MACHINE LEARNING APPROACHES FOR DEALING WITH LIMITED BILINGUAL TRAINING DATA IN STATISTICAL MACHINE TRANSLATION

by

Gholamreza Haffari

BSc, Sharif University of Technology, 2000

MSc, Sharif University of Technology, 2002

A Thesis submitted in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy
in the School
of
Computing Science

SIMON FRASER UNIVERSITY

Fall 2009

APPROVAL

Name: Gholamreza Haffari

Degree: Doctor of Philosophy

Title of Thesis: Machine Learning Approaches for dealing with Limited Bilingual Training Data in Statistical Machine Translation

Examining Committee: Dr. Torsten Möller, Associate Professor, SFU
Chair

Dr. Anoop Sarkar, Senior Supervisor
Assistant Professor of Computing Science, SFU

Dr. Greg Mori, Supervisor
Assistant Professor of Computing Science, SFU

Dr. Shaojun Wang, Supervisor
Assistant Professor of Computer Science and Eng'g
Wright State University

Dr. Valentine Kabanets, Supervisor
Assistant Professor of Computing Science, SFU

Dr. Oliver Schulte, Internal Examiner
Associate Professor of Computing Science, SFU

Dr. Kevin Knight, External Examiner
Senior Research Scientist and Fellow, ISI
Research Associate Professor of Computer Science, USC

Date Approved: _____

Abstract

Statistical Machine Translation (SMT) models learn how to translate by examining a bilingual parallel corpus containing sentences aligned with their human-produced translations. However, high quality translation output is dependent on the availability of massive amounts of parallel text in the source and target languages. There are a large number of languages that are considered *low-density*, either because the population speaking the language is not very large, or even if millions of people speak the language, insufficient online resources are available in that language. This thesis covers machine learning approaches for dealing with such situations in statistical machine translation where the amount of available bilingual data is limited.

The problem of learning from insufficient labeled training data has been dealt with in machine learning community under two general frameworks: (i) Semi-supervised Learning, and (ii) Active Learning. The complex nature of machine translation task poses severe challenges to most of the algorithms developed in machine learning community for these two learning scenarios. In this thesis, I develop semi-supervised learning as well as active learning algorithms to deal with the shortage of bilingual training data for Statistical Machine Translation task, specific to cases where there is shortage of bilingual training data.

This dissertation provides two approaches, unified in what is called the *bootstrapping* framework, to this problem. I assume that we are given access to a monolingual corpus containing large number of sentences in the source language, in addition to a small or moderate sized bilingual corpus. The idea is to take advantage of this readily available monolingual data in building a better SMT model in an iterative manner: By *selecting* an important subset of these monolingual sentences, *preparing* their translations, and using them together with the original sentence pairs to *re-train* the SMT model. When preparing the translation of the selected sentences, if we use a human annotator, then the framework fits into the

Active Learning scenario in machine learning. Instead if we use the SMT system generated translations, then we get the *self-training* framework which fits into the semi-supervised learning scenario in machine learning. The key points that I address throughout this thesis are (1) how to choose the important sentences, (2) how to provide their translations (possibly with as little effort as possible), and (3) how to use the newly collected information in training the SMT model. As a result, we have a fully automatic and general method to improve the phrase-based SMT models for the situation where the amount of bilingual training data is small.

The success of self-training in SMT and many other NLP problems raises the question why self-training works. I investigate this question by giving a theoretical analysis of the self-training for *decision lists*. I provide objective functions which are motivated by information theory for the resulting semi-supervised learning algorithms. These objective functions provide us with: (1) Insights about why and when we should expect self-training to work well, and (2) Proofs of the convergence of their corresponding algorithms.

I dedicate this thesis to my love, Zohreh

Acknowledgments

I would like to express my deep gratitude to my advisor, Prof. Anoop Sarkar who has had a profound influence on my research and studies. He introduced me to the fascinating field of Natural Language Processing and Statistical Machine Translation, and taught me a great deal of valuable research skills. Beside being a knowledgeable teacher, he has also been a helpful friend with a strong personality which was always inspiring to me.

I would like to thank the members of my thesis committee: Prof. Greg Mori, Prof. Shaojun Wang, Prof. Oliver Schulte, Prof. Valentine Kabanets, and Prof. Kevin Knight. Prof. Mori has been always there for help in difficult times of my graduate life. Prof. Wang has been a great friend and collaborator, and his sharp thinking on mathematical formulation of research problems was very inspiring. I have also benefited a lot from comments and feedbacks provided by Prof. Schulte and Kabanets on my thesis. It has been a great honor to have Prof. Knight as the external examiner of my thesis defense. I am inspired by his accomplishments and breakthroughs.

During my graduate studies, I have had the great opportunity to collaborate with the following wonderful people: Yasemin Altun, Martin Ester, Mohsen Jamali, Yudong Liu, Greg Mori, Maxim Roy, Nicola Ueffing, Shaojun Wang, Yang Wang, Yee Whye Teh. I look forward to future collaborations with these great individuals.

I would like to thank the following faculties in Simon Fraser University and University of British Columbia for the wonderful courses I have had with them: Jason Bell, Martin Ester, Razvan Fetecau, Michael Friedlander, Arvind Gupta, Valentine Kabanets, Kevin Murphy, and Steven Ruuth. Moreover, I am grateful for the opportunity of meeting great researchers, including but not limited to: Phil Blunsom, Chris Callison-Burch, Colin Cherry, and Hal Daume. I greatly benefited from thought provoking discussions with them at the past conferences. I am thankful to the past and current members of Natural Language

Lab at the School of Computing Sciences at Simon Fraser University: Fred Popowich, Ann Clifton, Chris Demwell, Akshay Gattani, Ajeet Grewal, Bohua Gu, Yudong Liu, Mehdi Mostafavi-Kashani, Majid Razmara, Maxim Roy, Baskaran Sankaran, Dong Song, Milan Tofiloski, Zhongmin Shi, Yang (Wendy) Wang, and Winona Wu.

During my graduate life in Vancouver, I enjoyed the company of many dear friends: Hossein Ahmadi, Ali Amiri, Ehsan Amiri, Hamidreza Chitsaz, Javad Ebrahimi, Alireza Entezari, Alireza Fathi, Roozbeh Farahbod, Alireza Hadj-khodabakhshi, Mohsen Jamali, Mehdi Javadi, Roozbeh Ghaffari, Mohammad-mehdi Karimi, Pouya Karimian, Moslem Kazemi, Amir Keyvanloo, Majid Khabbazian, Yaroslav Litus, Hossein Masserat, Roozbeh Mottaghi, Hossein Mohtasham, Flavia Moser, Mohammad Nouroozi, Mani Ranjbar, Hamidreza Vaezi, Hamidreza Younesi, Habil Zare, and many others.

Finally, I would like to thank my family. My parents have endowed me with the curiosity about the world, and supported me through my life even from thousands miles away. Last but not the least, I am indebted to my lovely wife, Zohreh. Her support through the ups and downs of this journey has kept me sane and always happy. Without her, this thesis would have never been born. This is for you, Zohreh, my love!

Contents

Approval **ii**

Abstract **iii**

Dedication **v**

Acknowledgments **vi**

Contents **viii**

List of Tables **xii**

List of Figures **xiv**

1 Introduction **1**

1.1 Contributions of This Thesis 3

1.2 An Overview of This Thesis 4

2 Background **6**

2.1 Supervised Learning 6

2.2 Semi-supervised Learning 8

2.2.1 Theory 9

2.2.2 Expectation Maximization (EM) 10

2.2.3 The Yarowsky Algorithm 12

2.2.4 Co-training 13

2.3 Active Learning 15

2.3.1 Theory 15

2.3.2 Query by Committee . . . 17
2.3.3 Query based on Uncertainty . . . 18
2.3.4 Hierarchical Sampling . . . 19
2.4 Phrase-based Statistical Machine Translation . . . 20
2.4.1 Phrase-Pair Extraction . . . 23
2.4.1.1 IBM Model 1 and IBM Model 2 . . . 25
2.4.2 Learning the Weights of the Log-Linear Model . . . 27
2.4.3 Solving the Search Problem . . . 29
2.4.4 Evaluation . . . 32
2.4.5 Software Packages . . . 33
2.5 Summary . . . 35

3 An Analysis for Self-Training 36
3.1 The Motivation . . . 36
3.2 The Modified Yarowsky Algorithm . . . 38
3.2.1 The Objective Function . . . 40
3.3 Specific Yarowsky algorithms . . . 40
3.3.1 The DL-0 algorithm . . . 41
3.3.2 The DL-1 Algorithm . . . 42
3.3.2.1 Analysis of the DL-1 Algorithm . . . 43
3.4 Extensions of the DL-1 Algorithm . . . 46
3.5 The DL-2 Algorithms . . . 50
3.5.1 Analysis of DL-2-S Algorithm . . . 51
3.5.2 Analysis of the DL-2-ML Algorithm . . . 53
3.6 Summary . . . 54

4 Bootstrapping for Statistical Machine Translation 55
4.1 The Motivation . . . 55
4.2 The Framework . . . 56
4.2.1 The Algorithm . . . 56
4.2.2 The Training Function . . . 58
4.2.3 The Scoring Function . . . 59
4.2.4 The Selection Function . . . 60
4.2.5 Filtering the Training Data . . . 61

4.3 Experimental Results . . . 62
4.3.1 Transductive Experiments . . . 63
4.3.2 Inductive Experiments . . . 64
4.3.3 Analysis . . . 67
4.4 Related Work . . . 69
4.4.1 Results on Chinese–English . . . 71
4.4.1.1 Transductive Experiments . . . 71
4.4.1.2 Inductive Experiments . . . 73
4.5 Summary . . . 77

5 Active Learning for SMT: Single Language Pair 78
5.1 The Motivation . . . 78
5.2 The Algorithmic Framework . . . 79
5.3 Sentence Selection: Specific to Phrase-based SMT . . . 81
5.3.1 Considering Phrases (Geom-Phrase, Arith-Phrase) . . . 81
5.3.2 Considering n-grams (Geom n-gram, Arith n-gram) . . . 82
5.4 Sentence Selection: General Techniques . . . 83
5.4.1 Length of the Source Sentences (Length) . . . 83
5.4.2 Similarity to the Bilingual Training Data (Similarity) . . . 83
5.4.3 Confidence of Translations (Confidence) . . . 83
5.4.4 Feature Combination (Combined) . . . 84
5.4.5 Hierarchical Adaptive Sampling (HAS) . . . 84
5.4.6 Reverse Model (Reverse) . . . 86
5.5 Experimental Results . . . 86
5.5.1 Simulated Low Density Language Pairs . . . 87
5.5.2 Realistic Low Density Language Pair . . . 90
5.5.3 Domain Adaptation . . . 92
5.5.4 Analysis . . . 93
5.6 Sentence Selection for Phrase-based SMT: Revisited . . . 94
5.6.1 Latent Segmentation of OOV Fragments (Model 1) . . . 95
5.6.2 Separate Models for OOV and Regular Phrases (Model 2) . . . 97
5.6.3 Sentence Scoring . . . 98
5.7 Experimental Results . . . 98

5.7.1 Simulated Low Density Language Pairs . . . 99
5.7.2 Analysis . . . 101
5.8 Related Work . . . 102
5.9 Summary . . . 103

6 Active Learning for SMT: Multiple Language Pairs 104
6.1 The Motivation . . . 104
6.2 The Algorithmic Framework . . . 105
6.3 Sentence Selection for Multiple Language Pairs . . . 107
6.3.1 Alternating Selection . . . 107
6.3.2 Combined Ranking . . . 107
6.3.3 Disagreement Among the Translations . . . 108
6.3.3.1 Consensus Finding . . . 108
6.4 Experimental Results . . . 109
6.4.1 Simulated Multiple Language Pairs Setting . . . 110
6.5 Related Work . . . 110
6.6 Summary . . . 111

7 Conclusions 113
7.1 Future Directions . . . 114

Bibliography 116

List of Tables

3.1 Summary of the notation. 39

4.1 French–English EuroParl corpora 62
4.2 EuroParl results based on the mixture of phrase tables with $\lambda = .1$. 64
4.3 Translation examples after using transductive learning. 64
4.4 French–English corpora. 65
4.5 Statistics of the phrase tables trained on the genres of the NIST test corpora (see Section 4.4.1 for more details about the corpora). Scoring: confidence estimation, selection: threshold. 69
4.6 The *semi-supervised* examples are taken from sampling-based sentence selection with normalized scores. Lower-cased output, punctuation marks tokenized. 70
4.7 Chinese–English corpora. 71
4.8 Translation quality using an additional adapted phrase table trained on the dev/test sets for Chinese–English task. $^{+}$: 4 refs., *: 1 ref., and IS: importance sampling. Best results are printed in **boldface**. 72
4.9 Translation quality using an additional phrase table trained on monolingual Chinese news data for Chinese–English task. Threshold on confidence scores is used for selecting good translations. $^{+}$: 4 refs., *: 1 ref. 73
4.10 Translation quality using an additional phrase table trained on monolingual Chinese news data. Chinese–English test sets. **Bold:** best result, *italic:* significantly better than baseline. 75
4.11 Translation quality on Chinese–English eval-04 test set, by genre. Same experimental setup as Table 4.10. 76
4.12 Number of sentences added in each iteration. Chinese–English. 77

5.1 Specification of different data sets we will use in experiments. The target language is English, and the source languages are either French, German, Spanish, or Bangla. 87
5.2 Phrase-based utility selection is compared with random sentence selection baseline with respect to BLEU, wer (word *error* rate), per (position independent word *error* rate), and the total number of words in the selected sentences across three language pairs. 90
5.3 Comparison of methods in domain adaptation scenario. The bold numbers show statistically significant improvement with respect to the baseline. The text size refer to the number of words in the all selected sentences during in the total AL iterations. 92
5.4 Average number of English phrases per source language phrase, average length of the source language phrases, number of source language phrases, and number of phrase pairs which has been seen once in the phrase tables across three language pairs (French text taken from Hansard is abbreviated by 'Ha'). From top to bottom in each row, the numbers belong to: before starting AL, and after finishing AL based on 'Geom Phrase', 'Confidence', and 'Random'. 95
5.5 The BLEU score and the size (number of words) of the selected text for different sentence selection strategies after the *last* iteration of the AL loop on three translation tasks. 99
5.6 For regular/OOV phrases, the KL-divergence between the true distribution (P^*) and the estimated (P) or uniform ($unif$) distributions are shown, where: $KL(P^* \parallel P) := \sum_x P^*(x) \log \frac{P^*(x)}{P(x)}$. 101

6.1 Comparison of multilingual selection methods with WER (word error rate), PER (position independent WER). 95% confidence interval for WER numbers is 0.7 and for PER numbers is 0.5. **Bold:** best result, *italic:* significantly better. 112

List of Figures

1.1 High level overview of Bootstrapping. Active Learning and Semi-supervised Learning can be viewed in the Bootstrapping framework (see the text for explanation). 2

2.1 The input data points lie on the x axis, and the black/white boxes show their labels. The nodes of the tree correspond to a hierarchical clustering of the data (Dasgupta and Hsu, 2008). 19
2.2 w represents the decision boundary resulted from a typical AL method which operates based on querying points having the most labeling uncertainty. However, w^* represents one of the optimum classifiers having the least error rate (Dasgupta and Hsu, 2008). 20
2.3 The machine translation pyramid (Hutchins and Somers, 1992). 21
2.4 Example of a phrase-based translation from Farsi to English. 22
2.5 Example of an alignment between the Farsi and English parallel sentences mentioned in Figure 2.4. 24
2.6 (a,b) show the piecewise constant functions for the first and second source sentences. (c) shows the final piecewise constant function resulted from the sum of these two functions. The horizontal axis corresponds to real values for γ and the vertical axis corresponds to a measure of translation quality. . . 29
2.7 The structure of the search graph based on the operations used in the decoder. In each step, an English phrase is generated, words in the source Farsi phrase are covered (marked by *), and the cost so far is updated (for the example in Figure 2.4). 31

2.8 A hypothesis is expanded to new hypotheses, and they are placed into new stacks according to the number of source sentence words which they cover. This is for our running example in Figure 2.4 in which the source English sentence has seven words/tokens. 32

3.1 A bipartite graph representing the relationship between data points X and features F. Each data point has been connected to its features by undirected edges, and some data points are initially labeled. 42

4.1 Translation quality for importance sampling with full re-training on train100k (left) and train150k (right). EuroParl French–English task. 63
4.2 Translation quality using an additional phrase table trained on monolingual French data. The plot shows the performance of the different sentence selection and scoring schemes on in-domain and out-of-domain corpora. 66
4.3 This example illustrates how the phrase tables becomes focused to the test set. Initially there is a high ambiguity in translating the source phrase 'A B'. In translating the test set, most of the time the target phrase 'c d' is being chosen as the translation of the source phrase 'A B' by the language model (LM), based on the signals received from the context. As the result, the probabilities of the phrase pairs in the phrase table are adaptively tuned towards the test sentences. 67
4.4 This example illustrates how new phrase pairs can be composed when monolingual sentences are translated. Part of the monolingual sentence 'A B C D E' is translated to 'a b c d e', and new phrase pairs are constructed depending on how to break these two sentence fragments. 68
4.5 Translation quality using an additional phrase table trained on monolingual Chinese news data. Chinese–English development set. 74

5.1 Adaptively sampling the sentences while constructing a hierarchical clustering of D_u. 85

5.2 BLEU scores for different sentence selection strategies per iteration of the AL algorithm. Plots in the left show the performance of sentence selection methods which depend on the target language in addition to the source language (hierarchical adaptive sampling, reverse model, decoder confidence, average and geometric phrase-based score), and plots in the right show methods which are independent of the target language (geometric 4-gram and 1-gram, similarity to L, and random sentence selection baseline). 88
5.3 BLEU scores for different sentence selection strategies per iteration of the AL algorithm. Plots in the left show the performance of sentence selection methods which depend on the target language in addition to the source language (hierarchical adaptive sampling, reverse model, decoder confidence, average and geometric phrase-based score), and plots in the right show methods which are independent of the target language (geometric 4-gram and 1-gram, similarity to L, and random sentence selection baseline). 89
5.4 Improving Bangla to English translation performance using active learning. . 91
5.5 Performance of different sentence selection methods for domain adaptation scenario. 93
5.6 Number of words in domain adaptation scenario. 94
5.7 The given sentence in (b) is segmented, based on the source side phrases extracted from the phrase table in (a), to yield regular phrases and OOV segment. The table in (c) shows the potential phrases extracted from the OOV segment "go to school" and their expected counts (denoted by $\overline{\text{count}}$) where the maximum length for the potential phrases is set to 2. In the example, "go to school" has 3 segmentations with maximum phrase length 2: *(go)(to school), (go to)(school), (go)(to)(school).* 96
5.8 The performance of different sentence selection strategies as the iteration of AL loop goes on for three translation tasks. Plots show the performance of sentence selection methods for single language pair in Section 5.6 compared to the "Geom Phrase" in Section 5.3 and random sentence selection baseline. 100

5.9 The log-log Zipf plots representing the true and estimated probabilities of a (source) *phrase* vs the rank of that phrase in the German to English translation task. The plots for the Spanish to English and French to English tasks are also similar to the above plots, and confirm a power law behavior in the true phrasal distributions. 102

6.1 Random sentence selection baseline using self-training and co-training (Germanic languages to English). The weights of the consensus scoring model (6.2) are trained *after* we have added the first 500 sentences, hence there is a jump in the performance of the co-training in the next iteration (where totally 1000 sentences are added). 109

6.2 The left/right plot show the performance of our AL methods for multilingual setting combined with self-training/co-training. The sentence selection methods from Sec. 6.3 are compared with random sentence selection baseline. The top plots correspond to Danish-German-Dutch-Swedish to English, and the bottom plots correspond to French-Spanish-Italian-Portuguese to English. 111

List of Algorithms

1 EM 11
2 Bootstrapping algorithm: classifier version 12
3 Co-training 14
4 The modified Yarowsky algorithm 39
5 Majority-Majority 48
6 Bootstrapping Algorithm for SMT 57
7 AL-SMT: Single Language Pair 81
8 Hierarchical-Adaptive-Sampling 86
9 AL-SMT: Multiple Language Pairs 107

Chapter 1

Introduction

Having a machine learn how to translate from a source language to a target language, Automatic Machine Translation, is one of the holy grails of Artificial Intelligence. Statistical machine translation (SMT) systems, which rely on the *data driven* approach, have made great improvements in translation quality. SMT casts the translation from a source language to a target language as a machine learning problem. SMT algorithms learn the translation process by examining a bilingual parallel corpus containing human-produced translations. However, high quality translation output is dependent on the availability of massive amounts of parallel text in the source and target languages. There are a large number of languages that are considered *low-density*, either because the population speaking the language is not very large, or even if millions of people speak the language, insufficient online resources are available in that language. This thesis covers machine learning approaches for dealing with such situations in statistical machine translation where the amount of available bilingual data is limited.

The problem of learning from insufficient labeled training data has been dealt with in machine learning community under two general frameworks: (i) Semi-supervised Learning, and (ii) Active Learning. The goal of semi-supervised learning is to take advantage of abundant and cheap unlabeled data, together with labeled data, to build a high quality mapping from examples (the input space) to labels (the output space). On the other hand, the goal of active learning is to reduce the amount of labeled data required to learn a high quality mapping by querying the user to label the most informative examples so that the mapping is learned with fewer training examples.

The complex nature of machine translation task poses severe challenges to most of

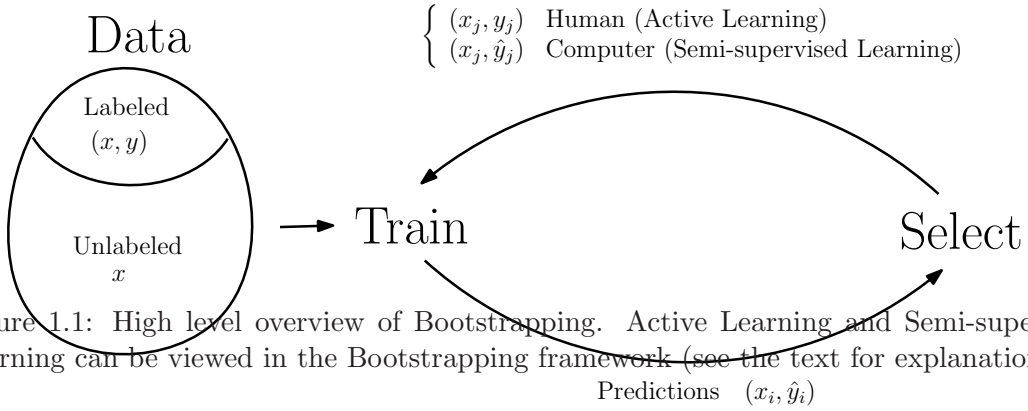


Figure 1.1: High level overview of Bootstrapping. Active Learning and Semi-supervised Learning can be viewed in the Bootstrapping framework (see the text for explanation).

the algorithms developed in machine learning community for these two learning scenarios. In this thesis, I develop semi-supervised learning as well as active learning algorithms to deal with the shortage of bilingual training data for the SMT task. I assume that we are given access to a monolingual corpus containing a large number of sentences in the source language, in addition to a small bilingual corpus. The idea is to take advantage of this readily available monolingual data in building a better SMT model using bootstrapping-style methods: By *selecting* an important subset of these monolingual sentences, *preparing* their translations, and using them together with the original sentence pairs to *re-train* the SMT model effectively (see Figure 1.1). When preparing the translation of the selected sentences, if we use a human annotator, then the framework fits into the Active Learning scenario in machine learning. Instead if we use the (system generated) translations produced by the SMT model itself, then the framework fits into the semi-supervised learning scenario in machine learning. The key points identified in this thesis: are: (1) how to choose the important sentences, (2) how to provide their translations (possibly with as little effort as possible), and (3) how to use the newly collected information in re-training the SMT model. The main contribution of this thesis is to present Bootstrapping-style algorithms for SMT

based on the three points mentioned above, and to introduce new methods and solutions for each of these three points.

The shortage of bilingual training data is not specific to low density language pairs. It may also happen because of change in the domains of training and test data. For example, consider a case where we have bilingual training sentences from the News domain, and we want to build an SMT system to translate text from the Politics domain. A statistical translation system can be *improved* and/or *adapted* by incorporating new training data in the form of parallel text in cases where there is shortage of bilingual training data. The methods that we introduce in this thesis cover both cases.

The basic question behind this work is the value of unlabeled data in learning a mapping from the input space to the output space. Intuitively, it seems that having more (unlabeled) data should be useful; however it is not necessarily true for all situations. For active learning, we see in Section 2.3.4 that *sampling bias* can occur if care is not taken when selecting data points for annotation. This may increase the precision but decrease the recall of the learned classifier (Schütze, Velipasaoglu, and Pedersen, 2006). For the semi-supervised learning, there are cases in the literature which have reported that adding unlabeled data to the labeled training data has caused performance degradation of the learned classifier (Nigam et al., 2000; Merialdo, 1993). These issues show that coming up with successful active/semi-supervised learning techniques for the complex task of SMT is challenging.

1.1 Contributions of This Thesis

The primary contributions of this thesis can be summarized as follows:

- I present a formal analysis of a bootstrapping style algorithm commonly used in NLP, known as the Yarowsky algorithm, for semi-supervised learning using Decision Lists (DL) for multiclass classification. This algorithm has a specific prescription for each of the three key points of the Bootstrapping framework: instance selection, instance annotation, and retraining of the underlying DL model (Figure 1.1). I show the proof of convergence for several variants of the Yarowsky algorithm by showing that these algorithms indeed optimize reasonable objective functions. Furthermore, these objective functions may shed light to the situations where we should expect these variants to perform well, and help us to make connection with other approaches to

semi-supervised learning suggested in the literature (consequently, we can have a unified view to these seemingly different approaches). These results are based on (Haffari and Sarkar, 2007).

- I propose bootstrapping style algorithms to improve/adapt state of the art phrase-based SMT models in the semi-supervised/transdutive learning scenario. Based on extensive experimental evaluations, I show the effectiveness of the proposed methods not only for the semi-supervised/transductive learning setting, but also for the domain adaptation setting. These results are based on (Ueffing, Haffari, and Sarkar, 2007a; Ueffing, Haffari, and Sarkar, 2007b; Ueffing, Haffari, and Sarkar, 2008).

- I devise effective sentence selection strategies for improving/adapting state of the art phrase-based SMT models in the active learning scenario. Furthermore, I define a novel active learning setting in which a new language is added to an existing multilingual parallel corpus, and I devise effective sentence selection strategies for this scenario. These results are based on (Haffari and Sarkar, 2009; Haffari, Roy, and Sarkar, 2009).

1.2 An Overview of This Thesis

The chapters in this thesis are organized as follows:

Chapter 2 introduces relevant background from the machine learning and phrase-based statistical machine translation literature. From the machine learning perspective, this chapter gives an introduction to supervised/semi-supervised/active learning. For semi-supervised/active learning scenarios, we start by looking at some core theoretical issues, and then review some algorithmic approaches to realize these theoretical viewpoints. For phrase-based SMT, this chapter provided background on how the model's feature functions and parameters are learned, how the model's output is evaluated, and which tools and software packages are going to be used in the experiments of the later chapters.

Chapter 3 presents my formal analysis of a self-training algorithm, known as the Yarowsky algorithm in computational linguistics literature. The Yarowsky algorithm (Yarowsky, 1995) is a rule-based semi-supervised learning algorithm that has been successfully applied to many problems in computational linguistics. The algorithm was not mathematically well understood until (Abney, 2004) which analyzed some specific variants of the algorithm, and

also proposed some new algorithms for bootstrapping. In this chapter, I extend Abney's work and present the analysis for some of his proposed algorithms as well as a new class of objective functions for rule-based semi-supervised learning. Our semi-supervised SMT approach in Chapter 4 is inspired by the Yarowsky algorithm.

Chapter 4 explores the use of bootstrapping (self-training) methods for the effective use of monolingual data from the source language in order to improve translation quality. I present transductive and semi-supervised methods for the effective use of monolingual data in the source language to improve translation quality for phrase-based SMT models. I present detailed experimental evaluations on the French-English and Chinese–English tasks, and show that the proposed algorithms improve the translation quality of the best known machine translation systems on large scale translation tasks.

Chapter 5 presents my active learning algorithms for SMT for the single language-pair case. It provides the first serious study of active learning for SMT, and introduces new highly effective sentence selection methods that improve AL for phrase-based SMT. I use active learning to improve the quality of a phrase-based SMT system when translating from a source language to a target language, given a large set of monolingual source sentences and a small/moderate set of bilingual sentences. I show significant improvements in translation quality compared to the random sentence selection baseline, when test and training data are taken from the same or different domains.

Chapter 6 introduces the active learning task of adding a new language to an existing multilingual set of parallel text and constructing high quality MT systems, from each language in the collection into this new target language I show that adding a new language using active learning to the EuroParl corpus provides a significant improvement compared to a random sentence selection baseline. I also provide new highly effective sentence selection methods that improve AL for phrase-based SMT in the multilingual setting.

Chapter 7 summarizes the specific results achieved in the previous chapters on the analysis of self-training and its application to SMT, as well as active learning for SMT. I then discuss both theoretical and practical avenues for future research.

Chapter 2

Background

The goal in this thesis is to deal with those situations where there is shortage of bilingual training data in Statistical Machine Translation. The problem of learning from insufficient labeled training data has been dealt with in the machine learning community under two general frameworks: (i) Semi-supervised Learning, and (ii) Active Learning. In this chapter, I give the necessary background on Semi-supervised Learning, Active Learning, and Statistical phrase-based Machine Translation. But first, let us start from Supervised learning and introduce some notation that is used in this thesis.

2.1 Supervised Learning

Having a training sample, the aim is to find a *good* function $h : \mathcal{X} \rightarrow \mathcal{Y}$ from a set of possible functions $\mathcal{H}$, which maps input instances $x \in \mathcal{X}$ to output $y \in \mathcal{Y}$ correctly. More formally, assume there is a loss function $loss(y, y^t, x)$ which gives the cost of assigning a (predicted) value y instead of the true value y^t to an input instance x. The ultimate goal of supervised/semi-supervised/active[1] learning is to produce the best function which has the minimum possible expected loss or *risk*:

$$\arg\min_{h \in \mathcal{H}} E_{(x',y') \sim P(x,y)}[loss(h(x'), y', x')] \tag{2.1}$$

The *loss* function specifies how bad is the prediction $h(x')$ compared to the truth y' for the input data point x'. For example, the performance measure that we would like to *maximize*

[1] For active learning, there is also the issue of annotation cost.

for the SMT task is called the BLEU score (see Section 2.4). In this case, we can consider "1 - BLEU" as the *loss* function.

The difference between fully supervised and semi-supervised learning is that, in the training sample for the semi-supervised learning we have access to labeled data $D_l := \{(x_1, y_1), ..., (x_l, y_l)\}$ as well as unlabeled data $D_u = \{x_{l+1}, ..., x_{l+u}\}$ but for supervised learning we only have access to labeled data. Indeed the labels $y_{l+1}, ..., y_{l+u}$ of unlabeled instances are hidden (latent). Furthermore, the difference between active learning and semi-supervised learning is that in active learning we are allowed to query the label of some data points from an oracle (e.g. a human). The active learning scenario we consider in this thesis is called the *pool-based* active learning scenario: We are given a pool of unlabeled instances and the goal is to ask the annotation of a subset of most informative instances from the oracle. This is explored further in Section 2.3.

Often it is assumed that there is a fixed but unknown probability distribution $P_{x,y} : \mathcal{X} \times \mathcal{Y} \to [0, 1]$ from which labeled instances are drawn independently; the so-called i.i.d. (independently and identically distributed) assumption. Moreover, it is assumed unlabeled instances are drawn i.i.d. from the marginal distribution $P_x(x') = \sum_{y' \in \mathcal{Y}} P_{x,y}(x', y')$. For simplicity, we will write $P(x', y')$ instead of $P_{x,y}(x', y')$ and $P(x')$ instead of $P_x(x')$.

Supervised learning problems and algorithms can be analyzed in the Probably Approximately Correct (PAC) model. Fix the target (true) hypothesis h^* (which is the subject of learning) in a hypothesis space $\mathcal{H}$. We say the learning algorithm A, PAC-learns the target hypothesis if it can produce a hypothesis $h \in \mathcal{H}$ such that

$$Pr\Big(A \text{ outputs } h \text{ such that } E_{x' \sim P(x)}[loss_1(h(x'), h^*(x'), x')] \leq \epsilon\Big) \geq 1 - \delta$$

where $loss_1(y, y', x)$ is defined to incur a loss of 1 if $y \neq y'$, and 0 otherwise. Note that $E[loss_1(h^*(x'), h(x'), x')] = P(h^*(x') \neq h(x'))$ is the probability of making an error with the output hypothesis h. So algorithm A, PAC-learns the target hypothesis h^* if with high confidence (with probability more than $1 - \delta$) it can output an approximately correct (with error less than ϵ) hypothesis. A target hypothesis is said to be PAC-learnable if there exists an algorithm A which can PAC-learn it. I use the PAC-model viewpoint to give high level theoretical perspectives to semi-supervised/active learning in the later sections of this chapter.

2.2 Semi-supervised Learning

The problem of learning in the presence of labeled and unlabeled data, also known as Semi-Supervised Learning, has recently attracted a great deal of attention; see for example (Zhu and Goldberg, 2009), and the references therein. In many real life machine learning tasks, providing a labeled training set is costly whereas there is a lot of unlabeled data which can be obtained easily. For example, in part-of-speech tagging it takes a lot of time to annotate sentences with their part of speech tags but a huge amount of unannotated sentences can easily be provided by crawling the web.

The natural question is the possibility of taking advantage of unlabeled data in order to construct more accurate classifiers. The focus of research on semi-supervised learning in machine learning community has been twofold: on one hand analyzing the situations in which unlabeled data can be useful, and on the other hand, providing learning algorithms which can actually do this. These proposed learning algorithms often make very different assumptions about the problem.

There exist several approaches to semi-supervised learning, and we limit our scope of attention to the *classifier based* methods as opposed to *data based* methods. Roughly speaking, classifier based approaches start from an initial classifier (or an ensemble of classifiers), and then iteratively try to produce new training data by injecting (noisy) labels to unlabeled data, and iteratively learn new classifier(s). On the other hand, data based methods try to reveal the geometry (if there is any) of the distribution of data[2] with the help of unlabeled data, make some assumptions about a good function class with respect to the geometry, and then find the best function in this function class.

This section is organized as follow. First we review a theoretical perspective to semi-supervised learning problem, and then see three popular algorithmic approaches to it: (i) Expectation Maximization (EM), (ii) The Yarowsky algorithm, and (iii) Co-training. Note that EM and the Yarowsky algorithm are *self-training* style algorithms, i.e. the prediction of the trained model is used to retrain the model again in an iterative manner. Whereas in Co-Training, we train *two* models and the output of each one is used to retrain the other one in an iterative manner.

[2]Like a *manifold* on which input data might be sitting.

2.2.1 Theory

The standard PAC framework is distribution free: The results do not depend on the input distribution. The PAC model assumes that the distribution of the data collected in the test time is the same as that in the training time; in fact this is the main point which connects training to testing, and makes learning possible in this framework. Loosely speaking, the PAC framework is only suitable for analyzing *supervised* learning algorithms since it does not take into account the input distribution $P(x)$. It is a good idea to change the standard PAC framework in such a way that the new framework enables us to think about semi-supervised learning problems and algorithms, since unlabeled data gives information about $P(x)$. This attempt has been done in (Balcan and Blum, 2005; Chapelle, Scholkopf, and Zien, 2006) by incorporating the distribution over the input space $\mathcal{X}$ into the PAC framework via the notion of *compatibility* function χ. The resulting framework is called "The Augmented PAC Model".

