MODEL ADAPTATION FOR STATISTICAL MACHINE TRANSLATION

by

Ajeet Grewal

B.Tech., Indian Institute of Technology Bombay, 2003

A THESIS SUBMITTED IN PARTIAL FULFILLMENT
OF THE REQUIREMENTS FOR THE DEGREE OF
MASTER OF SCIENCE
in the School
of
Computing Science

SIMON FRASER UNIVERSITY
Fall 2009

APPROVAL

Name: Ajeet Grewal

Degree: Master of Science

Title of thesis: Model Adaptation for Statistical Machine Translation

Examining Committee:

Dr. Martin Ester
Professor, School of Computing Science
Simon Fraser University
Chair

Dr. Anoop Sarkar
Associate Professor, School of Computing Science
Simon Fraser University
Senior Supervisor

Dr. Fred Popowich
Professor, School of Computing Science
Simon Fraser University
Supervisor

Dr. Greg Mori
Assistant Professor, School of Computing Science
Simon Fraser University
Examiner

Date Approved: _____

Abstract

Statistical machine translation (SMT) systems use statistical learning methods to learn how to translate from large amounts of parallel training data. Unfortunately, SMT systems are tuned to the domain of the training data and need to be adapted before they can be used to translate data in a different domain.

First, we consider a semi-supervised technique to perform model adaptation. We explore new feature extraction techniques, feature combinations and their effects on performance. In addition, we introduce an unsupervised variant of Minimum Error Rate Training (MERT), which can be used to tune the SMT model parameters. We do this by using another SMT model that translates in the reverse direction. We apply this variant of MERT to the model adaptation task. Both of the techniques we explore in this thesis produce promising results in exhaustive experiments we performed for translation from French to English in different domains.

Keywords: natural language processing; statistical machine translation; back translations; domain adaptation; model adaptation; transductive model adaptation; unsupervised MERT

To the love of my life

"This one's tricky. You have to use imaginary numbers, like eleventeen . . . "

— *Calvin,* Calvin & Hobbes

Acknowledgments

I owe my deepest gratitude to my senior supervisor, Dr. Anoop Sarkar. This work would have been impossible without his unparalleled expertise, keen insight and endless patience.

I would like to thank my supervisor, Dr. Fred Popowich for introducing me to the field of Natural Language Processing and for his thorough assessment of my thesis. I would also like to express my gratitude to my thesis examiner, Dr. Greg Mori for his insightful opinions regarding the thesis.

During the course of my Master's degree, I had the opportunity to work with many excellent faculty members at Simon Fraser University. I would especially like to thank Dr. Anoop Sarkar, Dr. Greg Mori, Dr. Alexandra Fedorova and Dr. James Delgrande whose excitement and enthusiasm in their respective fields was contagious.

I am also indebted to my friends and colleagues in the Natural Language Laboratory, especially Baskaran, Reza, Milan, Yudong, Ann, Maxim and Winona for their help and support throughout my stay.

Lastly, but most importantly, I would like to express my gratitude to my wife whose love and support carried me through this degree. I thank my parents and grandparents for their constant encouragement, their prayers and their blessings. I am indebted to my brother for providing welcome conversation on some dreary days. Finally, I would like to thank my in-laws for being there for us always.

Contents

List of Tables

List of Figures

Chapter 1

Phrase Based Statistical Machine Translation

1.1 Introduction

In computer science, statistical machine translation (SMT) refers to the task of automating the translation of text from one natural language to another, using statistical techniques.

Statistical Machine Translation (SMT) was first extensively studied in (Brown et al., 1990) and (Brown et al., 1993). They introduced the so called IBM models, which are based on the *source-channel model*. These models were word-based, in that sentences were translated one word at a time.

Improvements to these models led to the development of phrase-based SMT systems (Koehn, Och, and Marcu, 2003), which are described in detail in Section 1.2. The state-of-the-art SMT systems today are phrase-based.

1.1.1 Overview

- Chapter 1 provides a theoretical background of phrase based SMT systems, techniques used to train these systems and automated evaluation metrics.
- Chapter 2 introduces the problem of Model Adaptation and summarizes some of the existing literature in the field.

