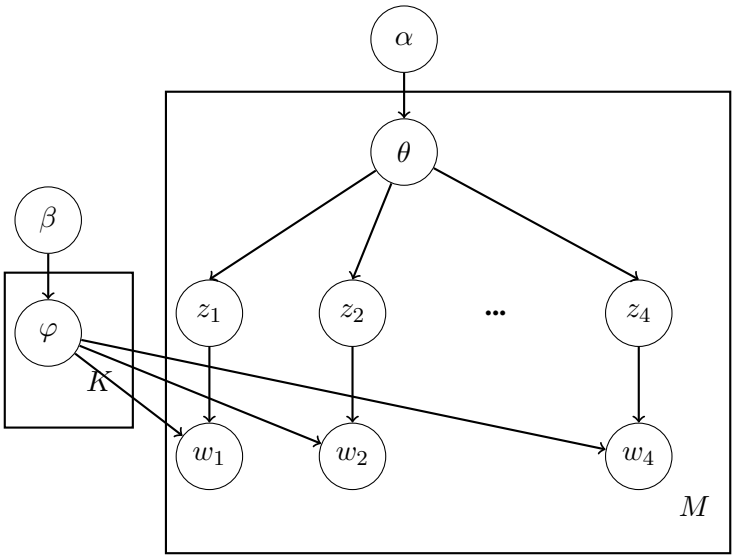



Figure 5.3: The graphical model for smoothed LDA with repeated word generation within a document (Gruber et al. 2007).

tation. The generative process of the LDA model for a corpus consisting of $M$ documents, where each document has the length of $N_i$, $i \in \{1, .., M\}$, can be described as follows:

1. Choose $\theta_i \sim Dirichlet(\alpha)$, where $Dirichlet(\alpha)$ is Dirichlet distribution with parameter $\alpha$

2. Choose $\varphi_k \sim Dirichlet(\beta)$, where $k \in \{1, .., K\}$, $K$ is the number of latent topics and $Dirichlet(\beta)$ is Dirichlet distribution with parameter $\beta$

3. For each of the word $w_{i,j}$, where $j \in \{1, .., N_i\}$ and $z$ is the topic assignment for each word:

   (a) Choose a topic $z_{i,j} \sim Multinomial(\theta_i)$

   (b) Choose a word $w_{i,j} \sim Multinomial(\varphi_{z_{i,j}})$

where each $\theta \in \{\theta_1, ...\theta_M\}$ is a multinomial distribution over topics given a document.

Figure 5.3 shows the repeated word generation within a document of Figure 5.2. As we can see, the word topics in LDA are assumed to be independent, i.e. $z_1, .., z_4$ (word topics) can be the same or different within a document. Gruber et al. (2007) claim that such

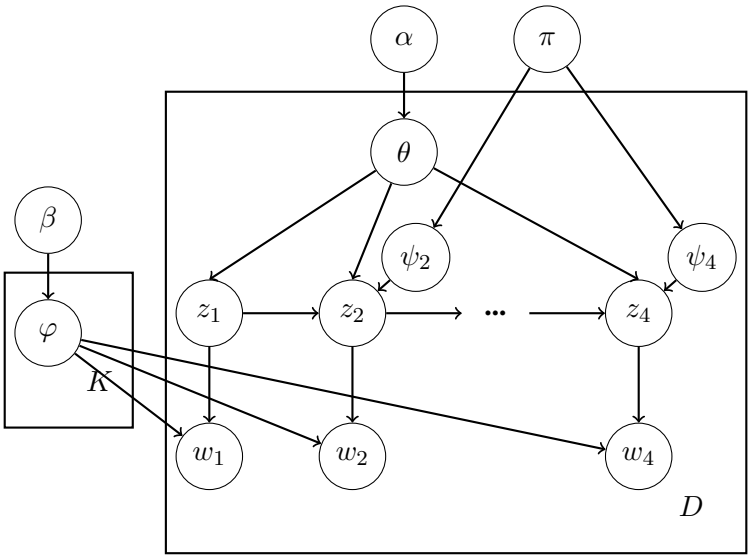



Figure 5.4: The graphical model for HTMM (Gruber et al. 2007).

an assumption is "an unrealistic oversimplification" and propose to use Hidden Markov Models to capture local dependencies between words when learning the topics. Such a topic detection algorithm is called HTMM, as seen in Figure 5.4. The transition probability depends on $\theta$ and a topic transition variable $\psi_N$, $\pi$ is the initial state of the Markov chain. When $\psi_N = 1$, we choose a new topic from $\theta$, when $\psi_N = 0$, the topic of the current word is the same as the previous one. Gruber et al. (2007) assume that "topic transitions can only occur between sentences", which means $\psi_N$ can only be nonzero for the first word in a sentence. In HTMM, if $\psi_N$ is forced to be 1 for all words, it becomes to be LDA since we always choose a new topic; if $\psi_N$ is set to be 0, all words in a document have the same topic.

Topic models are trained using documents, it is also common that sentences are inferred to represent mixtures of topics in Machine Translation (MT) (Eidelman et al. 2012, Xiong et al. 2015). For example, we can train two topic models given a bilingual data: a source topic model and a target topic model. In this work, we are interested in both LDA and HTMM algorithms.

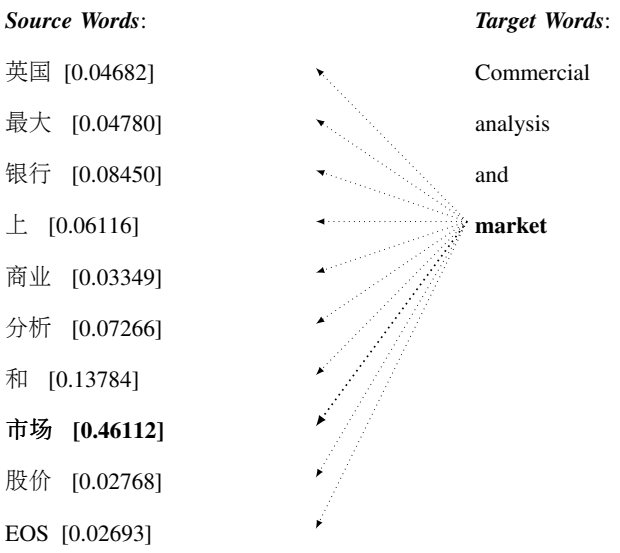



Figure 5.5: An example of "*soft*" alignment in NMT. The current translated word is "market", which is aligned to source word "市场" with a higher probability than the other source words, i.e. the probability is 0.46112.

## 5.3 Our Approach

In this section, we provide details on our proposed models. More specifically, given the topic distributions of source and target words, either using LDA or HTMM, we suggest three different approaches to incorporate the topic distributions into the NMT models.

### 5.3.1 Topic-based Encoder

In the original NMT model we presented in Chapter 2, the attention model (Bahdanau et al. 2015) can selectively pay attention on different parts of the source context that are relevant to the predicting target words. As a by-product of the attention model, a "*soft*" alignment between the source input and translation is also generated. For example in Figure 5.5, the current translated word is "market", the attention model is paying more attention to the source word "市场". In the "*soft*" alignment output, target word "market" is aligned to source word "市场" with a higher probability than the other source words.

Our intuition of the topic-based encoder is, if we can embed the topic distributions into the source context vector, i.e. $h_i$ in Equation (2.23), the decoder therefore can also pay

attention to the different parts of a source sentence with respect to their topics, the "*soft*" alignment can then be more accurate on the source context, and translation performance can be increased.

Therefore, we propose to argument the source context vectors with topic information $topic\_h_i^f$ as in Equation (5.1):

$$topic\_h_i^f = [h_i^f, \beta_i^f] \tag{5.1}$$

where $\beta_i^f$ is the topic distribution for source word $f_i$, which can be obtained from the topic model pre-trained with the source side of the NMT training data.[1] $[h_i^f, \beta_i^f]$ denotes concatenation operation on $h_i^f$ and $\beta_i^f$. For example, in our $topic\_h_i^f$ vector, the last $t$ elements represent the topic distributions where $t$ is the number of topics configured in topic modelling. Furthermore, we need to update Equation (2.23), (2.24) and (2.25) to Equation (5.2), (5.3) and (5.4), respectively, such as:

$$topic\_e_{ij} = v^T a(h_{j-1}^e, topic\_h_i^f) \tag{5.2}$$

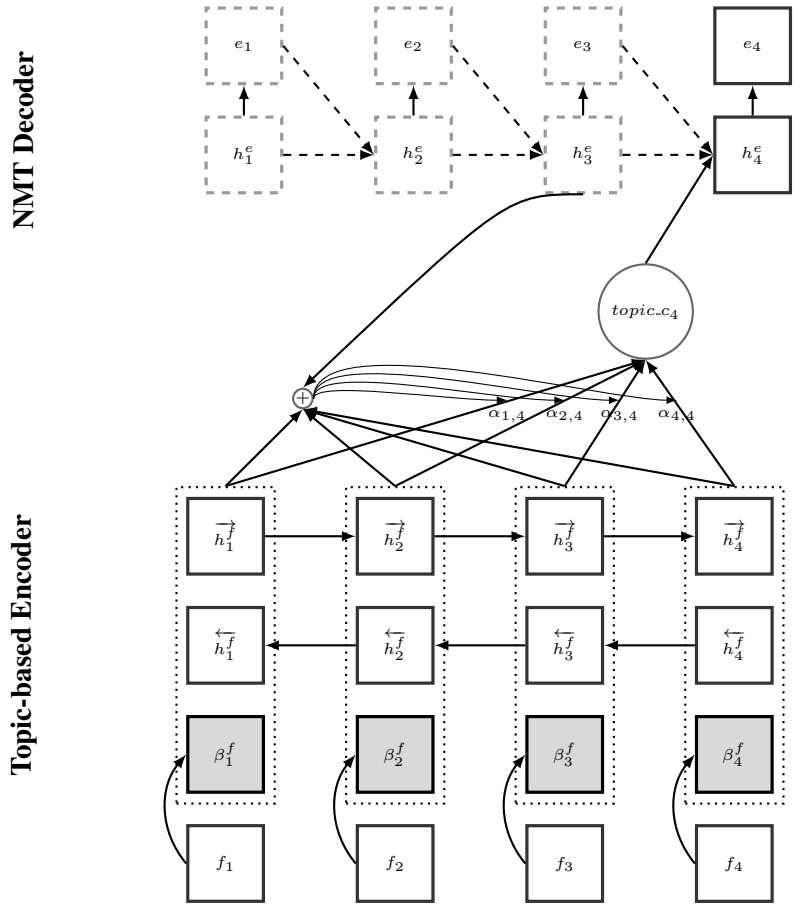
$$topic\_c_j = \sum_{i=1}^{m} topic\_\alpha_{ij}\ topic\_h_i^f \tag{5.3}$$

$$topic\_\alpha_{ij} = \frac{exp(topic\_e_{ij})}{\sum_{i=1}^{m} exp(topic\_e_{ij})} \tag{5.4}$$

in order to incorporate source topic information in the attention model.[2] Therefore, the decoder can pay attention to different parts of a source sentence with respect to the source topic distributions.

We then update Equation (2.26) and (2.27) by substituting $c_j$ with $topic\_c_j$ to obtain

[1] We use $f$ to indicate a context vector in encoder.
[2] We use $e$ to indicate a context vector in decoder.




***Notations:***

$f_1$,...,$f_4$: Source words
$h_1^f$,...,$h_4^f$: Source context vectors
$\beta_1^f$,...,$\beta_1^f$: Source word topic distributions
$topic\_c_4$: Topic-based source context vector
$\alpha_{1,4}$,...,$\alpha_{4,4}$: Soft-alignments
$e_1$,...,$e_4$: Target words
$h_1^e$,..., $h_4^e$: Target context vectors
$\oplus$: Attention model

Figure 5.6: Graphical illustration of the topic-based encoder with 4 source words and 4 target words. In this architecture, target word $e_4$ is conditioned on the target context vector $h_4^e$, which is computed by the topic-based source context vector $topic\_c_4$, the previous target context vector $h_3^e$ and the previous target word $e_3$. Dashed nodes indicate translation histories.

our topic-based encoder, as in Equation (5.5) and (5.6), respectively:

$$h_j^e = g(t_{j-1}, h_{j-1}^e, topic\_c_j) \tag{5.5}$$

$$p(e|e_{1:j-1}, F) = softmax(S(t_{j-1}, h_j^e, topic\_c_j)) \tag{5.6}$$

Figure 5.6 is the graphical illustration of the proposed topic-based encoder with 4 source words and 4 target words. In this architecture, target word $e_4$ is conditioned on target context vector $h_4^e$, which is computed by the topic-based source context vector $topic\_c_4$, the previous target context vector $h_3^e$ and the previous target word $e_3$.