In the standard PAC framework the only *a priori* assumption is that the output hypothesis must be chosen in some hypothesis space $\mathcal{H}$. The new framework goes further and assumes that the final hypothesis must be compatible with the input distribution based on the compatibility function $\chi : \mathcal{H} \times \mathcal{X} \to [0, 1]$. The function χ gets an input instance $x \in \mathcal{X}$ and a hypothesis $h \in \mathcal{H}$, and produces a number which shows how reasonable the hypothesis is w.r.t the input x. Furthermore, the quantity $\chi(h, P) = E_{x \sim p(x)}[\chi(h, x)]$ represents the plausibility of the hypothesis h w.r.t the distribution over the input space. The amount of incompatibility $1 - \chi(h, P)$ can be considered as the unlabeled error rate of the hypothesis h, i.e. how likely a priori we think that h is *not* the target hypothesis.

As an example suppose training data lives in some Euclidean space $\mathbb{R}^d$ and consider the class of linear separators as the hypothesis space. The prior belief might be that the target function should separate the two classes of data points with the margin at least γ. So the compatibility $\chi(h, x)$ is one if the distance of x to the hyperplane $h^T \cdot z = 0$ is greater than γ and is zero otherwise (here the hypothesis $h \in \mathbb{R}^d$ is just a vector of coefficients). Consequently, the incompatibility $1-\chi(h, P)$ shows how much mass of the input distribution is placed on the space within the distance γ from the hyperplane.

Intuitively unlabeled data helps to reduce the size of hypothesis space by just keeping the plausible ones, and labeled data is used to choose a good hypothesis from these plausible

hypotheses. Denote by $\mathcal{H}_{P,\chi}(\tau)$ the subset of the concept class $\mathcal{H}$ which have incompatibility less than τ. Often the input distribution is unknown so the *empirical* measure of compatibility (or incompatibility) $\hat{\chi}(h,S) = \frac{1}{l+u}\sum_1^{l+u}\chi(h,x_i)$ is calculated based on the available sample S; therefore $\mathcal{H}_{S,\chi}(\tau)$ denotes the subset of the hypothesis class $\mathcal{H}$ which have the empirical incompatibility less than τ. The sample complexity results (i.e. how much labeled and unlabeled data we need to build a good classifier) are expressed based on some measure of complexity of $\mathcal{H}_{S,\chi}(\tau)$ or $\mathcal{H}_{P,\chi}(\tau)$ as well as other quantities such as the allowed approximation error ϵ and confidence parameter δ of the output hypothesis. Several theorems are stated in (Balcan and Blum, 2005), and the conclusion is that unlabeled data is helpful when (i) the target hypothesis is highly compatible with the notion of compatibility χ, (ii) not many hypotheses in $\mathcal{H}$ have a low unlabeled error rate (which depends on the input distribution P as well as χ), and (iii) there is enough unlabeled data to approximate well the empirical unlabeled error rate.

The generative model approach to semi-supervised learning can be fit into this framework (Chapelle, Scholkopf, and Zien, 2006) as an extreme case. Usually generative model based approaches assume that the marginal distribution over the input space is an identifiable mixture:

$$P(x|\theta) = \sum_j \alpha_j P(x|\theta_j)$$

where $\sum_j \alpha_j = 1$ and all mixture components belong to a known parametric family of models. Moreover each mixture component is labeled with only one class. Therefore the strategy is to first discover these mixture components based on unlabeled data, and then assign labels to components based on labeled data. The scoring function which is used in the first phase (to find the mixture components) can be considered as the degree of compatibility of hypotheses. As an example, the likelihood score can be used as the compatibility function $\chi(h_\theta, D) = E_{x\sim P(x)}[\log P(x|\theta)]$ where h_θ is the resulting classifier based on $P(x|\theta)$.

2.2.2 Expectation Maximization (EM)

Perhaps Expectation Maximization is the most common method used to deal with incomplete data. In the setting of semi-supervised learning problem, missing data are the latent labels of unlabeled data. We assume a joint probability distribution over the space of input instances and their labels $P(x,y)$. Then we choose a parametric model $P(x,y|\theta)$ for modeling the input-output joint probability distribution and try to find the best $\hat{\theta}$ by maximizing

the incomplete log likelihood of the sample:

$$\mathcal{L}(\theta) = \sum_{i=1}^{l} \log P(x_i, y_i|\theta) + \sum_{j=l+1}^{l+u} \log P(x_j|\theta)$$

$$\hat{\theta} = \arg\max_{\theta} \mathcal{L}(\theta)$$

where marginal distribution $P(x|\theta) = \sum_{y \in \mathcal{Y}} P(x, y|\theta)$.

The idea in EM is to optimize the incomplete log likelihood of the observed data $\mathcal{L}(\theta)$ via the iteration of the two phases in Algorithm 1.

Algorithm 1 EM

1. **Expectation** step: Given the current model parameters and observed data, compute the probability distribution over the unobserved data $P(y_{l+1}, ..., y_{l+u}|D_l, D_u, \hat{\theta}^t)$.
2. **Maximization** step: Search for the optimum model parameters which maximizes the log likelihood of the *current complete* data $\hat{\theta}^{t+1} = \arg\max_\theta \sum_{i=1}^{l} \log P(x_i, y_i|\theta) + \sum_{j=l+1}^{l+u} E_{P(y_j|D_l, D_u, \hat{\theta}^t)}[\log P(x_j, y_j|\theta)]$.

Any optimization algorithm can be used to optimize the incomplete log likelihood but EM is widely used because it has an intuitive interpretation. In the first step it makes *inference* about the (distribution over) latent labels, and in the second step, it maximizes the parameters of the model based on the log likelihood of the *current complete* data. Intuitively speaking, for each unlabeled data point and each value of its latent label, we put a weighted labeled instance in the current complete data, where the weight is equal to the probability of the label for the unlabeled data point.

As noted in (Seeger, 2000) applying EM on $P(x, y)$ might be dangerous for the task of predicting y from x. EM tries to find the model which best fits $P(x, y)$. Since the fitness measure is not based on the discriminative ability (or power) of x in predicting y, i.e. $P(y|x)$, the selected model might not be a good predictor of class labels. Furthermore, the log-likelihood function has many local optima and it is highly probable that EM gets stuck in a local optimum. In contrast to EM, we will see that the variants of the Yarowsky algorithm do not waste their prediction power on learning $P(x)$. In contrast, they optimize objective functions which force them to become as close as possible to $P(y|x)$.

2.2.3 The Yarowsky Algorithm

The Yarowsky algorithm (Yarowsky, 1995) was proposed in the context of the word-sense disambiguation task. The model was a simple decision list classifier $h(x)$ where the class label $y \in \mathcal{Y}$ is predicted based on the *single* most likely feature extracted from the input x. The pseudo-code for this algorithm is shown in Algorithm 2.

Algorithm 2 Bootstrapping algorithm: classifier version

```
1: Initially set pool of training data Λ to D_l
2: repeat
3:    Train classifier h based on the current pool of training data Λ
4:    reset pool of training data Λ to D_l
5:    for each instance x in the unlabeled data do
6:       Construct prediction distribution π_x for instance x where π_x(y) shows the probability that classifier h predicts label y for x.
7:       Set ŷ := arg max_{y∈Y} π_x(y)
8:       if π_x(ŷ) > ζ then
9:          Add (x, ŷ) to Λ (do not remove x from unlabeled data)
10:      end if
11:   end for
12: until some condition is met
```

The key idea of the Yarowsky algorithm is to use an initial classifier which is built from the seed data, and has high precision but low recall (it cannot predict a label for most unlabeled examples). The reason for low recall is that if a feature extracted from input x has not been observed with any class label, this event is not assigned a smoothed probability estimate (unlike the common strategy in supervised learning). Hence for instances where all the features are events of this type, the classifier will not assign a label; it is as if the classifier assigns a special "I do not know" label $\perp$ to such instances. However, even if a single feature extracted from x is observed with a class label in the labeling step, this information can be used to make a prediction for future examples by the decision list classifier (which only uses the single most likely feature to predict the class label). This classifier is then used to label the unlabeled data and those examples labeled with high confidence (above a threshold) are used along with the labeled data to train a new classifier. This process was repeated iteratively until no new labels could be found for the unlabeled data set. Each iteration can be seen as increasing the recall of the classifier at the expense of decreasing its precision. In each iteration the classifier is able to provide labels for larger and larger subsets of the

unlabeled data.

We can think of some modifications of the original Yarowsky algorithm. For example, the labels assigned to the unlabeled data can be sticky. At the extreme case, we can set the convention that these labels cannot change during later iterations of the algorithm (removing line 4 in Algorithm 2, and changing line 9 to discard x from D_u after labeling it). But less harshly, we can set the convention that those instances which have been assigned a label cannot become unlabeled in the later iterations but their label may change (removing line 4 in Algorithm 2). This is the convention used in (Abney, 2004) and I will use it in the variants of the Yarowsky algorithm in Chapter 3. I will present a formal analysis of the resulting variants of the Yarowsky algorithm in Chapter 3, and show the objective functions being optimized by these algorithms during their iterations.

2.2.4 Co-training

Co-training (Blum and Mitchell, 1998) is one of the first algorithms proposed for semi-supervised learning, and analyzed thereafter by many papers, including (Balcan, Blum, and Yang, 2004; Dasgupta, Littman, and McAllester, 2001). Co-training assumes that there are two sets of features which are conditionally independent given the labels, and each of which is enough to build a good classifier[3]. Each feature set is called a *view*, and Co-training learns two classifiers where each of them corresponds to a single view.

Co-training is illustrated in Algorithm 3. In each iteration, the algorithm selects a subset of unlabeled data (N_p instances in D_u which are labeled positive, and N_n instances in D_u which are labeled negative) whose labels are assigned with high confidence by the current classifiers, and add them to the pool of labeled training data. The number of selected positive and negative instances is proportional to their ratio in the labeled sample. Then the classifiers are retrained based on this expanded training data. This process continues till it converges.

The conditional independence assumption means that if we label an unlabeled instance in one view, the resulting labeled instance is a random instance in the other view. As a result each view provides random labeled instances for the opposite view with some slight *noise*. We can use any machinery for building classifiers which works based on the (weak) i.i.d.

[3]This method is naturally suited in domains where the information about each instance comes from two separate sources or sensors.

Algorithm 3 Co-training

1: Initially, train classifiers h^1 and h^2 on labeled data D_l.
2: **repeat**
3: **for** each view w=1..2 **do**
4: Remove N_p elements with greatest $h^w(x_u)$ from D_u and add $(x_u, +1)$ to D_l.
5: **end for**
6: **for** each view w=1..2 **do**
7: Remove N_n elements with smallest $h^w(x_u)$ from D_u and add $(x_u, -1)$ to D_l.
8: **end for**
9: Retrain h^1 and h^2 using the updated D_l
10: **until** D_u becomes empty

assumption in collecting training instances. Furthermore, the conditional independence of the two views reduces the chance that both classifiers (with high confidence) label an unlabeled instance *wrongly* at the same time.

Co-training belongs to a broader class of semi-supervised learning algorithms which can be called "Agreement Maximization" algorithms. In co-training, each learner (corresponding to each view) gradually labels *some* unlabeled data which is mostly confident about their labels, adds them to the pool of training data, and then uses them in the next iteration to *inform* the other learner about its *opinion*. In other words, unlabeled data is a platform for the two learners to communicate their opinions. After a while, as a side effect of the algorithm, two learners are motivated to *agree* on unlabeled data. What if we make the agreement of the learners as the explicit goal of the algorithm? Does this help to learn from unlabeled data? In (Leskes, 2005; Collins and Singer, 1999) this outlook is explored further.

It is proven (Leskes, 2005) that reducing the disagreement, or equivalently maximizing the agreement, helps the process of learning from labeled and unlabeled data (in the PAC sense). Unlike Co-training which makes a commitment about the labels of unlabeled data, in the *disagreement reduction* we do not make decision about the label of unlabeled data. Instead unlabeled data is used for the (learned) hypotheses in different views to communicate and assess the plausibility of each other. To make the idea clear, suppose we have different views of the data, and each view is enough to learn the target concept. For each view we have a concept space, denote the concept corresponding to the target concept in each view as c^w_{opt}. It is clear that all c^w_{opt} must agree on the label of unlabeled data because essentially they are different representations of the (same) target concept. This causes the size reduction of the hypothesis space in each view. Hence, learning the target concept in

this reduced hypothesis space becomes easier.

2.3 Active Learning

The idea behind Active Learning (AL), also known as *experimental design* in Statistics (Santner, Williams, and Notz, 2003), is that a machine learning algorithm can achieve greater accuracy with fewer *labeled* training instances if it is allowed to choose the data from which it learns. An active learner may ask queries in the form of unlabeled instances to be labeled by an oracle, e.g. a human annotator. The AL problem setting that we consider in this thesis is referred to by *pool-based* AL, or sometimes by AL with *selective sampling*, in the literature. In this setting, the query points are selected from a large pool of given unlabeled data (Cohn, Atlas, and Ladner, 1994). This is in contrast to AL with *query synthesis* (Angluin, 1988) in which the learner can create the query points and then ask for their labels from the oracle. In what follows, we review active learning from theoretical perspective. Then we see several AL sample selection methods that are relevant to the future chapters in this thesis. For a comprehensive review of different query selection strategies in AL, see (Settles, 2009) and the references therein.

2.3.1 Theory

Thinking in the PAC model, the aim in AL is to reduce the number of labeled examples (or *label complexity*) needed to come up (with high probability) with a low error-rate hypothesis. There are two intuitions formalized in the literature about how AL may be helpful (Dasgupta and Hsu, 2008). The first intuition is to use AL for efficient search through a hypothesis space $\mathcal{H}$. By carefully selecting query points, the set of plausible hypotheses shrinks fast. We just need to maintain those hypotheses which are (almost) consistent[4] with the data seen so far, i.e. the so-called *version space* (Mitchell, 1982). For example, consider the best case where each new labeled data point cuts version space into half, and causes the version space to be shrunk greatly. This intuition has been formalized in several works to show the possibility of AL for different hypothesis classes in different conditions, e.g. when the target hypothesis does not belong to the hypothesis space (the so-called *agnostic* setting (Balcan, Beygelzimer, and Langford, 2009)).

[4]A hypothesis h is consistent with labeled data $D_l = \{(x_i, y_i)\}$, if $\forall i : h(x_i) = y_i$.

The second intuition is to use AL to exploit cluster structure in the data. If the data points belonging to one cluster have the same label, then asking the label of only one data point from each cluster suffices to learn the target hypothesis. However, it is not necessarily the case that clusters have pure labels, or even "they may not be aligned with labels at all" (Dasgupta and Hsu, 2008). Moreover, the data may exhibit cluster structure in different levels of granularity.

Here is an example (Dasgupta, 2004) to show the usefulness of AL via efficient search through the hypothesis space. Suppose the data lie on the real line, and consider the hypothesis space $\mathcal{H}$ in which the classifiers h_w are simple thresholding functions:

$$h_w(x) = \begin{cases} 1 & \text{if } x \geq w \\ 0 & \text{otherwise} \end{cases}$$

According to the PAC model, if the underlying distribution $P(x, y)$ can be classified perfectly by some hypothesis in $\mathcal{H}$ (called the *realizable* case), then in order to get a classifier with error rate at most ϵ (with high probability), it is enough to draw $m = O(\frac{1}{\epsilon})$ random labeled examples from $P(x, y)$ and to return any classifier consistent with them (Kearns and Vazirani, 1994). But suppose we instead draw m unlabeled samples from $P(x) = \sum_y P(x, y)$. If we lay these points down on the line, their hidden labels are a sequence of 0's followed by a sequence of 1's, and the goal is to discover the point $\hat{w}$ at which the transition occurs. This can be accomplished with a simple binary search which asks for just $\log(m)$ labels. Thus active learning gives us an exponential improvement in the number of labels needed: By adaptively querying $\log(m)$ labels, we can automatically infer the rest of them.

(Cohn, Ghahramani, and Jordan, 1996) takes a statistical perspective to AL and suggests to select instances so that the *expected error rate* of the learner over unseen data is reduced:

$$\int_x E\Big[\big(\hat{y}(x;\mathcal{D}) - y(x)\big)^2\Big] P(x)dx \tag{2.2}$$

where the expectation is taken over $P(y|x)$ and training sets $\mathcal{D}$. The expectation inside the integral, i.e. the expected error rate at a fixed point x, can be decomposed further into three terms:

$$\begin{aligned} E\Big[\big(\hat{y}(x;\mathcal{D}) - y(x)\big)^2\Big] &= E\Big[\big(y(x) - E[y|x]\big)^2\Big] \\ &+ \big(E[\hat{y}(x;\mathcal{D})] - E[y|x]\big)^2 \\ &+ E\Big[\big(\hat{y}(x;\mathcal{D}) - E[\hat{y}(x;\mathcal{D})]\big)^2\Big] \end{aligned}$$

In the above expressions, the first term is the variance of $P(y|x)$ which shows the amount of *noise* in the underlying distribution and does not depend on the learner. The second term is the squared *bias* of the learner which shows the difference between learner's expected prediction and the expected true label for the point x. The third term is the learner's *variance* on its prediction which shows the amount of uncertainty in learner's prediction on the input point x. Queries can be used to reduce either bias or variance-related objective functions (or ideally both), but noise cannot be reduced since it is independent of the learner. In (Cohn, 1995) queries are used to reduce the model bias, and in (Cohn, Ghahramani, and Jordan, 1996) queries are used to decrease the expected variance for mixtures of Gaussians and locally weighted regression.

2.3.2 Query by Committee

An efficient search strategy through the hypothesis space selects instances which would reduce greatly the number of plausible hypotheses if we had their labels. Based on this argument, informative instances are those which belong to the *disagreement* area of the input space with respect to the current set of plausible hypotheses. The disagreement area is a subset of the input space $\mathcal{X}$ containing those points whose labels are not agreed upon by the current plausible hypotheses. Ideally, we are interested in those points in the disagreement area which rule out half of the plausible hypotheses.

One way to locate such informative instances is to choose a committee of hypotheses from the version space, use their predictions on the pool of unlabeled data, and quantify the disagreement among the committee on the predicted labels for unlabeled instances. This method, which is called *query by committee*, has been proved theoretically (Seung, Opper, and Sompolinsky, 1992; Freund et al., 1997) and experimentally (Engelson and Dagan, 1996; Kauchak, 2006) useful for reducing the annotation effort.

A related setting is multi-view active learning (Muslea, Minton, and Knoblock, 2006; Wang and Zhou, 2008) in which instances are selected assuming that there exist disjoint subsets of features, each of which are sufficient to learn the target hypothesis (a setting similar to Co-training in Section 2.2.4). The informative instances are selected from the *contention points*[5]: Those data points whose labels are disagreed upon by at least two

[5]In contrast to the disagreement area which is specified by the hypotheses on a single view, the contention points are specified by hypotheses on different views.

learners, each of which corresponding to a separate view. Querying contention points shrinks the version spaces in multiple views and, together with agreement constraints imposed by multiple views on the hypotheses spaces, helps to speed up the learning. In contrast to the query by committee in which committee hypotheses are chosen from a single view, in this setting the hypotheses are chosen from different views.

2.3.3 Query based on Uncertainty

According to this approach, the unlabeled points for which the learner is most uncertain about the labels are queried first. This is in contrast to the committee approach which uses multiple hypotheses, in that it lets the learning algorithm to do its job and output a hypotheses. However it selects instances which are harder for the current hypothesis during the AL iterations to guide the learning algorithm towards better hypotheses.

The Shannon entropy is extensively used in information theory to quantify the amount of uncertainty contained in a probability distribution. In case that the underlying learning algorithm outputs a probabilistic classifier, we select an instance which maximizes the Shannon entropy of the prediction distribution $P(y|x, \theta)$:

$$\underset{x_j \in D_u}{\operatorname{argmax}} - \sum_y P(y|x_j, \theta) \log P(y|x_j, \theta)$$

This has been used in (Settles and Craven, 2008) and (Hwa, 2004) for instance selection in sequence labeling and parsing tasks based on Conditional Random Fields (CRFs) (Lafferty, McCallum, and Pereira, 2001) and Probabilistic Context Free Grammars (PCFGs) respectively, and in (Kim et al., 2006) where the entropy has been *approximated* based on the n-best list for CRFs.

Instead of Shannon entropy, we can use the *min-entropy* to measure the amount of uncertainty and select the next query as follows:

$$\underset{x_j \in D_u}{\operatorname{argmax}} \min_y \log \frac{1}{P(y|x_j, \theta)} = \underset{x_j \in D_u}{\operatorname{argmin}} \max_y P(y|x_j, \theta)$$

It means to choose instances for which the classifier is least confident about the best label. This strategy has been used in (Settles and Craven, 2008; Culotta and McCallum, 2005) for sequence labeling tasks using CRFs. The Shannon entropy and min-entropy are both instantiations of a broader class of entropy measures called Rényi entropy which is defined as $H_\alpha(P(z)) := \frac{1}{1-\alpha} \log \sum_z P^\alpha(z)$. It can be shown that $\lim_{\alpha \to 1} H_\alpha$ and $\lim_{\alpha \to \infty} H_\alpha$ correspond to the Shannon entropy and min-entropy, respectively (Arndt, 2001).

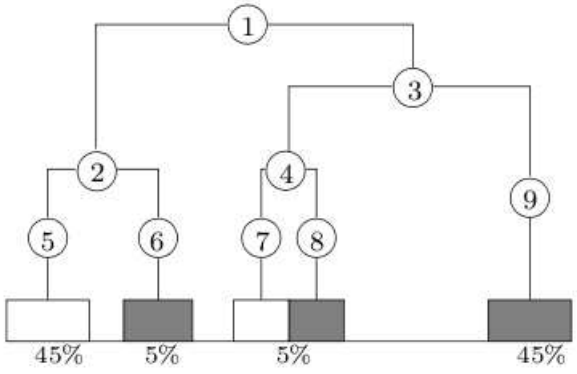


Figure 2.1: The input data points lie on the x axis, and the black/white boxes show their labels. The nodes of the tree correspond to a hierarchical clustering of the data (Dasgupta and Hsu, 2008).

2.3.4 Hierarchical Sampling

(Dasgupta and Hsu, 2008) proposes a technique for sample selection that, under certain assumptions, is guaranteed to be no worse than random sampling. The method exploits the cluster structure (if there is any) in the unlabeled data. Ideally, querying the label of only one of the data points in a cluster would be enough to determine the label of the other data points in that cluster. But the clusters may not be pure (in terms of labels), and we may need to go to a finer grained clustering of impure clusters with the hope of finding more pure sub-clusters. The aim is to go adaptively down in the clustering level of different regions of the input space until we can confidently say the resulted clusters are almost pure. This is done iteratively by the following two phrases (see Figure 2.3.4):

- Selecting a cluster to expand into finer grained clusters. This is done using the cluster impurity obtained from empirical estimates (based on the labeled data points in the cluster), and the size of the cluster (the number of all labeled and unlabeled data points in the cluster).
- Sampling some unlabeled points from the selected cluster (in the previous step), and ask the oracle for their labels.

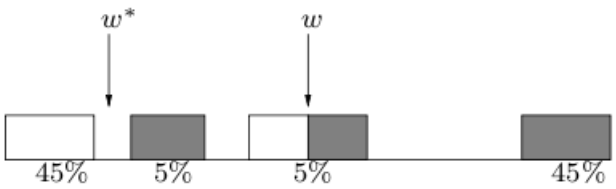


Figure 2.2: w represents the decision boundary resulted from a typical AL method which operates based on querying points having the most labeling uncertainty. However, w^* represents one of the optimum classifiers having the least error rate (Dasgupta and Hsu, 2008).

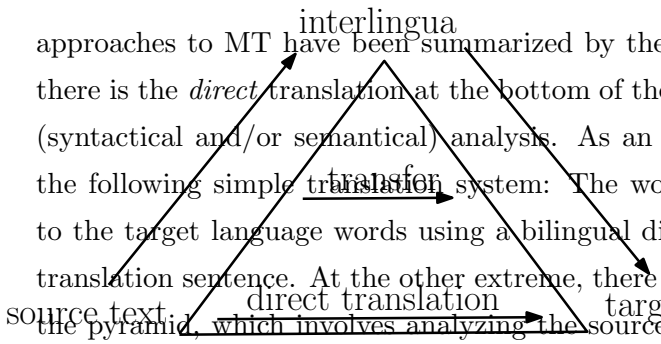
At the end, the majority label of each cluster is assigned to the rest of the points of the cluster, and any supervised learning algorithm can be used to learn a hypothesis from the input space to the labels using this *labeled* training data.

The motivation behind this strategy comes from a problem knows as the *sampling bias* inherent in many AL instance selection strategies (see Figure 2.2). Consider for example the query selection method based on the uncertainty (explained in the previous section) applied to Support Vector Machines (SVMs) (Tong and Koller, 2000). This amounts to query points which are close to the decision boundary, since they have the most uncertain labelings in the sense of min-entropy . Now suppose there are three clusters in the data, and initially we are given labeled points just from two of them which are more dense than the third one. Having an SVM trained on these labeled data in the first iteration of the AL loop, the future queries will focus on the points in the boundary of these two clusters to adjust the hyper-plane accordingly. This will overlook the third cluster and will not query its points since the queries are biased. The hierarchical sampling method instead diversifies its queries so that every cluster of points can have representatives in the labeled data, so that it can influence the finally trained classifier.

2.4 Phrase-based Statistical Machine Translation

In this section I present a brief overview of the phrase-based SMT models, but first let us have a high level view on different approaches to machine translation (MT). Traditionally,

Figure 2.3: The machine translation pyramid (Hutchins and Somers, 1992).



approaches to MT have been summarized by the pyramid in Figure 2.3. At one extreme, there is the *direct* translation at the bottom of the pyramid, which involves no intermediate (syntactical and/or semantical) analysis. As an example of the direct approach, consider the following simple translation system: The words of the source sentence are translated to the target language words using a bilingual dictionary, and then reordered to yield the translation sentence. At the other extreme, there is the *interlingua* approach at the apex of the pyramid, which involves analyzing the source sentence into an abstract representation and then generating the translation from this representation. The interlingua representation is supposed to be universal and to have all information needed to generate the translation without looking back into the source sentence. The path to that interlingua is long in the source side, and it is better to cut the monolingual analysis at some point to get an intermediate language-dependent representation. This representation is *transferred* to a representation in the target language, and then the translation is generated from it. As the pyramid shows, the more the source text is analyzed, the simpler the transfer will be. The phrase-based SMT models that we see in the following belong to the class of direct translation approaches.

To find the translation of a given source language sentence $\mathbf{f}$ in SMT, we choose the sentence with the highest probability among all possible target language sentences:

$$\begin{aligned} \hat{\mathbf{e}} &= \operatorname*{argmax}_{\mathbf{e}} P(\mathbf{e}|\mathbf{f}) \\ &= \operatorname*{argmax}_{\mathbf{e}} P(\mathbf{e}) \times P(\mathbf{f}|\mathbf{e}) \end{aligned} \tag{2.3}$$

The component $P(\mathbf{e})$ is the so-called *language model* and is responsible for the well-formedness

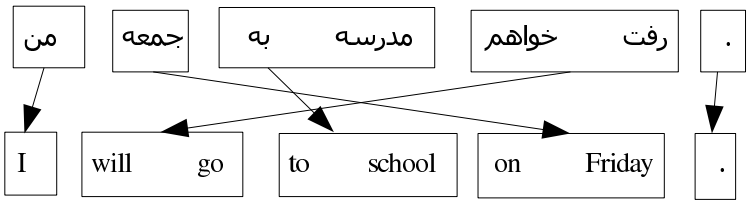


Figure 2.4: Example of a phrase-based translation from Farsi to English.

of the generated translation. The component $P(\mathbf{f}|\mathbf{e})$ is called the *translation model* and is meant to explain the process of generating the source sentence from a hypothetical translation in the target language. Intuitively speaking, the language model takes care of the *fluency* of the generated translation, and the translation model mainly takes care of the *content preservation*[6]. The decomposition in equation 2.3 into two sources of information is known as the source-channel approach to SMT.

One possibility to generate the target translation $\mathbf{e}$ from the source sentence $\mathbf{f}$ is the following: Segment the sentence $\mathbf{f}$ into phrases, look up the translations of the phrases in a bilingual phrase dictionary, and finally reorder the translated phrases (see Figure 2.4). Note that a phrase is a contiguous sequence of words, and is not necessarily a syntactic or semantic unit. Phrase pairs robustly capture local reordering and idiomatic terms, and are learned from substring pairs that are observed commonly in the bilingual training data.

Instead of using the source channel model and decomposing the posterior probability of the target sentence given the source sentence $P(\mathbf{e}|\mathbf{f})$ as equation (2.3), we can directly model the conditional probability by a *log-linear* model in the phrase-based SMT (Och and Ney, 2002):

$$P(\mathbf{e}|\mathbf{f}) \quad = \quad \frac{\exp\left[\sum_{k=1}^{K} \lambda_k h_k(\mathbf{e}, \mathbf{f})\right]}{Z_\lambda(\mathbf{f})} \tag{2.4}$$

where $Z_\lambda(\mathbf{f}) = \sum_{\mathbf{e}'} \exp \sum_{k=1}^{K} \lambda_k h_k(\mathbf{e}', \mathbf{f})$ is the source-sentence specific normalization constant, and h_k are *feature functions*. To find the translation of a given source sentence, we

[6]Of course the translation model in the phrase-based SMT systems enforces local fluency using phrases; however, the important point is that the main functionality of the translation model is to preserve the content.

just need to solve the following search problem:

$$\hat{\mathbf{e}} = \underset{\mathbf{e}}{\operatorname{argmax}} \sum_{k=1}^{K} \lambda_k h_k(\mathbf{e}, \mathbf{f}) \tag{2.5}$$

The log-linear model is a generalization of the source-channel approach. The feature functions which are commonly used are:

- **The Language Model Score**. Usually the score $\log P(\mathbf{e})$ of an n-gram language model is used, in which

  $$P(\mathbf{e}) = \prod_{i=1}^{|\mathbf{e}|} P_{LM}(e_i|e_{i-1}, \ldots, e_{i-n+1})$$

  The parameters of this model, i.e. the probability of the next word given the history of the previous $n-1$ words $P_{LM}(e_i|e_{i-1}, \ldots, e_{i-n+1})$ where n is a fixed number, are trained on a large monolingual text in the target language.

- **The Phrase Translation Scores.** It is the sum of the goodness score of individual phrase pairs, e.g. $\sum_k \log P(\bar{e}_k|\bar{f}_k)$ where $(\bar{e}_k, \bar{f}_k)$ is a phrase pair.

- **The Reordering Model Score.** It penalizes long distance movements in the reordering of the phrases in the target sentence. For example, it may assign a penalty based on the number of source words which are skipped when generating a new target phrase.

In addition to above basic feature functions, we can integrate any other feature function into the overall system, which highlights the advantage of using log-linear models for SMT. For example we can have a feature function to penalize long translation sentences and motivate short translations. In what follows, we see in more details how phrase-pairs are extracted from the bilingual corpus, how the weights of the log-linear model λ are trained, and how the search problem (2.5) is solved.

2.4.1 Phrase-Pair Extraction

Every subsequence of a parallel source and target sentence can be a candidate phrase pair. However, the space of all possible phrase pairs is huge[7] and is a computational bottleneck if

[7]It is $\mathcal{O}(|\mathbf{e}|^2.|\mathbf{f}|^2)$ for a sentence pair $(\mathbf{f}, \mathbf{e})$.

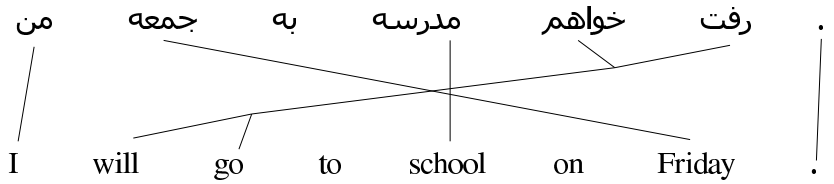


Figure 2.5: Example of an alignment between the Farsi and English parallel sentences mentioned in Figure 2.4.

we consider all of the candidates. Furthermore, most of these phrase pairs will not be useful for our SMT model since they are not good ones. To overcome these issues, the size of this space is reduced carefully using different mechanisms. The phrase extraction technique of our underlying phrase-based SMT system uses the *word alignment* between the source and target sentences to prune many of the useless phrase pairs. In the following, we first see what is word alignment and then how to use it for phrase pair extraction.

Intuitively a word alignment between a sentence pair represents a correspondence between the words of the two sentences; see Figure 2.5 for an example. Using the word alignment in this example, we can understand that the seventh word in the English sentence is the translation of the second word in the Farsi sentence. More formally, an alignment $\mathbf{a}$ is a subset of the Cartesian product of the word positions in the source and target sentences. Developing alignment models to deal with this general representation is hard (Och and Ney, 2003), so additional constraints are imposed on the alignment representations. As an example of such restrictions, each position in the source sentence may be assigned to exactly one position in the target sentence. This restriction is a characteristic of the so-called IBM word alignment models (Brown et al., 1993), and means that the alignment $\mathbf{a}$ is a vector of length $|\mathbf{f}|$ in which $a_j = i$ if f_j (the jth word in $\mathbf{f}$) is aligned with the e_i (the ith word in $\mathbf{e}$). The alignment $a_j = 0$ accounts for those words in $\mathbf{f}$ which are not aligned with any word in $\mathbf{e}$, i.e. those which are aligned with a special 'empty' word e_0. Note that the alignment is not necessarily a symmetric object and depends on the direction of the translation. I will explain computing word alignments based on IBM translation models 1 and 2 in Section 2.4.1.1, since these alignments are used in the phrase pair extraction of the underlying SMT model used in this thesis.

Starting from an alignment between the source and target sentences, only the phrase

pairs which are **consistent** with this alignment are extracted. Consistency means no word in one phrase should be aligned with words outside the other phrase, and there is at least one alignment link between the two phrases. This way, only the potential high-quality phrase pairs for which there exist some evidences according to the alignment are considered. Furthermore, the phrases with length above/below a threshold may be discarded to constraint more the space of phrase pairs. Often we compute the alignments for both translation directions, i.e. source to target and target to source, and then take the *intersection* of the two, which gives a high precision alignment with low recall. Conversely, taking the *union* of the alignments results in high recall but low precision. There are different heuristics, e.g. *grow-diag* or *grow-diag-final* (Koehn, Och, and Marcu, 2003), to start from the intersection of the alignments and grow carefully the alignment links. Indeed the phrase pairs are extracted based on their consistency with the links included in this modified alignment.

For each phrase pair $(\bar{f}, \bar{e})$, we need to specify some score(s) showing the quality of the phrase pair. Each score corresponds to one feature function in the log-linear model. Two popular scores are the conditional probabilities based on the aligned parallel sentences:

$$P(\bar{e}|\bar{f}) = \frac{Count(\bar{f}, \bar{e})}{\sum_{\bar{e}'} Count(\bar{f}, \bar{e}')} \qquad P(\bar{f}|\bar{e}) = \frac{Count(\bar{f}, \bar{e})}{\sum_{\bar{f}'} Count(\bar{f}', \bar{e})}$$

The *smoothed* versions of these probabilities (Foster, Kuhn, and Johnson, 2006) and/or the *lexical weight* of the phrase pairs (Koehn, Och, and Marcu, 2003) may also be used as additional feature functions. The extracted phrase pairs and their corresponding scores are kept in the form of a table called the *phrase table*. The source and target phrases are kept in the first two columns of the phrase table, and different phrase pair scores are kept in the following columns.

2.4.1.1 IBM Model 1 and IBM Model 2

IBM models are probabilistic models to transform a sentence $\mathbf{e}$ in one language to its translation $\mathbf{f}$ in another language. There are five IBM models which are originally introduced in the seminal paper (Brown et al., 1993), in increasing order of the translation process complexity. We just review the IBM models 1 and 2 here since they will be used in the phrase extraction process of the underlying phrase-based SMT system used in this thesis. Just for this section, let us assume that $\mathbf{e}$ is the source sentence and $\mathbf{f}$ is the target sentence, and we want to model the translation process from $\mathbf{e}$ to $\mathbf{f}$.