- Chapter 3 describes the transductive model adaptation algorithm and details experiments with different feature extraction methods.
- Chapter 4 introduces a novel technique for performing unsupervised minimum error rate training (MERT). This technique is then applied to the problem of model adaptation.
- Chapter 5 ends with conclusions and possible directions for future work.

The different software tools used in this work are discussed in Appendix B.

1.2 The Theory of Statistical Machine Translation

1.2.1 The Source-channel Model

The source channel model is also sometimes referred to as the noisy channel model. The reasoning is as follows. We denote the source language by S and the target language by T. Let $s_1^J = s_1, \ldots, s_j, \ldots, s_J$ be a sentence in the source language and $t_1^I = t1, \ldots, t_i, \ldots, t_I$ be a sentence in the target language. We assume that the person uttering s_1^J, had t_1^I in mind. However, this "signal" was corrupted by some noise and we got s_1^J as a result. Now, we would like to recover t_1^I as was originally intended. To do this we need to calculate the probability of uttering a sentence t_1^I and the probability of it getting converted into s_1^J.

The source-channel model can also be motivated using Bayes' rule. We would like to find the sentence t_1^I that maximizes $Pr(t_1^I|s_1^J)$. Using Bayes' rule and noting that $Pr(s_1^J)$ does not depend on t_1^I, the most likely translation $\hat{t}_1^I$ is given by,

$$\begin{aligned} \hat{t}_1^I &= \underset{t_1^I}{\operatorname{argmax}}\, Pr(t_1^I|s_1^J) && (1.1)\\ &= \underset{t_1^I}{\operatorname{argmax}}\, \frac{Pr(t_1^I)Pr(s_1^J|t_1^I)}{Pr(s_1^J)} && (1.2)\\ &= \underset{t_1^I}{\operatorname{argmax}}\, Pr(t_1^I)Pr(s_1^J|t_1^I) && (1.3) \end{aligned}$$

$Pr(t_1^I)$ represents the probability of a sequence of words t_1^I occurring in the language T. This probability is calculated using a "language model". The remaining factor $P(s_1^J|t_1^I)$ represents the probability of s_1^J being a translation of t_1^I. Note that due to Bayes' rule the

direction of translation has been reversed (we are modeling the prior probabilities). We obtain this component from the "translation model".

There are some disadvantages to the source-channel model.

1. We get the optimal solution in Equation 1.3 only if we use the true probability distributions for $Pr(t_1^I)$ and $Pr(s_1^J|t_1^I)$. However, we can only get approximations for these distributions from data. It is possible that other models might give better results.

2. As an example of the earlier point, in (Och et al., 1999) it was shown that using $Pr(t_1^I|s_1^J)$ instead of $Pr(s_1^J|t_1^I)$ gave comparable results. This approach, though lucrative, is hard to motivate theoretically in the source-channel framework.

3. It is quite difficult to extend this model by introducing additional dependencies.

1.2.2 The Log-Linear Model

The maximum entropy approach introduced by (Berger, Pietra, and Pietra, 1996) can be used as an alternative to the source-channel approach. This was described in (Papineni, Roukos, and Ward, 1997) and (Och and Ney, 2002). Here, we directly model the posterior probability $Pr(t_1^I|s_1^J)$ as follows.

$$Pr(t_1^I|s_1^J) = p_{\lambda_1^M}(t_1^I|s_1^J) \tag{1.4}$$

$$= \frac{\exp\left[\sum_{m=1}^{M} \lambda_m h_m(t_1^I, s_1^J)\right]}{\sum_{t'^I_1} \exp\left[\sum_{m=1}^{M} \lambda_m h_m(t'^I_1, s_1^J)\right]} \tag{1.5}$$

Here $h_m(t_1^I, s_1^J), m = 1, \cdots, M$ are feature functions and λ_m are the corresponding weights. We can now find the most likely translation solving the following decision rule.

$$\hat{t}_1^I = \underset{t_1^I}{\operatorname{argmax}} \left[\sum_{m=1}^{M} \lambda_m h_m(t_1^I, s_1^J)\right] \tag{1.6}$$