### 5.3.2 Topic-based Decoder

In the proposed topic-based encoder, our aim is to increase the performance of the attention model, which can consequently improve the translation quality. While in the target topic-based decoder, we try to maintain the topic consistency between the translating word with the previously translated words. Thus, translation options can be selected from the same domain.

Words in the NMT outputs are translated in the sequence from left to right. When a word is translating, we use three pieces of information: the source context, the target context of translated words and previously translated word, as seen in Equation (2.26). A natural choice of introducing topic information to the decoder is to also include the previous topic histories as an additional information for the proceeding word predictions, such as the current translation is also conditioned on the topics of previous translations, as in Equation (5.7):

$$h_j^e = g(t_{j-1}, h_{j-1}^e, c_j, H^\beta) \tag{5.7}$$

where $H^\beta$ denotes the target topic information. We propose to model $H^\beta$ using an Recurrent Neural Network (RNN) network, e.g. Gated Recurrent Unit (GRU) (Chung et al.

2014). The motivation is that previously translated words can show strong indications and influences to the current translation in NMT, if we can provide the (target) topic information of previously translated words to the NMT model, then the current translation of being the similar topic as the previous ones should receive higher chances to be selected.

As seen in Figure 5.5, word "市场" is translated into word "market". However, it can also be translated into words "mart" or "bazaar", which are the wrong translations in this example. We can provide previous topic distributions as a hint to the current word choice, e.g. informing the decoder that previous translations are in the *Financial* topic, the model can therefore give higher chances to the translations which are also in the same topic. Accordingly, we compute the previous topic histories as a sequence with the dependency relationships using GRU, as in Equation (5.8):

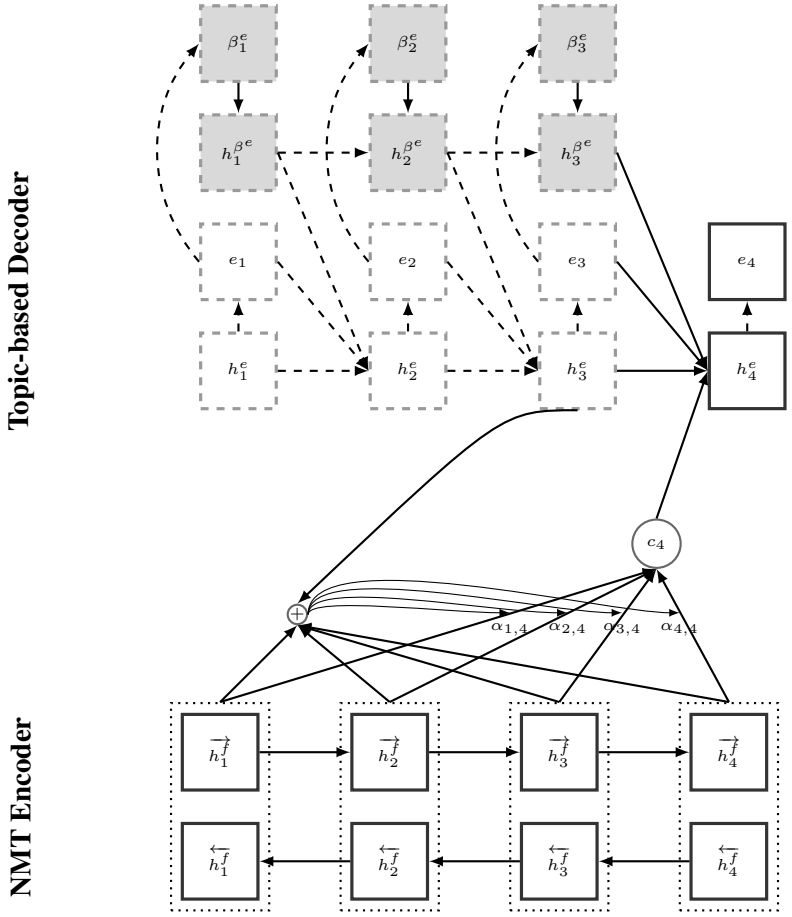
$$\begin{aligned} u_t &= sigmoid(W_u \beta_t^e + U_u h_{t-1}^{\beta^e} + b_u) \\ r_t &= sigmoid(W_r \beta_t^e + U_r h_{t-1}^{\beta^e} + b_r) \\ \tilde{h}_t^{\beta^e} &= tanh(W \beta_t^e + U(r_t \odot h_{t-1}^{\beta^e}) + b) \\ h_t^{\beta^e} &= (1 - u_t) \odot h_{t-1}^{\beta^e} + u_t \odot \tilde{h}_t^{\beta^e} \end{aligned} \tag{5.8}$$

where $\beta_t^e$ is the topic distribution for target word at time $t$, which is obtained from the topic model pre-trained on the target side of the NMT training data. $h_{t-1}^{\beta^e}$ is the context vector of this topic RNN, $\tilde{h}_t^{\beta^e}$ is the candidate activation (Chung et al. 2014) and $h_t^{\beta^e}$ is the context vector of this network. $W_u$, $U_u$, $W_r$, $U_r$, $W$ and $U$ are the weight parameters, and $b_u$, $b_r$ and $b$ are the bias values of the corresponding gates. Thus, we can substitute $H^{\beta}$ in Equation (5.7) to the corresponding topic context vector in Equation (5.8), as in Equation (5.9):

$$h_j^e = g(t_{j-1}, h_{j-1}^e, c_j, h_{t-1}^{\beta^e}) \tag{5.9}$$

to obtain the topic-based decoder.

Figure 5.7 is the graphical illustration of the proposed topic-based decoder with 4 source words and 4 target words. In this architecture, target word $e_4$ is conditioned on target




***Notations:***

| | |
|---|---|
| $h_1^f$ ,..., $h_4^f$: Source context vectors | $e_1$ ,..., $e_4$: Target words |
| $c_4$: Source context vector | $h_1^e$ ,..., $h_4^e$: Target context vectors |
| $\alpha_{1,4}$,...,$\alpha_{4,4}$: Soft-alignments | $\beta_1^e$ ,..., $\beta_1^e$: Target word topic distributions |
| $\oplus$: Attention model | $h_1^{\beta^e}$ ,..., $h_4^{\beta^e}$: Target topic context vectors |

Figure 5.7: This figure is the graphical illustration of the topic-based decoder with 4 source words and 4 target words. In this architecture, target word $e_4$ is conditioned on target context vector $h_4^e$, which is computed by source context vector $c_4$, target context vector $h_3^e$, previous target word $e_3$ and context vector of target topics $h_3^{\beta^e}$. Dashed nodes indicate translation histories.

context vector $h_4^e$, which is computed by the source context vector $c_4$, the target context vector $h_3^e$, the previous target word $e_3$ and the context vector of target topics $h_3^{\beta^e}$.

### 5.3.3 Topic-based NMT

We can also combine the topic-based encoder in Equation (5.5) and the topic-based decoder in Equation (5.9) to obtain the topic-based NMT, as in Equation (5.10):

$$h_j^e = g(t_{j-1}, h_{j-1}^e, topic\_c_j, h_{t-1}^{\beta^e}) \tag{5.10}$$

in which we use the previous translation $t_{j-1}$, the context vector of previous translations $h_{j-1}^e$, the topic-based source context vector $topic\_c_j$ and the target topic context vectors $h_{t-1}^{\beta^e}$ to compute a context vector $h_j^e$ for the current translating word. Later, a $softmax$ function can be applied to $h_j^e$ to output the probability distributions of the target vocabulary as in Equation (2.20).

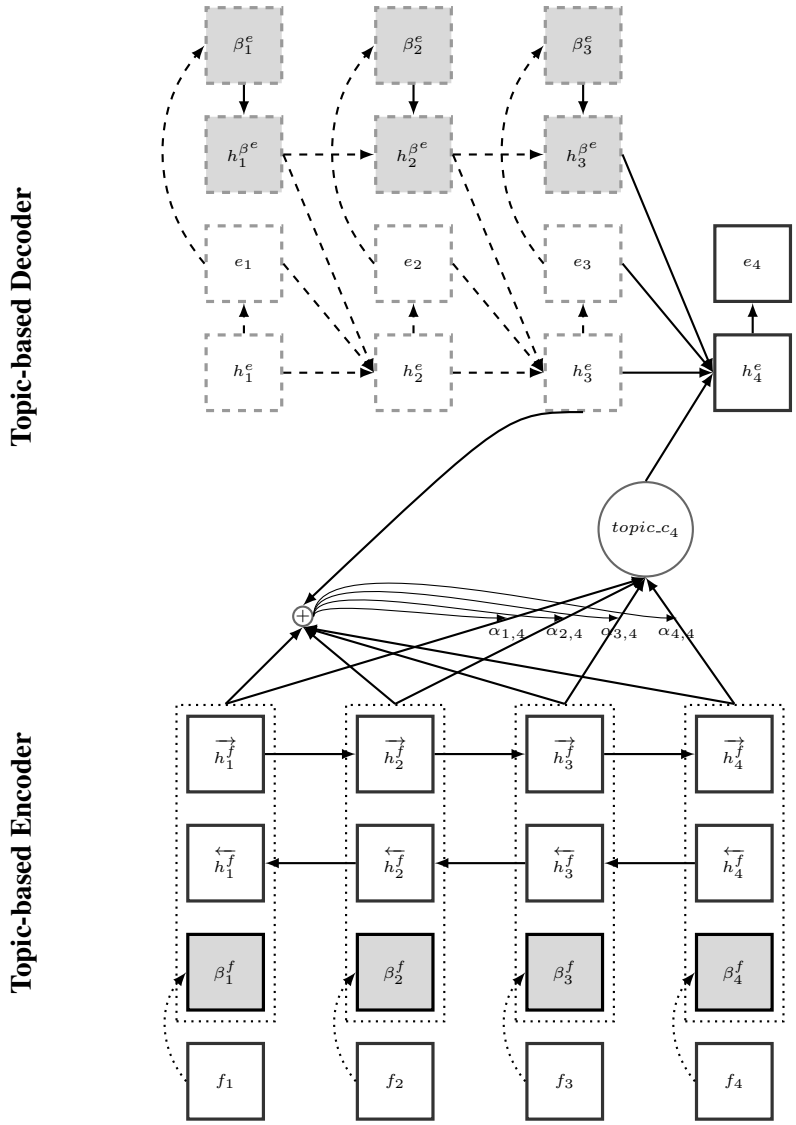
Figure 5.8 is the graphical illustration of the topic-based NMT with 4 source words and 4 target words. In this architecture, target word $e_4$ is conditioned on target context vector $h_4^e$, which is computed by topic-based source context vector $topic\_c_4$, target context vector $h_3^e$, previous target word $e_3$ and context vector of target topics $h_3^{\beta^e}$.

## 5.4 Experiments

### 5.4.1 Data and Experiment Models

We report our experimental results on the National Institute of Standards and Technology (NIST) evaluation data set in the Chinese-to-English translation direction. Our MT training data are extracted from Linguistic Data Consortium corpora, and NIST 2002 is used as our development set.[3] We use the NIST 2004 and 2005 as our test sets. The English training data are tokenized and lowercased using scripts in Moses (Koehn et al. 2007) MT

[3]LDC2002E18, LDC2003E07, LDC2003E14, LDC2004T07, the Hansards portion of LDC2004T08 and LDC2005T06




***Notations:***

| | |
|---|---|
| $f_1$,...,$f_4$: Source words | $e_1$,...,$e_4$: Target words |
| $h_1^f$,...,$h_4^f$: Source context vectors | $h_1^e$,..., $h_4^e$: Target context vectors |
| $\beta_1^f$,...,$\beta_1^f$: Source word topic distributions | $\beta_1^e$,...,$\beta_1^e$: Target word topic distributions |
| $topic\_c_4$: Topic-based source context vector | $h_1^{\beta^e}$,...,$h_4^{\beta^e}$: Target topic context vectors |
| $\alpha_{1,4}$,...,$\alpha_{4,4}$: Soft-alignments | $\oplus$: Attention model |

Figure 5.8: This figure is the graphical illustration of the topic-based NMT with 4 source words and 4 target words. In this architecture, target word $e_4$ is conditioned on target context vector $h_4^e$, which is computed by topic-based source context vector $topic\_c_4$, target context vector $h_3^e$, previous target word $e_3$ and target topic context vector $h_3^{\beta^e}$. Dashed nodes indicate translation histories.

framework. The Stanford Chinese word segmenter (Tseng et al. 2005) is used to segment the Chinese training data. In NMT, we limit our vocabularies to be the top 16,000 most frequent words, which covers 97.57% and 98.77% of the original words in the source and target training data, respectively. Outside the 16,000, vocabularies are mapped to the UNK token.