As we mentioned before, it is assumed that each target word f_j is translated from a source sentence word e_i or a special empty source word e_0 according to IBM models. Therefore, the alignment space $\mathcal{A}(\mathbf{e}, \mathbf{f})$ contains $(|\mathbf{e}|+1)^{|\mathbf{f}|}$ possible alignments, where $|\mathbf{s}|$ denotes the length (number of words) of the sentence $\mathbf{s}$. The probability $P(\mathbf{f}|\mathbf{e})$ in IBM models is decomposed as follows

$$\begin{aligned} P(\mathbf{f}|\mathbf{e}) &= \sum_{\mathbf{a}\in\mathcal{A}(\mathbf{e},\mathbf{f})} P(\mathbf{f},\mathbf{a}|\mathbf{e}) \\ &= \sum_{\mathbf{a}\in\mathcal{A}(\mathbf{e},\mathbf{f})} P(|\mathbf{f}| \mid \mathbf{e}) \prod_{j=1}^{|\mathbf{f}|} P(a_j|a_1^{j-1}, f_1^{j-1}, |\mathbf{f}|, \mathbf{e}).P(f_j|a_1^j, f_1^{j-1}, |\mathbf{f}|, \mathbf{e}) \end{aligned} \quad (2.6)$$

where a_j means the alignment for the jth position, $a_1^j = a_1, .., a_j$ means the alignments for the positions 1 to j, and similarly for f_j and f_1^j. Moreover the term $P(|\mathbf{f}| \mid \mathbf{e})$ refers to the probability of a particular length for the translation given the source sentence.

In IBM Model 1, the following assumptions are made to simplify the expression 2.6:

- $P(f_j|a_1^j, f_1^{j-1}, |\mathbf{f}|, \mathbf{e})$ only depends on the source word according to the alignment which corresponds to the target word f_j. It is denoted by $t(f_j|e_{a_j})$ and called the translation probability. $t(.|e)$ shows the probability of generating different target words conditioned on a particular source word e.
- $P(a_j|a_1^{j1}, f_1^{j-1}, |\mathbf{f}|, \mathbf{e})$ only depends on the length of the source sentence and is equal to $\frac{1}{|\mathbf{e}|+1}$.
- $P(|\mathbf{f}| \mid \mathbf{e})$ is a fixed number ϵ.

Consequently, the expression 2.6 is simplified to:

$$P(\mathbf{f}, \mathbf{a}|\mathbf{e}) = \frac{\epsilon}{(|\mathbf{e}|+1)^{|\mathbf{f}|}} \prod_{j=1}^{|\mathbf{f}|} t(f_j|e_{a_j}) \quad (2.7)$$

In training, we are interested to learn the translation probabilities $t(f|e)$, i.e. the *model parameters*, based on a given sample of parallel sentences $\{(\mathbf{e}, \mathbf{f})\}$. We use the EM algorithm for training, assuming that the true alignments are hidden. This amounts to doing the following two steps iteratively:

- Set $P(\mathbf{a}|\mathbf{f}, \mathbf{e}) \propto P(\mathbf{f}, \mathbf{a}|\mathbf{e})$ to be the probability of the alignments based on the equation 2.7 using current values of the model parameters.

- Update the model parameters as follows

$$t(f|e) \propto \sum_{(\mathbf{f},\mathbf{e})} \sum_{\mathbf{a}\in\mathcal{A}(\mathbf{f},\mathbf{e})} Count(f, e; \mathbf{a})P(\mathbf{a})$$

  where $Count(f, e; \mathbf{a})$ is the number of times f is aligned to e in the alignment $\mathbf{a}$. Although the size of the alignment space is exponential in the length of the target sentence, the expected counts $\sum_{\mathbf{a}\in\mathcal{A}(\mathbf{f},\mathbf{e})} Count(f, e; \mathbf{a})P(\mathbf{a})$ can be computed efficiently in polynomial time.

Now suppose we are given a pair of parallel sentences $(\mathbf{e}, \mathbf{f})$, and asked to find the best alignment for it based on the IBM Model 1. Having the translation probabilities $t(f|e)$, the procedure is as follows: For each position j in the target sentence, choose the index $\text{argmax}_{0\leq i\leq|\mathbf{e}|}\, t(f_j|e_i)$.

In the IBM model 2, the assumptions are the same as those in the IBM Model 1, except that the positions in the source sentence are not equally likely as alignment candidates for a fixed position in the target sentence. In other words, there are new parameters called *alignment probabilities* $a(a_j|j, |\mathbf{e}|, |\mathbf{f}|) = P(a_j|a_1^{j-1}, f_1^{j-1}, |\mathbf{f}|, \mathbf{e})$, which represent the plausibility of different source sentence positions for an alignment link conditioned on a target sentence position j and the length of the two sentences. Therefore, the expression 2.6 is simplified to:

$$P(\mathbf{f}, \mathbf{a}|\mathbf{e}) = \epsilon \prod_{j=1}^{|\mathbf{f}|} a(a_j|j, |\mathbf{f}|, |\mathbf{e}|).t(f_j|e_{a_j}) \tag{2.8}$$

The training for the translation probabilities in this case is similar to the one mentioned for the IBM model 1, except that we use equation 2.8 to compute the alignment probabilities in the first step of the iterative training process. The alignment probabilities are also estimated based on the EM in a similar manner using the normalized expected counts. Now suppose we are given a pair of parallel sentences $(\mathbf{e}, \mathbf{f})$, and asked to find the best alignment for it based on the IBM Model 2. Having the translation probabilities $t(f|e)$ and the alignment probabilities $a(i|j, |\mathbf{f}|, |\mathbf{e}|)$, the procedure is as follows: For each position j in the target sentence, choose the index $\text{argmax}_{0\leq i\leq|\mathbf{e}|}\, t(f_j|e_i)a(i|j, |\mathbf{f}|, |\mathbf{e}|)$.

2.4.2 Learning the Weights of the Log-Linear Model

Having provided the feature functions of the log-linear model (2.4), we need to specify their weights $\boldsymbol{\lambda}$. Instead of learning the model parameters to optimize the likelihood, the aim is

to learn them by directly optimizing for the translation quality since "[likelihood has] only a loose relation to the final translation quality on an unseen text" (Och, 2003a). However the cost surface of the typical measures for translation quality (see Section 2.4.4) is highly non-smooth and has lots of local minima. One strategy is to do grid-based search: Starting from a random point in $\mathbb{R}^K$ where K is the number of feature functions, we learn one parameter at a time while keeping the others fixed, and change the value of this parameter according to some step size. However, there is a tradeoff between the efficiency of this method and the quality of the final solution. Small steps may lead to a better parameter value but suffer from low convergence (and vice versa). The *minimum error rate training* (MERT) (Och, 2003a) is a specialized line search algorithm to find the weights of the log-linear model in phrase-based SMT, which chooses the appropriate step size smartly. In what follows, I explain this algorithm in more detail since we use it extensively in this thesis.

Let us assume that we are given a set of source sentences $\{\mathbf{f}_1, \ldots, \mathbf{f}_S\}$, and for each sentence $\mathbf{f}_s$, a set of candidate translations $\{\mathbf{e}_{s,1}, \ldots, \mathbf{e}_{s,n}\}$. The goal is to manipulate the model parameters so that, for each sentence, the best translation sentence (among the n candidate translations) is ranked first in the candidate list. Let $Q(\hat{\mathbf{e}}(\mathbf{f}_s, \boldsymbol{\lambda}))$ denote the quality measure of the first ranked sentence $\hat{\mathbf{e}}(\mathbf{f}_s, \boldsymbol{\lambda})$. It may possibly depend on the true translation (see Section 2.4.4) but we have removed the reference sentence from the notation for brevity. Usually the quality measure for all of the sentences is its sum for individual sentences, namely $\sum_{s=1}^{S} Q(\hat{\mathbf{e}}(\mathbf{f}_s, \boldsymbol{\lambda}))$, and this is the objective function which is going to be optimized. The objective function is not differentiable, so we cannot use gradient-based methods to learn the weights.

The idea of the MERT algorithm (Och, 2003a) is to start from a random starting point $\boldsymbol{\lambda}_0 \in \mathbb{R}^K$ and then move along a direction $\mathbf{d} \in \mathbb{R}^K$ with *appropriate* step size $\gamma^* \in \mathbb{R}$ to optimize $\sum_{s=1}^{S} Q(\hat{\mathbf{e}}(\mathbf{f}_s, \boldsymbol{\lambda}_0 + \gamma \mathbf{d}))$. In practice, the direction $\mathbf{d}$ is chosen to be parallel to axes. Note that $Q(\hat{\mathbf{e}}(\mathbf{f}_s, \boldsymbol{\lambda}_0 + \gamma \mathbf{d}))$ is a piecewise constant function with respect to γ. So the real line can be partitioned into segments at points $\gamma_1 \leq \gamma_2 \leq \ldots \leq \gamma_{n-1}$ such that the function $Q(\hat{\mathbf{e}}(\mathbf{f}_s, \boldsymbol{\lambda}_0 + \gamma \mathbf{d}))$ is constant in each segment. After finding these piecewise constant functions for (the translations of) each source sentence $\mathbf{f}_s$, we add them together to get one final piecewise constant function (see Figure 2.6 for an example). That is, the total quality measure in each individual final segment is the sum of the quality measure of each individual old segment. The best value for the step size, i.e. γ^*, is chosen by looking at the total quality measure in the final segments.

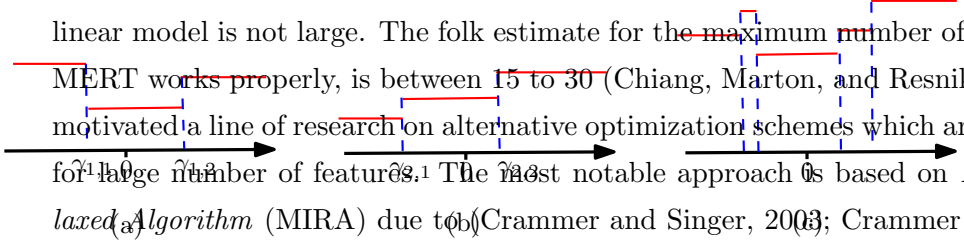


Figure 2.6: (a,b) show the piecewise constant functions for the first and second source sentences. (c) shows the final piecewise constant function resulted from the sum of these two functions. The horizontal axis corresponds to real values for γ and the vertical axis corresponds to a measure of translation quality.

The last step of the algorithm can be sped up by noticing that in each final segment, the quality measure is the sum of individual quality measures of the old segments. Therefore, we choose to record the *change* in the quality measure when (1) building the segmentations for each individual sentence, and (2) merging the segmentations of all of the source sentences. Consequently, computing the quality measure in each final segment becomes $\mathcal{O}(1)$ instead of $\mathcal{O}(S)$.

The MERT algorithm is appropriate when the number of feature functions in the log-linear model is not large. The folk estimate for the maximum number of features, for which MERT works properly, is between 15 to 30 (Chiang, Marton, and Resnik, 2008). This issue motivated a line of research on alternative optimization schemes which are more appropriate for large number of features. The most notable approach is based on *Margin Infused Relaxed Algorithm* (MIRA) due to (Crammer and Singer, 2003; Crammer et al., 2006) which maximizes the margin between the best translation and the candidate translations in the n-best list in an online manner (Tillmann and Zhang, 2006; Watanabe et al., 2007; Chiang, Marton, and Resnik, 2008; Chiang, Knight, and Wang, 2009).

2.4.3 Solving the Search Problem

In general, searching for the best translation for a given source sentence (or decoding) is a *hard* problem (Knight, 1999), and can be reformulated as a classic AI search problem. Often researchers resort to effective heuristic search algorithms developed in the AI community,

most notably *beam search* (Russell and Norvig, 1995), to address the decoding problem. In what follows, I describe briefly the stack-based beam search algorithm (Koehn, 2003) employed by PORTAGE which is the underlying phrase-based SMT system used in this thesis.

The idea is to generate the target sentence from left to right by repeatedly translating phrases from different parts of the source sentence. That is, we iterate through these two steps until all words of the source sentence are translated (or covered):

- Select a phrase from the untranslated part of the source sentence. Indeed it is this step which causes the exact decoding to become computationally intractable.
- Translate this phrase into one of its several translations according to the phrase table.

An incomplete covering of the source sentence plus its corresponding incomplete translation sentence is called a *hypothesis*. Each iteration through the above two steps may transform one hypothesis to several next hypotheses with some *costs* which come from feature functions and their weights in the log-linear model (objective function), e.g. reordering cost, translation cost, and language model cost (see Figure 2.7). The language model and phrase translation scores can be negated to reflect costs. The goal is to efficiently move on the resulting *search graph* to find high quality complete hypotheses, i.e. the ones which cover all words in the source sentence with low costs.

Each hypothesis in the search graph is associated with a *path-cost*: The sum of all costs along its path from the initial hypothesis. From a hypothesis in the search graph, we can choose to move to the next hypothesis which has the lowest path-cost among the candidates. However this greedy search strategy does not work well; it prefers to translate easy parts of the sentence first since it does not take into account the cost of translating the uncovered parts of the source sentence. To overcome this issue, an estimate of the *future-cost* is also assigned to each hypothesis, i.e. an estimate of its path to the closest lowest-cost complete hypothesis. The total cost of a hypothesis, based on which we select the next hypothesis, is the sum of the path-cost and future-cost estimate. If the future-cost estimate is admissible, i.e. it does not overestimate the true cost, then the search algorithm is the so-called A^* which is guaranteed to return the optimal solution. But providing an admissible future-cost estimate in a reasonable amount of time is usually not possible (in fact, this is the reason that makes the decoding computationally *hard* in the first place). Therefore, we resort to

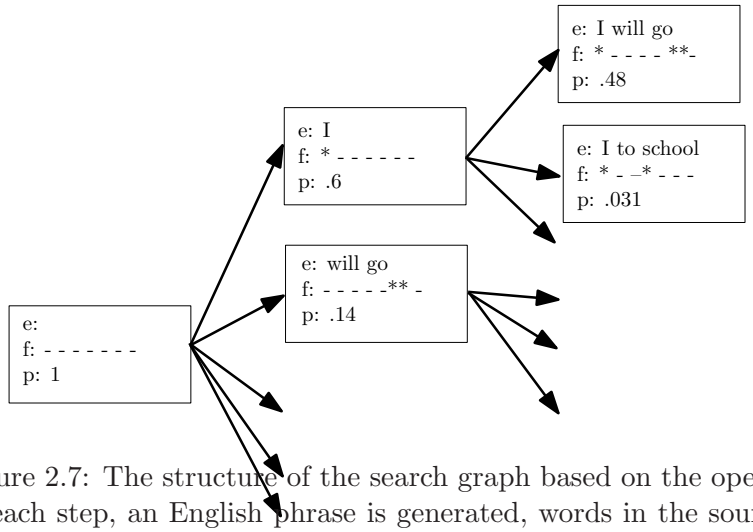


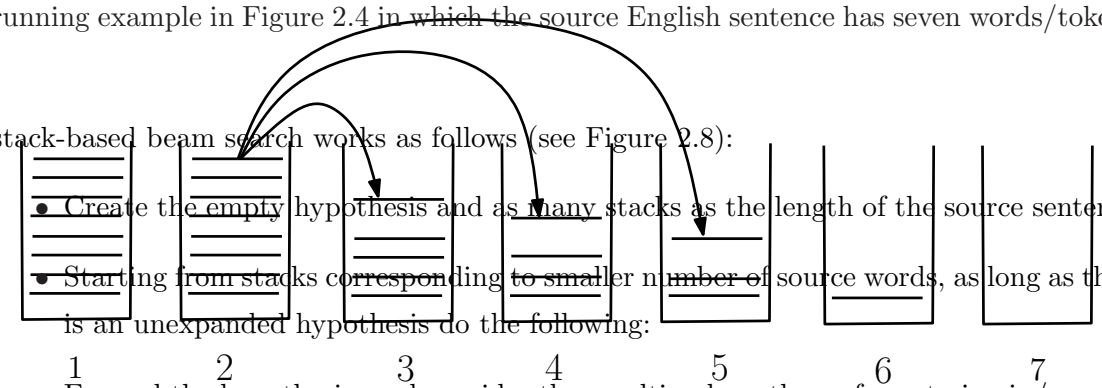
Figure 2.7: The structure of the search graph based on the operations used in the decoder. In each step, an English phrase is generated, words in the source Farsi phrase are covered (marked by *), and the cost so far is updated (for the example in Figure 2.4).

inadmissible future cost estimation which can be calculated efficiently, and keep a set of best hypothesis seen so far while traversing the search graph.

An estimate for the future cost of an hypothesis is computed efficiently by dynamic programming (DP) based on the language model and phrase translation costs (the contribution of the reordering model is ignored). For each phrase covering a contiguous sequence of uncovered words in the source sentence, we consider the cost of its best translation phrase by adding: (1) The cost of the phrase pair from the phrase table, and (2) The language model cost for the translation phrase. The cost of the best covering for all uncovered words of the source sentence can be efficiently computed by a DP once the DP table is initialized by the best translation phrase costs.

The beam search algorithm discards many low quality hypotheses and limits itself to a promising part of the search graph. The core of the algorithm is to keep a subset of the best hypotheses seen so far, expand them according to the operations mentioned before, and prune the inferior hypotheses according to their total costs. In pruning, only the hypotheses which cover the same *number* of source sentence words are compared. Algorithmically, this

Figure 2.8: A hypothesis is expanded to new hypotheses, and they are placed into new stacks according to the number of source sentence words which they cover. This is for our running example in Figure 2.4 in which the source English sentence has seven words/tokens.



stack-based beam search works as follows (see Figure 2.8):

- Create the empty hypothesis and as many stacks as the length of the source sentence.
- Starting from stacks corresponding to smaller number of source words, as long as there is an unexpanded hypothesis do the following:
- Expand the hypothesis, and consider the resulting hypotheses for entering in/pruning from the corresponding stacks. Go to the previous step.

To limit the number of hypotheses in one stack, we can either do threshold pruning or histogram pruning. In the former, the maximum capacity of a stack is set beforehand. But in the latter, the hypotheses whose costs are above a multiplicative factor of the best hypothesis in that stack are discarded.

2.4.4 Evaluation

Human judgment of the MT output is expensive and subjective, so automatic evaluation measures are a necessity. However, automatic evaluation of the MT output is a challenging task. In most cases, there is no single correct translation. Furthermore, it might be the case that two correct translations of the same sentence can have completely different words and sentence structure. Showing the deficiencies of the current automatic evaluation measures

and proposing new measures has been an area of active research in SMT community. In what follows, I describe three automatic evaluation measures which are extensively used in the community and also in our experiments in the later chapters. They compare the generated translation to one or more given reference translations.

- BLEU (bilingual evaluation understudy) (Papineni et al., 2002): The BLEU score is based on the notion of modified n-gram precision, for which all candidate n-gram counts in the translation are collected and clipped against their corresponding maximum reference counts. These clipped candidate counts are summed and normalized by the total number of candidate n-grams.

- WER (word error rate): The word error rate is based on the Levenshtein distance. It is computed as the minimum number of substitution, insertion and deletion operations that have to be performed to convert the generated translation into the reference translation. In the case where several reference translations are provided for a source sentence, we calculate the minimal distance to this set of references as proposed in (Nießen et al., 2000).

- PER (position-independent word error rate) (Nießen et al., 2000): A shortcoming of the WER is the fact that it requires a perfect word order. In order to overcome this problem, the position independent word can be used, comparing the words in the two sentences without taking the word order into account. Words that have no matching counterparts are counted as substitution errors, missing words are deletion and additional words are insertion errors. The PER is a lower bound for the WER.

Note that BLEU score measures translation quality, whereas WER and PER measure translation errors. Often, we also present 95%-confidence intervals calculated using bootstrap resampling.

2.4.5 Software Packages

The SMT system I use in the experiments of the Chapters 4,5, and 6 is PORTAGE (Ueffing et al., 2007). This is a state-of-the-art phrase-based translation system developed at the National Research Council Canada which has been made available to Canadian universities for research and education purposes. Essentially PORTAGE is an implementation of the phrase-based SMT system explained in the previous sections. I have used different releases

of this software up to 2008 for the experiments of this thesis. There is also another implementation of the described phrase-based SMT called *moses*, which is open source and can be used freely[8]. I provide here the basic features I use in the log-linear model with PORTAGE:

- One or several phrase table(s), which are smoothed using the methods described in (Foster, Kuhn, and Johnson, 2006). More specifically, for each pair of parallel sentences, the IBM model 2 alignments in the forward and backward directions are intersected and then grown to cover all words[9]. Then, the phrases which are consistent with these alignments are extracted while respecting the length constraints. The maximum length of each phrase is 8 words, and the difference between the lengths of the source and target phrases in a phrase pair cannot exceed 10. Finally, the one best IBM model 1 translations for source and target language words that occur in the given bitext but do not have translations in phrase table are added. I used the forward and backward *smoothed* versions of the phrase pairs raw probabilities based on Kneser-Ney and Zens-Ney smoothing methods (Foster, Kuhn, and Johnson, 2006).

- Tri-gram language models trained with the SRILM toolkit described in (Stolcke, 2002). I have used different releases of this software up to the version 1.5.6 for the experiments of this thesis.

- A distortion model which assigns a penalty based on the number of source words which are skipped when generating a new target phrase.

- A word penalty assigning constant cost to each generated target word. This constitutes a way to control the length of the generated translation.

In this thesis in addition to the above basic features, I often add a new phrase table with multiple feature functions to realize the ideas of the related chapters. The feature weights in the log-linear model are trained using PORTAGE implementation of the MERT algorithm, with respect to improvement of the BLEU score on a development set.

It is important to look more closely how the decoding is done when the decoder is given multiple phrase tables. When *canoe*, which is the decoder program in PORTAGE, is provided with multiple phrase tables, it internally builds a single phrase table that is the

[8]It can be downloaded from http://www.statmt.org/moses .

[9]Using the default option $IBMOchAligner3$ for the *gen_phrase_table* program.

union of all the phrase tables. For each phrase pair, there are as many scores as the total number of scores in the phrase tables. When a phrase pair $(\bar{e}, \bar{f})$ exists in a table T, the scores are those in T. But when $(\bar{e}, \bar{f})$ is not in T, the missing scores are filled in as a very small number. The decoding will be performed as explained in Section 2.4.3 using this final phrase table. This is crucial for our self-training algorithms to work properly: The decoding algorithm should be able to use a phrase pair even if it is not in the intersection of the two (or more) given phrase tables.

2.5 Summary

In this chapter, we reviewed all the background knowledge needed to follow the rest of the thesis, including: Semi-supervised Learning, Active Learning, and Phrase-based Statistical Machine Translation. The semi-supervised learning algorithms that we saw in this chapter belong to self-training style algorithms: those which learn from their own output in an iterative manner. This is similar to active learning with one major difference: In active learning the labels (for the selected instances) are provided by human whereas in self-training the predictions of the re-trained model are used as the *pseudo-labels*. In the following chapter, I will theoretically investigate why we should expect our model to learn from self-training.

Chapter 3

An Analysis for Self-Training

The Yarowsky algorithm (Yarowsky, 1995) is a rule-based semi-supervised learning algorithm that has been successfully applied to many problems in computational linguistics. The algorithm was not mathematically well understood until (Abney, 2004) which analyzed some specific variants of the algorithm, and also proposed some new algorithms for bootstrapping. In this chapter, I extend Abney's work and show that some of his proposed algorithms indeed optimize (an upper-bound on) an objective function based on a new definition of cross-entropy which is based on a particular instantiation of the Bregman distance between probability distributions. Moreover, I suggest some new algorithms for rule-based semi-supervised learning and show connections with harmonic functions and minimum multi-way cuts in graph-based semi-supervised learning. The results in this chapter are based on (Haffari and Sarkar, 2007).

3.1 The Motivation

The Yarowsky algorithm (see Algorithm 2 in Section 2.2.3) is an iterative bootstrapping algorithm where in each iteration a classifier is built based on the current set of labeled data, the learned classifier is used to produce label for unlabeled data, and a subset of newly labeled data contributes new features for the classifier trained for the next iteration. Despite success in various experimental studies, the Yarowsky algorithm was not mathematically well understood until (Abney, 2004) which tried to show that the Yarowsky algorithm minimizes a reasonable objective function. In fact (Abney, 2004) did not give an objective function for the original algorithm in (Yarowsky, 1995) but introduced some variants of the original

algorithm which were shown to optimize reasonable objective functions.

In this chapter, I extend Abney's work and show that some of his proposed algorithms, which are very similar to the original Yarowsky algorithm, optimize (an upper-bound on) a reasonable objective function based on a particular instantiation of the Bregman distance between probability distributions and a new definition of cross-entropy. My analysis and the contrast with the analysis in (Abney, 2004) is given in Section 3.3. With this analysis, I hope to provide a general theoretical framework for the analysis of bootstrapping (or self-training) algorithms for semi-supervised learning with the use of Bregman distances. In Section 3.4, I show how one can view variants of some well known models for graph-based semi-supervised learning, such as the models described in (Blum and Chawla, 2001; Zhu, Ghahramani, and Lafferty, 2003), as particular instantiations of our general model. In addition, in Section 3.5, I formulate a new variant of the Yarowsky algorithm which is suggested by my analysis in Section 3.3.

As we saw in Section 2.2.4, Co-training is another algorithm for learning from labeled and unlabeled data, which assumes that each data point can be described by two distinct *views*. Co-training learns two classifiers, one for each view, and the PAC-learning analysis by (Dasgupta, Littman, and McAllester, 2001) shows that the classifier trained on one view has low generalization error if it agrees on unlabeled data with the classifier trained on the other view. The assumption under which this result holds is that each view is conditionally independent of the other view given the class label. However, (Abney, 2002) shows that this assumption of view independence is remarkably powerful, and often violated in real data. In contrast to co-training, the Yarowsky algorithm uses only a single classifier. (Abney, 2004) shows that one can view the Yarowsky algorithm as minimizing the label disagreement between pairs of features in the classifier. The goal of this chapter is to further study the Yarowsky algorithm, as it could be an attractive alternative to the co-training algorithm in many settings in which one wishes to explore learning by bootstrapping.

It is instructive to have the following NLP problem in mind when reading this chapter: The "word sense disambiguation" problem, upon which Yarowsky originally introduced his algorithm (Yarowsky, 1995). Word sense disambiguation (WSD) is the process of identifying which sense (meaning) of a word is used in any given sentence, when the word has a number of distinct senses. For example, consider the word "plant" in the following two sentences:

- ... company said the *plant* is still operating .

- Although thousands of *plant* and animal species ...

It is clear that "plant" in the first and second sentences refer to different senses: A factory (sense A) and a living organism (sense B), respectively. We can represent the word in question by some important words which surround it, and then use the Decision Lists (DLs) to decide about the correct sense of the word. In the above example, the first occurrence of the word "plant" can be represented by *(company,operating)* and the second occurrence can be represented by *(animal,species)*. Then the first applicable rule in the following DL can be fired to specify one of the two senses:

Collocation	Sense
animal (within $\pm$1 words)	$\Rightarrow$ A
factory (within $\pm$2 words)	$\Rightarrow$ B
species (within $\pm$2 words)	$\Rightarrow$ A
...	

Equivalently, to each rule we can assign a score, and then among all applicable rules, accept the prediction of the rule which has the highest score. The rule weights/scores (which specify the ordering of the rules) are the parameters of the DL classifier and are subject of learning.

3.2 The Modified Yarowsky Algorithm

The modified Yarowsky algorithm is introduced in (Abney, 2004) and is shown in Algorithm 4. The algorithms that we analyze in this chapter are derived by instantiating different components of this algorithm. Table 3.1 summarizes the notation that I use throughout this chapter.

The modified Yarowsky algorithm differs from the original algorithm as follows (compare Algorithm 4 with Algorithm 2):

- Once an unlabeled example gets labeled, it stays labeled (its label may change after recomputing the labels)
- The labeling threshold is fixed to be $\frac{1}{L}$ (instead of an additional threshold parameter ζ as used in the original Yarowsky algorithm).

X	The set of labeled and unlabeled examples
L	The number of possible labels $\|\mathcal{Y}\|$
$\Lambda^{(t)}$	The (current) set of labeled examples
$V^{(t)}$	The (current) set of unlabeled examples
$\pi_x(y)$	Prediction distribution of x based on the model (see equation 3.3)
$\phi_x(y)$	Labeling distribution
X_f, Λ_f, V_f	Examples/labeled examples/unlabeled examples having the feature f
F_x	Features of example x
Λ_{fy}	Examples having the feature f and labeled y
$q_{fy}, \hat{q}_{fy}$	Precision/smoothed precision of the rule $f \to y$ (see equations 3.4 and 3.5)
θ_{fy}	Score for the rule $f \to y$; we view θ_f as the prediction distribution of f

Table 3.1: Summary of the notation.

Algorithm 4 The modified Yarowsky algorithm

Initially set pool of training data $\Lambda^{(0)}$ to D_l
for $t \in \{1, 2, \ldots\}$ **do**
  Train classifier $h^{(t)}$ based on the current pool of training data $\Lambda^{(t-1)}$
  $\Lambda^{(t)} := \Lambda^{(t-1)}$
  for each instance x in the unlabeled data **do**
    Construct prediction distribution $\pi_x^{(t)}$ for instance x where $\pi_x^{(t)}(y)$ shows the probability that classifier $h^{(t)}$ predicts label y for x.
    Set $\hat{y} := \arg\max_{y \in \mathcal{Y}} \pi_x^{(t)}(y)$
    if $\pi_x(\hat{y}) > \frac{1}{L}$ **or** x is labeled in $\Lambda^{(t-1)}$ **then**
      Add $(x, \hat{y})$ to $\Lambda^{(t)}$ (do not remove x from unlabeled data)
    end if
    if $\Lambda^{(t)} = \Lambda^{(t-1)}$ **then** Stop
  end for
end for

3.2.1 The Objective Function

Let $\phi_x(j)$ be the probability that instance x belong to the jth class. If a labeled instance x belongs to the class j, the labeling distribution ϕ_x has all of its mass on the label j, and if x is unlabeled the labeling distribution is uniform over all possible labels. The proposed objective function is the cross entropy between the *prediction distribution* of the model π_x and the *labeling distribution* ϕ_x over *all* (labeled and unlabeled) instances:

$$l(\phi, \theta) \stackrel{\triangle}{=} \sum_{x \in X} H(\phi_x \parallel \pi_x) \tag{3.1}$$

where $H(\phi_x \parallel \pi_x) = \sum_j \phi_x(j) \log \frac{1}{\pi_x(j)}$. The cross entropy is related to the Shannon entropy and the KL-divergence as follows

$$H(\phi_x \parallel \pi_x) = H(\phi_x) + D(\phi_x \parallel \pi_x) \tag{3.2}$$

where $H(\phi_x) = -\sum_j \phi_x(j) \log \phi_x(j)$ is the Shannon entropy, and $D(\phi_x \parallel \pi_x) = \sum_j \phi_x(j) \log \frac{\phi_x(j)}{\pi_x(j)}$ is the KL-divergence.

Consider the objective function (3.1). For labeled instances the entropy of the labeling distribution is zero, and the contribution to the objective function is $\sum_{x \in \Lambda} D(\phi_x \parallel \pi_x)$. This implies that the model parameters should be chosen so that the likelihood of the labeled data is maximized. For an unlabeled data point, the contribution is $H(\phi_x \parallel \pi_x) = H(\phi_x) + D(\phi_x \parallel \pi_x)$ which is minimized when the entropy goes to zero, i.e. it becomes labeled, and its assigned label agrees with the model prediction. Therefore minimizing the objective function forces unlabeled data to be labeled, and forces the model to maximize the likelihood of the (old and newly) labeled data. Our objective function (3.1) is different from the objective function of conditional entropy regularization (Grandvalet and Bengio, 2004):

$$\sum_{x \in \Lambda^{(0)}} H(\phi_x \parallel \pi_x) + \gamma \sum_{x \in V^{(0)}} H(\pi_x)$$

where $\Lambda^{(0)}$ is the labeled data, $V^{(0)}$ is the unlabeled data, and γ is used to control influence of unlabeled examples.

3.3 Specific Yarowsky algorithms

In this section I introduce some variants of Algorithm 4 defined in (Abney, 2004) which use a specific base learner. I then provide an analysis of one of these variants. We will see that

these variants optimize different objective functions, which may help us to discover different data conditions (i.e. properties of the data) that are well captured by these variants. In other words, the analysis may help us to formalize which variant is suitable for which data conditions.

3.3.1 The DL-0 algorithm

In the original Yarowsky algorithm (henceforth, the DL-0 algorithm where DL stands for Decision List), the base learner builds a decision list which consists of rules $f \to j$ showing that feature f predicts label j with the confidence score θ_{fj}. The number of all possible features is N and we assume that each instance x has a subset F_x of them. We can represent the relation between the features $F_x \in F$ and instances $x \in X$ as a bipartite graph as shown in Figure 3.1. Each example (in the right column) is connected to its features (in the left column). The labeled data have fixed labels, e.g. in Figure 3.1 one point is labeled '+' and one is labeled '−'. This representation is similar to the one used in (Corduneanu, 2006). We further develop the relation between DL-0 and this bipartite graph representation in Section 3.4.

Among the rules matching the features of an instance x, the DL-0 algorithm selects the one which has the highest score and assigns the label predicted with that rule to x. In our terminology, this is equivalent to defining

$$\pi_x(j) \propto \max_{f \in F_x} \theta_{fj} \tag{3.3}$$

and then assigning the label '$\arg\max_j \pi_x(j)$' to x. We assume the scores are non-negative and bounded, so they can be normalized for each instance to form a prediction distribution π_x.

The base learner in the DL-0 algorithm uses the smoothed precision to update θ_{fj}. Define the *precision* q_{fj} of a rule $f \to j$ to be

$$q_{fj} = \frac{|\Lambda_{fj}|}{|\Lambda_f|} \tag{3.4}$$

where Λ_{fj} is the set of all instances having the feature f and the label j, and Λ_f is the set of all instances having the feature f. *Smoothed precision* is defined by adding ϵ to the counts in the numerator:

$$\hat{q}_{fj} = \frac{|\Lambda_{fj}| + \epsilon}{|\Lambda_f| + L\epsilon} \tag{3.5}$$

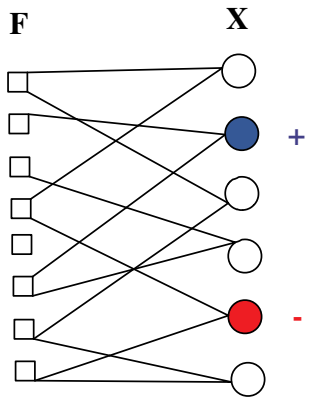


Figure 3.1: A bipartite graph representing the relationship between data points X and features F. Each data point has been connected to its features by undirected edges, and some data points are initially labeled.

In the Yarowsky algorithm, a rule's score is taken to be its smoothed precision $\theta_{fj} = \hat{q}_{fj}$. Note that θ_{fj} are the model parameters which characterize the classifier in a space of classifiers $\mathcal{H}$. Both raw precision and smoothed precision can be viewed as conditional probability distributions. We take the vector $\theta_f \equiv (\theta_{f1}, \ldots, \theta_{fL})$ to have the properties of a conditional probability distribution $\theta_{fj} = p(j \mid f)$ in the rest of this chapter.

It is not known which objective function is minimized by the DL-0 algorithm (with the definition (3.3) for the prediction distribution and updating rule (3.5) for the parameters).

3.3.2 The DL-1 Algorithm

The DL-1 algorithm differs in two ways from the DL-0 algorithm. The first difference is in the definition of the prediction distribution. For an instance x the prediction distribution is constructed as follows:

$$\pi_x(j) \propto \sum_{f \in F_x} \theta_{fj} \tag{3.6}$$

The scores of each feature sum to one, hence $\pi_x(j) = \frac{1}{|F_x|} \sum_{f \in F_x} \theta_{fj}$. The definition (3.6) says that for an instance x the prediction distribution is a mixture of its features' distributions.