We can see that Equation 1.6 is linear in terms of the feature functions. Since we are dealing with an exponent, any probabilities that we add as features are added in log-space. Hence we term this as a *log-linear* model. In particular, we note that Equations 1.6 and 1.3 yield identical solutions when,

$$M = 2 \tag{1.7}$$
$$h_1(t_1^I, s_1^J) = \log Pr(t_1^I) \tag{1.8}$$
$$h_2(t_1^I, s_1^J) = \log Pr(s_1^J | t_1^I) \tag{1.9}$$
$$\lambda_1 = 1 \tag{1.10}$$
$$\lambda_2 = 1 \tag{1.11}$$

Thus the *source-channel model can be expressed as a special case of the log-linear model.* However, we are not restricted to use only these features in the log-linear model. In fact, it is trivial to incorporate additional models into this log-linear model by introducing new feature functions $h_m(t_1^I, s_1^J)$. This property makes log-linear models a perfect fit for the task of model adaptation and hence, we use them exclusively in this work.

The feature functions used in this work are now described.

Language Model

In SMT, we would like to produce sentences in the target language T that "make sense". This notion is captured in a language model. A language model determines how likely a sentence is, in a given language i.e. it determines $Pr(t_1^I)$.

The feature is added into the log-linear model as

$$h_{lm}(t_1^I, s_1^J) = \log Pr(t_1^I) \tag{1.12}$$

It is easy to come up with sentences that have never been used before. So how can one tell, how probable a given sentence is? The idea is that instead of worrying about the probabilities of entire sentences, we worry ourselves with only parts of sentences.

A common solution to this problem is using *n-gram* probabilities. An *n-gram* is a sequence of n consecutive words in any sentence. If a sentence is made up of highly probable *n-grams* then the sentence is highly probably and vice versa. Using a large enough corpus, we can learn the distribution of these *n-grams*. If $n = 1$, we call the model a unigram model. Similarly for $n = 2, 3$ it is called a bigram and a trigram model respectively.

Mathematically, for a trigram model we determine the probability of a sequence of words t_1^I as follows.

$$Pr(t_1^I) = \prod_{i=1}^{I+1} Pr(t_i|t_{i-1}, t_{i-2}) \tag{1.13}$$

Here, $t_0 = t_{-1} = t_{I+1} = \$$, are special sentence boundary markers. t_{i-1} and t_{i-2} denote the *history* for t_i.

We describe how to train this language model in Section 1.3.2.

Phrase based Translation Model

A *phrase* is defined as a continuous sequence of words in a sentence. In this way, it is similar to the notion of an n-gram, except that we do not classify phrases based on their length. Importantly, a phrase can have a length of one.

The phrase translation model is another essential component that defines a distribution over phrases that are translations of each other. Let $\bar{s}$ denote a phrase in the source language and $\bar{t}$ denote a phrase in the target language. In a phrase based translation system, we chunk the source sentence s_1^J into phrases $\bar{s}_1^P = \bar{s}_1, \ldots, \bar{s}_p, \ldots, \bar{s}_P$ and then translate each of these phrases (possibly out of order as described by the distortion model in section 1.2.2). We then get P target phrases $\bar{t}_1^P = \bar{t}_1, \ldots, \bar{t}_p, \ldots, \bar{t}_P$ such that $\langle \bar{s}_p, \bar{t}_p \rangle$ are translations of each other. Now we compute the translation probability of the entire sentence as,

$$Pr(t_1^I|s_1^J) = \prod_{p=1}^{P} Pr(\bar{t}_p|\bar{s}_p) \tag{1.14}$$

In addition to using $Pr(s_1^J|t_1^I)$ like in the source-channel model, we compute additional features and add them into the log-linear model. A phrase based translation model consists of the following four log-linear features.