The MT training data is also used to determine the topic distributions. We train the all topic models with 200 iterations. The numbers of topic are {10, 20, 30, 40, 50, 80, 100, 150} for both languages. The LDA models training takes approximately 40 hours in average.[4] The HTMM models takes approximately 6 hours in average.[5] Special words, i.e. UNK and EOS, are set to have uniform topic distributions.

We compare our approaches with two baselines. The first one is the Statistical Machine Translation (SMT) baseline, which is trained using Moses (Koehn et al. 2007) machine translation framework, with a lexicalized reordering model (Koehn et al. 2005, Galley and Manning 2008) and a 5-gram KenLM (Heafield 2011) Language Model (LM). The LM is trained using the target side of the parallel training data. We use all default parameters in Moses. Our second baseline is an NMT system. We use the encoder-decoder framework with a single layer bidirectional GRU as the encoder and a single layer GRU with attention model as the decoder. Each word in the training corpora is converted into a 512-dimensional vector during training. The encoder and decoder contain 1,024 hidden units each. The bidirectional RNN is also used. We use beam search during translating, with a beam size of 5. We use a mini-batch (with batch size 32) Stochastic Gradient Descent algorithm together with Adadelta (Zeiler 2012) to train our models. All NMT models are trained up to 320,000 updates and the models are saved at each 1,000 updates. The training takes approximately 4 days on the NVIDIA GeForce GTX TITAN X GM200 GPU machine. We then choose the final model based on the Bilingual Evaluation Understudy (BLEU) (Papineni et al. 2002) score of the development data.

[4]The LDA implementations we used to training the models can be found at `https://radimrehurek.com/gensim/models/ldamulticore.html`

[5]The HTMM implementations we used to training the models can be found at `http://www.cs.toronto.edu/~amitg/`

### 5.4.2 Results

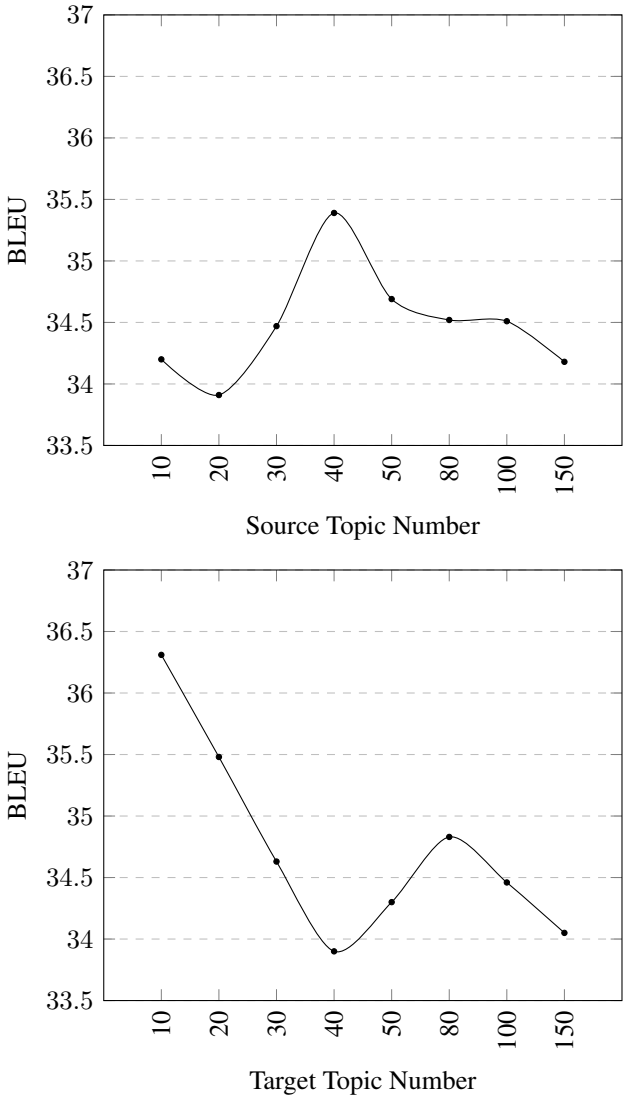



Figure 5.9: LDA topic numbers vs. translation BLEU scores on the NIST 2002 development dataset.

Table 5.1 presents the experiment results on the development and test data, the number beside each topic-based NMT system indicates the topic number used in the system (source topic number, target topic number or source and target topic numbers). ‡ and † indicate statistically significant (Koehn 2004) improvements upon the NMT baseline at the p = 0.01 and p = 0.05 level, respectively (with 1000 iterations).

The source and target topic numbers in Table 5.1 are experimentally chosen from {10,

| Topic Model | Systems | NIST 2002 (development) | NIST 2004 (test) | NIST 2005 (test) |
|---|---|---|---|---|
| | SMT | 33.42 | 32.36 | 30.11 |
| | NMT | 34.33 | 34.76 | 31.12 |
| **LDA** | Source Topic-based NMT (40) | 35.39 | 35.17† | 31.95‡ |
| | Target Topic-based NMT (10) | 36.31 | 35.43‡ | 32.50‡ |
| | Topic-based NMT (40,10) | 34.86 | 35.91‡ | **32.79**‡ |
| **HTMM** | Source Topic-based NMT (80) | 33.98 | 35.62‡ | 31.94‡ |
| | Target Topic-based NMT (80) | 34.73 | 36.02‡ | 31.77 † |
| | Topic-based NMT (80,80) | 35.47 | **36.43**‡ | 32.73‡ |

Table 5.1: BLEU scores of the trained SMT and NMT models, the topic distributions are learned using LDA or HTMM. We use ‡ and † to indicate statistically significant (Koehn,2004) improvements upon the NMT baseline model the p = 0.01 and p = 0.05 level, respectively. The significance testing uses bootstrapping (Koehn 2004) method with 1,000 iterations.

20, 30, 40, 50, 80, 100, 150} separately according to the development BLEU scores. As an example, Figure 5.9 presents the development BLEU scores when LDA is employed. We then combine the optimal topic numbers for each setting. For example, we use 40 source and 10 target topics in the LDA topic-based NMT (40,10) system, and 80 source and 80 target topics in the HTMM topic-based NMT (80,80) system.

In Table 5.1, significant improvements can be observed on the test data when LDA topic model is utilized. On NIST 2004, we observe improvements of 0.41 (absolute, 1.2% relative) and 0.67 (absolute, 1.9% relative) BLEU scores compared to the NMT baseline on the source/target topic-based NMT models, respectively. On NIST 2005, there is a gain of 0.83 (absolute, 2.7% relative) and 1.38 (absolute, 4.4% relative) BLEU scores compared to the NMT baseline on the source/target topic-based NMT models, respectively. This suggests that topic information can be independently used in our proposed models. However, topic-based NMT using target topics shows better results than using the source topics because that the target language is more directly involved with generating translation outputs and can benefit better to the translation qualities than the source language. When topic information is used jointly, models can achieve 35.91 and 32.79 BLEU scores on the NIST 2004 and NIST 2005 test data, respectively. These results are 1.15 (absolute, 3.3% relative) and 1.67 (absolute, 5.4% relative) higher in contrast with the NMT baseline model.

In Table 5.1, the similar trend can also be observed in the HTMM experiments. The

source topic-based NMT system can outperform the NMT baseline model with 0.86 (absolute, 2.5% relative) and 0.82 (absolute, 2.6% relative) BLEU score improvements on the NIST 2004 and 2005 test data, respectively. There is a further improvement of 0.4 absolute BLEU score gained in the target topic-based NMT model on the NIST 2014 test data. We notice a 0.17 absolute BLEU score decrease in the target topic-based NMT model on the NIST 2015 test data compared to the source topic-based NMT model. However, it still can outperform the baseline NMT by 0.65 (absolute, 2.0% relative) BLEU score. The best performed model in the HTMM experiments is the topic-based NMT (80&80) system, where achieves 36.43 and 32.73 BLEU scores on the two test sets, respectively.

Comparing the experiments of LDA or HTMM in Table 5.1, we think using HTMM is more congruous with our proposed approaches. The HTMM relaxes the assumption that words are independent in a sentence, believes that there are dependencies between the topic of words in a sentence which has the similar assumption with this work, i.e. the"topic consistent" behaviour in the training data.

## 5.5 Result Analysis

In this section, we use translation outputs to investigate the experimental results in three aspects: the attention model, the word choices and the lexical coverage in the proposed topic-based NMT.

### 5.5.1 Attention Model

Following the work at Bahdanau et al. (2015), we also use heat map to represent the "*soft*" alignments, as seen in Figure 5.10. Large values are presented by lighter colors. We use the first three source words as our example to compare the two alignments, e.g. "77", "家" and "私营". When the translated words are produced in the baseline system, e.g. "The UNK private", relatively large weights are all assigned to the first source word "77", but "家" and "私营". As a result, the scattered alignments can be one of the reasons that causes the wrong

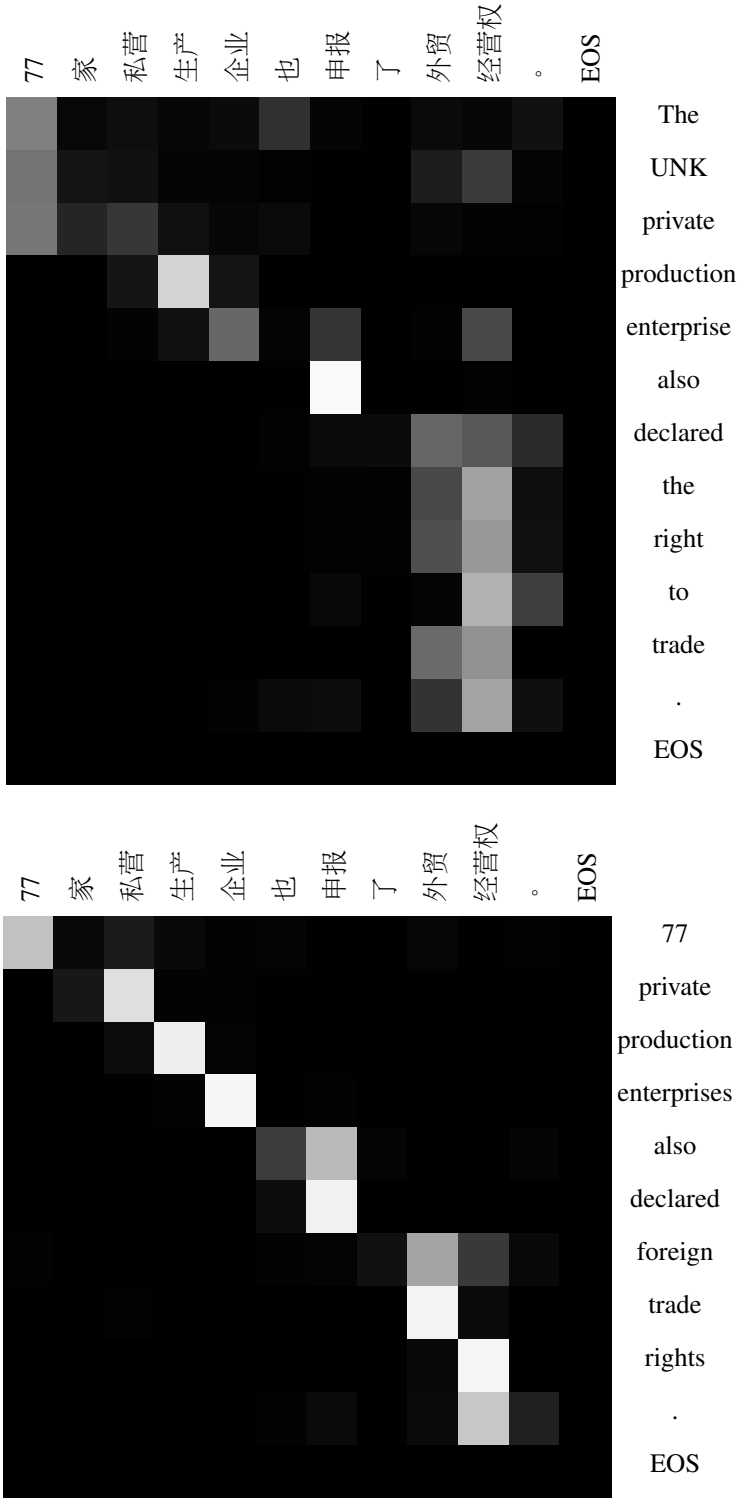



Figure 5.10: The comparison of alignments generated by NMT baseline (top) and topic-based NMT (bottom).

| | |
|---|---|
| **Source** | 他 当即 被 送往 医院 **抢救** 。 |
| **Reference** | He was taken immediately to a hospital for emergency treatment . |
| **Baseline NMT** | He was sent to the hospital for **rescue** . |
| **Topic-based NMT** | He was sent to hospital for **emergency treatment** . |
| | |
| **Source** | 过半 英国人 不 赞同 政府 支持 美 打 伊拉克 。 |
| **Reference** | Over half of all British voters disapprove British support for an American attack on Iraq . |
| **Baseline NMT** | The British people does not agree to support United States to **fight** Iraq . |
| **Topic-based NMT** | The British people disagree with the government 's support for US **war against** Iraq . |

Figure 5.11: The examples shows our observations that better word choices can be made in the topic-based NMT.

translation. In contrast, we include the topic information to the alignments calculations in the topic-based NMT. The operation of current translation selecting the most appropriate source words to translate is influenced also by the source topic distributions. As we can see, a better translation "77 private" is produced. Another example can also be observed for source words "外贸 经营权".