Second, the DL-1 algorithm[1] uses the updating rule for parameters which differs slightly

[1]In fact two variants of the DL-1 are introduced in (Abney, 2004): (i) DL-1-R which uses the the raw precision (3.4) for updating the parameters, and (ii) DL-1-VS. In our discussion, we just analyze DL-1-VS

from smoothed precision (3.5):

$$\theta_{fj} = \frac{|\Lambda_{fj}| + \frac{1}{L}|V_f|}{|\Lambda_f| + |V_f|} \tag{3.7}$$

where the smoothing constant is different for different features, in fact ϵ in (5) is replaced with $\epsilon_f = \frac{1}{L}|V_f|$.

3.3.2.1 Analysis of the DL-1 Algorithm

I prove in this section that the DL-1 algorithm optimizes the upper-bound K_{t^2} (which will be defined shortly) on the objective function (3.1) but based on a new definition of cross entropy. As we will see, the new definition of cross entropy is naturally derived based on a particular instantiation of the Bregman distance between two probability distributions.

Let ψ be a strictly convex real-valued function, the Bregman distance $B_\psi(\mathbf{p}, \mathbf{q})$ between two discrete probability distributions $\mathbf{p}$ and $\mathbf{q}$ is defined as follows (Lafferty, Pietra, and Pietra, 1997):

$$\begin{aligned} B_\psi(\mathbf{p}, \mathbf{q}) &\stackrel{\triangle}{=} \sum_i \psi(p_i) - \psi(q_i) - \psi'(q_i)(p_i - q_i) \\ &= \Psi(\mathbf{p}) - \Psi(\mathbf{q}) - \nabla\Psi(\mathbf{q}) \cdot (\mathbf{p} - \mathbf{q}) \end{aligned}$$

and the ψ-entropy of a discrete probability distribution $\mathbf{p}$ is defined as $H_\psi(\mathbf{p}) = -\sum_i \psi(p_i)$. When $\psi(t) = t^2$ we obtain the mean squared distance $B_{t^2}(\mathbf{p}, \mathbf{q}) = \sum_i (p_i - q_i)^2$, and the entropy as $H_{t^2}(\mathbf{p}) = -\sum_i p_i^2$. Moreover, if we let $\psi(t) = t \log t$, then $H_{t \log t}$ is the Shannon entropy and $B_{t \log t}(\mathbf{p}, \mathbf{q})$ is the KL-Divergence between these two probability distributions.

In the original formulation (3.2), cross entropy is equal to the Shannon entropy plus KL-Divergence. So we define the ψ-cross entropy between $\mathbf{p}$ and $\mathbf{q}$ to have the same relation:

$$\begin{aligned} H_\psi(\mathbf{p} \parallel \mathbf{q}) &\stackrel{\triangle}{=} H_\psi(\mathbf{p}) + B_\psi(\mathbf{p}, \mathbf{q}) \\ &= -\Psi(\mathbf{q}) - \nabla\Psi(\mathbf{q}) \cdot (\mathbf{p} - \mathbf{q}) \end{aligned}$$

In particular when $\psi(t) = t^2$, we will have $H_{t^2}(\mathbf{p} \parallel \mathbf{q}) = \sum_j q_j^2 - 2p_j q_j$. Let $l_{t^2}(\phi, \theta)$ refer to $l(\phi, \theta)$ when the new definition of cross entropy is used; then

$$\begin{aligned} l_{t^2}(\phi, \theta) &= \sum_{x \in X} H_{t^2}(\phi_x \parallel \pi_x) \\ &= \sum_{x \in X} \sum_j \pi_x^2(j) - 2\pi_x(j)\phi_x(j) \end{aligned} \tag{3.8}$$

---

and simply call it DL-1.

I will show that the DL-1 algorithm locally minimizes the following objective function

$$K_{t^2}(\phi, \theta) \stackrel{\triangle}{=} \sum_{x \in X} \sum_{f \in F_x} \Big[H_{t^2}(\phi_x \parallel \theta_f) \Big]$$

where $\frac{1}{m} K_{t^2}(\phi, \theta)$ is an upper-bound on $l_{t^2}(\phi, \theta)$ if we assume that instances have the same number of features, i.e. $|F_x|$ equals m for each instance x. This assumption is also used in (Abney, 2004) and is easy to satisfy when the instances do not have the same number of features by simply adding new features to the model.

Lemma 1. *$l_{t^2}(\phi, \theta)$ is upper-bounded by $\frac{1}{m} K_{t^2}(\phi, \theta)$.*

Proof.

$$\frac{1}{m} K_{t^2}(\phi, \theta) - l_{t^2}(\phi, \theta) =$$
$$\frac{1}{m^2} \sum_{x \in X} \sum_{j} \left[m \sum_{f \in F_x} \theta_{fj}^2 - \Big(\sum_{f \in F_x} \theta_{fj} \Big)^2 \right] \geq 0$$

which is true for all integers $m \geq 1$ based on the fact that $|F_x| = m$. □

Theorem 2. *The DL-1 algorithm minimizes the objective function $\frac{1}{m} K_{t^2}(\phi, \theta)$ which is an upper-bound for $l_{t^2}(\phi, \theta)$.*

Proof. In each iteration of the DL-1 algorithm, we first update the parameters θ_{fj} and then recompute the labeling distribution ϕ_x for each example $x \notin \Lambda^0$.

Having the parameters θ_{fj} fixed, we show that the labeling distributions chosen by the algorithms reduce the contribution of each instance x in $l_{t^2}(\phi, \theta)$ which is denoted by $l_{t^2}(\phi_x, \theta)$. There are three possible cases for the labeling of an instance $x \notin \Lambda^0$ in the previous and current iterations:
1) x is unlabeled according to $\phi_x^{(t-1)}$ and $\phi_x^{(t)}$. It means both distributions are uniform and $\phi_x^{(t-1)} = \phi_x^{(t)}$, so $l_{t^2}(\phi_x, \theta)$ has not been changed after recomputing the labeling distribution.
2) x is labeled according to $\phi^{(t-1)}$ and $\phi^{(t)}$. The label chosen by the DL-1 algorithms is $\arg\max_j \pi_x(j)$ which reduces $l_{t^2}(\phi_x, \theta)$ as much as possible. We view $l_{t^2}(\phi_x, \theta)$ as a function

of ϕ_x:

$$\begin{aligned} l_{t^2}(\phi_x, \theta) &= \sum_{f \in F_x} \sum_j \theta_{fj}^2 - 2\phi_x(j)\theta_{fj} \\ &= \text{Const} - 2|F_x| \sum_j \phi_x(j) \frac{\sum_{f \in F_x} \theta_{fj}}{|F_x|} \\ &= \text{Const} - 2|F_x| \sum_j \phi_x(j)\pi_x(j) \end{aligned}$$

3) x is unlabeled according to $\phi^{(t-1)}$ but is labeled according to $\phi^{(t)}$. Before recomputing the labeling distribution $l_{t^2}(\phi_x, \theta) = \text{Const} - 2|F_x|\frac{1}{L}$ since $\phi_x^{(t-1)}$ is a uniform distribution. After recomputing the labeling distribution $l_{t^2}(\phi_x, \theta) = \text{Const} - 2|F_x| \max_j \pi_x(j)$. It is clear that $\max_j \pi_x(j) > \frac{1}{L}$ (because the instance becomes labeled), therefore the value of $l_{t^2}(\phi_x, \theta)$ has been reduced by labeling the instance.

Now assume that ϕ is fixed; the goal is to prove that when the algorithms update the the parameters θ_{fj}, the objective function $l_{t^2}(\phi, \theta)$ is reduced. The Lagrangian for the constrained optimization problem is

$$\begin{aligned} \mathcal{L}(\theta, \lambda) = &\sum_{x \in X} \Big(\sum_{f \in F_x} \sum_j \theta_{fj}^2 - 2\theta_{fj}\phi_x(j) \Big) \\ &+ \sum_{f \in F} \lambda_f \Big(\sum_j \theta_{fj} - 1 \Big) \end{aligned}$$

The optimality condition $\partial\mathcal{L}(\theta\lambda)/\partial\theta_{fk} = 0$ yields

$$\begin{aligned} \partial\mathcal{L}(\theta, \lambda)/\partial\theta_{fk} &= \sum_{x \in X_f} 2\theta_{fk} - 2\phi_x(k) + \lambda_f = 0 \\ &\Rightarrow \quad \theta_{fk} = \frac{|\Lambda_{fk}| + \frac{1}{L}|V_f|}{|X_f|} \end{aligned} \tag{3.9}$$

which is exactly the updating rule for the parameters in DL-1 algorithm. □

If we do not assume that instances have the same number of features (namely m), it is still true that the DL-1 algorithm minimizes $K_{t^2}(\phi, \theta)$. However it would not necessarily be the case that l_{t^2} is upper-bounded by $\frac{1}{m}K_{t^2}$.

The analysis in (Abney, 2004) provided an upper-bound on the objective function for

DL-1, but based on the conventional definition of cross-entropy:

$$\begin{aligned} K(\phi,\theta) &\stackrel{\triangle}{=} \sum_{x\in X}\sum_{f\in F_x}\Big[H(\phi_x \parallel \theta_f)\Big] \\ &= \sum_{x\in X}\sum_{f\in F_x}\sum_{j}\phi_x(j)\log\frac{1}{\theta_{fj}} \end{aligned}$$

This definition leads to an error in the proof provided in (Abney, 2004) which attempts to show the relation between minimizing $K(\phi,\theta)$ and the label chosen by the DL-1 algorithm. The DL-1 algorithm chooses the label $\arg\max_j \pi_x(j) = \arg\max_j \sum_j \theta_{fj}$. However, K is minimized as a function of ϕ (which is done per instance x) by selecting the label $\arg\min_j \sum_{f\in F_x} \log\frac{1}{\theta_{fj}}$. In general, $\arg\min_j \sum_{f\in F_x} \log\frac{1}{\theta_{fj}} \neq \arg\max_j \sum_j \theta_{fj}$. Our main result in this section, using the framework of (Abney 2004), provides an objective function $K_{t^2}(\phi,\theta)$ that, when minimized, provides the same labeling and parameter updates as the DL-1 algorithm.

3.4 Extensions of the DL-1 Algorithm

In this section I generalize the ideas in the previous section to present a general formulation of the graph-based semi-supervised learning. Consider Figure 3.1 which represents the relationship between instances and features by a bipartite graph: Each feature induces a *bias* to enforce the label similarity of its neighboring instances. In other words, the label of data points adjacent to a common feature should be similar. By noticing that data points may have overlapping features, this induces constraints among the labels of data points.

For the set of instances X_f adjacent to a feature f, we assign a probability distribution (over labels) to f and require the label of all of its neighbors to have a small distance to it by minimizing $\sum_{x\in X_f} B_\psi(\theta_f,\phi_x)$. The overall objective function for minimization is:

$$\sum_{f\in F}\sum_{x\in X_f} B_\psi(\theta_f,\phi_x) \tag{3.10}$$

The above objective function is simply the sum over the distances of probability distributions of nodes connected by edges in the graph. We will show that some well-known models like (Zhu, Ghahramani, and Lafferty, 2003) can be seen as instantiations of this general model. In particular when $\psi(t) = t^2$, the objective function becomes:

$$\sum_{f\in F}\sum_{x\in X_f}\sum_{j=1}^{L}(\theta_{fj}-\phi_x(j))^2 \tag{3.11}$$

with the constraint that the labeling distribution of initially labeled points is fixed. It is not hard to see that this is a convex optimization program and at the optimum, the labeling distribution of every node in the graph should be the average of its neighbors. In the case of binary classification a real valued function is defined on the graph nodes, where the value of each node shows the probability of belonging to one of the output labels. In this setting the optimum function is *harmonic*, which is the mean of the Gaussian random field defined by the graph and can be computed by label propagation by performing a random walk on the graph (Zhu, Ghahramani, and Lafferty, 2003). Here the vertices of the underlying graph consists of both instances and features, in contrast in (Zhu, Ghahramani, and Lafferty, 2003) the vertices of the underlying graph are instances.

Suppose we want the labeling distribution of each data point to have a small entropy $-\sum_k \phi_x^2(k)$. We add the penalty term $-\sum_{x\in X} |F_x| \sum_j \phi_x^2(j)$ to the objective function (3.11) where the weights $|F_x|$ show the emphasis on different instances, and as the result we get the final objective function (3.8). The optimization problem is not convex, and the way that DL-1 algorithm propagates labels to reach a stationary solution is important. In addition to data points, imagine we further would like each feature to have a small entropy over its soft label. We can add another penalty term $-\sum_{f\in F}(|X_f| \sum_j \phi_x^2(j))$ to the objective function (the weights show the emphasis on different features), and with some simplification we get:

$$-2 \sum_{f\in F} \sum_{x\in X_f} \sum_{j=1}^{L} \phi_x(j) \times \theta_{fj} \tag{3.12}$$

In order to optimize the above objective function consider Algorithm 5 which is the modified Yarowsky algorithm specialized for this case. The function **majority**(S) returns a probability distribution over the labels. If all labels are supported by equal number of nodes then the resulting probability distribution is uniform, otherwise the label which is supported by the majority of nodes in S gets the total probability mass (if more than one label gets the highest number of supporters, one of them is chosen arbitrarily). Without loss of generality, we can ignore votes of the nodes in S having the uniform distribution over labels. Note that in Algorithm 5 once an unlabeled node is labeled in some iteration, it retains that label even if the majority function returns the uniform distribution in a future iteration.

It is easy to prove that the objective function (3.12) can be minimized, as a function of θ_f, by putting all of the mass of its probability distribution on the label k which has the largest contribution $\sum_{x\in X_f} \theta_{xk}$. The largest value for $\sum_{x\in X_f} \theta_{xk}$ corresponds to the label

Algorithm 5 Majority-Majority

```
repeat
  for f ∈ F do
    p ← majority(X_f)
    if p is not a uniform distribution then
      θ_f ← p
  end for
  for x ∈ X do
    q ← majority(F_x)
    if q is not a uniform distribution then
      φ_x ← q
  end for
until the labels do not change
```

which is the majority label in the neighbors of f. The same argument works for the second phase of the iterations where we update the label of instances. Moreover it can be shown that Algorithm 5 converges in polynomial time.

Lemma 3. *Suppose in some iteration in Algorithm 5, r nodes in the right and l nodes in the left columns are labeled in the bipartite graph (ignore the other nodes of the graph). The recomputation of the labels for these nodes (the main loop of the algorithm) terminates in no more than $l \times r$ iterations.*

Proof. Suppose there are only two labels: $+1$ and -1. Put all $+1$ nodes in the set A and all -1 nodes in the set B. Consider recomputing the label of a node v in the graph: it is given the label which majority of its neighbors have. It means that if majority of its neighbors are in the set A (or B) then this node is also put in the same set. After recomputing the label of a node v, if it is transferred from one set to another then the number of edges between sets A and B, i.e. the *cut* size, is reduced by at least one. Now consider recomputing the labels of all nodes in the right (left) column of the graph in parallel: it reduces the cut size because the right (left) nodes do not have any edge among them. Each iteration, if a different label is produced for at least one node, causes the cut size to be reduced by at least one. The maximum number of edges between the right and left nodes is $r \times l$, and in the worst case the cut size reaches zero after $l \times r$ iterations or the algorithm terminates before this point. □

Theorem 4. *Algorithm 5 stops after at most $\mathcal{O}(|F|^2|X|^2)$ iterations.*

Proof. Consider the situation in which r nodes in the right and l nodes in the left are labeled. Due to Lemma 3, at the worst case after $l \times r$ iterations their labels are stabilized. However, the addition of even one node from the unlabeled nodes to this set of currently labeled data (before label stabilization) can stop the convergence of the algorithm, and cause it to continue for more iterations. Consider the worst case where only one node is added in the $(l \times r)$th iteration: it may be added to the left nodes or to the right nodes. So the number of iterations needed for the convergence of the next round of the algorithm is upper-bounded by $l \times (r+1) + (l+1) \times r$. Continuing this reasoning, it can be seen that the number of iterations needed for the convergence of the algorithm is at most $\sum_{i=1}^{|F|} \sum_{j=1}^{|X|} i \times j = \mathcal{O}(|F|^2 |X|^2)$ □

Define a new operator **average**(S) that returns a probability distribution which is the (normalized) sum of probability distributions associated with nodes in the set S. Replacing **majority** in line 3 of Algorithm 5 with **average** gives the DL-1 algorithm, and replacing **majority** in lines 3 *and* 8 with **average** gives the harmonic function based method.

Suppose we arbitrarily break the ties and reach a stationary point where all nodes of the bipartite graph are labeled. Put all nodes from class 1 in one set, all nodes from class 2 in another set, and so on. Picking a node and changing its set will increase the cost of the L-way cut in the graph. In fact the algorithm greedily partitions the nodes into L sets and reaches a (local) minimum L-way cut that agrees with the labeled instances. In the binary case, it is similar to semi-supervised learning using mincut (Blum and Chawla, 2001) with the difference that here both instances and features are nodes of the underlying graph, but in (Blum and Chawla, 2001) only the instances are the vertices of the underlying graph.

Consider minimizing the objective function in (3.10) for the general class of the Bregman distances with the constraint that the labeling distributions of the initially labeled data are fixed. The minimization is done on the distributions associated with features and unlabeled data. We may add more constraints and require the probability distributions belong to specific classes of models, e.g. $\theta_f \in \mathcal{M}_{\mathcal{F}}$ where $\mathcal{M}_{\mathcal{F}}$ is a particular class of probability distributions. For the case where $\mathcal{M}_{\mathcal{F}}$ is unrestricted, the optimal probability distributions must satisfy the following optimality conditions

$$\forall f \in F : \theta_f^{opt} \propto \nabla \Psi^* \Big(\frac{1}{|X_f|} \sum_{x \in X_f} \nabla \Psi(\phi_x^{opt}) \Big)$$

where Ψ^* is the convex conjugate of the convex function Ψ, and

$$\forall x \in V : \sum_{f \in F_x} \left(\theta_{fj}^{opt} - \phi_x^{opt}(j)\right)\psi''(\phi_x^{opt}(j)) = c_x$$

$\forall j \in \{1, .., L\}$, where c_x is a constant associated with instance x.

3.5 The DL-2 Algorithms

In this section I propose and analyze a new family of specific Yarowsky algorithms called the DL-2 family of algorithms. The algorithms in this family are named DL-2-S and DL-2-ML. The prediction distribution for the DL-2 family of algorithms is constructed by:

$$\pi_x(j) \propto \prod_{f \in F_x} \theta_{fj} \tag{3.13}$$

with the normalization constant $Z_x = \sum_k \prod_{f \in F_x} \theta_{fk}$.

The intuition behind the prediction distribution (3.13) is as follows. Associated with each feature f, consider a biased die with L faces and the parameter θ_{fj} for the jth face coming up. For an instance x, we toss all of the dice associated with features $f \in F_x$; what is the probability of seeing face j in each die given we already know that all dice should show the same face? The answer is the expression given in (3.13). This way of combining predictions is similar to the Product of Experts (PoE) framework (Hinton, 1999). In PoE, prediction distributions of simple models, or "experts", are multiplied together and then normalized to produce a sharp high dimensional prediction distribution. While equation (3.13) is similar to PoE, our parameter updates are very different, and we use it in order to provide a novel semi-supervised algorithm.

The base learner in DL-2-S is very similar to the base learner in DL-0 in that it uses the smoothed precision to update the parameters:

$$\theta_{fj} = \frac{|\Lambda_{fj}| + \frac{1}{L}(|V_f| + \delta|X_f|)}{|\Lambda_f| + |V_f| + \delta|X_f|} \tag{3.14}$$

where δ is a given smoothness parameter. The smoothing constant is different for different features, in fact $\epsilon_f = \frac{1}{L}(|V_f| + \delta|X_f|)$.

In DL-2-ML, instead of learning a new classifier from scratch, the classifier in the previous iteration is improved to yield the new one. Updating the parameters is done in the direction of minimizing H. The only algorithm in this family which minimizes H given in (3.1) is DL-2-ML.

3.5.1 Analysis of DL-2-S Algorithm

I prove that DL-2-S minimizes the following objective function which is an upper-bound on H:

$$K_\delta \stackrel{\triangle}{=} \sum_{x \in X} \sum_{f \in F_x} \Big[H(\phi_x \parallel \theta_f) + \delta \cdot H(u \parallel \theta_f) \Big] \tag{3.15}$$

where u is the uniform distribution over the labels. We start the analysis by proving the following lemma.

Lemma 5. *H is upper-bounded by K_δ for all $\delta \geq 0$.*

Proof.

$$\begin{aligned} \prod_{f \in F_x} 1 &= \prod_{f \in F_x} \sum_j \theta_{fj} = \\ &= \underbrace{\sum_j \Big(\prod_{f \in F_x} \theta_{fj} \Big)}_{Z_x} + \text{some positive terms} \end{aligned}$$

hence $\forall x, Z_x \leq 1$. Now we prove the lemma:

$$\begin{aligned} H &= \sum_x H(\phi_x \parallel \pi_x) \\ &= \sum_x \sum_j \phi_x(j) \Big(\log Z_x - \log \prod_{f \in F_x} \theta_{fj} \Big) \\ &= \sum_x \Big(\log Z_x - \sum_f \sum_j \phi_x(j) \log \theta_{fj} \Big) \\ &= \sum_x \Big(\log Z_x + \sum_f H(\phi_x \parallel \theta_f) \Big) \\ &\leq \sum_x \sum_f H(\phi_x \parallel \theta_f) \end{aligned}$$

The inequality holds because $\log a \leq 0$ for $a \leq 1$.

Since $\delta \geq 0$ and cross-entropy is always a non-negative function, it follows that $H \leq K_\delta$. □

The following lemma is used to prove the main theorem of this section.

Lemma 6. *If the labeling distribution ϕ is fixed, then updating the parameters θ_{fj} based on the smoothed precision (3.14) minimizes K_δ.*

Proof. We should minimize K_δ subject to the constraints $\forall f, \sum_j \theta_{fj} = 1$. The Lagrangian associated with the optimization problem is

$$\begin{aligned}\mathcal{L}(\theta, \lambda) &= \sum_{x \in X} \sum_{f \in F_x} \Big(H(\phi_x \parallel \theta_f) + \delta \cdot H(u \parallel \theta_f) \Big) \\ &\quad + \sum_f \lambda_f \Big(\sum_j \theta_{fj} - 1 \Big) \\ &= \sum_{x \in X} \sum_{f \in F_x} \Big(\sum_j \phi_x(j) \log \frac{1}{\theta_{fj}} + \delta \sum_j \frac{1}{L} \log \frac{1}{\theta_{fj}} \Big) \\ &\quad + \sum_f \lambda_f \Big(\sum_j \theta_{fj} - 1 \Big)\end{aligned}$$

The optimality condition is $\partial \mathcal{L}(\theta, \lambda) / \partial \theta_{fk} = 0$, which yields

$$\begin{aligned}\partial \mathcal{L}(\theta, \lambda) / \partial \theta_{fk} = &- \sum_{x \in \Lambda_f} \Big[\frac{\phi_x(k)}{\theta_{fk}} + \frac{\delta}{L \theta_{fk}} \Big] \\ &- \sum_{x \in V_f} \Big[\frac{1}{L \theta_{fk}} + \frac{\delta}{L \theta_{fk}} \Big] + \lambda_f = 0\end{aligned}$$

hence the smoothed precision (3.14) updating rule is derived. □

Theorem 7. *DL-2-S minimizes the objective function K_δ which is an upper-bound on H for every $\delta \geq 0$.*

Proof. The proof is similar to the proof of Theorem 2. In Lemma 5, it was proved that K_δ is an upper-bound on H for every $\delta \geq 0$. In each iteration of the DL-2-S algorithm, we first update the parameters θ_{fj} and then recompute the labeling distribution ϕ_x for each example $x \notin \Lambda^0$. We show that each of these two phases reduces the objective function.

Having the parameters θ_{fj} fixed, the labeling distributions chosen by the algorithms reduces $K_\delta(x)$ for each instance x. If x is labeled according to $\phi^{(t-1)}$ and $\phi^{(t)}$, the label chosen by the algorithms is

$$\arg\max_j \pi_x(j) = \arg\max_j \sum_f \log \theta_{fj} \tag{3.16}$$

Now look at $K_\delta(x)$ as a function of ϕ_x:

$$\begin{aligned}K_\delta(x) &= \sum_{f \in F_x} H(\phi_x \parallel \theta_f) + \delta \cdot H(u \parallel \theta_f) \\ &= \sum_j \phi_x(j) \sum_{f \in F_x} \log \frac{1}{\theta_{fj}} + \text{Const}\end{aligned}$$

It is clear that the labeling distribution chosen by the algorithms in (3.16) reduces $K_\delta(x)$ as much as possible (the other two cases where changing the label distribution reduces $K_\delta(x)$ is easy to verify).

Now assume that ϕ is fixed. In Lemma 6 it is proved that the DL-2-S algorithm updates the parameters in such a way that K_δ is minimized, which concludes the proof. □

As a special case if we set $\delta = 0$, the algorithm minimizes the following objective function

$$\sum_{x \in X} \sum_{f \in F_x} H(\phi_x \parallel \theta_f)$$

based on the following update rule for the parameters

$$\theta_{fj} = \frac{|\Lambda_{fj}| + \frac{1}{L}|V_f|}{|\Lambda_f| + |V_f|}$$

When $\delta = 0$ we obtain a direct relationship with the parameter update rule for the DL-1 algorithm in (3.7).

3.5.2 Analysis of the DL-2-ML Algorithm

Using a log-linear formulation of π_x we find a simple formula for the gradient of H which can then be used by DL-2-ML to minimize H. Let $\theta_{fj} = \exp(w_{fj})$, in other words the model is parametrized by w_{fj}. Denotes the vector of parameters corresponding to the ith label by $\mathbf{w}_i$, i.e. $\mathbf{w}_i = (w_{f_1 i}, .., w_{f_N i})$. Let $\mathbf{w}$ to be the vector of all parameters in the model , i.e. $\mathbf{w} = (\mathbf{w}_1, .., \mathbf{w}_L) \in \mathcal{R}^{NL}$.

For each training pair (x,y) define

$$\Phi(x, y) = \lambda(y) \otimes x$$

where $\otimes$ denotes a tensor product, and $\lambda(y) \in \{0, 1\}^L$ is a vector having 1 in the position corresponding to y and zeros elsewhere. Writing out $\Phi(x, y)$, we get:

$$\Phi(x, y) = \begin{pmatrix} \lambda_1(y) \cdot x \\ \lambda_2(y) \cdot x \\ \cdots \\ \lambda_L(y) \cdot x \end{pmatrix} = \begin{pmatrix} \vdots \\ 0 \\ x \\ 0 \\ \vdots \end{pmatrix}$$

Based on the notation introduced so far, the prediction distribution (3.13) takes the following form

$$\pi_x(j) = \frac{\exp(\mathbf{w}^T \cdot \Phi(x, j))}{Z_x}$$

where $Z_x = \sum_{k=1}^{L} \exp(\mathbf{w}^T \cdot \Phi(x, k))$. Writing H in terms of $\mathbf{w}$ one gets

$$\begin{aligned} H &= \sum_{x \in X} H(\phi_x \parallel \pi_x) \\ &= \sum_{x \in \Lambda} \Big(\log Z_x - \mathbf{w}^T \cdot \Phi(x, y_x) \Big) \\ &+ \sum_{x \in V} \Big(\log Z_x - \frac{1}{L} \sum_{k} \mathbf{w}^T \cdot \Phi(x, k) \Big) \end{aligned} \tag{3.17}$$

where we have used y_x to denote the label of the instance x. As it can be seen, the above expression is the likelihood function based on a log-linear model over the labeled and unlabeled data. We may also regularize $\mathbf{w}$ to prefer vectors having small norm. It should be noted that here we drop the requirement that $\sum_j \theta_{fj} = 1$. The derivative of expression (3.17) can be used to update the parameters in a direction that reduces H:

$$\frac{\partial H}{\partial w_{fj}} = \sum_{x \in X_f} \left(\frac{\exp(\mathbf{w}^T \cdot \Phi(x, j))}{Z_x} \right) - \frac{|V_f|}{L} - |\Lambda_{fj}|$$

3.6 Summary

I analyzed several algorithms for rule-based semi-supervised learning. I showed that the DL-1 algorithm, which is a variant of the Yarowsky algorithm proposed in (Abney, 2004), minimizes an upper-bound on a new definition of cross entropy based on a specific instantiation of the Bregman distance. The use of Bregman distances provided us with a general framework which we used in order to show strong connections to the graph-based semi-supervised learning algorithms of (Zhu, Ghahramani, and Lafferty, 2003; Blum and Chawla, 2001). By extending the DL-1 algorithm, I provided a simple polynomial-time algorithm for label propagation in graph-based semi-supervised learning. I also introduced a new class of algorithms for semi-supervised learning, called the DL-2 algorithms, which is shown to optimize a reasonable objective function. In the next chapter I devise a self-training style algorithm, inspired by the Yarowsky algorithm, to improve the performance of the phrase-based SMT models.

Chapter 4

Bootstrapping for Statistical Machine Translation

In this chapter I explore the use of bootstrapping (self-training) methods for the effective use of monolingual data from the source language in order to improve translation quality. I propose several algorithms with this aim, and present the strengths and weaknesses of each one based on detailed experimental evaluations.

4.1 The Motivation

State-of-the-art SMT systems are trained on large collections of text which consist of bilingual corpora (to learn the parameters of the translation model), and of monolingual target language corpora (for the target language model). It has been shown that adding large amounts of target language text improves translation quality considerably (Brants et al., 2007), as improved language model estimates about potential output translations can be used by the decoder in order to improve translation fluency. However, the availability of monolingual corpora in the source language has not been shown to help improve the system's performance. I will show how such corpora can be used to achieve higher quality translations by improving content preservation and fluency in the translations generated by the underlying SMT system.

Even if large amounts of bilingual text are given, the training of the statistical models usually suffers from sparse data. The number of possible events, i.e. phrase pairs or pairs

of subtrees in the two languages, is too big to reliably estimate a probability distribution over such pairs. Another problem is that for many language pairs the amount of available bilingual text is very limited. In this chapter, I address this problem and propose a general framework to solve it. The hypothesis is that adding information from source language text can also provide improvements. Unlike adding target language text, this hypothesis renders the problem as a semi-supervised learning scenario. To tackle this problem, I propose algorithms for transductive/semi-supervised (inductive) learning. By transductive, I mean that the test set *is* the monolingual sentence set. We will see that such an approach can lead to better translations despite the fact that the test data are typically much smaller in size than typical training data for SMT systems.

Transductive learning can be seen alternatively as a means to *adapt* the SMT system to a new type of text. Say a system trained on newswire is used to translate weblog texts. The proposed method adapts the trained models to the style and domain of the new input without requiring bilingual data from this domain.

4.2 The Framework

4.2.1 The Algorithm

My bootstrapping algorithm for SMT, Algorithm 6, is inspired by the Yarowsky algorithm (see Section 2.2.3). I describe it here for (re-)training the translation-model component in our log-linear SMT system. However, the same algorithm can be used to (re-)train the language-model component of the SMT system.

The algorithm works as follows: First, the translation model is estimated based on the sentence pairs in the bilingual training data D_l. Then, a set of source language sentences, D_u, is translated based on the current model. A subset of good translations and their sources, T_i, is selected in each iteration and added to the training data. These selected sentence pairs are replaced in each iteration, and only the original bilingual training data, D_l, is kept fixed throughout the algorithm. The process of generating sentence pairs, selecting a subset of good sentence pairs, and updating the model is continued until a stopping condition is met. We may run this algorithm in a transductive/inductive setting which means that the set of sentences D_u is drawn either from the test set that will be used eventually to evaluate the SMT system or from additional data which is relevant to the test set. However, the evaluation step is still done just once at the end of our learning process, and all optimization

Algorithm 6 Bootstrapping Algorithm for SMT

Input: training set D_l of parallel sentence pairs. // Bilingual training data.
Input: unlabeled set D_u of source text. // Monolingual source language data.
Input: number of iterations R, and size of N-best list.
$T_{-1} := \{\}$. // Additional bilingual training data.
$i := 0$. // Iteration counter.
repeat
  Training step: $M^{(i)}_{F \rightarrow F} := \textbf{train}(D_l, T_{i-1})$.
  $X_i := \{\}$.
  // The set of generated translations for this iteration.
  for sentence $\mathbf{f} \in D_u$ **do**
    Labeling step: Translate (decode) $\mathbf{f}$ using $\pi^{(i)} := M^{(i)}_{F \rightarrow F}$ to obtain N best sentence pairs with their scores
    $X_i := X_i \cup \{(\mathbf{e}_n, \mathbf{f}, \pi^{(i)}(\mathbf{e}_n|\mathbf{f}))_{n=1}^{N}\}$
  end for
  Scoring step: $S_i := \textbf{score}(X_i)$
  // Assign a score to sentence pairs $(\mathbf{e}, \mathbf{f})$ from X.
  Selection step: $T_i := \textbf{select}(X_i, S_i)$
  // Choose a subset of *good* sentence pairs $(\mathbf{e}, \mathbf{f})$ from X.
  $i := i + 1$.
until $i > R$

steps (e.g. for learning the weights of the log-linear model) are carried out on development data. In Algorithm 6, changing the definition of **train**, **score** and **select** will give the different bootstrapping algorithms that I will discuss in this chapter.

Given the probability model $P(\mathbf{e}|\mathbf{f})$, consider the distribution over all possible translations $\mathbf{e}$ for a particular input sentence $\mathbf{f}$. We can initialize this probability distribution to the uniform distribution for each sentence $\mathbf{f}$ in the unlabeled data D_u; thus, this distribution will have the maximum entropy. Under certain precise conditions (as discussed in Chapter 3), we can analyze Algorithm 6 as minimizing the entropy of the distribution over translations of sentences in D_u while respecting the (true) translations of the sentences given in D_l. However, this is true only when the functions **train**, **score** and **select** have very prescribed definitions (Haffari and Sarkar, 2007; Abney, 2004). In this chapter, rather than analyze the convergence of Algorithm 6, I run it for a fixed number of iterations and instead focus on finding useful definitions for **train**, **score** and **select** that can be experimentally shown to improve MT performance.

4.2.2 The Training Function

I introduce the following different definitions for **train** in Algorithm 6:

- **Full Re-training** (of the model): If **train**(D_l, T) estimates the model parameters based on $D_l \cup T$, then we have a bootstrapping algorithm that re-trains a model on the original training data plus the sentences decoded in the last iteration.

- **Additional Model:** If, on the other hand, a new model is learned on T only and then added as a new component in the log-linear SMT model, we have an alternative to the full re-training of the model on labeled and unlabeled data which can be very expensive if D_l is very large. This additional model is small and specific to the development or test set it is trained on. As the analysis of such an additional phrase table in (Ueffing, Haffari, and Sarkar, 2007a) shows, it overlaps with the original phrase tables, but also contains many new phrase pairs.

- **Mixture Model:** Another alternative for **train** is to create a mixture model of the original model probabilities with the newly trained one. In the case of the phrase translation model, this yields

$$P(\mathbf{f}|\mathbf{e}) = \lambda \cdot P_{D_l}(\mathbf{f}|\mathbf{e}) + (1 - \lambda) \cdot P_T(\mathbf{f}|\mathbf{e}) \tag{4.1}$$

  where P_{D_l} and P_T are phrase table probabilities estimated on D_l and T, respectively. In cases where new phrase pairs are learned from T, they get added into the merged phrase table.

There is a delicate but important point about the **Additional Model** way of incorporating the new information into the SMT log-linear model. As pointed out, the new phrase table may have many phrase pairs which do not exist in the original phrase tables (and vise versa). In addition to the phrases in the intersection of the old and the new phrase tables, we should make sure that the decoder uses the phrase pairs which exist in *either* of the phrase tables while scoring the candidate hypotheses during the decoding process. For example, it may consider a small score as the contribution of a phrase table (in the log-linear model) for a phrase pair which is missing in that phrase table.