1. Phrase Translation probabilities:

   We add the phrase translation probabilities for both the directions to the log-linear model. The features added are shown below.

$$h_{tr_{S\to T}}(t_1^I, s_1^J) = \log Pr(t_1^I|s_1^J) = \log \prod_{p=1}^{P} Pr(\bar{t}_p|\bar{s}_p) \tag{1.15}$$

$$h_{tr_{T\to S}}(t_1^I, s_1^J) = \log Pr(s_1^J|t_1^I) = \log \prod_{p=1}^{P} Pr(\bar{s}_p|\bar{t}_p) \tag{1.16}$$

We describe how to obtain $Pr(\bar{t}_p|\bar{s}_p)$ and $Pr(\bar{s}_p|\bar{t}_p)$ in Section 1.3.3.

2. Lexical Weighting:

   It was shown in (Koehn, Och, and Marcu, 2003) that adding a feature that indicates how well individual words translate to each other, improves the performance of the log-linear model. This new feature is called *lexical weighting*. The lexical weight p_w for the entire sentence pair is computed from the product of the individual phrases, as follows

$$p_w(t_1^I|s_1^J) = \prod_{p=1}^{P} p_w(\bar{t}_p|\bar{s}_p) \tag{1.17}$$

   The features added to the log-linear model are shown below

$$h_{lw_{S\to T}} = \log p_w(t_1^I|s_1^J) = \log \prod_{p=1}^{P} p_w(\bar{t}_p|\bar{s}_p) \tag{1.18}$$

$$h_{lw_{T\to S}} = \log p_w(s_1^J|t_1^I) = \log \prod_{p=1}^{P} p_w(\bar{s}_p|\bar{t}_p) \tag{1.19}$$

   We describe how to obtain $p_w(\bar{t}_p|\bar{s}_p)$ and $p_w(\bar{s}_p|\bar{t}_p)$ in Section 1.3.3.

Distortion Model

The distortion model captures the reordering of the phrases in the target language output. During translation, one need not proceed left-to-right and translate consecutive phrases in the source language. Instead, we can jump back and forth between phrases in the source sentence, possibly choosing to translate them out of order. The output sentence is constructed by translating source phrases in this distorted order. This is what allows for the reordering of the phrases in the output.

To model this distortion, we maintain scores for each of these "jumps". Suppose a_p denotes the start position of the source phrase that was translated into the pth target phrase. Let b_{p-1} be the end position of the source phrase that was translated into the $(p-1)$th target phrase. Then the distortion between these two phrases is $d(a_p - b_{p-1})$. This distortion can be modeled by a simple exponential model,

$$d(a_p - b_{p-1}) = \alpha^{|a_p - b_{p-1} - 1|} \tag{1.20}$$

α is a suitably chosen parameter. We subtract 1 to denote that there should be no distortion if phrases are translated consecutively. Finally, the following feature is added into the log-linear model.

$$h_d = \log \prod_{p=2}^{P} d(a_p - b_{p-1}) \tag{1.21}$$

Word Penalty

The word penalty feature is used to calibrate the output length. Sometimes, the output sentence is disproportionately larger than the input sentence. Adding this feature helps alleviate this problem.

The length of the output sentence t_1^I is simply added as a feature into the log-linear model. If there are I words in the target sentence, the feature added is

$$h_{wp} = I \tag{1.22}$$

Phrase Penalty

The phrase penalty feature is used to calibrate whether the log-linear model should prefer a small number of large phrases, or a large number of smaller phrases. To do this, we simply add the count of phrase pairs $\langle \bar{s}_p, \bar{t}_p \rangle$ used in the translation as a feature into the log-linear model. If a total of P phrase pairs were used, the feature added to the log-linear model is

$$h_{pp} = P \tag{1.23}$$

1.3 Training

In this section, we describe how to train the various components used in the log-linear model. To train a language model, we need monolingual data in the target language T, whereas translation models need sentence aligned bilingual data also known as *parallel corpora*.

1.3.1 Pre-processing

While training both the language model and translation model, we *pre-process* the data to make our resulting models more robust. Typically, this pre-processing involves *tokenization* and *lowercasing* the training data. This is done to lower the perplexity of the resulting models and to increase their accuracy. For example, we do not want the language model to assign different probabilities to "canadian" and "Canadian". The former would occur in the middle of sentences, while the latter could occur at the start of sentences. Similarly, we can motivate tokenization by considering the "'s" suffix, which signifies ownership e.g. "John*'s* car". Tokenization would separate this suffix into a distinct token consistently across the data and we would be able to model it better, and assign it to its correct counterpart in a different language. In addition, this reduces perplexity, by eliminating the choice between "John's" and "John" for example.