### 5.5.2 Topic Consistent

Figure 5.11 shows examples that the topic consistent is maintained in the translations produced in the topic-based NMT model. For example in the baseline NMT translations, source words "抢救" is translated into "rescue" and "打" is translated into "fight", which are not the best word choices. In the topic-based NMT, models are trained with additional topic information, as seen in Equation (5.10), we can maintain the topic consistent in the translation outputs. In this way, the current word can be predicted with the influence of the topic distributions from previous words and better translation can be produced. For example, in the translation examples produced by the topic-based NMT in Figure 5.11, word "抢救" is translated into "emergency treatment", and "打" is translated into "war against", which are

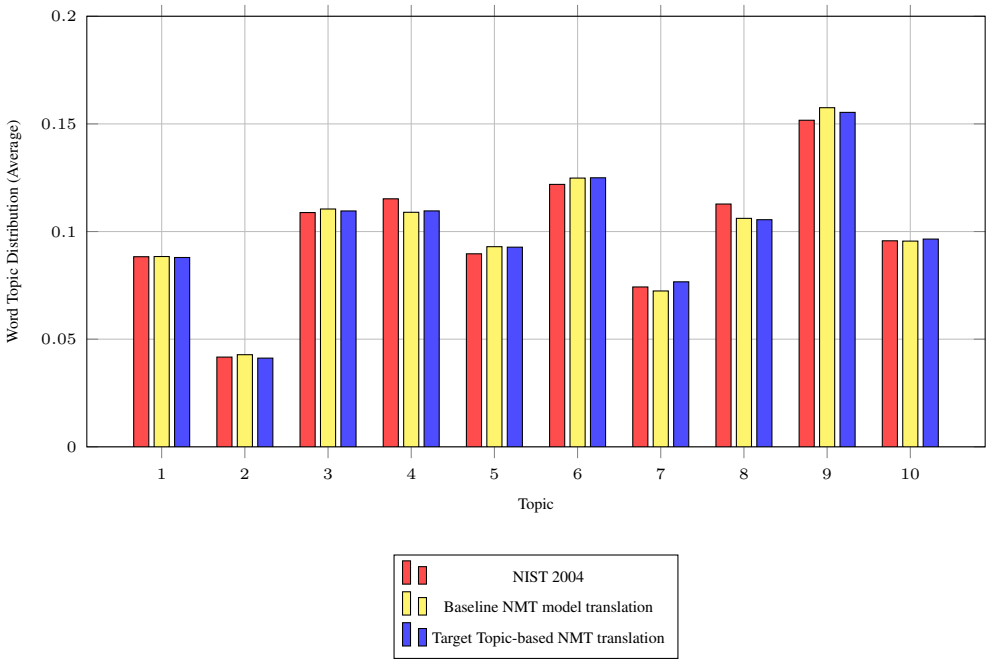



Figure 5.12: The average topic distribution for words between NIST 2004, baseline NMT model translations and target topic-informed NMT model translations.

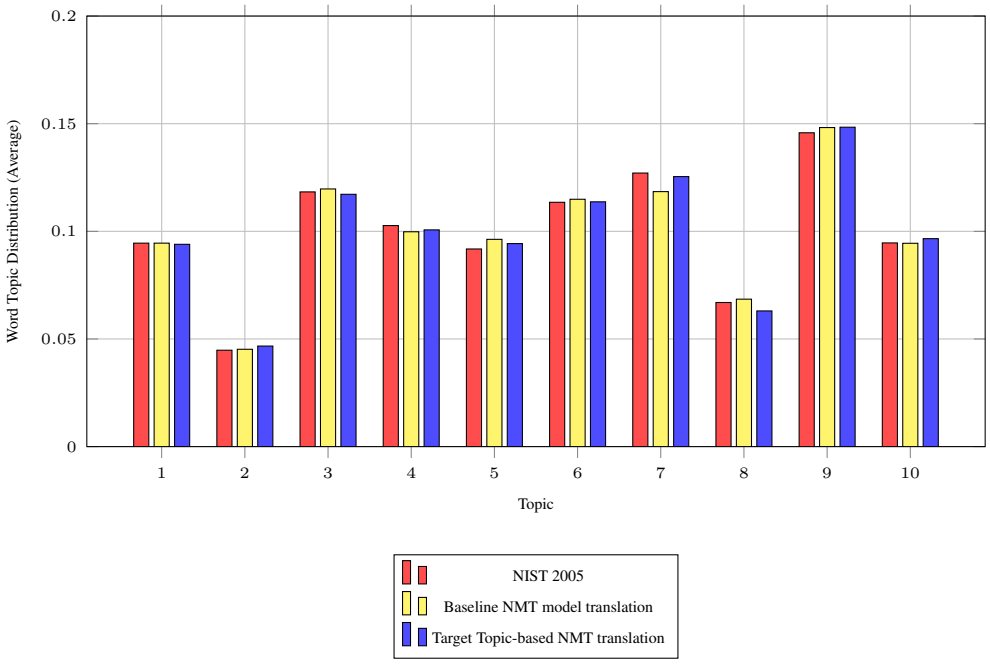



Figure 5.13: The average topic distribution for words between NIST 2005, baseline NMT model translations and target topic-informed NMT model translations.

the better translations in the given the context.

Figure 5.12 and 5.13 illustrate the average word topic distribution $H_T$ between the reference, baseline NMT and target topic-based NMT (Target Topic-based NMT (10) in Table 5.1) model translations. $H_T$ is computed as in Equation (5.11):

$$H_T(E) = \frac{1}{N} \sum_{w \in E} \beta_w^e \tag{5.11}$$

where $E$ is the translation, $w$ is a word in $E$, $N$ is the word count in $E$ and $\beta_w^e$ is topic distribution of $w$ obtained from the topic model. We can observe that the distributions between the translations produced by the topic-based NMT model and the reference data are more similar, such as in topic 3, 4, 6, 7, 8 and 9.

### 5.5.3 Lexical Coverage

While another drawback of NMT systems is that the lexical coverage in translation outputs is low. This is due to the trade off between efficiency and quality. Observing the translation outputs, we find that some of the source words are translated into the UNK token even though the correct translations can be found in the target vocabulary list in both the baseline and topic-based NMT systems. However, comparing the UNK token numbers produced between the two systems, we think the topic-based NMT can handle the UNK token better. As presented in Table 5.2, there are 2.3% and 2.7% words are UNK token in the translations of NIST 2014 and NIST 2015 produced by the baseline model, respectively. On the contrary, the percentages are reduced to 1.9% and 2.3% in the topic-based NMT system in the translations of NIST 2014 and NIST 2015, respectively.

We think the reason is that lower frequency words (Frequency $<k$, where $k$ is a predefined threshold) are grouped into a special UNK token in NMT, as seen in Figure 5.14. As a consequences, the UNK token is not in lower frequency any more (as seen in Figure 5.15) as its frequency is the sum of all the lower frequency words before grouping in Figure 5.14. Recall that the UNK token is set to have an uniform distribution in our topic models,

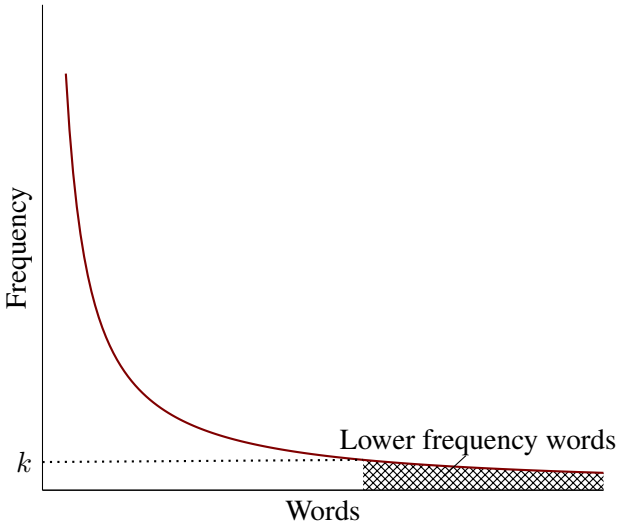



Figure 5.14: We map lower frequency words to the special UNK word in NMT, e.g. Frequency $<k$, where $k$ is a pre-defined threshold.

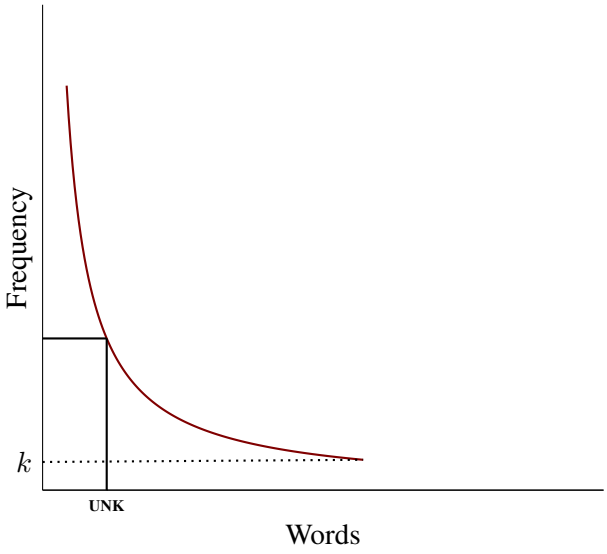



Figure 5.15: The frequency of word UNK is increased. For example, UNK is ranked as the third and eighth highest frequency word in the Chinese and English training corpus, respectively. This is because the frequency of word UNK is the sum of all the lower frequency words in Figure 5.12.