4.2.3 The Scoring Function

In Algorithm 6, the **score** function assigns a score to each translation hypothesis $\mathbf{e}$. In what follows, I describe my proposed scoring functions:

- **Length-normalized Score:** Each translated sentence pair $(\mathbf{e}, \mathbf{f})$ is scored according to the model probability $P(\mathbf{e}|\mathbf{f})$ normalized by the length $|\mathbf{e}|$ of the target sentence:

$$\mathbf{score}(\mathbf{e}, \mathbf{f}) \quad = \quad P(\mathbf{e}|\mathbf{f})^{\frac{1}{|\mathbf{e}|}} \tag{4.2}$$

- **Confidence Estimation:** The goal of confidence estimation is to estimate how reliable a translation is, given the corresponding source sentence. The confidence estimation which I use in my experiments follows the approaches in (Blatz et al., 2003; Ueffing and Ney, 2007): The confidence score of a target sentence $\mathbf{e}$ is calculated as a log-linear combination of several different sentence scores. These scores are Levenshtein-based word posterior probabilities, phrase posterior probabilities, and a target language model score. The posterior probabilities are determined over the N-best list generated by the SMT system.

Here is more explanation about the Confidence Estimation method. The word posterior probabilities are calculated on the basis of the Levenshtein alignment between the hypothesis under consideration and all other translations contained in the N-best list. The Levenshtein alignment is performed between a given hypothesis $\mathbf{e}$ and every sentence $\mathbf{e}_n$ contained in the N-best list individually. To calculate the posterior probability of target word e occurring in position i of the translation, the probabilities of all sentences containing e in position i or in a position Levenshtein-aligned to i is summed up. This sum is then normalized by the total probability mass of the N-best list. Let $\mathcal{L}(\mathbf{e}, \mathbf{e}_n)$ be the Levenshtein alignment between sentences $\mathbf{e}$ and $\mathbf{e}_n$, and $\mathcal{L}_i(\mathbf{e}, \mathbf{e}_n)$ that of word e in position i in $\mathbf{e}$.

Consider the following example: Calculating the Levenshtein alignment between the sentences $\mathbf{e}$ ="A B C D E" and $\mathbf{e}_n$ ="B C G E F" yields

$$\mathcal{L}(\mathbf{e}, \mathbf{e}_n) = \text{“– B C G E”}$$

where "–" represents insertion of the word A into $\mathbf{e}$ and in the above alignment F is deleted from $\mathbf{e}_n$. Using this representation, the word posterior probability of word e occurring in a

position Levenshtein-aligned to i is given by

$$p_{\text{lev}}(e|\mathbf{f},\mathbf{e},\mathcal{L}) = \frac{\sum_{n=1}^{N} \delta(t,\mathcal{L}_i(\mathbf{e},\mathbf{e}_n)) \cdot p(\mathbf{f},\mathbf{e}_n)}{\sum_{n=1}^{N} p(\mathbf{f},\mathbf{e}_n)} \tag{4.3}$$

To obtain a score for the whole target sentence, the posterior probabilities of all target words are multiplied. The sentence probability is approximated by the probability which the SMT system assigns to the sentence pair. More details on computing word posterior probabilities are available in (Ueffing and Ney, 2007).

The phrase posterior probabilities are determined in a similar manner by summing the sentence probabilities of all translation hypotheses in the N-best list which contain the phrase pair. The segmentation of the sentence into phrases is provided by the SMT system. Again, the single values are multiplied to obtain a score for the whole sentence.

The log-linear combination of the different sentence scores into one confidence score is optimized w.r.t. sentence classification error rate (CER) on the development corpus. The weights in this combination are optimized using the Downhill Simplex algorithm (Press et al., 2002). In order to carry out the optimization, reference classes are needed which label a given translation as either correct or incorrect. These are created by calculating the word error rate (WER) of each translation and labeling the sentence as incorrect if the WER exceeds a certain value, and correct otherwise. Then the confidence score $c(\mathbf{e})$ of translation $\mathbf{e}$ is computed, and the sentence is classified as correct or incorrect by comparing its confidence to a threshold τ:

$$c(\mathbf{e}) \begin{cases} > \tau & \Rightarrow \mathbf{e} \text{ correct} \\ \leq \tau & \Rightarrow \mathbf{e} \text{ incorrect} \end{cases}$$

The threshold τ is optimized to minimize CER. Then the assigned classes are compared to the reference classes, determine the CER and update the weights accordingly. This process is iterated until the CER converges.

4.2.4 The Selection Function

The **select** function in Algorithm 6 is used to create the additional training data T_i which will be used in the next iteration $i+1$ by **train** to augment the original bilingual training data. I introduce the following selection functions:

- **Importance Sampling:** For each sentence **f** in the set of unlabeled sentences D_u, the Labeling step in Algorithm 6 generates an N-best list of translations, and the subsequent Scoring step assigns a score for each translation **e** in this list. Let the set of generated translations for all sentences in D_u be the event space and use the decoder scores to put a probability distribution over this space, simply by renormalizing the scores described in Section 4.2.3. We select K translations by sampling from this distribution, which is henceforth referred to as *importance sampling*. The sampling is done with replacement which means that the same translation may be chosen several times. Furthermore, several different translations of the same source sentence can be sampled from the N-best list. The K sampled translations and their associated source sentences make up the additional training data T_i.

- **Selection using a Threshold:** This method compares the score of each single-best translation to a threshold. The translation is considered reliable and added to the set T_i if its score exceeds the threshold. Otherwise, it is discarded and not used in the additional training data. The threshold is optimized on the development set beforehand. Since the scores of the translations change in each iteration, the size of T_i also changes.

- **Keep All:** This method does not perform any selection at all. It is simply assumed that all translations in the set X_i are reliable, and none of them are discarded. Thus, in each iteration, the result of the selection step will be $T_i = X_i$. This method is introduced mainly as a baseline for comparison with other selection methods.

4.2.5 Filtering the Training Data

In general, having more training data improves the quality of the trained models. However, when it comes to the translation of a particular test set, the question is whether *all* of the available training data are relevant to the translation task or not. Moreover, working with large amounts of training data requires more computational power. So if we can identify a subset of training data which are relevant to the current task and use only this to re-train the models, we can reduce computational complexity significantly.

I propose to **Filter** the training data, either bilingual or monolingual text, to identify the parts which are relevant to the test set. This filtering is based on n-gram coverage. For a source sentence **f** in the training data, its n-gram coverage over the sentences in the test

corpus	use	sentences
EuroParl	phrase table + language model	688K
dev06	dev1	2,000
test06	test in-domain / out-of-domain	2,000 / 1,064

Table 4.1: French–English EuroParl corpora

set is computed:

$$\text{coverage}(\mathbf{f}, D, n) := \frac{\#\ n\text{-grams in } \mathbf{f} \text{ which also appear in } D}{\#\ n\text{-grams in } \mathbf{f}}$$

The average over several n-gram lengths is used as a measure of relevance of this training sentence to the test corpus:

$$\text{Score}(\mathbf{f}, D) := \sum_{n=1}^{N} \alpha_n \ \text{coverage}(\mathbf{f}, D, n)$$

where the weights α_n specify the importance of individual n-gram coverage scores. Based on this, I select the top K source sentences or sentence pairs. In the experiments of this chapter, I use $N = 4$ and $\alpha_n = \frac{1}{4}$.

4.3 Experimental Results

I use the EuroParl corpus as distributed for the shared task in the NAACL 2006 workshop on statistical machine translation (WMT)[1], and the Hansards corpus as distributed by ISI[2]. The bilingual training data from the EuroParl corpus is used to train translation and language models, and the development sets are used to optimize the weights of the SMT log-linear model. In the transductive experiments, I use the test set as the monolingual set. Whereas in the inductive experiments, I use the English sentences from Hansard corpus as the monolingual sentence set. The final evaluation is done on the test set using BLEU score (Papineni et al., 2002) with 95%-confidence intervals which is calculated using bootstrap resampling.

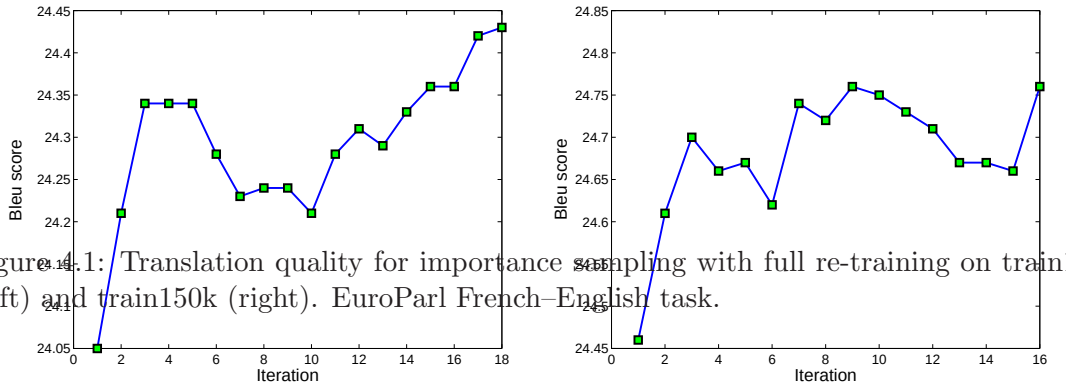


Figure 4.1: Translation quality for importance sampling with full re-training on train100k (left) and train150k (right). EuroParl French–English task.

4.3.1 Transductive Experiments

For the experiments of this section, the corpus statistics are shown in Table 4.1. In a first set of experiments, I explore the behavior of the transductive learning algorithm during its iterations. The selection and scoring is carried out using importance sampling with normalized scores. I use the 100K and 150K training sentences filtered according to n-gram coverage over the test set. The phrase table is fully re-trained on these data and 8,000 test sentence pairs which are sampled from 20-best lists in each iteration. The results on the test set are shown in Figure 4.1. The BLEU score increases, although with slight variation, over the iterations. In total, it increases from 24.1 to 24.4 for the 100K filtered corpus, and from 24.5 to 24.8 for 150K, respectively. We see that the BLEU score of the system using 100K training sentence pairs and transductive learning is the same as that of the one trained on 150K sentence pairs. So the information extracted from untranslated test sentences is equivalent to having an additional 50K sentence pairs.

In a second set of experiments, I use the whole EuroParl corpus and the sampled sentences to fully re-train the phrase tables in each iteration. I run the algorithm for three iterations and the BLEU score increased from 25.3 to 25.6. Even though this is a small increase, it shows that the unlabeled data contains some information which can be explored

[1] www.statmt.org/wmt06/shared-task/

[2] www.isi.edu/natural-language/download/hansard/

test set	selection method	BLEU[%]
French–English EuroParl (1 ref.)		
in-domain	baseline	28.1 ±0.8
	importance sampling	28.4
	keep all	28.3
out-of-domain	baseline	18.8 ±0.8
	importance sampling	18.9
	keep all	19

Table 4.2: EuroParl results based on the mixture of phrase tables with $\lambda = .1$.

in transductive learning.

In a third experiment, I use the mixture model idea as explained in Section 4.2.2. The initially learned phrase table is merged with the learned phrase table in each iteration with a weight of $\lambda = 0.1$. This value for λ was found based on cross validation on a development set. I run the experiments on in-domain and out-of-domain sentences separately, and the results can be seen in Table 4.2. Again this is a small increase in the performance of the model, but it shows that the unlabeled data contains some information which can be exploited in the transductive learning. Some of the translation examples are shown in Table 4.3, which highlight the quality of the generated translation after transductive learning.

baseline	the opportunities to achieve this are good *but are special efforts are indispensable .*
transductive	the chances of achieving this are good **but special effort are still necessary .**
reference	there is a good chance of this , but particular efforts are still needed .
baseline	*this does not want to say first of all , as a result .*
transductive	**it does not mean that everything is going on .**
reference	this does not mean that everything has to happen at once .

Table 4.3: Translation examples after using transductive learning.

4.3.2 Inductive Experiments

The corpus statistics for the experiments in this section are shown in Table 4.4. I will carry out evaluation separately for the in-domain/out-of-domain test *and* monolingual set

corpus	use	sentences
EuroParl-training	phrase table + language model	688K
EuroParl-test2006	in-domain dev1	500
EuroParl-test2006	out-of-domain dev2	500
EuroParl-devtest2006	dev3	2,000
Hansards-training	monolingual source data	1130K
EuroParl-test2006	in-domain test1	1,500
EuroParl-test2006	out-of domain test2	500

Table 4.4: French–English corpora.

with respect to the bilingual training data to investigate the adaptation capabilities of the methods.

The experiments in this section are designed to explore the behavior of the semi-supervised learning algorithm with respect to the different sentence selection methods on in-domain and out-of-domain test sentences.

I have partitioned the NAACL 2006 WMT shared task's test set into two sets (S_{in} and S_{out}) to separate in-domain and out-of-domain test sentences. S_{in} includes the first 2000 sentences and S_{out} includes the last 1000 sentences of this test set. Then we used the first 500 sentences in S_{in} and S_{out} as the development sets dev1 and dev2, and used the rest as the test sets. As additional monolingual source sentences, we used the French sentences in the training set of the Canadian Hansards corpus as provided by ISI. The monolingual French sentences were sorted according to their n-gram overlap (see Section 4.2.5) with the development corpora dev1 and dev2 for in-domain and out-of-domain experiments, and 5,000 French sentences were added in each iteration of the semi-supervised algorithm.

The scoring and selection of the translations (see Algorithm 6) are performed using:

- confidence score with importance sampling,
- length normalized translation score with importance sampling,
- confidence score with a threshold, and
- keeping the top-K sentence pairs having the highest length normalized translation scores.

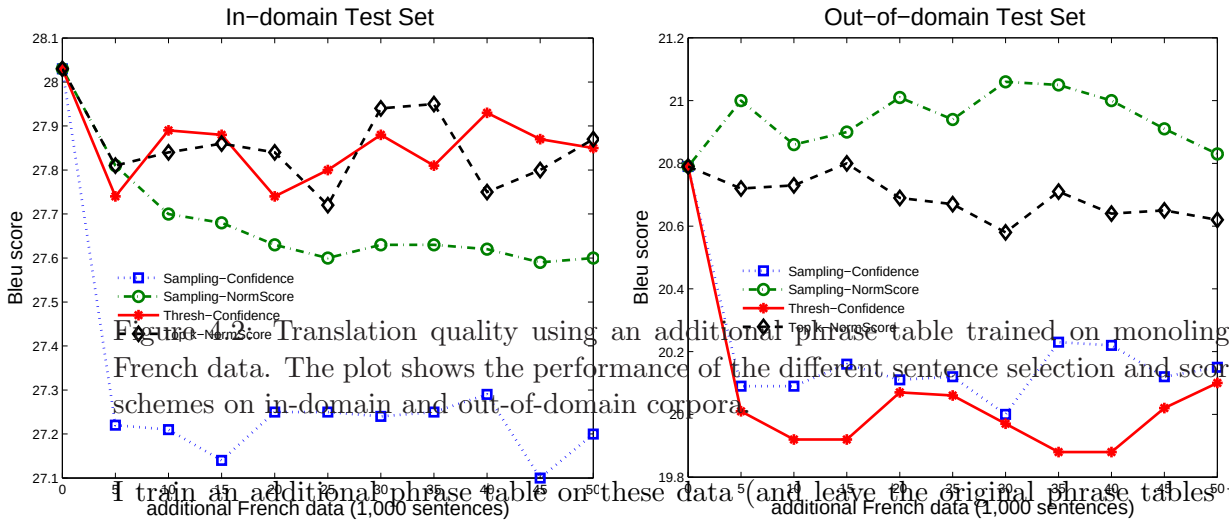


Figure 4.2: Translation quality using an additional phrase table trained on monolingual French data. The plot shows the performance of the different sentence selection and scoring schemes on in-domain and out-of-domain corpora.

I train an additional phrase table on these data (and leave the original phrase tables unmodified) which is added as a new component in the log-linear model. The weight of this new component is optimized based on dev1 and dev2 for in-domain and out-of-domain experiments. Moreover, I use the development set dev3 for estimating the parameters of the confidence estimation model and the threshold (see Section 4.3.2).

The results of the four sentence selection methods are shown in Figure 4.2. The semi-supervised algorithm deteriorates the performance of the initial SMT system for all cases except sampling with normalized translation scores. The performance of top-K sentence pair selection based on the normalized scores is encouraging for both in-domain and out-of-domain experiments. It is probable that by choosing K in a more elaborate way, this method would outperform the baseline and other methods. Note that it just uses the normalized translation scores which are already generated by the decoder. Using dev1 and dev2 to train the confidence estimation models for in-domain and out-of-domain experiments, may help the methods which use the confidence to boost their performance. Some of the translation examples are shown in Table 4.6, which highlight the quality of the generated translation by the semi-supervised learning.

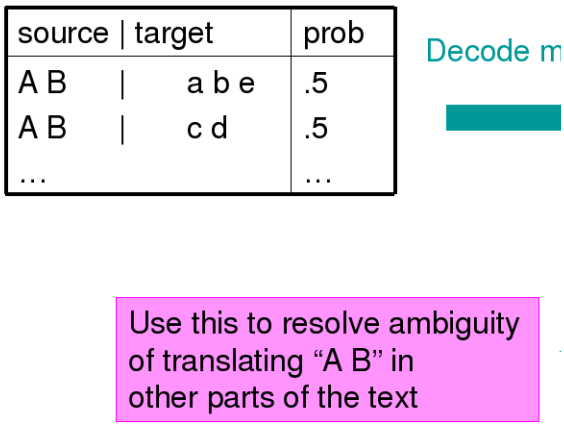


Figure 4.3: This example illustrates how the phrase tables becomes focused to the test set. Initially there is a high ambiguity in translating the source phrase 'A B'. In translating the test set, most of the time the target phrase 'c d' is being chosen as the translation of the source phrase 'A B' by the language model (LM), based on the signals received from the context. As the result, the probabilities of the phrase pairs in the phrase table are adaptively tuned towards the test sentences.

4.3.3 Analysis

It may not be intuitively clear why the SMT system can learn something from its own output and improve through semi-supervised learning. There are two main reasons for this improvement. Firstly, the selection step provides important feedback for the system. The confidence estimation, for example, discards translations with low language model scores or posterior probabilities. The selection step discards bad machine translations and reinforces phrases of high quality. As a result, the probabilities of low-quality phrase pairs, such as noise in the table or overly confident singletons, degrade. The experiments comparing the various settings for transductive learning shows that selection clearly outperforms the method which keeps all generated translations as additional training data. The selection methods investigated here have been shown to be well-suited to boost the performance of semi-supervised learning for SMT.

Secondly, the bootstrapping algorithm constitutes a way of adapting the SMT system

Original parallel corpus	Additiona
'A B', 'C D E'	'A B C D

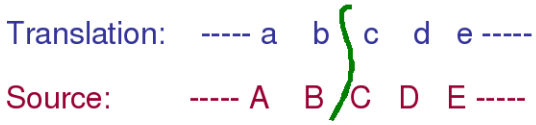


Figure 4.4: This example illustrates how new phrase pairs can be composed when monolingual sentences are translated. Part of the monolingual sentence 'A B C D E' is translated to 'a b c d e', and new phrase pairs are constructed depending on how to break these two sentence fragments.

to a new domain or style without requiring bilingual training or development data. Those phrases in the existing phrase tables which are relevant for translating the new data are reinforced. The probability distribution over the phrase pairs thus gets more focused on the (reliable) parts which are relevant for the test data (see Figure 4.3). Moreover, new phrase pairs can be constructed when the monolingual sentences are translated by the current SMT model, which can contribute to the model in the later iterations (see Figure 4.4). Table 4.5 shows the statistics about the number of phrase pairs with adapted probabilities, and the number of newly generated phrase pairs. It also shows the number of times that they have been used in the decoder for generating translations of the test sentences. As it can seen from the Table 4.5, new phrases are used rarely, hence most of the benefit comes from focused probability distributions (Ueffing, Haffari, and Sarkar, 2007b).

eval-04	Editorials	Newswire	Speeches
Size of adapted phrase table	1,981	3,591	2,321
Adapted phrases used	707	1,314	815
New phrases	679	1,359	657
New phrases used	23	47	25

eval-06	Broadcast s conversations	Broadcast news	Newsgroup	Newswire
Size of adapted phrase table	2,155	4,027	2,905	2,804
Adapted phrases used	759	1,479	1,077	1,115
New phrases	1,058	1,645	1,259	1,058
New phrases used	90	86	88	41

Table 4.5: Statistics of the phrase tables trained on the genres of the NIST test corpora (see Section 4.4.1 for more details about the corpora). Scoring: confidence estimation, selection: threshold.

4.4 Related Work

Semi-supervised learning has been previously applied to improve word alignments. In (Callison-Burch, Talbot, and Osborne, 2004), a generative model for word alignment is trained using unsupervised learning on parallel text. In addition, another model is trained on a small amount of hand-annotated word alignment data. A mixture model provides a probability for word alignment. Experiments showed that putting a large weight on the model trained on labeled data performs best. Along similar lines, (Fraser and Marcu, 2006) combine a generative model of word alignment with a log-linear discriminative model trained on a small set of hand aligned sentences. The word alignments are used to train a standard phrase-based SMT system, resulting in increased translation quality .

In (Callison-Burch, 2002) co-training is applied to MT. This approach requires several source languages which are sentence-aligned with each other and all translate into the same target language. One language pair creates data for another language pair and can be naturally used in a (Blum and Mitchell, 1998)-style co-training algorithm. Experiments on the EuroParl corpus show a decrease in WER. However, the selection algorithm applied there is actually supervised because it takes the reference translation into account. Moreover, when the algorithm is run long enough, large amounts of co-trained data injected too much noise and performance degraded.

baseline	*the so-called ' grandfather bulldozer become the most of the israelis* and the final asset of western diplomacy and for the americans , surprisingly , for europeans too .
semi-supervised	**'bulldozer' became the grandfather of most of the israelis** and the final asset of western diplomacy and for the americans , surprisingly , for europeans too .
reference	the " bulldozer " had become the grandfather of most israelis and the last card of western diplomacy , for americans and , surprisingly , for europeans , too .
baseline	*these are not all their exceptional periods which create bonaparte* , which is probably better because leaders exceptional can provide the illusion that all problems have their solution , which is far from true .
semi-supervised	**these are not all periods which create their exceptional bonaparte** , which is probably better because leaders exceptional can provide the illusion that all problems have their solution which is far from being true .
reference	not all exceptional periods create their bonapartes , and this is probably a good thing , for exceptional leaders may give the illusion that all problems have solutions , which is far from true .
baseline	*enjoy an initial period of growth and innovation to experiment* with on these fronts may prove heavily paying subsequently .
semi-supervised	**given an initial period to experiment with growth and innovation** on these fronts may prove strong paying subsequently .
reference	using an initial period of growth to experiment and innovate on these fronts can pay high dividends later on .

Table 4.6: The *semi-supervised* examples are taken from sampling-based sentence selection with normalized scores. Lower-cased output, punctuation marks tokenized.

The idea of using monolingual source-language data to improve MT performance was first introduced in IWSLT Workshop (Ueffing, 2006) which was presented at the same time with some of this chapter's ideas, as two independent talks, in a NIPS Workshop (Goutte et al., 2006). After meeting at that workshop, we merged and pushed the ideas further (Ueffing, Haffari, and Sarkar, 2007a; Ueffing, Haffari, and Sarkar, 2007b; Ueffing, Haffari, and Sarkar, 2008). We also applied the framework to the Chinese-English translation task which has a larger scale compared to the French-English translation experiments presented in this chapter. In what follows, I summarize some of our key findings for Chinese-English translation task based on (Ueffing, Haffari, and Sarkar, 2007a; Ueffing, Haffari, and Sarkar,

corpus	use	sentences	domains
non-UN	phrase table + language model	3.2M	news, magazines, laws
UN	phrase table + language model	5.0M	UN Bulletin
English Gigaword	language model	11.7M	news
Chinese Gigaword	additional source data	50K	news
multi-p3	optimize decoder (dev1)	935	news
multi-p4	optimize rescoring (dev2)	919	news
eval-04	test	1,788	newswire , editorials, political speeches
eval-06 GALE	test	2,276	broadcast conversations, broadcast news, newsgroups, newswire
eval-06 NIST	test	1,664	broadcast news, newsgroups, newswire

Table 4.7: Chinese–English corpora.

2007b; Ueffing, Haffari, and Sarkar, 2008).

4.4.1 Results on Chinese–English

For the Chinese–English translation task, we used the corpora distributed for the large-data track in the 2006 NIST evaluation (see Table 4.7). We used the LDC segmenter for Chinese. A subset of the English Gigaword corpus was used as additional LM training material. Data from the Chinese Gigaword was filtered and used as additional monolingual source-language data for semi-supervised learning. The multiple translation corpora multi-p3 and multi-p4 were used as development corpora. Evaluation was performed on the 2004 and 2006 test sets. The 2006 test set consists of two sections: the NIST section which was created for the NIST machine translation evaluation in 2006 and which is provided with four English references, and the GALE section which was created in the DARPA project GALE and comes with one English reference.

4.4.1.1 Transductive Experiments

In all experiments in this section, Algorithm 6 has been run for one iteration. We have also investigated the use of an iterative procedure here, but this did not yield any improvement in translation quality.

Table 4.8 presents translation results on NIST with different versions of the scoring and

corpus	system	scoring	BLEU[%]	mWER[%]	mPER[%]
eval-04^{+}	baseline		31.8 ±0.7	66.8 ±0.7	41.5 ±0.5
	keep all		33.1	66.0	41.3
	IS	norm.	**33.5**	65.8	40.9
		conf.	33.2	65.6	**40.4**
	threshold	norm.	**33.5**	65.9	40.8
		conf.	**33.5**	**65.3**	40.8
eval-06 GALE*	baseline		12.7 ±0.5	75.8 ±0.6	54.6 ±0.6
	keep all		12.9	75.7	55.0
	IS	norm.	13.2	74.7	54.1
		conf.	12.9	74.4	53.5
	threshold	norm.	12.7	75.2	54.2
		conf.	**13.6**	**73.4**	**53.2**
eval-06 NIST^{+}	baseline		27.9 ±0.7	67.2 ±0.6	44.0 ±0.5
	keep all		28.1	66.5	44.2
	IS	norm.	28.7	66.1	43.6
		conf.	28.4	65.8	**43.2**
	threshold	norm.	28.3	66.1	43.5
		conf.	**29.3**	**65.6**	**43.2**

Table 4.8: Translation quality using an additional adapted phrase table trained on the dev/test sets for Chinese–English task. $^{+}$: 4 refs., *: 1 ref., and IS: importance sampling. Best results are printed in **boldface**.

selection methods introduced in Section 4.2. In these experiments, the unlabeled data for Algorithm 6 is the development or test corpus. For the corpus D_u, 5,000-best lists were generated using the baseline SMT system. Since re-training the full phrase tables is not feasible here, a (small) additional phrase table, specific to D_u, was trained and plugged into the SMT system as an additional model. The decoder weights thus had to be optimized again to determine the appropriate weight for this new phrase table. This was done on the dev1 corpus, using the phrase table specific to dev1. Every time a new corpus is to be translated, an adapted phrase table is created using transductive learning and used with the weight which has been learned on dev1. In the first experiment presented in Table 4.8, all of the generated 1-best translations were kept and used for training the adapted phrase tables. This method yields slightly higher translation quality than the baseline system. The second approach we studied is the use of importance sampling over 20-best lists, based either on length-normalized sentence scores (norm.) or confidence scores (conf.). As the results in

Table 4.8 show, both variants outperform the first method, with a consistent improvement over the baseline across all test corpora and evaluation metrics. The third method uses a threshold-based selection method. Combined with confidence estimation as scoring method, this yields the best results. All improvements over the baseline are significant at the 95

Table 4.9 shows the translation quality achieved on the NIST test sets when additional source language data from the Chinese Gigaword corpus comprising newswire text is used for transductive learning. These Chinese sentences were sorted according to their n-gram overlap with the development corpus, and the top 5,000 Chinese sentences were used. The selection and scoring in Algorithm 6 were performed using confidence estimation with a threshold. Again, a new phrase table was trained on these data. As can be seen in Table 4.9, this system outperforms the baseline system on all test corpora. The error rates are significantly reduced in all three settings, and BLEU score increases in all cases. A comparison with Table 4.8 shows that transductive learning on the development set and test corpora, adapting the system to their domain and style, is more effective in improving the SMT system than the use of additional source language data.

system	BLEU[%]	mWER[%]	mPER[%]
eval-04^{+}			
baseline	31.8±0.7	66.8±0.7	41.5±0.5
add Chin. data	32.8	65.7	40.9
eval-06 GALE*			
baseline	12.7±0.5	75.8±0.6	54.6±0.6
add Chin. data	13.1	73.9	53.5
eval-06 NIST^{+}			
baseline	27.9±0.7	67.2±0.6	44.0±0.5
add Chin. data	28.1	65.8	43.2

Table 4.9: Translation quality using an additional phrase table trained on monolingual Chinese news data for Chinese–English task. Threshold on confidence scores is used for selecting good translations. $^{+}$: 4 refs., *: 1 ref.

4.4.1.2 Inductive Experiments

For these experiments, we used additional source language data from the Chinese Gigaword corpus comprising newswire text. The Chinese Gigaword sentences are sorted according to their n-gram overlap with the development corpus. It is assumed that the test set is unknown

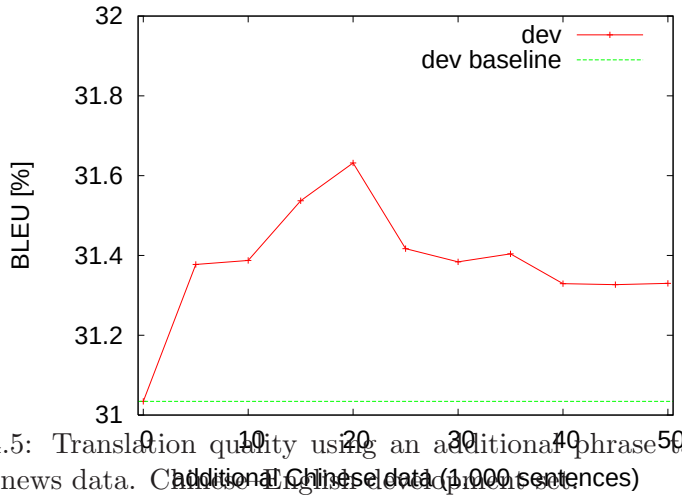


Figure 4.5: Translation quality using an additional phrase table trained on monolingual Chinese news data. Chinese-English development set.

at this point. The Chinese sentences are then divided into chunks of 5,000 sentences; one of these is added in each iteration as described in Algorithm 6.

Figure 4.5 shows the BLEU score on the development set over the iterations. As to be expected, the biggest jump occurs after the first iteration when the decoder weights are re-optimized. Up to iteration 4, which is equivalent to the use of 20,000 additional Chinese sentences in semi-supervised learning, the BLEU score increases. But after that, it drops and levels off at a point which is above the baseline, but does not significantly differ from it anymore. So it seems that if the semi-supervised training runs too long, it adds noise into the model rather than improving it. The overlap between the additional data and the development corpus decreases over the iterations, so that the added data might be less relevant and thus actually hurt translation quality rather than improving it.

After analyzing the results obtained on the development corpus, we evaluated three different systems on the test corpora: the system after the first iteration, which used 5,000 additional Chinese sentences in semi-supervised training and for which the weight optimization was carried out; the system after iteration 4 which performed best on the development set; and the system after the last iteration which had the highest number of additional Chinese sentences, namely 50,000, available for semi-supervised learning. The last setup was

system		BLEU[%]	WER[%]	PER[%]
eval-04 (4 refs.)				
baseline		31.8±0.7	66.8±0.7	41.5±0.5
add Chinese data	iteration 1	***32.8***	***65.7***	***40.9***
	iteration 4	*32.6*	*65.8*	***40.9***
	iteration 10	32.5	66.1	41.2
eval-06 GALE (1 ref.)				
baseline		12.7±0.5	75.8±0.6	54.6±0.6
add Chinese data	iteration 1	**13.1**	***73.9***	***53.5***
	iteration 4	13.0	*75.0*	*53.9*
	iteration 10	12.7	75.4	54.9
eval-06 NIST (4 refs.)				
baseline		27.9±0.7	67.2±0.6	44.0±0.5
add Chinese data	iteration 1	28.1	***65.8***	***43.2***
	iteration 4	**28.2**	*65.9*	*43.4*
	iteration 10	27.7	*66.4*	43.8

Table 4.10: Translation quality using an additional phrase table trained on monolingual Chinese news data. Chinese–English test sets. **Bold:** best result, *italic:* significantly better than baseline.

tested only for comparison, as the results presented in Figure 4.5 indicate already that this system might perform worse than the other two. Table 4.10 shows the translation quality achieved on the Chinese–English test sets with these three systems. The best system is clearly the one obtained after the first iteration. The translation quality is significantly better than the baseline in most cases. The system from iteration 4 (which performs best on the development set) shows a very similar performance in terms of translation quality as the first one. The error rates it achieves are slightly higher than those of the first system, but still significantly better than those of the baseline system. The third system, however, does not outperform the baseline system. As mentioned above, this was to be expected.

Table 4.11 analyzes the translation results achieved on the eval-04 test corpus separately for each genre: editorials, newswire, and political speeches. The most significant improvement in translation quality is achieved on the newswire section of the test corpus. All three semi-supervised systems perform very similarly on this genre, whereas performance decreases on the other two genres as the number of iterations increases. This difference among the genres can be explained by the fact that the additional data is drawn from the

Chinese Gigaword corpus which contains newswire data.

system		BLEU[%]	WER[%]	PER[%]
editorials	baseline	30.7±1.2	67.0±1.1	42.3±0.9
	iteration 1	31.3	65.9	41.8
	iteration 4	30.9	66.2	42.0
	iteration 10	30.8	66.6	42.3
newswire	baseline	30.0±0.9	69.1±0.8	42.7±0.8
	iteration 1	31.1	68.1	42.0
	iteration 4	31.1	67.9	42.0
	iteration 10	31.3	68.1	42.1
speeches	baseline	36.1±1.4	62.5±1.2	38.6±0.9
	iteration 1	37.3	61.3	38.0
	iteration 4	36.8	61.5	38.0
	iteration 10	36.3	61.8	38.4

Table 4.11: Translation quality on Chinese–English eval-04 test set, by genre. Same experimental setup as Table 4.10.

In the previous section, we saw the results of the transductive approach on the same Chinese–English test sets. The translation quality achieved by the transductive approach is higher than that of the inductive method presented here (yielding a BLEU score which is 0.5-1.1 points higher). We see two reasons for this difference: Firstly, transductive learning on the development or test corpus yields a model which is more focused on this corpus. It adapts the system directly to the domain and style by creating an additional phrase table which is specific to the development or test corpus and matches it very well. Secondly, the transductive approach adapts the SMT system to each of the genres. In the work presented here, the additional Chinese data came from the newswire domain only, and this yields a higher boost in translation quality for this genre than for the other ones. It would be interesting to see how the system performs if data from all domains in the test corpus are available for semi-supervised learning. We also investigated a combination of the two self-training methods: using additional source language data as well as the development or test corpus for transductive learning. Unfortunately, the gains achieved by the two methods do not add up, and this system does not outperform the transductively trained one.

Table 4.12 shows how many translations were identified as confident by the scoring and selection algorithm and used to extend the additional phrase table. In the first iteration, this

iteration	1	2	3	4	5
# confident sentences	3,141	3,196	3,017	2,889	2,981
iteration	6	7	8	9	10
# confident sentences	2,890	2,520	2,423	2,324	2,427

Table 4.12: Number of sentences added in each iteration. Chinese–English.

is approximately two thirds of the data added in the iteration. But as the semi-supervised training proceeds, the number of confident sentences decreases. After ten iterations of the algorithm, less than half of the translations are kept. This confirms our assumption that noise is introduced into the procedure by running the algorithm for too long.