The translations obtained using these models are also lowercased and tokenized. In a complete SMT system, these translations are usually re-cased and de-tokenized to produce natural looking sentences. In this work, the focus is on improving the model adaptation. So in order simplify the pipeline and reduce unintentional sources of errors, we do not perform any post-processing. Instead we perform our evaluations on the lowercased and tokenized output.

1.3.2 Language Model

To compute the language model, we use the SRILM toolkit (Stolcke, 2002). In this section, we describe how SRILM computes the language model probabilities.

We saw how to determine the probability of an entire sentence by decomposing it into its component n-grams, as shown in Equation 1.13. In order to learn these n-gram probabilities, we gather counts from large monolingual data sources. The n-gram probabilities are then simply the relative frequency estimates. Continuing the example of a trigram model, we have

$$Pr(t_i|t_{i-1}, t_{i-2}) = \frac{count(t_i, t_{i-1}, t_{i-2})}{\sum_{t'_i} count(t'_i, t_{i-1}, t_{i-2})} \tag{1.24}$$

Unfortunately, even though the data sets used may be large, there could be some n-grams that are not seen in the training data, but might be encountered later. We can alleviate this data sparsity problem by smoothing the counts. In this thesis, we use Kneser-Ney

smoothing described in (Kneser and Ney, 1995). If an n-gram is not found or has a low count, we obtain an approximation from lower order models. For example, if we don't have sufficient data for a trigram, we remove one word from the history and consider only the bigram.

1.3.3 Translation Model

To train a translation model, we need a *parallel corpus* which is set of sentences in the source language S aligned with their corresponding reference sentences in the target language T. Using this parallel corpus, we can gather statistical information which we can use to translate new sentences.

The word alignment, phrase alignment and lexical weighting features described in this section are learned using Giza++ (Och and Ney, 2003). We describe the techniques used in Giza++ to learn these features.

Word Alignment

In phrase based translation models, we want to gather information about which phrases are translations of each other. To do this, we first obtain *word alignments* between source and target sentences. A *word alignment* is just a mapping from words in the source sentence to words in the target sentence. Words that are aligned are translations of each other.

An alignment A between a sentence pair $\langle s_1^J, t_1^I \rangle$, consists of pairs of indices (j, i) (called alignment points) such that there exists only one alignment point for every word in the target string. Each alignment point (j, i) indicates that the source word s_j is translated to the target word t_i.

Please refer to Figure 1.1, which shows an example alignment in a French-English sentence pair, inspired by (Knight, 1999a). It is possible for words in the target sentence to be unaligned to any source word. To model such *spurious* words, we align them to a *NULL* source word. Although it is not shown in the example, multiple target words could be aligned to the same source word.

There are several different methods for performing word alignment. In (Och and Ney, 2003), a detailed comparison of the different alignment methods is provided. We note that in this work, we use Giza++ that generates alignments using Model 6. We now describe the different models for learning word alignments.

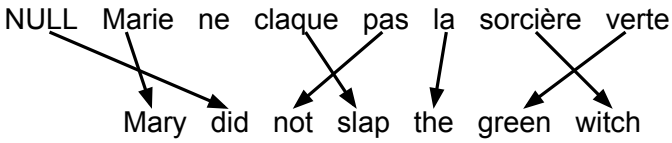

Figure 1.1: Example word alignment that could be observed between a French and an English sentence. Note that the word *did* is aligned with the *NULL* source word and is called a *spurious* word. Also note that the French word *ne* is not aligned to any English word.

- *Hidden Markov Model*:

  In the Hidden Markov Model (HMM) based alignment models, the alignment A is introduced as a hidden parameter into the translation probability $Pr(s_1^J|t_1^I)$. A first-order dependence of the alignments between consecutive words is assumed.