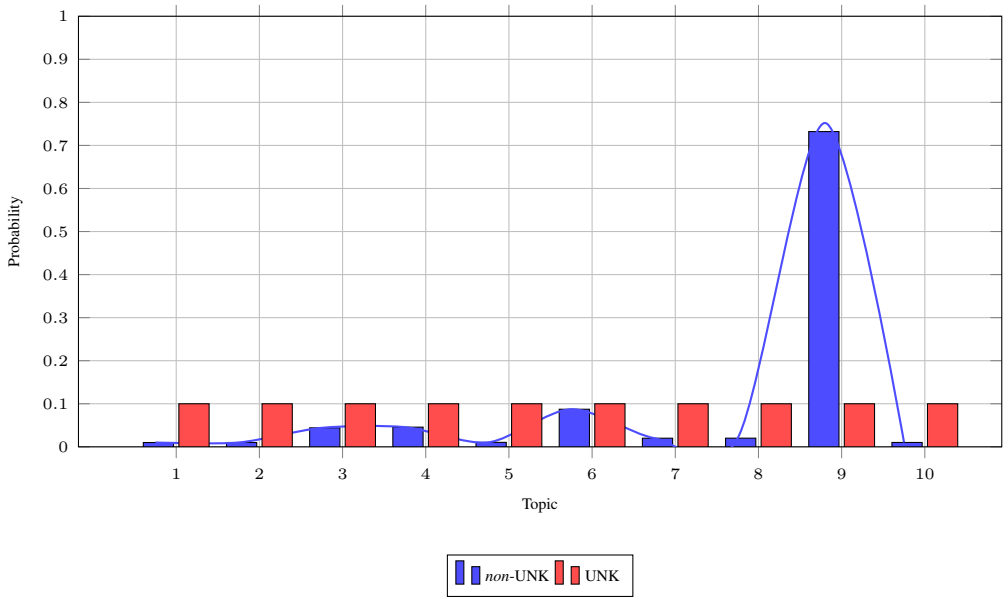



Figure 5.16: An example of UNK vs. *non*-UNK word topic distribution. The UNK word has an uniform distribution, whereas *non*-UNK words have their favoring topics. In this example, the *non*-UNK word is the word "speaker" and the topic distribution is extracted from our English topic model (the number of topics is 10). Other words which can be found in the same topic as word "speaker" are "committe", "economic", "international", etc.

| | **UNK Percentage** | |
|---|---|---|
| | **Baseline** | **Topic-based** |
| NIST 2004 | 2.3% | 1.9% |
| NIST 2005 | 2.7% | 2.3% |

Table 5.2: The percentage of *UNK* produced in translation outputs comparing between the baseline NMT and the topic-based NMT

| | **Word Number and Brevity Penalty (BP) in Translations** | | | | |
|---|---|---|---|---|---|
| | **Reference** | **Baseline** | **Baseline BP** | **Topic-based** | **Topic-based BP** |
| NIST 2004 | 51,967 | 50,020 | 0.988 | 50,333 | 0.994 |
| NIST 2005 | 34,563 | 34,441 | 1.000 | 34,610 | 1.000 |

Table 5.3: The number of words produced and brevity penalty (BP) calculated in translation outputs comparing between the baseline NMT and the topic-based NMT.

and other words (*non*-UNK words) are learned to have a topic distribution with a preferred topic, as seen in Figure 5.16. Therefore, when a translation is being chosen between the UNK token and a *non*-UNK word, the topic distributions of previously translated words can influence that a *non*-UNK word will have a higher chance to be selected since the UNK token is not favouring to any topic.

| | |
|---|---|
| **Source** | 印尼 国会 **议长** 出庭 受审 。 |
| **Reference** | Indonesia congress **speaker** on trial . |
| **Baseline NMT** | **UNK** of Indonesia's congress in court . |
| **Topic-based NMT** | Indonesia parliament **speaker** is trial . |
| | |
| **Source** | 卡伊达 组织 领导层 也 开始 活跃 起来 。 |
| **Reference** | The leadership of Qaida group began to *be active* . |
| **Baseline NMT** | The leadership of the **UNK** city began to **UNK** . |
| **Topic-based NMT** | The leadership of the **UNK** organization has begun to be active . |

Figure 5.17: The examples shows our observations that less number of UNK can be produced in the topic-based NMT.

In NMT, the translation process is terminated if an EOS token is produced. The EOS is also set to have an uniform distribution in the topic models. Accordingly, to confirm our conclusion about the UNK token, EOS token should also have a lower probability to be selected than the other words. Essentially, this means that our topic-based model is expected to produce longer translations than the baseline system. Table 5.3 shows the number of words in the translations produced by the baseline NMT and topic-based NMT models. We find that the topic-based NMT model produces more words in translations than the baseline model, e.g. 50,333 words and 34,610 words in NIST 2004 and NIST 2005, respectively. In contrast, the baseline system produces 50,020 and 34,441 words in NIST 2004 and NIST 2005, respectively. As a conclusion, topic-based NMT can reduce the UNK token number even with more words are produced in the translation outputs.

Inspecting the translation examples in Figure 5.17, "议长" and "活跃" fail to be translated by the baseline NMT. However, topic-based NMT is able to use the topic information to produce correct translations. For example, source words "议长" and "活跃" are translated into "speaker" and "active", respectively. The source word "卡伊达" does not appear in the parallel training data, thus both systems produce a UNK token in this case.

The optimal number of topics used in the source topic-based NMT and target topic-based NMT is not necessarily the same. For example, the best performed source topic-

| Systems | NIST 2002 (development) | NIST 2004 (test) | NIST 2005 (test) |
|---|---|---|---|
| LDA Topic-based NMT (40,10) | 34.86 | 35.91 | 32.79 |
| *Random* Topic-based NMT (40,10) | 33.84 | 34.12 | 30.68 |
| HTMM Topic-based NMT (80,80) | 35.47 | 36.43 | 32.73 |

Table 5.4: BLEU score comparison between NMT models trained using LDA, HTMM and the *random* topic models, where results of LDA Topic-based NMT (40,10) and HTMM Topic-based NMT (80,80) models are the same as Table 5.1.

based NMT and target topic-based NMT are configured with topic number 40 and 10 in the LDA experiments, respectively. We suggest two possible reasons that can cause such outcomes. Firstly, the topic information is employed differently in these two systems. As we presented in Section 5.3.1 and Section 5.3.2, the source topic information is appended to each source word as extra elements to the source context vector in the encoding phase, whereas the target topic distributions are fed into an RNN in the decoder. Another reason is that the motivation of the two systems are different. The source topic-based NMT tries to improve the efficiency of the attention model, and the target topic-based NMT aims to maintain the "topic-consistency" in the translation outputs.

An interesting way to confirm that the topic information is effective in the proposed topic-based NMT models is to train a NMT model using a "low-quality" (*random*) topic model. For this experiment, we used the same configurations as reported for the LDA topic model experiment but assigning a dirichlet distribution for each word in the training data. Table 5.4 shows the experimental result when a *random* topic model is used in the topic-based NMT model.

## 5.6 Summary

NMT has a lot of potential as a new approach to MT. We present in this chapter, a novel approach of integrating topic knowledge into the existing NMT architecture to answer our **RQ3**:

> **RQ3** *How word topic distributions can be used to improve translation quality for NMT models in a domain-unawareness scenario?*

We relied on topic model algorithms to discovery word topic distributions in the training data in a *domain-unawareness* scenario. Following previous work assumptions that words in the same sentences sharing the same (or similar) topic, we propose three different integration approaches to avail the topic information in an NMT system.

Through our experiments, we demonstrated that translation quality can be improved, which our topic-based NMT can achieve 1.15 and 1.67 absolute improvement in terms of BLEU score on two different test data sets compared to the baseline NMT system. The experiment results not only demonstrate the effectiveness of the proposed model, but also show an approach to enrich the representation of the context vector produced by encoder and decoder. We showed that introducing topic information in the NMT can produce translations with better word choices, and less number of UNK. We provided concrete examples to support our observations. Also, better word alignments can also be observed which consequently benefits translation quality in the proposed model. Furthermore, we made comparisons between the two topic-modelling algorithms and found that the HTMM is congruous with our proposed approaches.

# Chapter 6

# Conclusion

In this chapter, we provide conclusions for the previous chapters and revisit the research questions with the answers we have provided to them. We then summarise the contributions of our work in this thesis. Later in this chapter, we explore the various possibilities for further research.

## 6.1 Conclusion and Research Questions Answered

In Chapter 1, we provided the motivations underpinning our study of domain adaptation in Machine Translation (MT). We noted that domains depend on several factors, e.g. provenance, genres, topics, dialects or styles, and even the combination of all those factors. Moreover, because the training data of an MT system could be collected from various resources, i.e. it could come from different domains with or without domain labels, or in separate files or in a large file where all data are concatenated, we assumed two scenarios in this thesis:

***Domain-awareness:*** The domain information is given explicitly in the training data.

***Domain-unawareness:*** The domain information is not given explicitly in the training data.

We then presented our specific research questions, as follows:

**RQ1** *In a domain-awareness scenario, how can we further improve the current domain adaptation method of an Statistical Machine Translation (SMT) by availing of the domain-likeness of the context in which a word or a phrase appears?*

**RQ2** *Whether the vector model trained using General-Domain (GD) data can be used in domain adaptation in a domain-awareness scenario?*

**RQ3** *How word topic distributions can be used to improve translation quality for Neural Machine Translation (NMT) models in a domain-unawareness scenario?*

In Chapter 2, we first provide an overview of the MT models and algorithms related to this thesis, including Phrase-based Statistical Machine Translation (PBSMT) and NMT. We then reviewed the word vector models, Recurrent Neural Network (RNN) models and Gated Recurrent Unit. We also highlighted previous domain-adaptation approaches in three categories: data selection, domain adaptation for SMT, and domain adaptation for NMT.

In Chapter 3, we addressed **RQ1** by presenting an efficient translation model combination approach which extends the previous domain adaptation work of Bisazza et al. (2011) in a *domain-awareness* scenario. In Bisazza et al. (2011), a binary feature indicating the provenance of the phrase pairs was used in the fill-up model combination approach. However, we think that a GD translation model is often trained using a large corpus which comprises different domains. Some GD data may be very similar/dissimilar with the In-Domain (ID) data. The provenance feature may cause potential ID phrase pairs to be unfairly penalised.

Motivated by the distributional hypothesis that word meanings are implied by the context rather than by the words themselves (Banchs 2014), contextual information can be used to measure the closeness of bilingual phrase pairs from ID to GD. We first proposed a domain-likeness model that can be used to estimate the domain-likeness values for bilingual sentence pairs in three simple heuristics. The estimated values are probabilistic in nature which can be interpreted as the distance of a phrase pair from ID to GD. We then

applied these probabilistic values as an additional feature in the fill-up model combination approach to gave more attention to the phrase pairs which are in the GD translation model but are close to ID.

We designed two experimental setups: (i) the GD dataset being significantly larger than the ID data, and (ii) the two datasets being similar in size. We demonstrated that our fill-up approach can significantly improve translation quality in both experiments compared to the previous fill-up method. We provided analysis of the probabilistic feature values on the GD translation model. We also compared our approach with a data selection approach (Axelrod et al. 2011) and found that our method can outperform or produce comparable translation results.

Recent studies have shown remarkable results in applying neural networks in Natural Language Processing (NLP), especially using RNN. In the field of MT, significant improvements have also been observed (Bahdanau et al. 2015, Luong et al. 2015b, He et al. 2016, Tu et al. 2016) when applying neural network training. However, there is very little work to be found in the neural Language Model (LM) or NMT literature to address the domain-adaptation challenge. Thus, we moved our attention to proposing domain adaptation techniques to the most recently proposed neural network training approach in **RQ2** and **RQ3**.

In Chapter 4, we answered **RQ2** by proposing a domain-adaptation approach which avails of large pre-trained word vector models, e.g. the Google *word2vec* model (Mikolov et al. 2013a) and the Global Vectors for Word Representation (Pennington et al. 2014) model, in a *domain-awareness* scenario.

The word vector representations (Mikolov et al. 2013a) have a very good generalization ability (Mikolov et al. 2013c) and are used to represent words for almost all NLP tasks when neural network is applied. These models are trained with a large amount of data and can be used in the situation when the relevant data is limited in NLP tasks. We think the pre-trained and the task-specific-trained word vector models are complementary to one another in neural network training. The pre-trained word vector models can be a useful supplement

when the ID training data is too small.

However, if the task-specific-trained word vector models are substituted for the pre-trained one directly, LM performance will decrease since the word vectors are trained on different domains (as seen in Table 4.3). We therefore designed three domain-adaptation methods, namely adaptation on word vectors, adaptation on context vectors, and Gated Domain Adaptation (GDA). We found that the GDA approach outperforms the baseline models and the other two domain-adaptation methods in the LM experiments. Furthermore, we also employed the GDA method on the SMT re-ranking task and NMT framework to demonstrate its effectiveness.

In Chapter 5, we addressed **RQ3** by investigating a novel approach to integrate topic knowledge into the existing NMT architecture. We were confronted with a different challenge in **RQ3**; the domain information is not given explicitly in the training data in a *domain-unawareness* scenario. The training data may come from tens or even hundreds of different resources without well-defined domain labels to distinguish them, which is common in MT training. We rely on the well-established topic-learning algorithms, e.g. Latent Dirichlet Allocation (LDA) (Blei et al. 2003) or Hidden Topic Markov Model (HTMM) (Gruber et al. 2007), to discover the domain information.

Motivated by the previous study of Su et al. (2015) and our observation regarding "topic consistent" behaviour (where words in the training data often belong to the same or similar topic), we hypothesized that this can be a strong indication to influence NMT models. Intuitively, if we can identify the topic information of previously translated words, we then can provide such information to a translation system and train it to maintain the consistency of topics in the translation output. In other words, we can regard topics as domains in this scenario to guide the translation process. In our experiments, we demonstrated that translation quality can be improved. By introducing topic information in NMT, we observed better word choices, and fewer Unknown (UNK) words were produced. Furthermore, we also observed that better word alignments can be learned which are beneficial to translation quality. We provided a comparison between LDA and HTMM topic modelling algorithms

in the proposed approach and found that the HTMM can achieve better results.