4.5 Summary

In this chapter, we saw how re-training an SMT system based on its own output can improve the quality of the translation. As an example, the amount of information that we could extract from monolingual sentences using self-training was almost equivalent to that of an additional 50K sentences (see Figure 4.1). The key point in the success of this idea though is to identify and filter out *bad* translations, since they inject much noise into the training data which may lead to the degradation of the SMT system performance. I introduced different filtering methods so that we can select high quality *automatically*-generated bitext to improve the SMT system after re-training. I presented a detailed experimental evaluation of these ideas on French–English and Chinese–English translation tasks. In Chapters 5 and 6, I devise methods to identify important sentences from monolingual text so that we can use them together with their *human*-generated translations to maximally improve the SMT system performance.

Chapter 5

Active Learning for SMT: Single Language Pair

This chapter provides the first serious study of active learning for SMT, and provides new highly effective sentence selection methods that improve AL for phrase-based SMT in the single language pair setting. I use active learning to improve the quality of a phrase-based SMT system when translating from a source language to a target language. I show significant improvements in translation compared to a random sentence selection baseline, when test and training data are taken from the same or different domains.

5.1 The Motivation

A statistical translation system can be improved or adapted by incorporating new training data in the form of parallel text. However, human translations are very expensive, and it would be prudent to limit the amount we wish to get translated while getting the maximum payoff for the amount we do translate. In this chapter, I propose several novel *active learning* (AL) strategies for statistical machine translation in order to attack this problem.

Active learning is framed as an iterative learning process. In each iteration, new human labeled instances (manual translations) are added to the training data based on their expected training quality. However, if we start with only a small amount of initial parallel data for the new target language, then translation quality is very poor and requires a very large injection of human labeled data to be effective. To deal with this problem, I use a

novel framework for active learning: I assume we are given a small amount of parallel text and a large amount of monolingual source language text; using these resources, I create a large noisy parallel text which I then iteratively improve using small injections of human translations.

Conventional techniques for AL of classifiers are problematic in the SMT setting. Selective sampling of sentences for AL may lead to a parallel corpus where each sentence does not share any phrase pairs with the others. Thus, new sentences cannot be translated since we lack evidence for how phrase pairs combine to form novel translations. I take the approach of exploration vs. exploitation. That is, I select sentences so that (i) Novel translation pairs are added to the SMT system, hence its coverage is improved, and (ii) Already existing rare translation pairs are added to the SMT system, hence the statistics of the system about these translation pairs is improved.

There may be evidence to show that AL is useful even when we have massive amounts of parallel training data. (Turchi, Bie, and Cristianini, 2008) presents a comprehensive learning curve analysis of a phrase-based SMT system, and one of the conclusions they draw is, "The first obvious approach is an effort to identify or produce data sets on demand (active learning, where the learning system can request translations of specific sentences, to satisfy its information needs)."

Despite the promise of active learning for SMT there has been very little work published on this issue (see Sec. 5.8). I provide many useful and novel features useful for AL in SMT, and leverage a whole new set of features that were out of reach for classification systems. I devise features that look at the source language, but also devise features that make an estimate of the potential utility of translations from the source, e.g. phrase pairs that could be extracted. Furthermore, I show that AL can be useful in domain adaptation, and provide the first experimental evidence in SMT that active learning can be used to inject carefully selected translations in order to improve SMT output in a new domain.

5.2 The Algorithmic Framework

Starting from an SMT model trained initially on bilingual data, the problem is to minimize the human effort in translating new sentences which will be added to the training data to make the *retrained* SMT model achieves a certain level of performance. Thus, given a bitext $D_l := \{(\mathbf{f}_i, \mathbf{e}_i)\}$ and a monolingual source text $D_u := \{\mathbf{f}_j\}$, the goal is to select a subset of

highly informative sentences from D_u to present to a human expert for translation. Highly informative sentences are those which, together with their translations, help the retrained SMT system *quickly* reach a certain level of translation quality. As we saw in Section 2.3, this learning scenario is known as AL with Selective Sampling or pool-based AL.

Algorithm 7 describes the experimental setup I propose for active learning in SMT. I train the initial MT system on the bilingual corpus D_l, and use it to translate *all* monolingual sentences in D_u. I denote sentences in D_u together with their translations as D_u^+ (line 4 of Algorithm 7). Then we retrain the SMT system on $D_l \cup D_u^+$ and use the resulting model $M_{F \rightarrow E}$ to decode the test set. Afterwards, we select and remove a subset of highly informative sentences from D_u, and add those sentences together with their human-provided translations to D_l. This process is continued iteratively until a certain level of translation quality, which in our case is measured by the BLEU score, is met. The setup in Algorithm 7 helps us to investigate how to maximally take advantage of human effort (for sentence translation) when learning an SMT model from the available data, that includes bilingual and monolingual text.

When (re-)training the model, two phrase tables are learned: one from D_l and the other one from D_u^+. The phrase table obtained from D_u^+ is added as a new feature function in the log-linear translation model (see Section 4.2.2). The alternative is to ignore D_u^+ as in a conventional AL setting, however, it turns out in my experiments that using more bilingual data, even noisy data, results in better translations. Phrase tables from D_u^+ will get a 0 score in minimum error rate training if they are not useful, so this method is more general. Also, we saw in Chapter 4 that this method is more effective empirically than (1) using the weighted combination of the two phrase tables from D_l and D_u^+, or (2) combining the two sets of data and training from the bitext $D_l \cup D_u^+$.

The sentence selection strategies can be divided into two categories: (1) those which are independent of the target language and just look into the source language, and (2) those which also take into account the target language. From the description of the methods, it will be clear to which category they belong to. Furthermore, I group the sentence selection methods according to whether they are (i) specific to phrase-based SMT model, or (ii) applicable to the general SMT models, e.g. syntactic SMT models. Clearly methods in (i) belong also to the class of methods in (2) since the nature of *phrase* in phrase-based SMT depends on both the source and target languages. We will see in Section 5.5 that the most promising sentence selection strategies for phrase-based SMT belong to (i). In the baseline,

Algorithm 7 AL-SMT: Single Language Pair

1: Given bilingual corpus D_l, and monolingual corpus D_u.
2: $M_{F\rightarrow E} = \textbf{train}(D_l, \emptyset)$
3: **for** $t = 1, 2, ...$ **do**
4: $D_u^+ = \textbf{translate}(D_u, M_{F\rightarrow E})$
5: Select k sentence pairs from D_u^+, and ask a human for their *true* translations.
6: Remove the k sentences from D_u, and add the k sentence pairs (translated by human) to D_l
7: $M_{F\rightarrow E} = \textbf{train}(D_l, D_u^+)$
8: Monitor the performance on the test set D_t
9: **end for**

against which I compare the sentence selection methods, the sentences are chosen *randomly*.

5.3 Sentence Selection: Specific to Phrase-based SMT

Phrases are basic units of translation in phrase-based SMT models. The phrases potentially extracted from a sentence indicate its informativeness. The more new phrases a sentence can offer, the more informative it is. Additionally phrase translation probabilities need to be estimated accurately, which means sentences that contain rare phrases are also informative. When selecting new sentences for human translation, we need to pay attention to this tradeoff between *exploration* and *exploitation*, i.e. selecting sentences to discover new phrases vs estimating accurately the phrase translation probabilities. A similar argument can be made that emphasizes the importance of words rather than phrases for any SMT model. Also we should take into account that smoothing is a means for accurate estimation of translation probabilities when events are rare. In this work, I focus on methods that effectively expand the lexicon or set of phrases of the model.

5.3.1 Considering Phrases (Geom-Phrase, Arith-Phrase)

The more frequent a phrase is in the *unlabeled* data, the more important it is to know its translation; since it is more likely to occur in the test data (especially when the test data is in-domain with respect to unlabeled data). The more frequent a phrase is in the *labeled* data, the more unimportant it is; since probably we have observed most of its translations.

Based on the above observations, the importance score of a sentence is measured as:

$$\begin{aligned} \phi_g^p(s) &:= \Big[\prod_{\mathbf{x}\in X_s^p} \frac{P(\mathbf{x}|D_u)}{P(\mathbf{x}|D_l)}\Big]^{\frac{1}{|X_s^p|}} \\ &= \frac{1}{|X_s^p|}\sum_{\mathbf{x}\in X_s^p} \log \frac{\boldsymbol{\theta}_{D_u}(\mathbf{x})}{\boldsymbol{\theta}_{D_l}(\mathbf{x})} \end{aligned} \tag{5.1}$$

where X_s^p is the set of possible phrases that sentence s can offer, and $\boldsymbol{\theta}_{\mathcal{D}}$ is the multinomial distribution representing $P(\mathbf{x}|\mathcal{D})$ where: $\boldsymbol{\theta}_{\boldsymbol{\theta}}(\mathbf{x}) = \frac{Count(\mathbf{x},\mathcal{D})+\epsilon}{\sum_{\mathbf{x}\in X_{\mathcal{D}}^p} Count(\mathbf{x},\mathcal{D})+\epsilon}$ and $Count(\mathbf{x},\mathcal{D})$ is the frequency of the phrase $\mathbf{x}$ in data $\mathcal{D}$. The score (5.1) is the averaged *probability ratio* of the set of candidate phrases, i.e. the probability of the candidate phrases under a probabilistic phrase model based on D_u divided by that based on D_l. In addition to the geometric average in (5.1), we may also consider the arithmetic average score:

$$\phi_a^p(s) := \frac{1}{|X_s^p|}\sum_{\mathbf{x}\in X_s^p} \frac{P(\mathbf{x}|D_u)}{P(\mathbf{x}|D_l)} \tag{5.2}$$

which is similar to (5.2) with the difference of additional log.

In parallel data D_l, phrases are the ones which are extracted by the usual phrase extraction algorithm; but what are the candidate phrases in the unlabeled data? Considering the n-best list of translations can tell us the possible phrases the input sentence may offer. For each translation, we have access to the phrases used by the decoder to produce that output. However, there may be islands of out-of-vocabulary (OOV) words that were not in the phrase table and not translated by the decoder as a phrase. We group together such groups of OOV words to form an OOV phrase. The set of possible phrases we extract from the decoder output contain those coming from the phrase table (from labeled data D_l) and those coming from OOVs. OOV phrases are also used in our computation, where $P(\mathbf{x} \mid D_l)$ for an OOV phrase x is the uniform probability over all OOV phrases.

5.3.2 Considering n-grams (Geom n-gram, Arith n-gram)

As an alternative to phrases, we may consider n-grams as basic units of generalization. The resulting score is the weighted combination of the n-gram based scores:

$$\phi_g^N(s) := \sum_{n=1}^{N} \frac{w_n}{|X_s^n|}\sum_{x\in X_s^n} \log \frac{P(x|D_u,n)}{P(x|D_l,n)} \tag{5.3}$$

where X_s^n denotes n-grams in the sentence s, and $P(x|\mathcal{D}, n)$ is the probability of x in the set of n-grams in $\mathcal{D}$. The weights w_n adjust the importance of the scores of n-grams with different lengths. In addition to taking geometric average, we also consider the arithmetic average:

$$\phi_a^N(s) := \sum_{n=1}^{N} \frac{w_n}{|X_s^n|} \sum_{x \in X_s^n} \frac{P(x|D_u, n)}{P(x|D_l, n)} \tag{5.4}$$

As a special case when $N = 1$, the score motivates selecting sentences which increase the number of unique words with new words appearing with higher frequency in D_u than D_l. In the experiments, I will refer to the scores in equations 5.3 and 5.4 as Geom n-gram and Arith n-gram, respectively.

5.4 Sentence Selection: General Techniques

5.4.1 Length of the Source Sentences (Length)

The simplest way to improve the lexicon set and reordering component of the underlying SMT system is to choose longer sentences from the monolingual set of sentences. It should be noted that cost of translating longer sentences is more than shorter ones.

5.4.2 Similarity to the Bilingual Training Data (Similarity)

The simplest way to expand the lexicon set is to choose sentences from D_u which are as dissimilar as possible to D_l. We measure the similarity using weighted n-gram coverage (see Section 4.2.5).

5.4.3 Confidence of Translations (Confidence)

The decoder produces an output translation $\mathbf{e}$ using the probability $p(\mathbf{e} \mid \mathbf{f})$. This probability can be treated as a confidence score for the translation and used to select sentences (see Section 2.3.3). To make the confidence score for sentences with different lengths comparable, we normalize using the sentence length (as we did in Section 4.2.3).

5.4.4 Feature Combination (Combined)

The idea is to take into account the information from several simpler methods, e.g. those mentioned in Sections 5.3–5.4.3, when producing the final ranking of sentences. We can either merge the output rankings of those simpler models[1], or use the scores generated by them as input *features* for a higher level ranking model. We use a linear model:

$$F(s) = \sum_k a_k \phi_k(s) \tag{5.5}$$

where a_k are the model parameters, and $\phi_k(.)$ are the feature functions from Sections 5.3–5.4.3, e.g. confidence score, similarity to D_l, and score for the utility of translation units. Using 20K of Spanish unlabeled text we compared the r^2 correlation coefficient between each of these scores which, apart from the arithmetic and geometric versions of the same score, showed low correlation. So the information they provide should be complementary to each other.

We train the parameters in (5.5) using two bilingual development sets dev1 and dev2, the sentences in dev1 can be ranked with respect to the amount by which each particular sentence improves the BLEU score of the retrained[2] SMT model on dev2. Having this ranking, we look for the weight vector which produces the same ordering of sentences. As an alternative to this method (or its computationally demanding generalization in which instead of a single sentence, several sets of sentences of size k are selected and ranked) we use a hill climbing search on the surface of dev2's BLEU score. For a fixed value of the weight vector, dev1 sentences are ranked and then the top-k output is selected and the amount of improvement the retrained SMT system gives on dev2's BLEU score is measured. Starting from a random initial value for a_k's, we improve one dimension at a time and traverse the discrete grid placed on the values of the weight vector. Starting with a coarse grid, we make it finer when we get stuck in local optima during hill climbing.

5.4.5 Hierarchical Adaptive Sampling (HAS)

We saw in Section 2.3.4 the Hierarchical Sampling (Dasgupta and Hsu, 2008) method for instance selection, which requires a tree representing a hierarchical clustering of the data

[1]To see how different rankings can be combined, see (Reichart et al., 2008) which proposes this for multi-task AL.

[2]Here the retrained SMT model is the one learned by adding a particular sentence from dev1 into D_l.

to be given. In AL for SMT, such a unique clustering of the unlabeled data would be inappropriate or ad-hoc. For this reason, we present a new algorithm inspired by the rationale provided in (Dasgupta and Hsu, 2008) that can be used in our setting, where we construct a tree-based partition of the data dynamically[3]. This dynamic tree construction allows us to extend the HAS algorithm from classifiers to the SMT task.

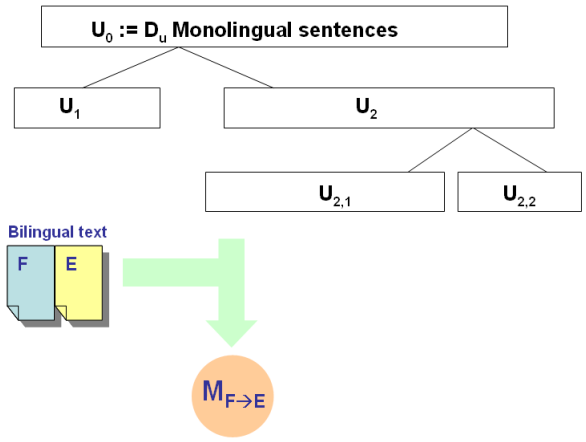


Figure 5.1: Adaptively sampling the sentences while constructing a hierarchical clustering of D_u.

The algorithm adaptively samples sentences from D_u while building a hierarchical clustering of the sentences in D_u (see Fig. 5.1 and Algorithm 8). At any iteration, first we retrain the SMT model and translate all monolingual sentences. At this point one monolingual set of sentences represented by one of the tree leaves is chosen for further partitioning: a leaf H is chosen which has the lowest average decoder confidence score for its sentence translations. We then rank all sentences in H based on their similarity to D_l and put the top $\alpha|H|$ sentences in H_1 and the rest in H_2. To select K sentences, we randomly sample βK sentences from H_1 and $(1-\beta)K$ sentences from H_2 and ask a human for their translations.

[3]The dynamic nature of the hierarchy comes from two factors: (1) selecting a leaf node for splitting, and (2) splitting a leaf node based on its similarity to the growing D_l.

Algorithm 8 Hierarchical-Adaptive-Sampling

$M_{F\to E} = \textbf{train}(D_l, \emptyset)$
Initialize the tree T by setting its root to D_u
$v := \text{root}(T)$
for $t = 1, 2, ...$ **do**
  $X_1, X_2 := \text{Partition}(D_l, v, \alpha)$ // rank and split sentence in v
  $Y_1, Y_2 := \text{Sampling}(X_1, X_2, \beta)$ // randomly sample and remove sents from X_i
  $T := \text{UpdateTree}(X_1, X_2, v, T)$ // make X_i children of node v in the tree T
  $D_l := D_l \cup Y_1^+ \cup Y_2^+$ // Y_i^+ has sents in Y_i together with human trans
  $M_{F\to E} = \textbf{train}(D_l, D_u)$
  for all leaves $l \in T$ **do**
    $Z[l] :=$ Average normalized confidence scores of sentence translations in l
  end for
  $v := \text{BestLeaf}(T, Z)$
  Monitor the performance on the test set D_t
end for

5.4.6 Reverse Model (Reverse)

While a translation system $M_{F\to E}$ is built from language F to language E, we also build a translation system in the reverse direction $M_{E\to F}$. To measure how informative a monolingual sentence $\mathbf{f}$ is, we translate it to English by $M_{F\to E}$ and then project the translation back to French using $M_{E\to F}$. Denote this *reconstructed* version of the original French sentence by $\tilde{\mathbf{f}}$. Comparing $\mathbf{f}$ with $\tilde{\mathbf{f}}$ using BLEU (or other measures) can tell us how much information has been lost due to our direct and/or reverse translation systems. The sentences with higher information loss are selected for translation by a human.

5.5 Experimental Results

The weights of the feature functions in the log-linear SMT model are optimized w.r.t. BLEU score using the MERT algorithm (see Section 2.4.2). This is done on a development corpus referred to as dev1 in this chapter.

The weight vectors in n-gram and similarity methods are set to $(.15, .2, .3, .35)$ to emphasize longer n-grams. We set $\alpha = \beta = .35$ for HAS, and use the 100-best list of translations when identifying candidate phrases while setting the maximum phrase length to 10. We set $\epsilon = .5$ to smooth probabilities when computing scores based on translation units.

corpus	language	use	sentences
EuroParl	French,German,Spanish	in-dom D_l	5K
		in-dom D_u	20K
		in-dom dev	2K
		in-dom test	2K
See Section 5.5.2	Bangla	in-dom D_l	11K
		in-dom D_u	20K
		in-dom dev	450
		in-dom test	1K
Hansards	French	out-dom D_l	5K

Table 5.1: Specification of different data sets we will use in experiments. The target language is English, and the source languages are either French, German, Spanish, or Bangla.

5.5.1 Simulated Low Density Language Pairs

I use three language pairs (French-English, German-English, Spanish-English) to compare all of the proposed sentence selection strategies in a simulated AL setting. The training data comes from the EuroParl corpus as distributed for the shared task in the NAACL 2006 workshop on statistical machine translation (WSMT06). For each language pair, the first 5K sentences from its bilingual corpus constitute D_l, and the next 20K sentences serve as D_u where the target side translation is ignored. The size of D_l was taken to be 5K in order to be close to a realistic setting in SMT. We use the first 2K sentences from the test sets provided for WSMT06, which are in-domain, as our test sets. The corpus statistics are summarized in Table 5.1.

After building the initial MT systems, I select and remove 200 sentences from D_u in each iteration and add them together with translations to D_l for 25 iterations. Each experiment which involves randomness, such as random sentence selection baseline and HAS, is averaged over three independent runs. The results are shown in Figure 5.2 and Figure 5.3, where the former presents the performance in terms of the iteration number of the AL loop. However, the latter presents the performance in terms of the size (number of words) of the selected sentences as the AL loop goes on.

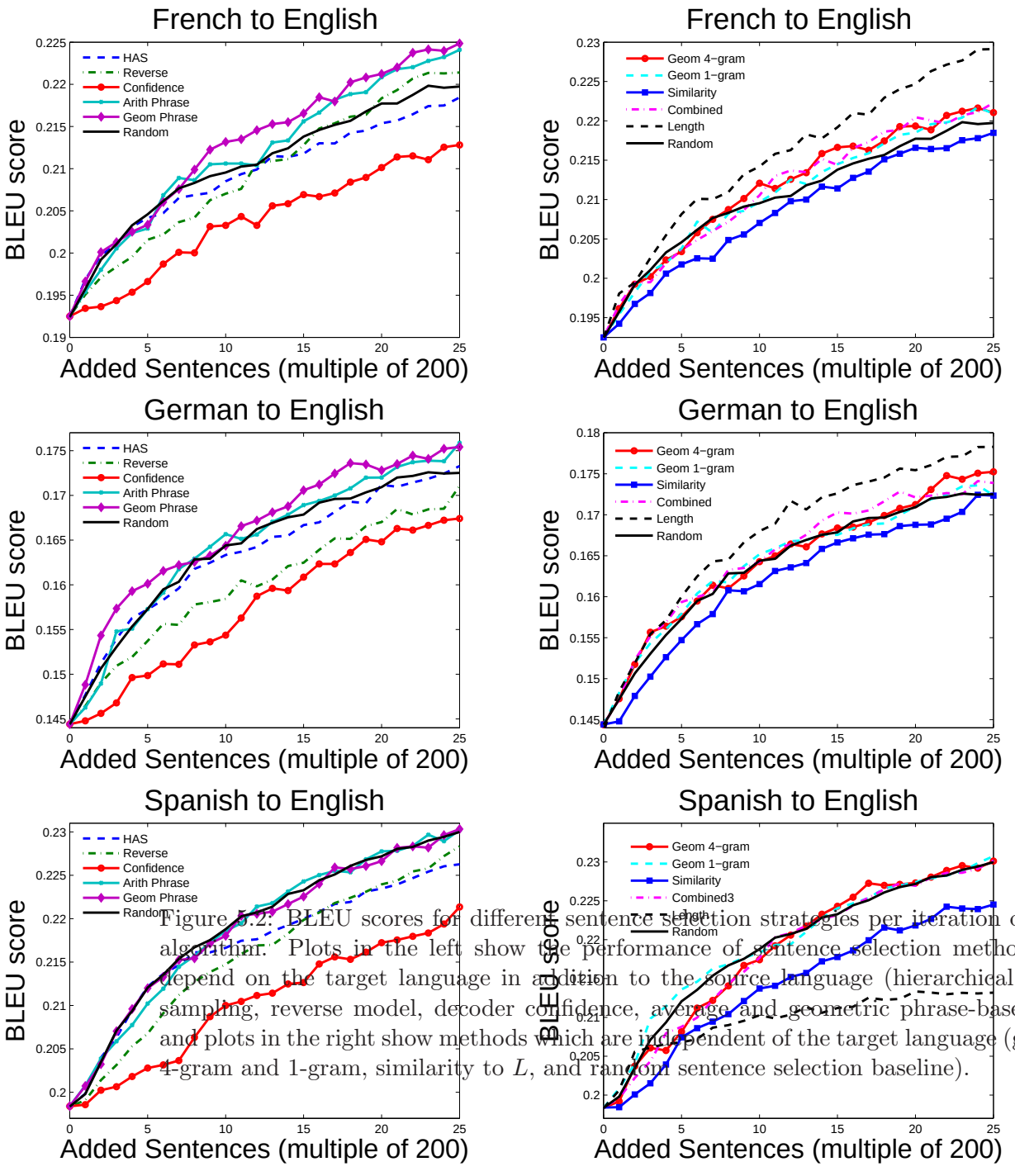


Figure 5.2: BLEU scores for different sentence selection strategies per iteration of the AL algorithm. Plots in the left show the performance of sentence selection methods which depend on the target language in addition to the source language (hierarchical adaptive sampling, reverse model, decoder confidence, average and geometric phrase-based score), and plots in the right show methods which are independent of the target language (geometric 4-gram and 1-gram, similarity to L, and random sentence selection baseline).

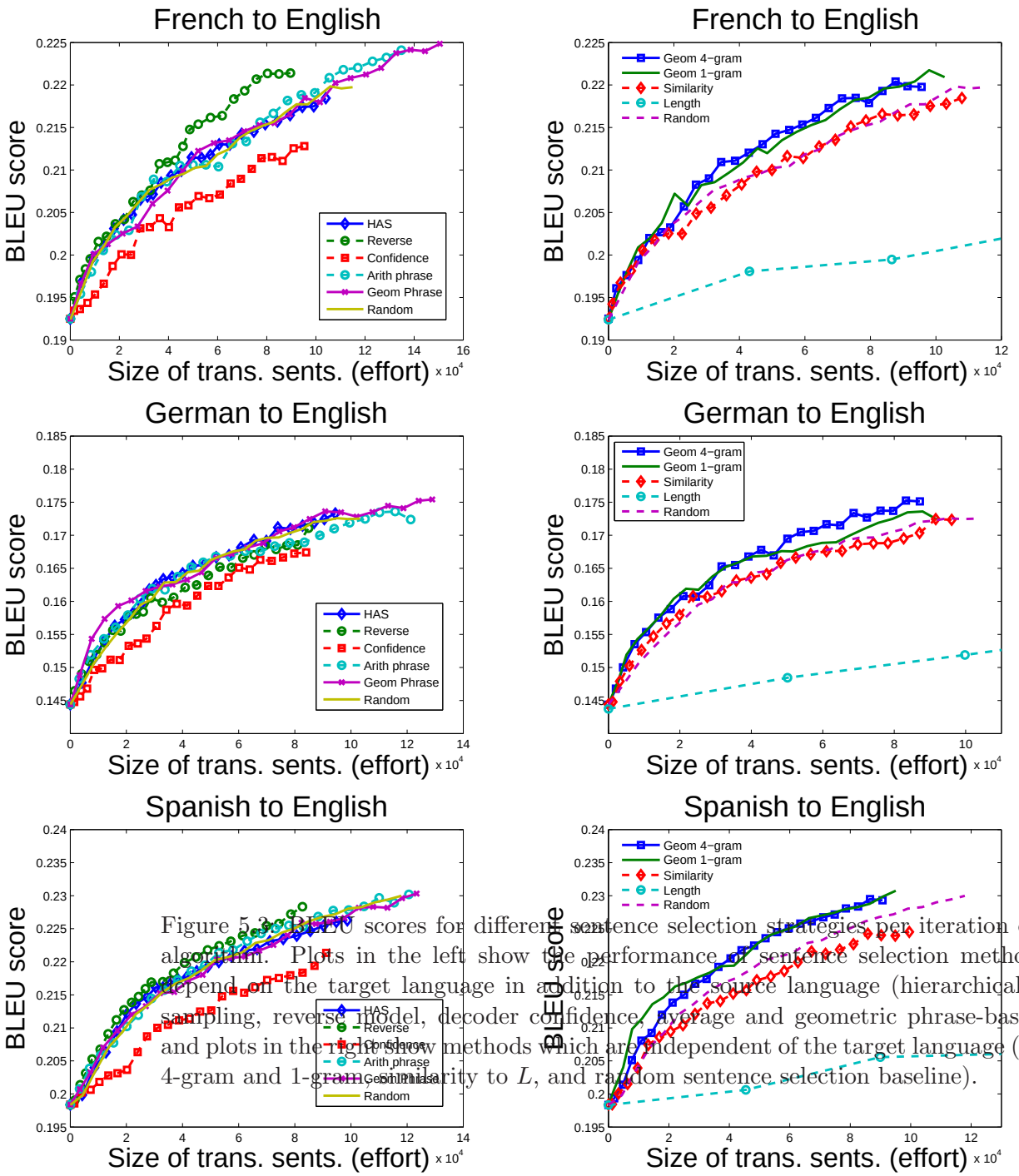


Figure 5.3: BLEU scores for different sentence selection strategies per iteration of the AL algorithm. Plots in the left show the performance of sentence selection methods which depend on the target language in addition to the source language (hierarchical adaptive sampling, reverse model, decoder confidence, average and geometric phrase-based score), and plots in the right show methods which are independent of the target language (geometric 4-gram and 1-gram similarity to L, and random sentence selection baseline).

Lang. Pair	Geom Phrase				Random (baseline)			
	bleu%	per%	wer%	text size	bleu%	per%	wer%	text size
French-En	22.49	27.99	38.45	150477	21.97	28.31	38.80	114950
German-En	17.54	31.51	44.28	128967	17.25	31.63	44.41	103210
Spanish-En	23.03	28.86	39.17	123383	23.00	28.97	39.21	117920

Table 5.2: Phrase-based utility selection is compared with random sentence selection baseline with respect to BLEU, wer (word *error* rate), per (position independent word *error* rate), and the total number of words in the selected sentences across three language pairs.

First we analyze the results in terms of the number of selected sentences (Figure 5.2). Selecting sentences based on the phrase-based utility score outperforms the strong random sentence selection baseline and other methods (Table 5.2) . Decoder confidence performs poorly as a criterion for sentence selection in this setting, and HAS which is built on top of confidence and similarity scores outperforms both of them. Although choosing sentences based on their n-gram score ignores the relationship between source and target languages, this methods outperforms random sentence selection.

Second, we analyze the results in terms of the number of words in the selected sentences (Figure 5.3). Surprisingly, the methods which outperform the strong random sentence selection baseline are 'Geom 4-gram' and 'Geom 1-gram'. We can attribute this to the rapid expansion of the lexicon set based on these methods. The methods which compete closely the random sentence selection baseline are 'Geom Phrase', 'Arith Phrase', 'HAS', and 'Reverse'. We notice that 'Geom Phrase' and 'Arith Phrase' tend to select longer sentences and eventually they outperform the baseline (Table 5.2). Decoder confidence performs poorly as a criterion for sentence selection in terms of the size of the selected sentences.

5.5.2 Realistic Low Density Language Pair

I apply active learning to the Bangla-English machine translation task. Bangla is the official language of Bangladesh and second most spoken language in India. It has more than 200 million speakers around the world. However, Bangla has few available language resources, and lacks resources for machine translation. In our experiments, we use training data provided by the Linguistic Data Consortium[4] containing ∼11k sentences. It contains newswire text from the BBC Asian Network and some other South Asian news websites. A bilingual

[4]LDC Catalog No.: LDC2008E29.

Bangla-English dictionary collected from different websites was also used as part of the training set which contains around 85k words. Our monolingual corpus[5] (Haffari, Roy, and Sarkar, 2009; Roy, 2009) is built by collecting text from the *Prothom Alo* newspaper, and contains all the news available for the year of 2005 – including magazines and periodicals. The corpus has 18,067,470 word tokens and 386,639 word types. For our language model we used data from the English section of EuroParl. The development set used to optimize the model weights in the decoder, and test set used for evaluation was taken from the same LDC corpus mentioned above.

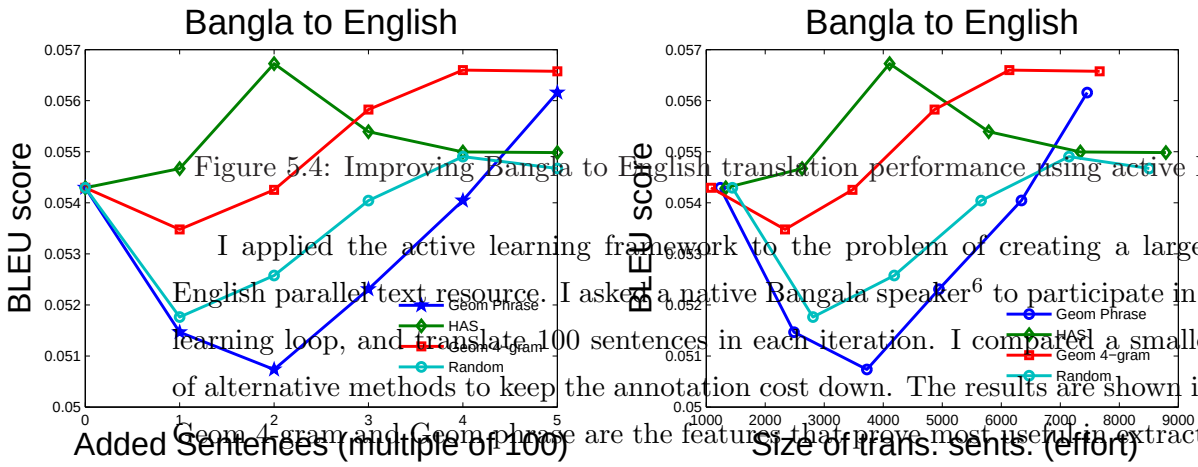


Figure 5.4: Improving Bangla to English translation performance using active learning.

I applied the active learning framework to the problem of creating a larger Bangla-English parallel text resource. I asked a native Bangala speaker[6] to participate in the active learning loop, and translate 100 sentences in each iteration. I compared a smaller number of alternative methods to keep the annotation cost down. The results are shown in Fig. 5.4. Geom 4-gram and Geom phrase are the features that prove most useful in extracting useful sentences for the human expert to translate. Unlike the simulated setting, in this realistic setting for AL, adding more human translation does not always result in better translation performance. This is likely due to the fact that the translator in the AL loop was not the same as the original translator for the labeled data. Another related issue, which in itself is an interesting research topic, is the effect of context on human translator in the real AL

[5]Provided by the Center for Research on Bangla Language Processing, BRAC University, Bangladesh.

[6]Thanks to Maxim Roy to translate Bangala sentences to English.

method	bleu%	per%	wer%	text size
Geom 1-gram	**14.92**	34.83	46.06	135524
Confidence	**14.74**	35.02	46.11	148169
Random (baseline)	14.11	35.28	46.47	149930

Table 5.3: Comparison of methods in domain adaptation scenario. The bold numbers show statistically significant improvement with respect to the baseline. The text size refer to the number of words in the all selected sentences during in the total AL iterations.

setting[7]. Being aware of the context may improve the quality of the human translations. However, the human translator in this experiment indicated that the task was hard enough and reading the context would have slowed him down further.

5.5.3 Domain Adaptation

In this section, I investigate the behavior of the proposed methods when unlabeled data D_u and test data D_t are in-domain and parallel training text D_l is out-of-domain. For example, we may have access to some parliamentary parallel text, but this information needs to be adapted to a novel translation domain, such as for example, speech transcripts, OCR output or blog postings. This scenario can easily be encountered: there is bilingual data for a particular domain, and the goal is to build a translation system for another domain where in-domain monolingual data can be collected by filtering a big pool of source language sentences.

I report experiments for French to English translation task where D_t and development sets are the same as those in section 5.5.1 but the bilingual training data come from Hansards[8] corpus. The domain is similar to EuroParl, but the vocabulary is very different. The results are shown in Fig. 5.5, and summarized in Table 5.3. As expected, unigram based sentence selection performs well in this scenario since it quickly expands the lexicon set of the bilingual data in an effective manner (Fig 5.6). By ignoring sentences for which the translations are already known based on D_l, it does not waste resources. On the other hand, it raises the importance of high frequency words in D_u. Interestingly, decoder confidence is also a good criterion for sentence selection in this particular case.

[7]Personal communication with Kevin Knight.