- *Models 1 and 2*:

  Models 1 and 2 are similar to HMM based alignment models, but instead of a first-order dependence, a zero-order dependence is assumed. In Model 1, all alignments are considered equally likely. Therefore the word order does not affect alignment in Model 1. In Model 2, we remedy this by explicitly modeling the conditional dependence of a word alignment to the word position in the source sentence. In both cases, the alignments do not depend on the previous alignment, as in the case of HMM.

- *Models 3, 4 and 5*:

  Models 3, 4 and 5 are *fertility* based alignments models and are more complicated than Models 1 and 2. The fertility ϕ_i of a target word t_i is defined as the number of source words aligned to it. Fertility based alignment models explicitly model the probability $Pr(\phi|t)$ that the target word t is aligned to ϕ words.

  In Model 3, the alignments for each target word are assumed to be independent of each other. This is remedied in Model 4, where we assume a first order dependence between alignments. Here, the alignment of each target word depends on the alignment of the previous word. Also, each word is dependent on the *word classes* of the surrounding words.

  Models 3 and 4 are *deficient* formulations as the sum of the probabilities of all valid alignments does not sum to one. They ignore the dependence on whether a source

word has already been chosen for alignment. Model 5 is a reformulation of Model 4 with additional alignment parameters that fix these issues.

- *Model 6*:

  The best alignments are obtained from models that assume a first-order dependence (HMM, Models 4 and 5). The HMM model predicts the distance between consecutive source language positions, while the Models 4 and 5 predict the distance between consecutive target language positions. This suggests that these models make use of locality in the source and target languages respectively. Model 6 combines these factors by taking a log-linear combination of the HMM model with Model 4. Model 4 is chosen instead of Model 5 as it performs better in experiments (Och and Ney, 2003).

The word alignment probabilities for a given model are computed using the EM algorithm (Baum, 1972). Models 1, 2 and HMM have a comparatively simple mathematical form and the EM algorithm can be implemented efficiently. This is not the case for fertility-based alignment models as there is no efficient way to avoid explicit summation over all possible alignments. To make things simpler for Models 3 to 6, the counts are only collected for a small number of good alignments.

To keep the exposition simple, we show the steps involved in training a Model 1 alignment using EM. To help the reader understand the process, a corresponding worked-out example for Model 1 is provided in Appendix A.

1. *Initialization*: We first assign uniform probabilities to all possible word-to-word translations, $Pr(t|s)$.

2. *Compute alignment probabilities*: These word translation probabilities $Pr(t|s)$ are used to estimate the alignment probabilities of entire sentence pairs for all possible alignments, $Pr(a|t_1^I, s_1^J)$ where a denotes an alignment between the sentences.

3. *Revise word translation probabilities*: The alignment probabilities $Pr(a|t_1^I, s_1^J)$ are used to re-estimate the word translation probabilities $Pr(t|s)$ that were initially assumed to be uniform.

4. *Repeat until convergence*: Steps 2-3 are repeated until convergence to get the final word translation probabilities $Pr(t|s)$.

Now for any sentence pair, the final word alignment is the Viterbi alignment found using the model learned by EM. Here again, there are efficient polynomial time solutions to finding Viterbi alignments for Models 1, 2 and the HMM model (Vogel, Ney, and Tillmann, 1996). However for fertility based models (Models 3 to 6), the corresponding search problem is NP-complete (Knight, 1999b). In these models, the greedy search algorithm presented in (Brown et al., 1993) is used to find the best alignment. This algorithm works by first finding the Viterbi alignment for a simpler model (Model 2 or HMM) and then iteratively improving the alignment by using the alignment probabilities from the fertility based alignment model being used.

Phrase Alignment

We learn phrase alignments using the approach proposed in (Och et al., 1999) and (Koehn, Och, and Marcu, 2003) to find phrase alignments from word alignments.

In word alignment described earlier, we cannot align a source word to two or more target words. To address this, we compute the word alignment in *both directions* (source-to-target and target-to-source), using the word alignment technique discussed earlier. We denote the set of alignments as A_1 and A_2; each element of these sets indicates an alignment between two specific words. If we take an intersection ($A_1 \cap A_2$) of the word-to-word alignments found in both directions, we have a set of alignments that have been predicted by both Viterbi alignments and are therefore highly reliable. The union ($A_1 \cup A_2$) of the two alignments contains some alignments that were learned by one of the Viterbi alignments but not the other.