## 6.2 Contributions

In this thesis, we have investigated novel approaches of domain adaptation for MT. The contributions of our work can be summarized as follows:

- **Domain-likeness model**. We presented an approach to measure the closeness of bilingual phrases from ID and GD. Furthermore, we extended the previous binary feature to a probabilistic feature in the fill-up combination approach. We compared the effectiveness of our approach with previous related work and found that our approach can improve the translation quality significantly. In our experiments, we also confirmed several previous research findings, including that (i) data selection is a heavy-handed approach, (ii) the unselected GD data can still make a good contribution to the MT system, and (iii) it is harmful to translation quality if a large proportion of GD data is concatenated with ID data for SMT training.

- **GDA**. We designed a network specifically for the domain-adaptation challenge in the sequence prediction task when an RNN is used. Our work is the first to propose model adaptation by adapting large pre-trained word vector models (as the GD data) to the task of ID model training in the domain-adaptation literature. The proposed approach is fast and does not require additional task-oriented data. In addition, the GDA technique can be used in many sequential neural network applications when an RNN is used. We have shown that using the GDA technique in training can outperform the baseline models on two datasets in the neural LM experiments. Moreover, we also demonstrated that the translation quality can be significantly improved when the proposed GDA technique is used in the state-of-the-art NMT model.

- **Topic-based NMT**. We proposed a novel approach to perform domain adaptation when the domain information is not explicitly given in the training data. We integrated topic knowledge into an existing state-of-the-art NMT model and achieved

significant improvements in translation quality compared to the baseline models. Our findings include the discovery that applying topic information in NMT can not only influence correction words to be produced, but also lower the number of UNK tokens appearing in the translations.

## 6.3 Future Work

There are several possible extensions to the models presented in this thesis. We summarize them in the following sections.

### 6.3.1 Limitations on the Domain-likeness Model

The proposed domain-likeness model in Chapter 3 has several limitations. Firstly, the training data of the domain-likeness model needs to be bilingual as the training features are based on previous bilingual data selection approach (Axelrod et al. 2011). However, it is possible to use other features which do not require bilingual training data but rather monolingual data which is much cheaper to obtain. Secondly, the training of the domain-likeness model can be addressed as a classification task. Therefore, a deeper investigation into the effect of different classification algorithms with translation quality is necessary to further improve the solution, in particular with the recent neural network data selection approaches (Chen and Huang 2016, Peris et al. 2016). Furthermore, linguistic knowledge can provide an indication of how to distinguish sentence pairs between domains (Toral 2013). Therefore, a possible extension of the domain-likeness model is to also incorporate linguistic knowledge in the training phase.

### 6.3.2 Deeper Analysis on the GDA Technique

In Section 4.3.4, we demonstrated the effectiveness of the GDA technique, showing that the improvement of GDA over the baseline model is obtained in the early training iterations. However, it is also interesting to see how often gates are completely open or closed in

the GDA technique in the related tasks. In an extreme scenario, the gates may be always open in some situations, which means ID word vectors are completely ignored. In contrast, the gates may be always closed, which means pre-trained word vectors are completely ignored. Moreover, we believe there is a strong relationship between the status (open/close) of the gates and word frequencies in the training data, as lower frequency words have less opportunity to be seen in training, and so the proposed approach may be beneficial to them. Further work is required to analyse these aspects. Moreover, we experimental demonstrated that the GDA technique can be used in Gated Recurrent Unit (GRU) (Chung et al. 2014) network. However, it is also possible to apply GDA on a Long Short-Term Memory (LSTM) (Hochreiter and Schmidhuber 1997) network. One further work is to make comparison on the performance between the GDA technique is applying on GRU and LSTM networks. Other future work on the GDA technique includes experiments on other sequence prediction tasks, such as Part-of-Speech Tagging, and translation of low-resource languages.

### 6.3.3 Joint Training for a Topic-based NMT

The topic-based NMT model we proposed in Chapter 5 needs topic models to be trained in both source and target languages in advance. Recent researches (Dong et al. 2015, Luong et al. 2015a) show that multiple tasks can be trained jointly with NMT training to achieve significantly higher translation quality over an individually trained NMT model. Therefore, an interesting extension of the topic-based NMT is to combine topic modelling training (Cao et al. 2015) with NMT model training. Moreover, we can also combine or extend the current monolingual topic model to multilingual topic models (Mimno et al. 2009, Ni et al. 2011) which would allow us to discover the topic information across the source and target languages. For example, the prediction of the current word also depends on the bilingual topic information discovered from the source sentence and previously translated words.

# Appendix A

# Open Source Tools

Instead of reimplementing the software used to train baseline systems reported in this thesis, we adapt the existing tested open source tools into our work. We list all the software used in our work as follows:

- Word Alignment

  The alignment tool we use is MGIZA (Gao and Vogel 2008), which can be downloaded from `https://github.com/moses-smt/mgiza.git`. MGIZA extends GIZA++ (Och and Ney 2003) alignment tool with the support of multi-threading.

- Phrase-based Statistical Machine Translation (PBSMT)

  We use the PBSMT implementation in the Moses 3.0 framework (Koehn et al. 2007) for our PBSMT baselines. The software can be downloaded from `https://github.com/moses-smt/mosesdecoder/tree/RELEASE-3.0`.

- *n*-gram Language Model (LM)

  The *n*-gram LMs in this thesis are trained using KenLM LM toolkit (Heafield 2011). It can be downloaded from `https://kheafield.com/code/kenlm/`.

- Neural Machine Translation (NMT)

Our NMT baseline implementation is from the dl4mt-tutorial.[1] The implementation can be downloaded from `https://github.com/nyu-dl/dl4mt-tutorial`.

- Recurrent Neural Network (RNN)LM

  The RNNLM baseline is implemented using Tensorflow (version 6.0) (Abadi et al. 2015). The code can be downloaded from `https://github.com/tensorflow/tensorflow/blob/0.6.0/tensorflow/models/rnn/ptb/ptb_word_lm.py`.

- Machine Translation (MT) Evaluation Metrics

  We use the Bilingual Evaluation Understudy (BLEU) implementation in the Moses 3.0 framework, it can be found at `https://github.com/moses-smt/mosesdecoder/tree/RELEASE-3.0/scripts/generic/multi-bleu.perl`.

[1] `http://dl4mt.computing.dcu.ie`

# Bibliography

Abadi, M., Agarwal, A., Barham, P., Brevdo, E., Chen, Z., Citro, C., Corrado, G. S., Davis, A., Dean, J., Devin, M., Ghemawat, S., Goodfellow, I., Harp, A., Irving, G., Isard, M., Jia, Y., Jozefowicz, R., Kaiser, L., Kudlur, M., Levenberg, J., Mané, D., Monga, R., Moore, S., Murray, D., Olah, C., Schuster, M., Shlens, J., Steiner, B., Sutskever, I., Talwar, K., Tucker, P., Vanhoucke, V., Vasudevan, V., Viégas, F., Vinyals, O., Warden, P., Wattenberg, M., Wicke, M., Yu, Y., and Zheng, X. (2015). TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems. Software available from tensorflow.org.

Axelrod, A., He, X., and Gao, J. (2011). Domain Adaptation via Pseudo In-Domain Data Selection. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 355–362, Edinburgh, UK.

Bahdanau, D., Cho, K., and Bengio, Y. (2015). Neural Machine Translation by Jointly Learning to Align and Translate. In *Proceedings of the International Conference on Learning Representations 2015, ICLR 2015, May 07-09, 2015, San Diego, California, USA*.

Banchs, R. E. (2014). A Principled Approach to Context-Aware Machine Translation. In *Proceedings of the 3rd Workshop on Hybrid Approaches to Machine Translation, HyTra@EACL 2014, April 27, 2014, Gothenburg, Sweden*, pages 70–74.

Banerjee, P., Naskar, S. K., Roturier, J., Way, A., and van Genabith, J. (2012). Translation Quality-Based Supplementary Data Selection by Incremental Update of Translation

Models. In *COLING 2012, 24th International Conference on Computational Linguistics, Proceedings of the Conference: Technical Papers, 8-15 December 2012, Mumbai, India*, pages 149–166.

Banerjee, S. and Lavie, A. (2005). METEOR: An Automatic Metric for MT Evaluation with Improved Correlation with Human Judgments. In *Intrinsic and Extrinsic Evaluation Measures for Machine Translation and/or Summarization, Proceedings of the ACL-05 Workshop*, pages 65–72, University of Michigan, Ann Arbor, Michigan, USA.

Bengio, Y., Ducharme, R., Vincent, P., and Janvin, C. (2003). A Neural Probabilistic Language Model. *J. Mach. Learn. Res.*, 3:1137–1155.

Bengio, Y., Louradour, J., Collobert, R., and Weston, J. (2009). Curriculum Learning. In *Proceedings of the 26th Annual International Conference on Machine Learning*, ICML '09, pages 41–48, New York, NY, USA.

Bengio, Y., Simard, P., and Frasconi, P. (1994). Learning Long-Term Dependencies with Gradient Descent is Difficult. *IEEE Transactions on Neural Networks*, 5(2):157–166.

Bertoldi, N. and Federico, M. (2009). Domain Adaptation for Statistical Machine Translation with Monolingual Resources. In *Proceedings of the fourth workshop on Statistical Machine Translation, March 30-31, 2009*, pages 182–189, Athens, Greece. Association for Computational Linguistics.

Biçici, E. and Yuret, D. (2011). Instance Selection for Machine Translation using Feature Decay Algorithms. In *Proceedings of the Sixth Workshop on Statistical Machine Translation*, pages 272–283, Edinburgh, UK.

Bisazza, A. and Federico, M. (2013). Dynamically Shaping the Reordering Search Space of Phrase-Based Statistical Machine Translation. *Transactions of the Association for Computational Linguistics TACL*, 1:327–340.

Bisazza, A., Ruiz, N., and Federico, M. (2011). Fill-up versus Interpolation Methods for

Phrase-based SMT Adaptation. In *2011 International Workshop on Spoken Language Translation, IWSLT 2011, San Francisco, CA, USA, December 8-9, 2011*, pages 136–143.

Blei, D. M., Ng, A. Y., and Jordan, M. I. (2003). Latent Dirichlet Allocation. *The Journal of Machine Learning Research*, 3:993–1022.

Bojar, O., Chatterjee, R., Federmann, C., Graham, Y., Haddow, B., Huck, M., Jimeno-Yepes, A., Koehn, P., Logacheva, V., Monz, C., Negri, M., Névéol, A., Neves, M. L., Popel, M., Post, M., Rubino, R., Scarton, C., Specia, L., Turchi, M., Verspoor, K. M., and Zampieri, M. (2016). Findings of the 2016 Conference on Machine Translation. In *Proceedings of the First Conference on Machine Translation, WMT 2016, colocated with ACL 2016, August 11-12, Berlin, Germany*, pages 131–198.

Brown, P., Cocke, J., Pietra, S. D., Pietra, V. D., Jelinek, F., Lafferty, J., Mercer, R., and Roosin, P. (1990). A Statistical Approach to Machine Translation. *Computational Linguistics*, 16(2):79–85.

Brown, P. F., Pietra, V. J. D., Pietra, S. A. D., and Mercer, R. L. (1993). The Mathematics of Statistical Machine Translation: Parameter Estimation. *Computational Linguistics*, 19(2):263–311.

Cao, Z., Li, S., Liu, Y., Li, W., and Ji, H. (2015). A Novel Neural Topic Model and Its Supervised Extension. In *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence, January 25-30, 2015, Austin, Texas, USA.*, pages 2210–2216.

Carpuat, M. and Wu, D. (2007). Improving Statistical Machine Translation Using Word Sense Disambiguation. In *EMNLP-CoNLL 2007, Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning, June 28-30, 2007, Prague, Czech Republic*, pages 61–72.

Cettolo, M., Niehues, J., Stüker, S., Bentivogli, L., Cattoni, R., and Federico, M. (2015).

The IWSLT 2015 Evaluation Campaign. In *Proceedings of the International Workshop on Spoken Language Translation*, pages 76–79, Da Nang, Vietnam.

Cettolo, M., Niehues, J., Stüker, S., Bentivogli, L., Cattoni, R., and Federico, M. (2016). The IWSLT 2016 Evaluation Campaign. In *Proceedings of the International Workshop on Spoken Language Translation*, Seattle, USA.