[8]The transcription of official records of the Canadian Parliament as distributed at http://www.isi.edu/natural-language/download/hansard

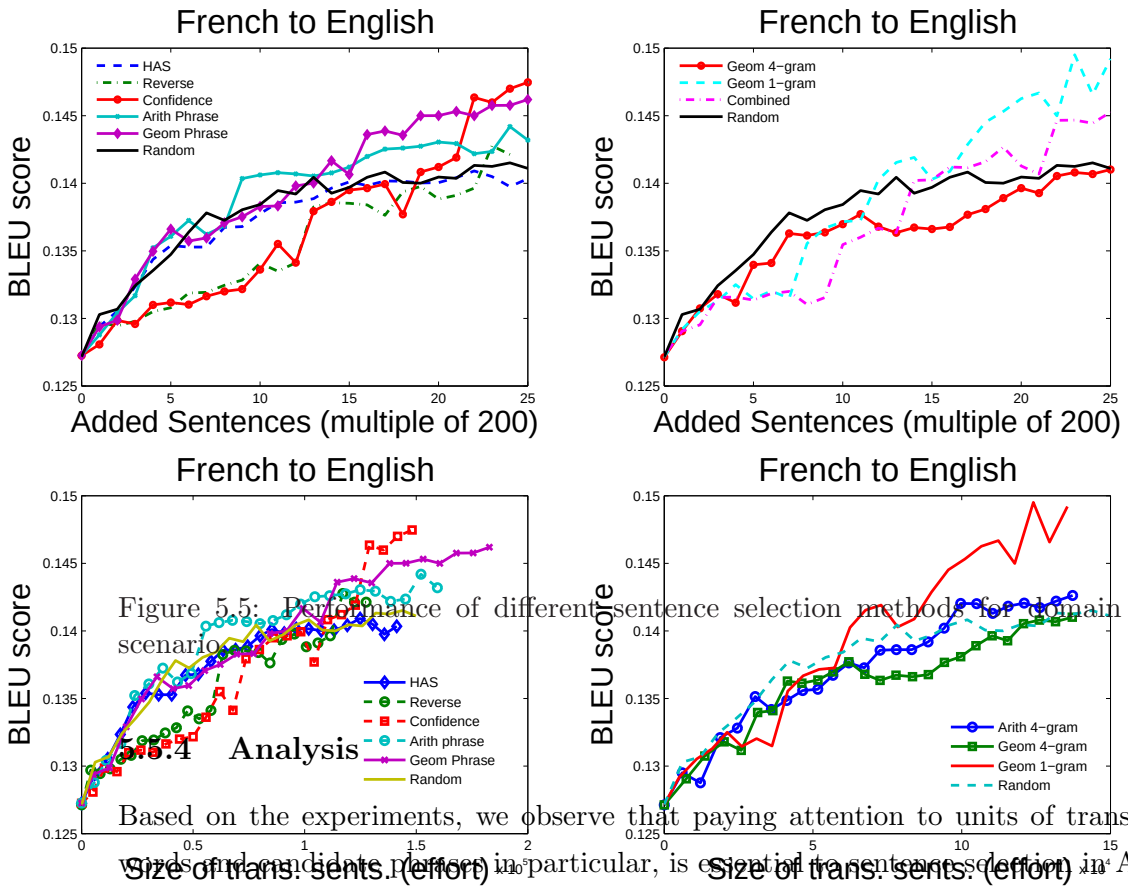


Figure 5.5: Performance of different sentence selection methods for domain adaptation scenarios.

5.5.4 Analysis

Based on the experiments, we observe that paying attention to units of translations, i.e. words and candidate phrases in particular, is essential to sentence selection in AL-SMT. In this section, I present some statistics about the phrases and words added to the underlying phrase-based SMT systems using different sentence selection methods.

Figure 5.6 shows the number of new words added to the SMT system as the active learning loop increases. We see that the following three sentence selection methods increase the number of words the most: Geom Phrase, Geom 1-gram, and Random. Remembering the performances of these methods in terms of the increase in the BLEU score, we can infer

that increasing the coverage of the bilingual training data is important but is not the only factor in a successful sentence selection method. As another evidence, decoder confidence has low coverage (in terms of new words), but performs well in the domain adaptation scenario and performs poorly otherwise. Therefore, we need to look more closely into the methods in terms of the phrases they bring into the system.

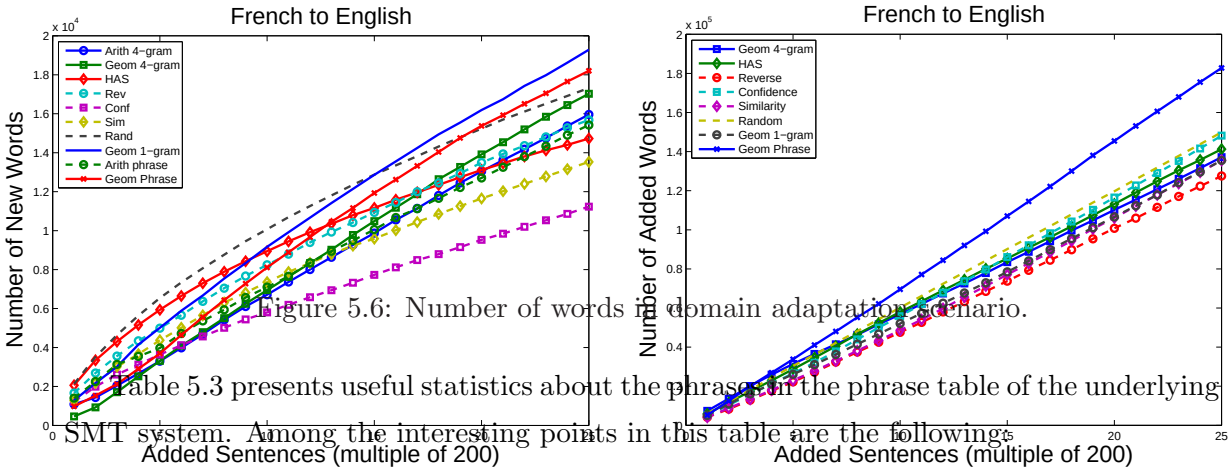


Figure 5.6: Number of words in domain adaptation scenario.

Table 5.3 presents useful statistics about the phrases in the phrase table of the underlying SMT system. Among the interesting points in this table are the following:

- The average source-language phrase length (the second big row in the table) which is low for "Geom Phrase" method compared to "Confidence" and "Random" methods. It shows that "Geom Phrase" motivates having short source-side phrases.

- The number of unique phrase-pairs in the phrase table (the fourth big row in the table) which is higher for "Geom Phrase" method compared to "Confidence" and "Random" methods. It shows that "Geom Phrase" method uses its ability effectively to collect diverse phrase pairs.

5.6 Sentence Selection for Phrase-based SMT: Revisited

Recall the phrase-based scoring schemes in Section 5.3 which was based in the idea that the phrases which may potentially be extracted from a sentence indicate its informativeness. In

	Fr2En	Ge2En	Sp2En	Ha2En
Avg # of trans	1.30	1.26	1.27	1.30
	1.24	1.25	1.20	1.26
	1.22	1.23	1.19	1.24
	1.22	1.24	1.19	1.24
Avg phrase len	2.85	2.56	2.85	2.85
	3.47	2.74	3.54	3.17
	3.95	3.34	3.94	3.48
	3.58	2.94	3.63	3.36
# of phrases	27,566	29,297	30,750	27,566
	78,026	64,694	93,593	108,787
	79,343	63,191	93,276	115,177
	77,394	65,198	94,597	115,671
# unique events	31,824	33,141	34,937	31,824
	103,124	84,512	125,094	117,214
	86,210	69,357	100,176	127,314
	84,787	72,280	101,636	128,912

Table 5.4: Average number of English phrases per source language phrase, average length of the source language phrases, number of source language phrases, and number of phrase pairs which has been seen once in the phrase tables across three language pairs (French text taken from Hansard is abbreviated by 'Ha'). From top to bottom in each row, the numbers belong to: before starting AL, and after finishing AL based on 'Geom Phrase', 'Confidence', and 'Random'.

the labeled data D_l, phrases are the ones which are extracted by the SMT models; but what are the candidate phrases in the unlabeled data D_u? We can use the currently trained SMT models to answer this question. Each translation in the n-best list of translations (generated by the SMT models) corresponds to a particular segmentation of a sentence, which breaks that sentence into several fragments (see Fig. 5.7). Some of these fragments are the source language part of a phrase pair available in the phrase table, which we call *regular phrases* and denote their set by X_s^{reg} for a sentence s. However, there are some fragments in the sentence which are *not* covered by the phrase table – possibly because of the OOVs (out-of-vocabulary words) or the constraints imposed by the phrase extraction algorithm – called X_s^{oov} for a sentence s. Each member of X_s^{oov} offers a set of *potential phrases* (also referred to as *OOV phrases*) which are not observed due to the *latent* segmentation of this fragment. We present three generative models for the phrases and show how to estimate and use them for sentence selection.

5.6.1 Latent Segmentation of OOV Fragments (Model 1)

In the first model, the generative story is to generate phrases for each sentence based on independent draws from a multinomial. The sample space of the multinomial consists of

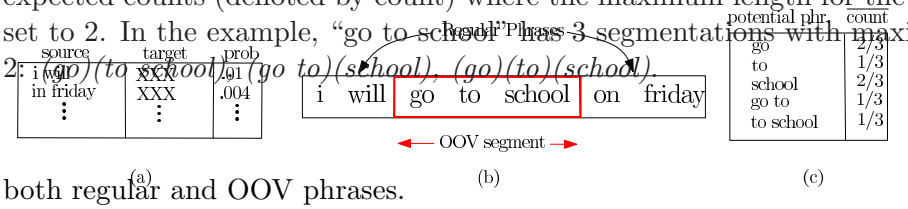


Figure 5.7: The given sentence in (b) is segmented, based on the source side phrases extracted from the phrase table in (a), to yield regular phrases and OOV segment. The table in (c) shows the potential phrases extracted from the OOV segment "go to school" and their expected counts (denoted by $\overline{\text{count}}$) where the maximum length for the potential phrases is set to 2. In the example, "go to school" has 3 segmentations with maximum phrase length 2: *(go)(to school), (go to)(school), (go)(to)(school)*.

both regular and OOV phrases.

We build two models, i.e. two multinomials, one for labeled data and the other one for unlabeled data. Each model is trained by maximizing the log-likelihood of its corresponding data:

$$\mathcal{L}_{\mathcal{D}} := \sum_{\mathbf{f} \in \mathcal{D}} \tilde{P}(\mathbf{f}) \sum_{\mathbf{x} \in X_{\mathbf{f}}} \log P(\mathbf{x}|\boldsymbol{\theta}_{\mathcal{D}}) \tag{5.6}$$

where $\mathcal{D}$ is either D_l or D_u, $\tilde{P}(\mathbf{f})$ is the *empirical* distribution of the sentences[9], and $\boldsymbol{\theta}_{\mathcal{D}}$ is the parameter vector of the corresponding probability distribution. When $\mathbf{x} \in X^{oov}_{\mathbf{f}}$, we will have

$$\begin{aligned} P(\mathbf{x}|\boldsymbol{\theta}_{D_u}) &= \sum_{h \in H_{\mathbf{x}}} P(\mathbf{x}, h|\boldsymbol{\theta}_{D_u}) \\ &= \sum_{h \in H_{\mathbf{x}}} P(h) P(\mathbf{x}|h, \boldsymbol{\theta}_{D_u}) \\ &= \frac{1}{|H_{\mathbf{x}}|} \sum_{h \in H_{\mathbf{x}}} \prod_{\mathbf{y} \in Y^h_{\mathbf{x}}} \boldsymbol{\theta}_{D_u}(\mathbf{y}) \end{aligned} \tag{5.7}$$

where $H_{\mathbf{x}}$ is the space of all possible segmentations for the OOV fragment $\mathbf{x}$, $Y^h_{\mathbf{x}}$ is the resulting phrases from $\mathbf{x}$ based on the segmentation h, and $\boldsymbol{\theta}_{D_u}(\mathbf{y})$ is the probability of the OOV phrase $\mathbf{y}$ in the multinomial associated with D_u. We let $H_{\mathbf{x}}$ to be all possible segmentations of the fragment $\mathbf{x}$ for which the resulting phrase lengths are not greater

[9] $\tilde{P}(\mathbf{f})$ is the number of times that the sentence $\mathbf{f}$ is seen in $\mathcal{D}$ divided by the number of all sentences in $\mathcal{D}$.

than the maximum length constraint for phrase extraction in the underlying SMT model. Since we do not know anything about the segmentations a priori, we have put a uniform distribution over such segmentations.

Maximizing (5.6) to find the maximum likelihood parameters for this model is an extremely difficult problem[10]. Therefore, we maximize the following lower-bound on the log-likelihood which is derived using Jensen's inequality:

$$\mathcal{L}_{\mathcal{D}} \geq \sum_{\mathbf{f} \in \mathcal{D}} \tilde{P}(\mathbf{f}) \Big[\sum_{\mathbf{x} \in X_{\mathbf{f}}^{reg}} \log \boldsymbol{\theta}_{\mathcal{D}}(\mathbf{x}) + \sum_{\mathbf{x} \in X_{\mathbf{f}}^{oov}} \sum_{h \in H_{\mathbf{x}}} \frac{1}{|H_{\mathbf{x}}|} \sum_{\mathbf{y} \in Y_{\mathbf{x}}^{h}} \log \boldsymbol{\theta}_{\mathcal{D}}(\mathbf{y}) \Big] \tag{5.8}$$

Maximizing (5.8) amounts to set the probability of each regular/potential phrase proportional to its count/*expected* count in the data $\mathcal{D}$.

Let $\rho_k(\mathbf{x}_{i:j})$ be the number of possible segmentations from position i to position j of an OOV fragment $\mathbf{x}$, and k is the maximum phrase length;

$$\rho_k(\mathbf{x}_{1:|\mathbf{x}|}) = \begin{cases} 0, & \text{if } |\mathbf{x}| = 0 \\ 1, & \text{if } |\mathbf{x}| = 1 \\ \sum_{i=1}^{k} \rho_k(\mathbf{x}_{i+1:|\mathbf{x}|}), & \text{otherwise} \end{cases}$$

which gives us a dynamic programming algorithm to compute the number of segmentation $|H_{\mathbf{x}}| = \rho_k(\mathbf{x}_{1:|\mathbf{x}|})$ of the OOV fragment $\mathbf{x}$. The expected count of a potential phrase $\mathbf{y}$ based on an OOV segment $\mathbf{x}$ is (see Fig. 5.7.c):

$$E[\mathbf{y}|\mathbf{x}] = \frac{\sum_{i \leq j} \delta_{[\mathbf{y}=\mathbf{x}_{i:j}]} \rho_k(\mathbf{x}_{1:i-1}) \rho_k(\mathbf{x}_{j+1:|\mathbf{x}|})}{\rho_k(\mathbf{x})}$$

where $\delta_{[C]}$ is 1 if the condition C is true, and zero otherwise. I have used the fact that the number of occurrences of a phrase spanning the indices $[i, j]$ is the product of the number of segmentations of the left and the right sub-fragments, which are $\rho_k(\mathbf{x}_{1:i-1})$ and $\rho_k(\mathbf{x}_{j+1:|\mathbf{x}|})$ respectively.

5.6.2 Separate Models for OOV and Regular Phrases (Model 2)

In the second model, we consider a mixture model of two multinomials responsible for generating phrases in each of the labeled and unlabeled data sets. To generate a phrase,

[10]Setting partial derivatives of the Lagrangian to zero amounts to finding the roots of a system of multivariate polynomials (a major topic in Algebraic Geometry).

we first toss a coin and depending on the outcome we either generate the phrase from the multinomial associated with regular phrases $\boldsymbol{\theta}^{reg}_{D_u}$ or potential phrases $\boldsymbol{\theta}^{oov}_{D_u}$:

$$P(\mathbf{x}|\boldsymbol{\theta}_{D_u}) := \beta_{D_u}\boldsymbol{\theta}^{reg}_{D_u}(\mathbf{x}) + (1-\beta_{D_u})\boldsymbol{\theta}^{oov}_{D_u}(\mathbf{x})$$

where $\boldsymbol{\theta}_{D_u}$ includes the mixing weight β and the parameter vectors of the two multinomials. The mixture model associated with L is written similarly. The parameter estimation is based on maximizing a lower-bound on the log-likelihood which is similar to what was done for the Model 1.

5.6.3 Sentence Scoring

The sentence score is a linear combination of two terms: one coming from regular phrases and the other from OOV phrases:

$$\phi_1(\mathbf{f}) := \frac{\lambda}{|X^{reg}_{\mathbf{f}}|}\sum_{\mathbf{x}\in X^{reg}_{\mathbf{f}}}\log\frac{P(\mathbf{x}|\boldsymbol{\theta}_U)}{P(\mathbf{x}|\boldsymbol{\theta}_L)}$$
$$+\frac{1-\lambda}{|X^{oov}_{\mathbf{f}}|}\sum_{\mathbf{x}\in X^{oov}_{\mathbf{f}}}\sum_{h\in H_{\mathbf{x}}}\frac{1}{|H_{\mathbf{x}}|}\log\prod_{\mathbf{y}\in Y^h_{\mathbf{x}}}\frac{P(\mathbf{y}|\boldsymbol{\theta}_U)}{P(\mathbf{y}|\boldsymbol{\theta}_L)}$$

where we use either Model 1 or Model 2 for $P(.|\boldsymbol{\theta}_{\mathcal{D}})$. The first term is the log probability ratio of regular phrases under phrase models corresponding to unlabeled and labeled data, and the second term is the *expected* log *probability ratio* (ELPR) under the two models. Another option for the contribution of OOV phrases is to take log *of expected probability ratio* (LEPR):

$$\phi_2(\mathbf{f}) := \frac{\lambda}{|X^{reg}_{\mathbf{f}}|}\sum_{\mathbf{x}\in X^{reg}_{\mathbf{f}}}\log\frac{P(\mathbf{x}|\boldsymbol{\theta}_U)}{P(\mathbf{x}|\boldsymbol{\theta}_L)}$$
$$+\frac{1-\lambda}{|X^{oov}_{\mathbf{f}}|}\sum_{\mathbf{x}\in X^{oov}_{\mathbf{f}}}\log\sum_{h\in H_{\mathbf{x}}}\frac{1}{|H_{\mathbf{x}}|}\prod_{\mathbf{y}\in Y^h_{\mathbf{x}}}\frac{P(\mathbf{y}|\boldsymbol{\theta}_U)}{P(\mathbf{y}|\boldsymbol{\theta}_L)}$$

It is not difficult to prove that there is no difference between Model 1 and Model 2 when ELPR scoring is used for sentence selection. However, the situation is different for LEPR scoring: the two models produce different sentence rankings in this case.

5.7 Experimental Results

The weights of the feature functions in the log-linear SMT model are optimized w.r.t. BLEU score using the MERT algorithm (see Section 2.4.2) based on a development corpus. I use

method	French to En		Spanish to En		German to En	
	BLEU	text size	BLEU	text size	BLEU	text size
Model 2 - LEPR	23.6581	165506	24.9526	205589	17.7815	156471
Model 1 - ELPR	23.5468	170661	25.0770	216997	17.8449	162458
Geom Phrase	23.4680	183519	24.8133	196408	17.6219	171046
Random	23.4058	160520	24.6138	192308	17.4936	137260

Table 5.5: The BLEU score and the size (number of words) of the selected text for different sentence selection strategies after the *last* iteration of the AL loop on three translation tasks.

add-ϵ smoothing where $\epsilon = .5$ to smooth the probabilities in Section 5.6; moreover $\lambda = .4$ for ELPR and LEPR sentence scoring and maximum phrase length k is set to 4.

5.7.1 Simulated Low Density Language Pairs

I use three language pairs in these experiments: French-English, German-English, and Spanish-English from the dataset described in Section 6.4. The reason for using this dataset is that I am going to use the best sentence selection method developed in single language pair setting for the multilingual setting, so it is better to see how the methods behave on this dataset.

After building the initial MT system for each experiment, we select and remove 500 sentences from U and add them together with translations to L for 10 total iterations. Compared to the experiments in Section 5.5.1, I set the number of iterations to 10 instead of 25 since I also use it for multilingual experiments in Section 6.4 (to reduce the running time of the multilingual experiments). However, I set the number of selected sentences to 500 instead of 200 so that eventually the number of selected sentences in both sets of experiments becomes equal (i.e. 5000 sentences).

I compare the new sentence selection methods proposed in Section 5.6 to the "Geom Phrase" which was the best method in Section 5.5.1. It differs from the new methods in that it considers each individual OOV fragments as a single OOV phrase (and does not consider latent segmentations). I also compare against random sentence selection baselines which are averaged over 3 independent runs. The results are presented in Figure 5.8 and Table 5.7.1. Selecting sentences based on the new methods outperform the random sentence selection baseline and "Geom Phrase". I suspect for the situations where L is out-of-domain and the average phrase length is relatively small, the gap between performances becomes even more.

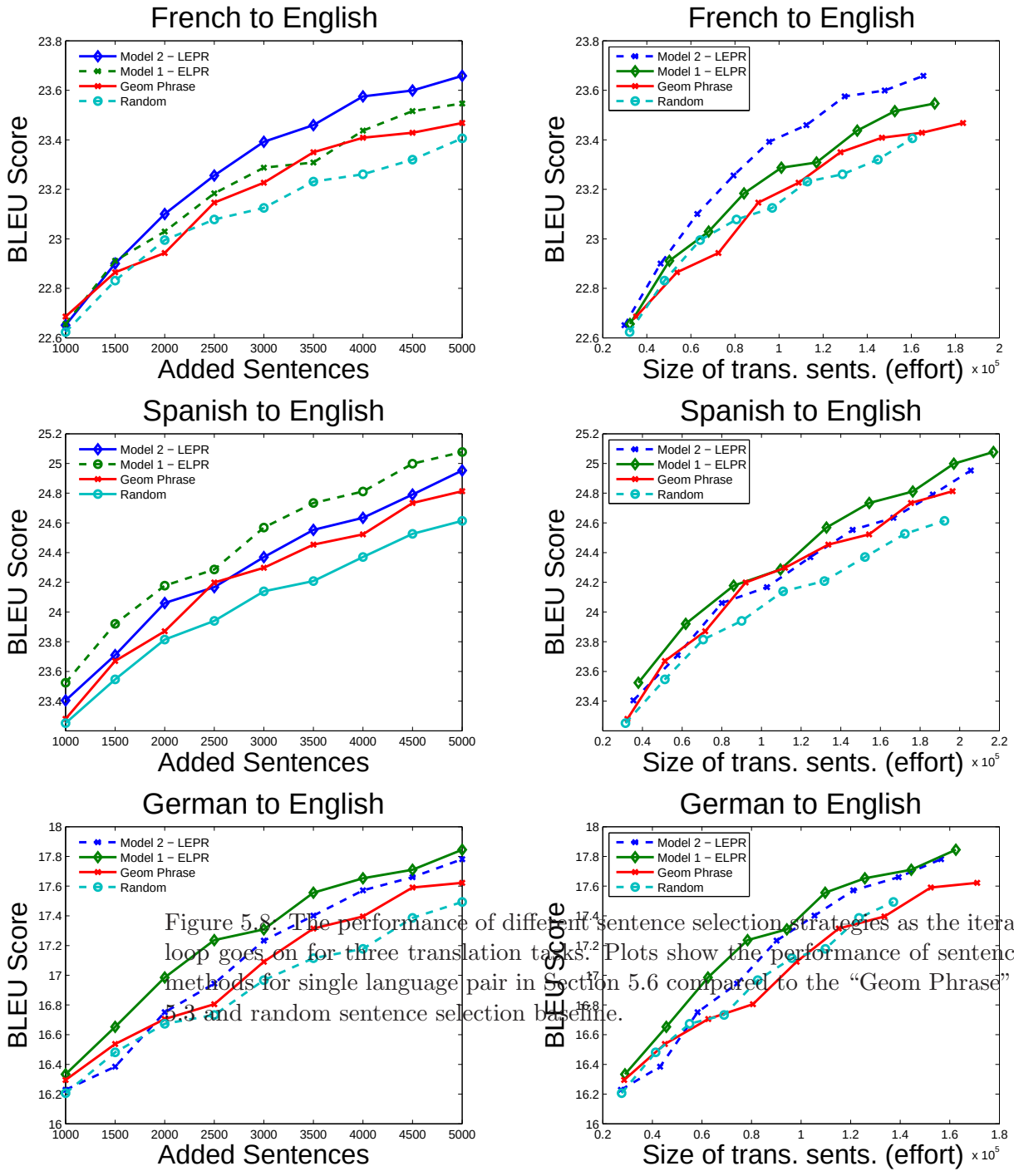


Figure 5.8: The performance of different sentence selection strategies as the iteration of AL loop goes on for three translation tasks. Plots show the performance of sentence selection methods for single language pair in Section 5.6 compared to the "Geom Phrase" in Section 5.3 and random sentence selection baseline.

5.7.2 Analysis

The basis for the proposed new methods has been the popularity of regular/OOV phrases in D_u and their unpopularity in D_l, which is measured by $\frac{P(\mathbf{x}|\boldsymbol{\theta}_{D_u})}{P(\mathbf{x}|\boldsymbol{\theta}_{D_l})}$. We need $P(\mathbf{x}|\boldsymbol{\theta}_{D_u})$, the *estimated* distribution of phrases in D_u, to be as similar as possible to $P^*(\mathbf{x})$, the *true* distribution of phrases in D_u. We investigate this issue for regular/OOV phrases as follows:

- Using the output of the initially trained MT system on D_l, we extract the regular/OOV phrases as described in Section 5.6. The smoothed relative frequencies give us the regular/OOV phrasal distributions.
- Using the *true* English translation of the sentences in D_u, we extract the true phrases. Separating the phrases into two sets of regular and OOV phrases defined by the previous step, we use the smoothed relative frequencies and form the *true* OOV/regular phrasal distributions.

	De2En	Fr2En	Es2En
$KL(P^*_{reg} \parallel P_{reg})$	4.37	4.17	4.38
$KL(P^*_{reg} \parallel unif)$	5.37	5.21	5.80
$KL(P^*_{oov} \parallel P_{oov})$	3.04	4.58	4.73
$KL(P^*_{oov} \parallel unif)$	3.41	4.75	4.99

Table 5.6: For regular/OOV phrases, the KL-divergence between the true distribution (P^*) and the estimated (P) or uniform ($unif$) distributions are shown, where: $KL(P^* \parallel P) := \sum_x P^*(x) \log \frac{P^*(x)}{P(x)}$.

I use the KL-divergence to see how dissimilar are a pair of given probability distributions. As Table 5.6 shows, the KL-divergence between the true and estimated distributions are less than that between the true and uniform distributions, in all three language pairs. Since uniform distribution conveys no information, this is evidence that there is some information encoded in the estimated distribution about the true distribution. However I noticed that the true distributions of regular/OOV phrases exhibit Zipfian (power law) behavior[11] which is not well captured by the estimated distributions (see Fig. 5.9). Enhancing the estimated distributions to capture this power law behavior would improve the quality of the proposed sentence selection methods.

[11]This observation is at the *phrase* level and not at the *word* (Zipf, 1932) or even n-gram level (Ha et al., 2002).

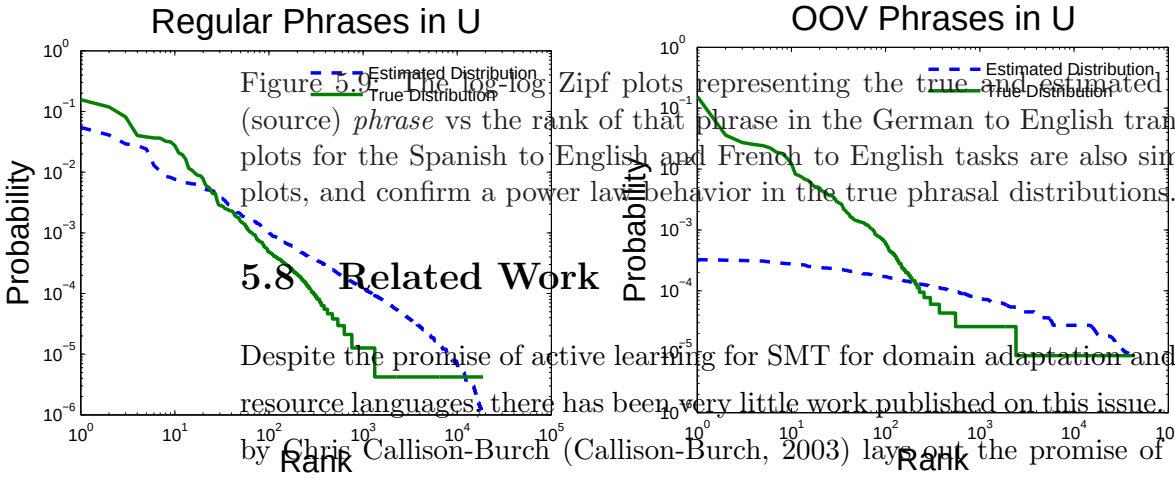


Figure 5.9: The log-log Zipf plots representing the true and estimated probabilities of a (source) *phrase* vs the rank of that phrase in the German to English translation task. The plots for the Spanish to English and French to English tasks are also similar to the above plots, and confirm a power law behavior in the true phrasal distributions.

5.8 Related Work

Despite the promise of active learning for SMT for domain adaptation and low-density/low-resource languages, there has been very little work published on this issue. A Ph.D. proposal by Chris Callison-Burch (Callison-Burch, 2003) lays out the promise of AL for SMT and proposes some methods. More specifically, he proposes to use the entropy of the N-best list of translations to select sentences. I have experimented with entropy-based sentence selection method but the results are not satisfactory. Partly, the reason may be the repetition of generated translations in the N-best list. More principled ways of approximating the entropy of the translation distribution may produce better results. Moreover, (Callison-Burch, 2003) suggests using a language model trained on the source side bilingual data to identify monolingual sentences which are most dissimilar to the sentences encountered in the training data. This method is similar to the one mentioned in Section 5.4.2 which uses n-gram coverage instead of a language model.

(Mohit and Hwa, 2007) provide a technique to classify phrases as difficult to translate (DTP), and incorporate human translations for these phrases. Their approach is different from AL: they use human translations for DTPs in order to improve translation output in the decoder. There is work on sampling sentence pairs for SMT (Kauchak, 2006; Eck,

Vogel, and Waibel, 2005) but the goal has been to limit the amount of training data in order to reduce the memory footprint of the SMT decoder. To compute this score, (Eck, Vogel, and Waibel, 2005) use n-gram features very different from the n-gram features proposed in this chapter. (Ronald and Barnard, 2007) implement an AL system for SMT for language pairs with limited resources (En-Xhosa, En-Zulu, En-Setswana and En-Afrikaans), but the experiments are on a very small simulated data set. The only feature used is the confidence score of the SMT system, which we showed in our experiments is not a reliable feature.

5.9 Summary

I provided a novel active learning framework for SMT which utilizes both labeled and unlabeled data. Several sentence selection strategies were proposed and comprehensively compared across three simulated language pairs and a realistic setting of Bangla-English translation with scarce resources. The experimental results support that paying attention to units of translations, i.e. words and candidate phrases in particular, is essential for successful sentence selection in AL-SMT. I performed detailed experimental evaluation of the proposed sentence selection methods, and studied their performance in the domain adaptation scenario as well the usual translation scenario.

Chapter 6

Active Learning for SMT: Multiple Language Pairs

This chapter introduces an active learning task of adding a new language to an existing multilingual set of parallel text and constructing high quality MT systems, from each language in the collection into this new target language which we refer to as active learning for multiple language pairs. I show that adding a new language using active learning to the EuroParl corpus provides a significant improvement compared to a random sentence selection baseline. I also provide new highly effective sentence selection methods that improve AL for phrase-based SMT in the multiple language-pairs setting. By "multiple language-pairs" setting, I mean multiple source-languages to a single target-language setting in the rest of this section.

6.1 The Motivation

The main source of training data for statistical machine translation (SMT) models is a parallel corpus. In many cases, the same information is available in multiple languages simultaneously as a multilingual parallel corpus, e.g., European Parliament (EuroParl) and U.N. proceedings. I consider how to use active learning (AL) in order to add a new language to such a multilingual parallel corpus and at the same time to construct a high-quality MT system from each language in the original corpus into this new target language. I introduce a novel combined measure of translation quality for multiple target language outputs (the

same content from multiple source languages).

The multilingual setting provides new opportunities for AL over and above a single language pair. This setting is similar to the multi-task AL scenario (Reichart et al., 2008). In our case, the multiple tasks are individual machine translation tasks for several language pairs. The nature of the translation processes vary from any of the source languages to the new language depending on the characteristics of each source-target language pair, hence these tasks are competing for annotating the same resource. However it may be that in a single language pair, AL would pick a particular sentence for annotation, but in a multilingual setting, a different source language might be able to provide a good translation, thus saving annotation effort. I explore how multiple MT systems can be used to effectively pick instances that are more likely to improve training quality.

When we build multiple MT systems from multiple source languages to the new target language, each MT system can be seen as a different 'view' on the desired output translation. Thus, we can train our multiple MT systems using either *self-training* or *co-training* (Blum and Mitchell, 1998). In self-training, each MT system is re-trained using human labeled data plus its own noisy translation output on the unlabeled data. In co-training, each MT system is re-trained using human labeled data plus noisy translation output from the other MT systems in the ensemble. We use consensus translations (He et al., 2008; Rosti et al., 2007; Matusov, Ueffing, and Ney, 2006) as an effective method for co-training between multiple MT systems.

6.2 The Algorithmic Framework

Consider a multilingual parallel corpus, such as EuroParl, which contains parallel sentences for several languages. Our goal is to add a new language to this corpus, *and* at the same time to construct high quality MT systems from the existing languages (in the multilingual corpus) to the new language. This goal is formalized by the following objective function:

$$\mathcal{O} = \sum_{\ell=1}^{L} \alpha_\ell \times TQ(M_{F^\ell \rightarrow E}) \tag{6.1}$$

where F^ℓ's are the source languages in the multilingual corpus (L is the total number of languages), and E is the new language. The translation quality is measured by TQ for individual systems $M_{F^\ell \rightarrow E}$; it can be BLEU score or WER/PER (Word error rate

and position independent WER) which induces a maximization or minimization problem, respectively. The non-negative weights α_ℓ reflect the importance of the different translation tasks and $\sum_\ell \alpha_\ell = 1$. AL-SMT formulation for *single* language pair is a special case of this formulation where only one of the α_d's in the objective function (1) is one and the rest are zero.

I denote the large *unlabeled* multilingual corpus by $\mathbb{D}_u := \{(\mathbf{f}_j^1, .., \mathbf{f}_j^L)\}$, and the small *labeled* multilingual corpus by $\mathbb{D}_l := \{(\mathbf{f}_i^1, .., \mathbf{f}_i^L, \mathbf{e}_i)\}$. We overload the term *entry* to denote a tuple in $\mathbb{D}_l$ or in $\mathbb{D}_u$ (it should be clear from the context).

Algorithm 9 represents our AL approach for the multilingual setting, which is similar to the Algorithm 7 for AL-SMT in a single language pair setting. We train our initial SMT systems $\{M_{F^\ell \rightarrow E}\}_{\ell=1}^L$ on the multilingual corpus $\mathbb{D}_l$, and use them to translate *all* monolingual sentences in $\mathbb{D}_u$. We denote sentences in $\mathbb{D}_u$ together with their multiple translations by $\mathbb{D}_u^+$ (line 4 of Algorithm 9). Then we retrain the SMT systems on $\mathbb{D}_l \cup \mathbb{D}_u^+$ and use the resulting model to decode the test set. Afterwards, we select and remove a subset of highly informative sentences from $\mathbb{D}_u$, and add those sentences together with their human-provided translations to $\mathbb{D}_l$. This process is continued iteratively until a certain level of translation quality is met (we use the BLEU score, WER and PER) (Papineni et al., 2002).

When (re-)training the models, two phrase tables are learned for each SMT model: one from the labeled data $\mathbb{D}_l$ and the other one from *pseudo-labeled* data $\mathbb{D}_u^+$ (which we call the *main* and *auxiliary* phrase tables respectively). Since each entry in $\mathbb{D}_u^+$ has multiple translations, there are two options when building the auxiliary table for a particular language pair (F^ℓ, E): (i) to use the corresponding translation $\mathbf{e}^\ell$ of the source language in a *self-training* setting, or (ii) to use the *consensus* translation among all the translation candidates $(\mathbf{e}^1, .., \mathbf{e}^L)$ in a *co-training* setting (sharing information between multiple SMT models).