Chang, C.-C. and Lin, C.-J. (2011). LIBSVM: A library for support vector machines. *ACM Transactions on Intelligent Systems and Technology*, 2:27:1–27:27. Software available at `http://www.csie.ntu.edu.tw/~cjlin/libsvm`.

Chen, B. and Huang, F. (2016). Semi-supervised Convolutional Networks for Translation Adaptation with Tiny Amount of In-domain Data. In *Proceedings of the 20th SIGNLL Conference on Computational Natural Language Learning, CoNLL 2016, Berlin, Germany, August 11-12, 2016*, pages 314–323.

Chen, B., Kuhn, R., and Foster, G. (2013). Vector Space Model for Adaptation in Statistical Machine Translation. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics*, pages 1285–1293, Sofia, Bulgaria.

Chen, B., Kuhn, R., and Foster, G. (2014). A Comparison of Mixture and Vector Space Techniques for Translation Model Adaptation. In *Proceedings of the Eleventh Conference of the Association for Machine Translation in the Americas (AMTA)*.

Chen, B., Kuhn, R., Foster, G., Cherry, C., and Huang, F. (2016). Bilingual Methods for Adaptive Training Data Selection for Machine Translation. In *AMTA 2016: Proceedings of the 12$^{th}$ Conference of the Association for Machine Translation in the Americas*, pages 93 – 106, Austin, Texas, USA.

Chen, B., Zhang, M., Aw, A., and Li, H. (2008). Exploiting N-best Hypotheses for SMT Self-enhancement. In *Proceedings of the 46th Annual Meeting of the Association for Computational Linguistics on Human Language Technologies: Short Papers*, HLT-Short '08, pages 157–160, Stroudsburg, PA, USA.

Chen, S. F. and Goodman, J. (1996). An Empirical Study of Smoothing Techniques for Language Modeling. In *34th Annual Meeting of the Association for Computational Linguistics, 24-27 June 1996, University of California, Santa Cruz, California, USA, Proceedings.*, pages 310–318.

Cho, K., van Merrienboer, B., Bahdanau, D., and Bengio, Y. (2014). On the Properties of Neural Machine Translation: Encoder–Decoder Approaches. In *Proceedings of SSST@EMNLP 2014, Eighth Workshop on Syntax, Semantics and Structure in Statistical Translation, Doha, Qatar, 25 October 2014*, pages 103–111.

Chung, J., Gülçehre, Ç., Cho, K., and Bengio, Y. (2014). Empirical Evaluation of Gated Recurrent Neural Networks on Sequence Modeling. *ArXiv e-prints*, http://arxiv.org/abs/1412.3555.

Collobert, R., Weston, J., Bottou, L., Karlen, M., Kavukcuoglu, K., and Kuksa, P. P. (2011). Natural Language Processing (Almost) from Scratch. *Journal of Machine Learning Research*, 12:2493–2537.

Cortes, C. and Vapnik, V. (1995). Support-Vector Networks. *Machine Learning*, 20(3):273–297.

Dempster, A. P., Laird, N. M., and Rubin, D. B. (1977). Maximum Likelihood from Incomplete Data via the Em Algorithm. *Journal of the Royal Statistical Society. Series B (Methodological)*, 39(1):pp. 1–38.

Dimitriadou, E., Hornik, K., Leisch, F., Meyer, D., Weingessel, A., and Leisch, M. F. (2009). Package 'e1071'. *R Software package, available at http://cran.rproject.org/web/packages/e1071/index.html*.

Dong, D., Wu, H., He, W., Yu, D., and Wang, H. (2015). Multi-Task Learning for Multiple Language Translation. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural*

*Language Processing of the Asian Federation of Natural Language Processing, ACL 2015, July 26-31, 2015, Beijing, China, Volume 1: Long Papers*, pages 1723–1732.

Duh, K., Neubig, G., Sudoh, K., and Tsukada, H. (2013). Adaptation Data Selection using Neural Language Models: Experiments in Machine Translation. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics*, pages 678–683, Sofia, Bulgaria.

Eidelman, V., Boyd-Graber, J., and Resnik, P. (2012). Topic Models for Dynamic Translation Model Adaptation. In *50th Annual Meeting of the Association for Computational Linguistics*, Jeju Island, Korea.

Foster, G., Chen, B., and Kuhn, R. (2013). Simulating Discriminative Training for Linear Mixture Adaptation in Statistical Machine Translation. *Proceedings of the XIV Machine Translation Summit*, pages 183–190.

Foster, G., Goutte, C., and Kuhn, R. (2010). Discriminative Instance Weighting for Domain Adaptation in Statistical Machine Translation. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, pages 451–459. Association for Computational Linguistics.

Foster, G. and Kuhn, R. (2007). Mixture-Model Adaptation for SMT. In *Proceedings of the Second Workshop on Statistical Machine Translation*, StatMT '07, pages 128–135, Stroudsburg, PA, USA.

Freitag, M. and Al-Onaizan, Y. (2016). Fast Domain Adaptation for Neural Machine Translation. *ArXiv e-prints*, http://arxiv.org/abs/1612.06897.

Galley, M. and Manning, C. D. (2008). A Simple and Effective Hierarchical Phrase Reordering Model. In *2008 Conference on Empirical Methods in Natural Language Processing, A meeting of SIGDAT, a Special Interest Group of the ACL*, pages 848–856, Honolulu, Hawaii, USA.

Gao, Q. and Vogel, S. (2008). Parallel Implementations of Word Alignment Tool. In *Software Engineering, Testing, and Quality Assurance for Natural Language Processing*, SETQA-NLP '08, pages 49–57, Stroudsburg, PA, USA.

Giménez, J. and Màrquez, L. (2007). Context-aware Discriminative Phrase Selection for Statistical Machine Translation. In *Proceedings of the Second Workshop on Statistical Machine Translation*, StatMT '07, pages 159–166.

Gruber, A., Weiss, Y., and Rosen-Zvi, M. (2007). Hidden Topic Markov Models. In *Proceedings of the Eleventh International Conference on Artificial Intelligence and Statistics, AISTATS 2007, San Juan, Puerto Rico, March 21-24, 2007*, volume 2 of *JMLR Proceedings*, pages 163–170. JMLR.org.

Haddow, B. and Koehn, P. (2012). Analysing the Effect of Out-of-Domain Data on SMT Systems. In *Proceedings of the Seventh Workshop on Statistical Machine Translation, WMT@NAACL-HLT 2012, June 7-8, 2012, Montréal, Canada*, pages 422–432.

Hasler, E., Haddow, B., and Koehn, P. (2012). Sparse Lexicalised Features and Topic Adaptation for SMT. In *2012 International Workshop on Spoken Language Translation, IWSLT 2012, Hong Kong, December 6-7, 2012*, pages 268–275.

Hasler, E. C. (2015). Dynamic Topic Adaptation for Improved Contextual Modelling in Statistical Machine Translation. *PhD thesis*.

He, W., He, Z., Wu, H., and Wang, H. (2016). Improved Neural Machine Translation with SMT Features. In *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence, February 12-17, 2016, Phoenix, Arizona, USA.*, pages 151–157.

Heafield, K. (2011). KenLM: Faster and Smaller Language Model Queries. In *Proceedings of the EMNLP 2011 Sixth Workshop on Statistical Machine Translation*, pages 187–197, Edinburgh, Scotland, United Kingdom.

Hinton, G. E., McClelland, J. L., and Rumelhart, D. E. (1986). Parallel Distributed Processing: Explorations in the Microstructure of Cognition, vol. 1. In *Distributed Representations*, pages 77–109. MIT Press, Cambridge, MA, USA.

Hochreiter, S. and Schmidhuber, J. (1997). LONG SHORT-TERM MEMORY. *Neural Computation*, 9(8):1735–1780.

Kim, Y. (2014). Convolutional Neural Networks for Sentence Classification. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing, EMNLP 2014, October 25-29, 2014, Doha, Qatar, A meeting of SIGDAT, a Special Interest Group of the ACL*, pages 1746–1751.

Kim, Y., Jernite, Y., Sontag, D., and Rush, A. M. (2016). Character-Aware Neural Language Models. In *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence, February 12-17, 2016, Phoenix, Arizona, USA.*, pages 2741–2749.

Kirchhoff, K. and Bilmes, J. (2014). Submodularity for Data Selection in Statistical Machine Translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing, EMNLP 2014, October 25-29, 2014, Doha, Qatar, A meeting of SIGDAT, a Special Interest Group of the ACL*, pages 131–141.

Kobus, C., Crego, J. M., and Senellart, J. (2016). Domain Control for Neural Machine Translation. *ArXiv e-prints*, http://arxiv.org/abs/1612.06140.

Koehn, P. (2004). Statistical Significance Tests for Machine Translation Evaluation. In *The 2004 Conference on Empirical Methods on Natural Language Processing*, pages 388–395, Barcelona, Spain.

Koehn, P. (2005). Europarl: A Parallel Corpus for Statistical Machine Translation. In *MT Summit X: The Tenth Machine Translation Summit*, pages 79–86, Phuket, Thailand.

Koehn, P., Axelrod, A., Mayne, A. B., Callison-Burch, C., Osborne, M., and Talbot, D. (2005). Edinburgh System Description for the 2005 IWSLT Speech Translation Evalua-

tion. In *International Workshop on Spoken Language Translation: Evaluation Campaign on Spoken Language Translation*, Pittsburgh, PA, USA.

Koehn, P., Hoang, H., Birch, A., Callison-Burch, C., Federico, M., Bertoldi, N., Cowan, B., Shen, W., Moran, C., Zens, R., et al. (2007). Moses: Open Source Toolkit for Statistical Machine Translation. In *Proceedings of the 45th Annual Meeting of the Association for Computational Linguistics on Interactive Poster and Demonstration Sessions*, pages 177–180, Prague, Czech Republic.

Koehn, P., Och, F. J., and Marcu, D. (2003). Statistical Phrase-Based Translation. In *Human Language Technology Conference of the North American Chapter of the Association for Computational Linguistics, HLT-NAACL 2003, Edmonton, Canada, May 27 - June 1, 2003*, pages 48–54.

Koehn, P. and Schroeder, J. (2007). Experiments in Domain Adaptation for Statistical Machine Translation. In *Proceedings of the Second Workshop on Statistical Machine Translation*, StatMT '07, pages 224–227, Stroudsburg, PA, USA.

Lü, Y., Huang, J., and Liu, Q. (2007). Improving Statistical Machine Translation Performance by Training Data Selection and Optimization. In *EMNLP-CoNLL 2007: Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 343–350, Prague, Czech Republic.

Luong, M., Le, Q. V., Sutskever, I., Vinyals, O., and Kaiser, L. (2015a). Multi-task Sequence to Sequence Learning. *ArXiv e-prints*, http://arxiv.org/abs/1511.06114.

Luong, T. and Manning, C. D. (2015). Stanford Neural Machine Translation Systems for Spoken Language Domains. In *Proceedings of the International Workshop on Spoken Language Translation*, pages 76–79, Da Nang, Vietnam.

Luong, T., Pham, H., and Manning, C. D. (2015b). Effective Approaches to Attention-based Neural Machine Translation. In *Proceedings of the 2015 Conference on Empirical*

*Methods in Natural Language Processing, EMNLP 2015, Lisbon, Portugal, September 17-21, 2015*, pages 1412–1421.

Mansour, S., Wuebker, J., and Ney, H. (2011). Combining Translation and Language Model Scoring for Domain-Specific Data Filtering. In *Proceedings of the Eighth International Workshop on Spoken Language Translation (IWSLT)*, pages 222–229, San Francisco, CA, USA.

Marcus, M. P., Santorini, B., and Marcinkiewicz, M. A. (1993). Building a Large Annotated Corpus of English: The Penn Treebank. *Computational Linguistics*, 19(2):313–330.

Meng, F., Lu, Z., Tu, Z., Li, H., and Liu, Q. (2015). Neural transformation machine: A new architecture for sequence-to-sequence learning. *ArXiv e-prints*, abs/1506.06442.

Mikolov, T., Chen, K., Corrado, G., and Dean, J. (2013a). Efficient Estimation of Word Representations in Vector Space. *ArXiv e-prints*, http://arxiv.org/abs/1301.3781.

Mikolov, T., Karafiát, M., Burget, L., Cernocký, J., and Khudanpur, S. (2010). Recurrent Neural Network Based Language Model. In *INTERSPEECH 2010, 11th Annual Conference of the International Speech Communication Association, Makuhari, Chiba, Japan, September 26-30, 2010*, pages 1045–1048.