A whole range of methods exist in the literature for combining the output translations of multiple MT systems for a *single* language pair, operating either at the sentence, phrase, or word level (He et al., 2008; Rosti et al., 2007; Matusov, Ueffing, and Ney, 2006). The method that we use in this work operates at the sentence level, and picks a single high quality translation from the union of the n-best lists generated by multiple SMT models. Section 6.4 gives more details about features which are used in our consensus finding method, and how it is trained. In the following section, I push the ideas in Section 5.3 further and introduce a more elaborate phrase-based sentence scoring for a AL-SMT for a single language pair. Armed with this enhanced sentence selection strategy, I address the important question

Algorithm 9 AL-SMT: Multiple Language Pairs

1: Given multilingual corpora $\mathbb{D}_l$ and $\mathbb{D}_u$
2: $\{M_{F^\ell \to E}\}_{\ell=1}^{L} = \textbf{multrain}(\mathbb{D}_l, \emptyset)$
3: **for** $t = 1, 2, ...$ **do**
4: $\quad \mathbb{U}^+ = \textbf{multranslate}(\mathbb{D}_u, \{M_{F^\ell \to E}\}_{\ell=1}^{L})$
5: $\quad$ Select k sentences from $\mathbb{U}^+$, and ask a human for their *true* translations.
6: $\quad$ Remove the k sentences from $\mathbb{U}$, and add the k sentence pairs (translated by human) to $\mathbb{L}$
7: $\quad \{M_{F^\ell \to E}\}_{\ell=1}^{L} = \textbf{multrain}(\mathbb{D}_l, \mathbb{D}_u^+)$
8: $\quad$ Monitor the performance on the test set
9: **end for**

of selecting highly informative sentences (step 5 in the Algorithm 9) for the AL-SMT in multiple language-pairs in Section 6.3.

6.3 Sentence Selection for Multiple Language Pairs

The goal is to optimize the objective function (6.1) with minimum human effort in providing the translations. This motivates selecting sentences which are *maximally* beneficial for *all* the MT systems. In this section, we present several protocols for sentence selection based on the combined information from multiple language pairs.

6.3.1 Alternating Selection

The simplest selection protocol is to choose k sentences (entries) in the first iteration of AL which improve maximally the first model $M_{F^1 \to E}$, while ignoring other models. In the second iteration, the sentences are selected with respect to the second model, and so on (Reichart et al., 2008).

6.3.2 Combined Ranking

Pick any AL-SMT scoring method for a *single* language pair (see Sec. 5.3). Using this method, we rank the entries in unlabeled data $\mathbb{D}_{\approx}$ for each translation task defined by language pair (F^ℓ, E). This results in several ranking lists, each of which represents the importance of entries with respect to a particular translation task. We combine these

rankings using a combined score:

$$Score\big((\mathbf{f}^1,..,\mathbf{f}^L)\big) = \sum_{\ell=1}^{L} \alpha_\ell \text{Rank}_\ell(\mathbf{f}^\ell)$$

$\text{Rank}_\ell(.)$ is the ranking of a sentence in the list for the ℓ^{th} translation task (Reichart et al., 2008).

6.3.3 Disagreement Among the Translations

Disagreement among the candidate translations of a particular entry is evidence for the difficulty of that entry for different translation models. The reason is that disagreement increases the possibility that most of the translations are not correct. Therefore it would be beneficial to ask human for the translation of these hard entries.

Now the question is how to quantify the notion of disagreement among the candidate translations $(\mathbf{e}^1,..,\mathbf{e}^L)$. We propose two measures of disagreement which are related to the portion of shared n-grams ($n \leq 4$) among the translations:

- Let $\mathbf{e}^c$ be the consensus among all the candidate translations, then define the disagreement as:

  $$\sum_{\ell} \alpha_\ell \big(1 - \text{BLEU}(\mathbf{e}^c, \mathbf{e}^\ell)\big)$$

  We will see shortly how a consensus translation can be determined among a set of generated translations.

- Based on the disagreement of every pair of candidate translations:

  $$\sum_{\ell} \alpha_\ell \sum_{\ell'} \big(1 - \text{BLEU}(\mathbf{e}^{\ell'}, \mathbf{e}^\ell)\big)$$

6.3.3.1 Consensus Finding

Let T be the union of the n-best lists of translations for a particular sentence. The consensus translation $\mathbf{e}^c$ is:

$$\arg\max_{\mathbf{e}\in T} w_1 \frac{LM(\mathbf{e})}{|\mathbf{e}|} + w_2 \frac{Q_d(\mathbf{e})}{|\mathbf{e}|} + w_3 R_d(\mathbf{e}) + w_{4,d} \tag{6.2}$$

where $\text{LM}(\mathbf{e})$ is the score from a 3-gram language model, $Q_d(\mathbf{e})$ is the translation score generated by the decoder for $M_{F^d \rightarrow E}$ if $\mathbf{e}$ is produced by the dth SMT model, $R_d(\mathbf{e})$ is the

rank of the translation in the n-best list produced by the dth model, $w_{4,d}$ is a bias term for each translation model to make their scores comparable, and $|\mathbf{e}|$ is the length of the translation sentence. The number of weights w_i is 3 plus the number of source languages, and they are trained using MERT (see Section 2.4.2) to maximize the BLEU score on a development set.

6.4 Experimental Results

I pre-processed the EuroParl corpus (*http://www.statmt.org/europarl*) (Koehn, 2005) and built a multilingual parallel corpus with 653,513 sentences, excluding the Q4/2000 portion of the data (2000-10 to 2000-12) which is reserved as the test set. I subsampled 5,000 sentences as the labeled data $\mathbb{L}$ and 20,000 sentences as $\mathbb{U}$ for the pool of untranslated sentences (while hiding the English part). The test set consists of 2,000 multi-language sentences and comes from the multilingual parallel corpus built from Q4/2000 portion of the data.

I use add-ϵ smoothing where $\epsilon = .5$ to smooth the probabilities in Section 5.6; moreover $\lambda = .4$ for ELPR and LEPR sentence scoring and maximum phrase length k is set to 4. For the multilingual experiments (which involve four source languages), I set $\alpha_d = .25$ to make the importance of individual translation tasks equal.

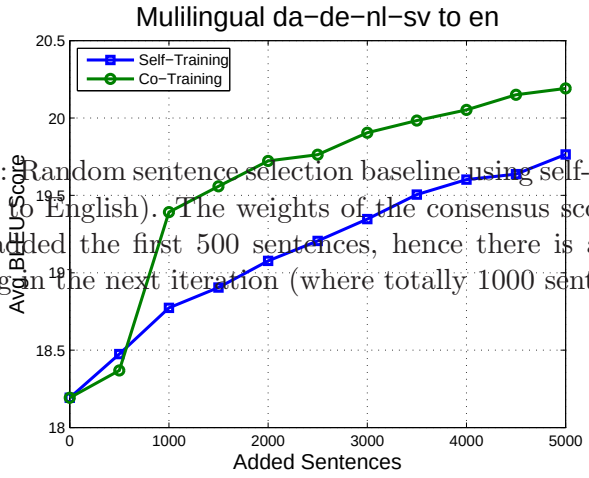


Figure 6.1: Random sentence selection baseline using self-training and co-training (Germanic languages to English). The weights of the consensus scoring model (6.2) are trained *after* we have added the first 500 sentences, hence there is a jump in the performance of the co-training on the next iteration (where totally 1000 sentences are added).

6.4.1 Simulated Multiple Language Pairs Setting

I use Germanic (German, Dutch, Danish, Swedish) and Romance (French, Spanish, Italian, Portuguese languages as the source and English as the target language as two sets of experiments.[1] Figure 6.1 shows the performance of random sentence selection for AL combined with self-training/co-training for the multi-source translation from the four Germanic languages to English. It shows that the co-training mode outperforms the self-training mode by almost 1 BLEU point.

The results of selection strategies in the multilingual setting are presented in Figure 6.2 and Table 6.1. Having noticed that Model 1 with ELPR performs well in the single language pair setting, I use it to rank entries for individual translation tasks. Then these rankings are used by 'Alternate' and 'Combined Rank' selection strategies in the multilingual case. The 'Combined Rank' method outperforms all the other methods including the strong random selection baseline in both self-training and co-training modes. The disagreement-based selection methods underperform the baseline for translation of Germanic languages to English, so I omitted them for the Romance language experiments.

6.5 Related Work

(Reichart et al., 2008) introduces multi-task active learning where unlabeled data require annotations for multiple tasks, e.g. they consider named-entities and parse trees, and showed that multiple tasks helps selection compared to individual tasks. The setting of this chapter is different in that the target language is the same across multiple MT tasks, which we exploit to use consensus translations and co-training to improve active learning performance. (Callison-Burch and Osborne, 2003b; Callison-Burch and Osborne, 2003a) provide a co-training approach to MT, where one language pair creates data for another language pair. In contrast, the co-training approach in this chapter uses consensus translations and this setting for active learning is very different from their semi-supervised setting. While I use consensus translations (He et al., 2008; Rosti et al., 2007; Matusov, Ueffing, and Ney, 2006) as an effective method for co-training in this chapter, unlike consensus for system combination, the source languages for each of our MT systems are different. This rules out

[1] Choice of Germanic and Romance for the experimental setting is inspired by results in (Cohn and Lapata, 2007)

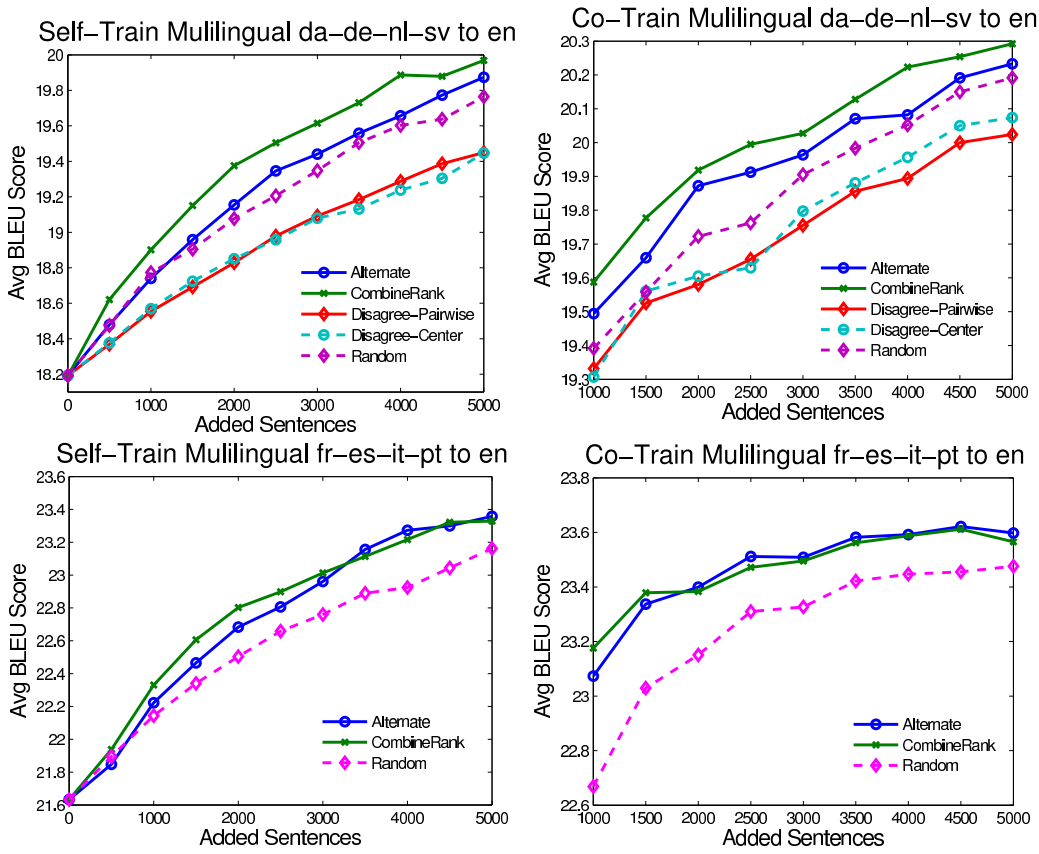


Figure 6.2: The left/right plot show the performance of our AL methods for multilingual setting combined with self-training/co-training. The sentence selection methods from Sec. 6.3 are compared with random sentence selection baseline. The top plots correspond to Danish-German-Dutch-Swedish to English, and the bottom plots correspond to French-Spanish-Italian-Portuguese to English.

a set of popular methods for obtaining consensus translations which assume translation for a single language pair. Finally, we briefly note that triangulation (see (Cohn and Lapata, 2007)) is orthogonal to the use of co-training in this chapter, since it only enhances each MT system in our ensemble by exploiting the multilingual data.

6.6 Summary

This chapter introduced the novel active learning task of adding a new language to an existing multilingual set of parallel text. We construct SMT systems from each language

Germanic languages to English

mode	self-train		co-train	
Method	wer	per	wer	per
Combined Rank	***40.2***	***30.0***	**40.0**	***29.6***
Alternate	41.0	*30.2*	40.1	*30.1*
Disagree-Pairwise	41.9	32.0	40.5	30.9
Disagree-Center	41.8	31.8	40.6	30.7
Random Baseline	41.6	31.0	40.5	30.7

Romance languages to English

mode	self-train		co-train	
Method	wer	per	wer	per
Combined Rank	***37.7***	***27.3***	***37.3***	***27.0***
Alternate	***37.7***	***27.3***	***37.3***	***27.0***
Random Baseline	38.6	28.1	38.1	27.6

Table 6.1: Comparison of multilingual selection methods with WER (word error rate), PER (position independent WER). 95% confidence interval for WER numbers is 0.7 and for PER numbers is 0.5. **Bold:** best result, *italic:* significantly better.

in the collection into the new target language. I showed that we can take advantage of multilingual corpora to decrease annotation effort thanks to the highly effective sentence selection methods devised for AL in the single language-pair setting in Chapter 5. In simulated experiments we show that using AL to add a new language to the EuroParl corpus provides a significant improvement compared to the random sentence selection baseline. Furthermore, a novel co-training method for active learning in SMT is proposed using consensus translations which outperforms AL-SMT with self-training.

Chapter 7

Conclusions

To conclude the thesis, first I describe the contributions, and then mention avenues for future research in Section 7.1. To summarize, in this thesis I have presented:

- A formal analysis of the Yarowsky algorithm for semi-supervised learning using rule-based models for the multiclass classification problem. The algorithm was not mathematically well understood until (Abney, 2004) which analyzed some specific variants of the algorithm, and also proposed some new algorithms for bootstrapping. I extended Abney's work and presented the analysis for some of his proposed algorithms as well as new class of objective functions for rule-based semi-supervised learning.

- Bootstrapping (self-training) algorithms for the effective use of monolingual data from the source language in order to improve translation quality. I proposed transductive and semi-supervised methods for the effective use of monolingual data in the source language to improve translation quality for phrase-based SMT models. I presented detailed experimental evaluations on the French-English and Chinese–English tasks, and show that the proposed algorithms improve the translation quality of the best known machine translation systems on large scale translation tasks in the semi-supervised/transductive learning scenarios as well as domain adaptation scenario.

- Active learning algorithms for SMT for the bilingual case. I provided the first serious study of active learning for SMT, and new effective sentence selection methods that improve AL for phrase-based SMT in the single language pair setting. I used active learning to improve the quality of a phrase-based SMT system when translating from a

source language to a target language, given a large set of monolingual source sentences and a small/moderate set of bilingual sentences. I showed significant improvements in translation compared to a random sentence selection baseline, when test and training data are taken from the same or different domains (domain adaptation scenario).

- The active learning task of adding a new language to an existing multilingual set of parallel text and constructing high quality MT systems, from each language in the collection into this new target language I showed that adding a new language using active learning to the EuroParl corpus provides a significant improvement compared to a random sentence selection baseline. I also provided new highly effective sentence selection methods that improve AL for phrase-based SMT in the multilingual setting.

7.1 Future Directions

From the theoretical point of view, the performance guarantees of the self-training algorithms I proposed in Chapter 3 are open questions. It seems arguments based on the cluster assumption (Chapelle, Scholkopf, and Zien, 2006) and PAC-Bayes theorem (McAllester, 1999) – where the distribution over hypotheses comes from hypotheses' margin distribution on unlabeled data – are applicable here. In addition to help further understanding of the self-training algorithms, these generalization bounds may help to set the free parameters of the learning algorithms in a principled manner.

I foresee several directions for research on dealing with shortage of bilingual data for SMT. Firstly, some of the successful active learning strategies which I proposed in this thesis are designed particularly for phrased based SMT models by incorporating different statistics about the phrases. Recently there has been much progress and interest in syntax oriented SMT, most notably (Chiang, 2007) and (Shen, Xu, and Weischedel, 2008). Since the unit of translation changes from simple phrases to super phrases or tree fragments in these new models, novel techniques are required for effective active learning. Secondly we can set up the active learning scenario for learning better alignment models, and hope for the improvement in translation quality upon having better alignments (Ganchev, Graa, and Taskar, 2008). Other than (Fraser and Marcu, 2006) which trains an alignment model in a semi-supervised manner, most of the alignment models are trained in unsupervised manner since simply there is not labeled data available for this task (sentence pairs annotated with alignments). Finally, we can use a third language, parallel to the source and target

languages, to get more information and extract new phrase pairs. This idea which is known as triangulation (Cohn and Lapata, 2007) can be employed to narrow down more the space of unknown phrases, which consequently affects sentence selection strategies in the AL setting. Overall, although we saw several successful methods for dealing with shortage of bilingual training data for SMT in this thesis, I believe these attempts are first steps toward more effective strategies which are waiting to be discovered.

It is worth noticing that the Bootstrapping framework, in which we developed our strategies to attack the shortage of bilingual training data in SMT, is very general. For example, this framework has already been used in Statistical Parsing (Reichart and Rappoport, 2007) and Word Sense Disambiguation (Yarowsky, 1995) in NLP, and Object Detection in Computer Vision (Rosenberg, 2004; Rosenberg, Hebert, and Schneiderman, 2005). Bootstrapping framework elegantly encompasses Semi-supervised/Transductive/Active Learning approaches to situations where the annotated training data is short, and I believe it has the potential to be applied successfully to many problems in NLP and other fields.

References

Abney, Steven. 2002. Bootstrapping. In *Annual Meeting on Association for Computational Linguistics (ACL)*.

Abney, Steven. 2004. Understanding the yarowsky algorithm. *Computational Linguistics*.

Angluin, Dana. 1988. Queries and concept learning. *Machine Learning*, 2(4).

Arndt, Cristoph. 2001. *Information Measures*. Springer.

Balcan, Maria-Florina, Alina Beygelzimer, and John Langford. 2009. Agnostic active learning. *Journal of Computer and System Sciences*, 75(1).

Balcan, Nina and Avrim Blum. 2005. A pac-style model for learning from labeled and unlabeled data. In *Computational Learning Theory (COLT)*.

Balcan, Nina, Avrim Blum, and Ke Yang. 2004. Co-training and expansion: Towards bridging theory and practice. In *Neural Information Proccessing Systems (NIPS)*.

Blatz, John, Erin Fitzgerald, George Foster, Simona Gandrabur, Cyril Goutte, Alex Kulesza, Alberto Sanchis, and Nicola Ueffing. 2003. Confidence estimation for machine translation. Final report, JHU/CLSP Summer Workshop. `http://www.clsp.jhu.edu/ws2003/groups/estimate/`.

Blum, Avrim and Shuchi Chawla. 2001. Learning from labeled and unlabeled data using graph mincuts. In *International Conference on Machine Learning (ICML)*.

Blum, Avrim and Tom Mitchell. 1998. Combining Labeled and Unlabeled Data with Co-Training. In *Computational Learning Theory (COLT)*.

Brants, Thorsten, Ashok C. Popat, Peng Xu, Franz J. Och, and Jeffrey Dean. 2007. Large language models in machine translation. In *Empirical Methods on Natural Language Processing (EMNLP)*.

Brown, Peter F., Vincent J.Della Pietra, Stephen A. Della Pietra, and Robert. L. Mercer. 1993. The mathematics of statistical machine translation: parameter estimation. *Computational Linguistics*, 19(2).

Callison-Burch, Chris. 2002. Co-training for statistical machine translation. Master's thesis, School of Informatics, University of Edinburgh.

Callison-Burch, Chris. 2003. Active learning for statistical machine translation. In *PhD Proposal, The University of Edinburgh*.

Callison-Burch, Chris and Miles Osborne. 2003a. Bootstrapping parallel corpora. In *NAACL workshop: Building and Using Parallel Texts: Data Driven Machine Translation and Beyond*.

Callison-Burch, Chris and Miles Osborne. 2003b. Co-training for statistical machine translation. In *Proceedings of the 6th Annual CLUK Research Colloquium*.

Callison-Burch, Chris, David Talbot, and Miles Osborne. 2004. Statistical machine translation with word- and sentence-aligned parallel corpora. In *Annual Meeting of the Association for Computational Linguistics (ACL)*.

Chapelle, Olivier, Bernhard Scholkopf, and Alexander Zien, editors. 2006. *Semi-Supervised Learning*. MIT Press.

Chiang, David. 2007. Hierarchical phrase-based translation. *Computational Linguistics*, 33(2).

Chiang, David, Kevin Knight, and Wei Wang. 2009. 11,001 new features for statistical machine translation. In *North American Chapter of the Association for Computational Linguistics - Human Language Technologies (NAACL-HLT)*.

Chiang, David, Yuval Marton, and Philip Resnik. 2008. Online large-margin training of syntactic and structural translation features. In *Empirical Methods on Natural Language Processing (EMNLP)*.

Cohn, David. 1995. Minimizing statistical bias with queries. Technical Report AI Lab memo AUM-1552, MIT.

Cohn, David, Les Atlas, and Richard Ladner. 1994. Improving generalization with active learning. In *Machine Learning Journal*.

Cohn, David, Zoubin Ghahramani, and Michael Jordan. 1996. Active learning with statistical models. *Journal of Artificial Intelligence Research (JAIR)*, 4.

Cohn, Trevor and Mirella Lapata. 2007. Machine translation by triangulation: Making effective use of multi-parallel corpora. In *ACL*.

Collins, Michael and Yoram Singer. 1999. Unsupervised models for named entity classification. In *Empirical Methods on Natural Language Processing (EMNLP)*.

Corduneanu, Adrian. 2006. *The Information Regularization Framework for Semi-Supervised Learning*. Ph.D. thesis, MIT.

Corduneanu, Adrian and Tommi Jaakkola. 2001. Stable mixing of complete and incomplete informaation. Technical report, CSAIL, MIT.

Corduneanu, Adrian and Tommi Jaakkola. 2002. Continuation methods for mixing hetrogenuous sources. In *Uncertainty in Artificial Intelligence (UAI)*.

Corduneanu, Adrian and Tommi Jaakkola. 2003. On information regulrization. In *Uncertainty in Artificial Intelligence (UAI)*.

Corduneanu, Adrian and Tommi Jaakkola. 2004. Distributed information regularization on graphs. In *Neural Information Proccessing Systems (NIPS)*.

Crammer, Koby, Ofer Dekel, Joseph Keshet, Shai Shalev-Shwartz, and Yoram Singer. 2006. Online passive-aggressive algorithms. *Journal of Machine Learning Research*, 7.

Crammer, Koby and Yoram Singer. 2003. Ultraconservative online algorithms for multiclass problems. *Journal of Machine Learning Research*, 3.

Culotta, Aron and Andrew McCallum. 2005. Reducing labeling effort for structured prediction tasks. In *American Association for Artificial Intelligence (AAAI)*.

Dasgupta, Sanjoy. 2004. Analysis of a greedy active learning strategy. In *Advances in Neural Information Processing Systems (NIPS)*.

Dasgupta, Sanjoy and Daniel Hsu. 2008. Hierarchical sampling for active learning. In *International Conference on Machine Learning (ICML)*.

Dasgupta, Sanjoy, Michael L. Littman, and David A. McAllester. 2001. Pac generalization bounds for co-training. In *Neural Information Proccessing Systems (NIPS)*.

Eck, Matthias, Stephan Vogel, and Alex Waibel. 2005. Low cost portability for statistical machine translation based in n-gram frequency and tf-idf. In *International Workshop on Spoken Language Translation (IWSLT)*.

Engelson, Sean and Ido Dagan. 1996. Minimizing manual annotation cost in supervised training from corpora. In *Annual meeting on Association for Computational Linguistics (ACL)*.

Foster, George, Roland Kuhn, and Howard Johnson. 2006. Phrasetable smoothing for statistical machine translation. In *Empirical Methods on Natural Language Processing (EMNLP)*.

Fraser, Alez and Daniel Marcu. 2006. Semi-supervised training for statistical word alignment. In *Annual Meeting of the Association for Computational Linguistics (ACL)*.

Freund, Yoav, H. Sebastian Seung, Eli Shamir, and Naftali Tishby. 1997. Selective sampling using the query by committee algorithm. *Machine Learning*, 28(2-3).

Ganchev, Kuzman, Joo Graa, and Ben Taskar. 2008. Better alignments = better translations? In *Association for Computational Linguistics (ACL)*.

Goutte, Cyril, Nicola Cancedda, Marc Dymetman, and George Foster. 2006. Machine learning for multilingual information access. In *NIPS Workshop*.

Grandvalet, Yves and Yoshua Bengio. 2004. Semi-supervised learning by entropy minimization. In *Proceedings of Neural Information Proccessing Systems (NIPS)*.

Ha, Le Quan, E. I. Sicilia-Garcia, Ji Ming, and F.J. Smith. 2002. Extension of zipf's law to words and phrases. In *International Conference on Computational linguistics (COLING)*.

Haffari, Gholamreza, Maxim Roy, and Anoop Sarkar. 2009. Active learning for statistical phrase-based machine translation. In *North American Chapter of the Association for Computational Linguistics - Human Language Technologies (NAACL-HLT)*.

Haffari, Gholamreza and Anoop Sarkar. 2007. Analysis of semi-supervised learning with the yarowsky algorithm. In *Uncertainty in Artificial Intelligence (UAI)*.

Haffari, Gholamreza and Anoop Sarkar. 2009. Active learning for multilingual statistical machine translation. In *Annual Meeting of Association for Computational Linguistics (ACL)*.

He, Xiaodong, Mei Yang, Jianfeng Gao, Patrick Nguyen, and Robert Moore. 2008. Indirect-hmm-based hypothesis alignment for combining outputs from machine translation systems. In *Empirical Methods on Natural Language Processing (EMNLP)*.

Hinton, Geoffrey. 1999. Products of experts. In *International Conference on Artificial Neural Networks*.

Hutchins, W.John and Harold L. Somers. 1992. *An Introduction to Machine Translation*. Academic Press.

Hwa, Rebecca. 2004. Sample selection for statistical parsing. *Computational Linguistics*, 30.

Kauchak, David. 2006. Contribution to research on machine translation. In *PhD Thesis, University of California at San Diego (UCSD)*.

Kearns, Michael and Umesh Vazirani. 1994. *An Introduction to Computational Learning Theory*. MIT Press.

Kim, Seokhwan, Yu Song, Kyungduk Kim, Jeongwon Cha, and Gary Geunbae Lee. 2006. Mmr-based active machine learning for bio named entity recognition. In *North American Chapter of the Association for Computational Linguistics - Human Language Technologies (NAACL-HLT)*.

Knight, Kevin. 1999. Decoding complexity in word-replacement translation models. *Computational Linguistics*, 25.

Koehn, Philipp. 2003. *Noun Phrase Translation*. Ph.D. thesis, University of Southern California.

Koehn, Philipp. 2005. Europarl: A parallel corpus for statistical machine translation. In *MT Summit*.

Koehn, Philipp, Franz Och, and Daniel Marcu. 2003. Statistical phrase-based translation. In *North American Chapter of the Association for Computational Linguistics (NAACL)*.

Lafferty, John, Andrew McCallum, and Fernando Pereira. 2001. Conditional random fields: Probabilistic models for segmenting and labeling sequence data.

Lafferty, John, Stephen Della Pietra, and Vincent Della Pietra. 1997. Statistical learning algorithms based on bregman distances. In *Canadian Workshop on Information Theory*.

Leskes, Boaz. 2005. The value of agreement: A new boosting algorithm. In *Computational Learning Theory (COLT)*.

Lopez, Adam. 2008. Statistical machine translation. *ACM Computing Surveys*, 40(3).

Matusov, Evgeny, Nicola Ueffing, and Hermann Ney. 2006. Computing consensus translation from multiple machine translation systems using enhanced hypotheses alignment. In *European Chapter of the Association for Computational Linguistics (EACL)*.

McAllester, David. 1999. Some pac-bayesian theorems. *Machine Learning*, 37(3).

Merialdo, Bernard. 1993. Tagging english text with a probabilistic model. *Computational Linguistics*, 20.

Mitchell, Tom. 1982. Generalization as search. *Artificial Intelligence*, 18(2).

Mohit, Behrang and Rebecca Hwa. 2007. Localization of difficult-to-translate phrases. In *proceedings of the 2nd ACL Workshop on Statistical Machine Translations*.

Muslea, Ion, Steven Minton, and Craig Knoblock. 2006. Active learning with multiple views. *Journal of Artificial Intelligence Research (JAIR)*, 27.

Nießen, Sonja, Franz Och, Gregor Leusch, and Hermann Ney. 2000. An evaluation tool for machine translation: Fast evaluation for MT research. In *Proceedings of LREC-2000*, Athens, Greece.

Nigam, Kamal and Rayid Ghani. 2000. Analyzing the effectiveness and applicability of co-training. In *Conference on Information and Knowledge Management (CIKM)*.

Nigam, Kamal, Andrew McCallum, Sebastian Thrun, and Tom Mitchell. 2000. Text classification from labeled and unlabeled documents using em. *Machine Learning*, 39(2-3).

Och, Franz. 2003a. Minimum error rate training in statistical machine translation. In *Annual Meeting on Association for Computational Linguistics (ACL)*.

Och, Franz. 2003b. Minimum error rate training in statistical machine translation. In *Annual Meeting of the Association for Computational Linguistics (ACL)*.

Och, Franz and Hermann Ney. 2002. Discriminative training and maximum entropy models for statistical machine translation. In *Annual Meeting on Association for Computational Linguistics (ACL)*.

Och, Franz and Hermann Ney. 2003. A systematic comparison of various statistical alignment models. *Computational Linguistics*, 29(1).

Papineni, Kishore, Salim Roukos, Todd Ward, and Wei jing Zhu. 2002. BLEU: a method for automatic evaluation of machine translation. In *Annual Meeting of the Association for Computational Linguistics (ACL)*.

Press, William, Saul Teukolsky, William Vetterling, and Brian Flannery. 2002. *Numerical Recipes in C++*. Cambridge University Press, Cambridge, UK.

Reichart, Roi and Ari Rappoport. 2007. Self-training for enhancement and domain adaptation of statistical parsers trained on small datasets. In *Association for Computational Linguistics (ACL)*.

Reichart, Roi, Katrin Tomanek, Udo Hahn, and Ari Rappoport. 2008. Multi-task active learning for linguistic annotations. In *Annual Meeting of the Association for Computational Linguistics (ACL)*.

Ronald, Kato and Etienne Barnard. 2006. Statistical translation with scarce resources: a south african case study. In *17th Annual Symposium of the Pattern Recognition Association of South Africa*.

Ronald, Kato and Etienne Barnard. 2007. Statistical translation with scarce resources: a south african case study. *SAIEE Africa Research Journal*, 98(4).

Rosenberg, Charles. 2004. *Semi-Supervised Training of Models for Appearance-Based Statistical Object Detection Methods*. Ph.D. thesis, CMU.

Rosenberg, Charles, Martial Hebert, and Henry Schneiderman. 2005. Semi-supervised self-training of object detection models. In *Seventh IEEE Workshop on Applications of Computer Vision*.

Rosti, Antti-Veikko, Necip Fazil Ayan, Bing Xiang, Spyros Matsoukas, Richard M. Schwartz, and Bonnie Jean Dorr. 2007. Combining outputs from multiple machine translation systems. In *North American Chapter of the Association for Computational Linguistics (NAACL)*.

Roy, Maxim. 2009. A semi-supervised approach to bengali-english phrase-based statistical machine translation. In *Canadian AI Conference*.

Russell, Stuart and Peter Norvig. 1995. *Artificial Intelligence: A Modern Approach*. Prentice-Hall.

Santner, Thomas, Brian Williams, and William Notz. 2003. *The Design and Analysis of Computer Experiments*. Springer Verlag.

Schütze, Hinrich, Emre Velipasaoglu, and Jan Pedersen. 2006. Performance thresholding in practical text classification. In *Proceedings of the 15th ACM international conference on Information and knowledge management (CIKM)*.

Seeger, Matthias. 2000. Learning with labeled and unlabeled data. Technical report, The University of Edinburgh.

Settles, Burr. 2009. Active learning literature survey. Computer Sciences Technical Report 1648, University of Wisconsin–Madison.

Settles, Burr and Mark Craven. 2008. An analysis of active learning strategies for sequence labeling tasks. In *Empirical Methods on Natural Language Processing (EMNLP)*.

Seung, H. Sebastian, Manfred Opper, and Haim Sompolinsky. 1992. Query by committee. In *Computational Learning Theory (COLT)*.

Shen, Libin, Jinxi Xu, and Ralph Weischedel. 2008. A new string-to-dependency machine translation algorithm with a target dependency language model. In *Association for Computational Linguistics (ACL)*.

Stolcke, Andreas. 2002. SRILM - an extensible language modeling toolkit. In *International Conference on Spoken Language Processing (ICSLP)*.

Tillmann, Christoph and Tong Zhang. 2006. A discriminative global training algorithm for statistical mt. In *Annual Meeting of the Association for Computational Linguistics (ACL)*.

Tong, Simon and Daphne Koller. 2000. Support vector machine active learning with applications to text classification. In *International Conference on Machine Learning (ICML)*.

Turchi, Marco, Tijl De Bie, and Nello Cristianini. 2008. Learning performance of a machine translation system: a statistical and computational analysis. In *proceedings of the Third Workshop on Statistical Machine Translation*. Association for Computational Linguistics (ACL).

Ueffing, Nicola. 2006. Using monolingual source-language data to improve MT performance. In *International Workshop on Spoken Language Translation (IWSLT)*.

Ueffing, Nicola, Gholamreza Haffari, and Anoop Sarkar. 2007a. Transductive learning for statistical machine translation. In *proceedings of Annual Meeting of the Association for Computational Linguistics (ACL)*.

Ueffing, Nicola, Golamreza Haffari, and Anoop Sarkar. 2007b. Semi-supervised model adaptation for statistical machine translation. *Machine Translation*, 21(2).

Ueffing, Nicola, Golamreza Haffari, and Anoop Sarkar. 2008. Semi supervised learning for statistical machine translation. In *In: C. Goutte, N. Cancedda, M. Dymetman and G. Foster, Editors, Learning Machine Translation*. MIT Press.

Ueffing, Nicola and Hermann Ney. 2007. Word-level confidence estimation for machine translation. *Computational Linguistics*, 33(1).

Ueffing, Nicola, Michel Simard, S. Larkin, and J. H. Johnson. 2007. NRC's Portage system for WMT 2007. In *Proceedings of ACL Workshop on SMT*.

Wang, Wei and Zhi-Hua Zhou. 2008. On multi-view active learning and the combination with semi-supervised learning. In *International Conference on Machine Learning (ICML)*.

Watanabe, Taro, Jun Suzuki, Hajime Tsukuda, , and Hideki Isozaki. 2007. Online large-margin training for statistical machine translation. In *Empirical Methods on Natural Language Processing (EMNLP)*.

Yarowsky, David. 1995. Unsupervised word sense disambiguation rivaling supervised methods. In *Association for Computational Linguistics (ACL)*.

Zhu, Xiaojin, Zoubin Ghahramani, and John Lafferty. 2003. Semi-supervised learning using gaussian fields and harmonic functions. In *International Conference on Machine Learning (ICML)*.

Zhu, Xiaojin and Andrew B. Goldberg. 2009. *Introduction to Semi-Supervised Learning*. Morgan and Claypool Publishers.

Zipf, George. 1932. *Selective Studies and the Principle of Relative Frequency in Language*. Harvard University Press.