Mikolov, T., Le, Q. V., and Sutskever, I. (2013b). Exploiting Similarities among Languages for Machine Translation. *ArXiv e-prints*, http://arxiv.org/abs/1309.4168.

Mikolov, T., Yih, W., and Zweig, G. (2013c). Linguistic Regularities in Continuous Space Word Representations. In *Human Language Technologies: Conference of the North American Chapter of the Association of Computational Linguistics, Proceedings, June 9-14, 2013, Westin Peachtree Plaza Hotel, Atlanta, Georgia, USA*, pages 746–751.

Mikolov, T. and Zweig, G. (2012). Context Dependent Recurrent Neural Network Language Model. In *2012 IEEE Spoken Language Technology Workshop (SLT), Miami, FL, USA, December 2-5, 2012*, pages 234–239.

Miller, G. A. (1995). WordNet: A Lexical Database for English. *Commun. ACM*, 38(11):39–41.

Mimno, D. M., Wallach, H. M., Naradowsky, J., Smith, D. A., and McCallum, A. (2009). Polylingual Topic Models. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing, EMNLP 2009, 6-7 August 2009, Singapore, A meeting of SIGDAT, a Special Interest Group of the ACL*, pages 880–889.

Moore, R. C. and Lewis, W. (2010). Intelligent Selection of Language Model Training Data. In *Proceedings of the ACL 2010 Conference Short Papers*, pages 220–224, Uppsala, Sweden.

Nakov, P. (2008). Improving English-Spanish Statistical Machine Translation: Experiments in Domain Adaptation, Sentence Paraphrasing, Tokenization, and Recasing. In *Proceedings of the Third Workshop on Statistical Machine Translation*, pages 147–150, Columbus, Ohio.

Neale, S., Gomes, L., Agirre, E., de Lacalle, O. L., and Branco, A. (2016). Word Sense-Aware Machine Translation: Including Senses as Contextual Features for Improved Translation Models. In *Proceedings of the Tenth International Conference on Language Resources and Evaluation LREC 2016, Portorož, Slovenia, May 23-28, 2016.*

Ni, X., Sun, J., Hu, J., and Chen, Z. (2011). Cross Lingual Text Classification by Mining Multilingual Topics from Wikipedia. In *Proceedings of the Forth International Conference on Web Search and Web Data Mining, WSDM 2011, Hong Kong, China, February 9-12, 2011*, pages 375–384.

Och, F. J. (2003). Minimum Error Rate Training in Statistical Machine Translation. In *Proceedings of the 41st Annual Meeting of the Association for Computational Linguistics, 7-12 July 2003, Sapporo Convention Center, Sapporo, Japan.*, pages 160–167.

Och, F. J. and Ney, H. (2002). Discriminative Training and Maximum Entropy Models for Statistical Machine Translation. In *Proceedings of the 40th Annual Meeting of the*

*Association for Computational Linguistics, July 6-12, 2002, Philadelphia, PA, USA.*, pages 295–302.

Och, F. J. and Ney, H. (2003). A Systematic Comparison of Various Statistical Alignment Models. *Computational Linguistics*, 29(1):19–51.

Papineni, K., Roukos, S., Ward, T., and Zhu, W.-J. (2002). BLEU: a Method for Automatic Evaluation of Machine Translation. In *Proceedings of the 40th Annual Meeting on Association for Computational Linguistics*, pages 311–318, Philadelphia, PA, USA.

Park, J., Liu, X., Gales, M. J. F., and Woodland, P. C. (2010). Improved Neural Network Based Language Modelling and Adaptation. In *INTERSPEECH 2010, 11th Annual Conference of the International Speech Communication Association, Makuhari, Chiba, Japan, September 26-30, 2010*, pages 1041–1044.

Pecina, P., Toral, A., Papavassiliou, V., Prokopidis, P., Van Genabith, J., and Athena, R. (2012). Domain Adaptation of Statistical Machine Translation using Web-Crawled Resources: A Case Study. In *Proceedings of the 16th Annual Conference of the European Association for Machine Translation*, pages 145–152.

Pennington, J., Socher, R., and Manning, C. D. (2014). GloVe: Global Vectors for Word Representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing, EMNLP 2014, October 25-29, 2014, Doha, Qatar, A meeting of SIGDAT, a Special Interest Group of the ACL*, pages 1532–1543.

Peris, Á., Chinea-Rios, M., and Casacuberta, F. (2016). Neural Networks Classifier for Data Selection in Statistical Machine Translation. *ArXiv e-prints*, http://arxiv.org/abs/1612.05555.

Qu, L., Ferraro, G., Zhou, L., Hou, W., Schneider, N., and Baldwin, T. (2015). Big Data Small Data, In Domain Out-of Domain, Known Word Unknown Word: The Impact of Word Representations on Sequence Labelling Tasks. In *Proceedings of the 19th Confer-*

*ence on Computational Natural Language Learning, CoNLL 2015, Beijing, China, July 30-31, 2015*, pages 83–93.

Robbins, H. and Monro, S. (1951). A Stochastic Approximation Method. *The Annals of Mathematical Statistics*, 22(3):400–407.

Rumelhart, D. E., Durbin, R., Golden, R., and Chauvin, Y. (1995). Backpropagation. In *Backpropagation*, chapter Backpropagation: The Basic Theory, pages 1–34. L. Erlbaum Associates Inc., Hillsdale, NJ, USA.

Rumelhart, D. E., Hinton, G. E., and Williams, R. J. (1986). Parallel Distributed Processing: Explorations in the Microstructure of Cognition, vol. 1. In *Learning Internal Representations by Error Propagation*, pages 318–362. MIT Press, Cambridge, MA, USA.

Rumelhart, D. E., Hinton, G. E., and Williams, R. J. (1988). Neurocomputing: Foundations of research. In *Neurocomputing: Foundations of Research*, chapter Learning Representations by Back-propagating Errors, pages 696–699. MIT Press.

Schuster, M. and Paliwal, K. K. (1997). Bidirectional Recurrent Neural Networks. *IEEE Transactions on Signal Processing*, 45(11):2673–2681.

Schwenk, H. (2008). Investigations on Large-Scale Lightly-Supervised Training for Statistical Machine Translation. In *IWSLT*, pages 182–189.

Sennrich, R. (2012). Perplexity Minimization for Translation Model Domain Adaptation in Statistical Machine Translation. In *Proceedings of the 13th Conference of the European Chapter of the Association for Computational Linguistics*, pages 539–549, Avignon, France.

Sennrich, R., Haddow, B., and Birch, A. (2016). Controlling Politeness in Neural Machine Translation via Side Constraints. In *NAACL HLT 2016, The 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, San Diego California, USA, June 12-17, 2016*, pages 35–40.

Servan, C., Crego, J. M., and Senellart, J. (2016). Domain Specialization: a Post-training Domain Adaptation for Neural Machine Translation. *ArXiv e-prints*, http://arxiv.org/abs/1612.06141.

Shi, Y., Larson, M., and Jonker, C. M. (2013). K-component Recurrent Neural Network Language Models using Curriculum Learning. In *Automatic Speech Recognition and Understanding (ASRU), 2013 IEEE Workshop on*, pages 1–6. IEEE.

Snover, M., Dorr, B., Schwartz, R., Makhoul, J., and Micciula, L. (2006). A Study of Translation Error Rate with Targeted Human Annotation. In *AMTA 2006: Proceedings of the 7$^{th}$ Conference of the Association for Machine Translation in the Americas: Visions for the Future of Machine Translation*, pages 223–231, Cambridge, Massachusetts, USA.

Stolcke, A., Konig, Y., and Weintraub, M. (1997). Explicit Word Error Minimization in N-best List Rescoring. In *Fifth European Conference on Speech Communication and Technology, EUROSPEECH 1997, Rhodes, Greece, September 22-25, 1997*.

Su, J., Wu, H., Wang, H., Chen, Y., Shi, X., Dong, H., and Liu, Q. (2012). Translation Model Adaptation for Statistical Machine Translation with Monolingual Topic Information. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Long Papers-Volume 1*, pages 459–468. Association for Computational Linguistics.

Su, J., Xiong, D., Liu, Y., Han, X., Lin, H., Yao, J., and Zhang, M. (2015). A Context-Aware Topic Model for Statistical Machine Translation. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing of the Asian Federation of Natural Language Processing, ACL 2015, July 26-31, 2015, Beijing, China, Volume 1: Long Papers*, pages 229–238.

Sutskever, I., Vinyals, O., and Le, Q. V. (2014). Sequence to Sequence Learning with Neural Networks. In *Advances in Neural Information Processing Systems 27: Annual Confer-*

*ence on Neural Information Processing Systems 2014, December 8-13 2014, Montreal, Quebec, Canada*, pages 3104–3112.

Ter-Sarkisov, A., Schwenk, H., Bougares, F., and Barrault, L. (2015). Incremental Adaptation Strategies for Neural Network Language Models. In *Proceedings of the 3rd Workshop on Continuous Vector Space Models and their Compositionality (CVSC), July 26-31, 2015, Beijing, China*, pages 48–56.

Tiedemann, J. (2012). Parallel data, tools and interfaces in OPUS. In *Proceedings of the Eighth International Conference on Language Resources and Evaluation, LREC 2012, Istanbul, Turkey, May 23-25, 2012*, pages 2214–2218.

Tillmann, C. (2004). A unigram orientation model for statistical machine translation. In *HLT-NAACL 2004: Human Language Technology Conference of the North American Chapter of the Association for Computational Linguistics, Proceedings of the Main Conference*, pages 101–104, Boston, Massachusetts, USA.

Toral, A. (2013). Hybrid Selection of Language Model Training Data Using Linguistic Information and Perplexity. In *Proceedings of the Second Workshop on Hybrid Approaches to Translation*, pages 8–12, Sofia, Bulgaria.

Tseng, H., Chang, P., Andrew, G., Jurafsky, D., and Manning, C. (2005). A Conditional Random Field Word Segmenter. In *In Fourth SIGHAN Workshop on Chinese Language Processing*.

Tu, Z., Lu, Z., Liu, Y., Liu, X., and Li, H. (2016). Modeling Coverage for Neural Machine Translation. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, ACL 2016, August 7-12, 2016, Berlin, Germany, Volume 1: Long Papers*.

Ueffing, N. (2006). Using Monolingual Source-Language Data to Improve MT Performance. In *IWSLT 2006: Proceedings of the 3rd International Workshop on Spoken Language Translation*, pages 174–181, Palulu Plaza, Kyoto, Japan.

Wang, M., Lu, Z., Li, H., Jiang, W., and Liu, Q. (2015). $gen$CNN: A Convolutional Architecture for Word Sequence Prediction. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing of the Asian Federation of Natural Language Processing, ACL 2015, July 26-31, 2015, Beijing, China, Volume 1: Long Papers*, pages 1567–1576.

Wang, W., Macherey, K., Macherey, W., Och, F., and Xu, P. (2012). Improved Domain Adaptation for Statistical Machine Translation. In *10th Biennial Conference of the Association for Machine Translation in the Americas (AMTA)*, San Diego, CA.

Xiao, X., Xiong, D., Zhang, M., Liu, Q., and Lin, S. (2012). A Topic Similarity Model for Hierarchical Phrase-based Translation. In *The 50th Annual Meeting of the Association for Computational Linguistics, Proceedings of the Conference, July 8-14, 2012, Jeju Island, Korea - Volume 1: Long Papers*, pages 750–758.

Xiong, D., Liu, Q., and Lin, S. (2006). Maximum Entropy Based Phrase Reordering Model for Statistical Machine Translation. In *ACL 2006, 21st International Conference on Computational Linguistics and 44th Annual Meeting of the Association for Computational Linguistics, Proceedings of the Conference, Sydney, Australia, 17-21 July 2006*.

Xiong, D., Zhang, M., and Wang, X. (2015). Topic-based Coherence Modeling for Statistical Machine Translation. *The IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 23(3):483–493.

Zeiler, M. D. (2012). ADADELTA: An Adaptive Learning Rate Method. *ArXiv e-prints*, http://arxiv.org/abs/1212.5701.

Zhang, J., Liu, S., Li, M., Zhou, M., and Zong, C. (2014a). Bilingually-constrained Phrase Embeddings for Machine Translation. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics, ACL 2014, June 22-27, 2014, Baltimore, MD, USA, Volume 1: Long Papers*, pages 111–121.

Zhang, M., Xiao, X., Xiong, D., and Liu, Q. (2014b). Topic-based Dissimilarity and Sensitivity Models for Translation Rule Selection. *Journal of Artificial Intelligence Research*, 50:1–30.