We consider a matrix with the source words forming one dimension, the target words forming the other. Please refer to Figure 1.2 for a visual example. Each element in this matrix refers to an alignment between the corresponding source and target words. We define the "neighbours" of an alignment point as those points that are horizontally, vertically or diagonally adjacent to that alignment point in this matrix.

We start with the intersection of the two alignments, $A = A_1 \cap A_2$. We add those alignments from A_1 or A_2 that do not exist in A and that are "neighbours" of some alignment point in A. We do this iteratively until no more alignment points can be added. The final alignment set A lies somewhere between the two sets,

$$A_1 \cap A_2 \subseteq A \subseteq A_1 \cup A_2 \tag{1.25}$$

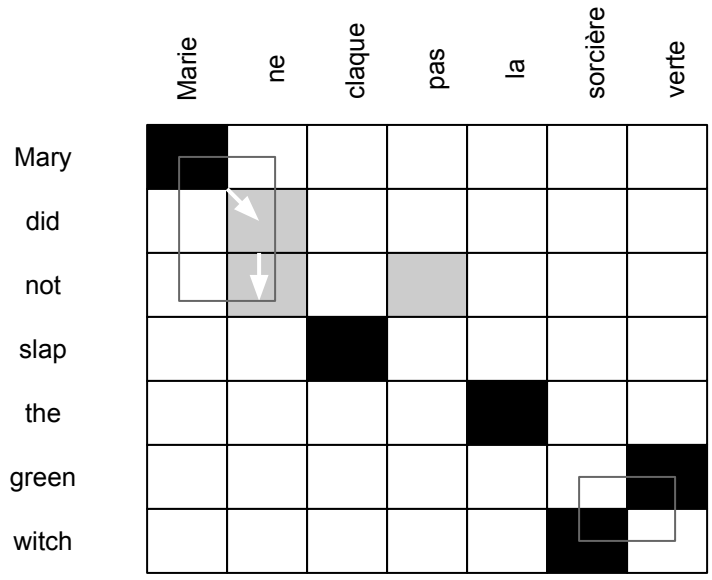

Figure 1.2: Phrase alignment for an example sentence pair. We omit the *NULL* word for simplicity. The black squares indicate alignments found in both directions ($A_1 \cap A_2$). The grey squares indicate the additional alignment points that were found in $A_1 \cup A_2$. The white arrows indicate the "neighbours" of existing alignment points that are added iteratively. Finally, we show two of the learned phrase alignments by the dark grey rectangles.

We now collect aligned phrase pairs that are consistent with this alignment. The words in a legal phrase pair are only aligned to each other and not to words outside the phrase boundary. Figure 1.2 shows some examples of such aligned phrase pairs.

From our parallel corpus, we select all legal phrase pairs from every given sentence pair. We can now estimate the phrase translation probability by their relative frequency of occurrence in the corpus. If $\bar{s}$ is a phrase in the source language and $\bar{t}$ is one in the target language, then the phrase translation probability is given by,

$$Pr(\bar{t}|\bar{s}) = \frac{count(\bar{t}, \bar{s})}{\sum_{\bar{t}'} count(\bar{t}', \bar{s})} \tag{1.26}$$

Lexical Weighting

As discussed earlier, (Koehn, Och, and Marcu, 2003) showed that adding a *lexical weighting* feature that indicates how well individual words translate to each other, improves the

performance of the log-linear model.

We first estimate a lexical translation probability distribution ($w(t|s)$) using the word alignments (found during phrase alignment), using their relative frequency as follows.

$$w(t|s) = \frac{count(t,s)}{\sum_{t'} count(t',s)} \tag{1.27}$$

Now, for a phrase pair $\langle \bar{t}, \bar{s} \rangle$, and a corresponding word alignment a between the target word positions $i = 1, \ldots, n$ and source word positions $j = 0, 1, \ldots, m$, (0 indicates the *NULL* source word) the lexical weight is computed as

$$p_w(\bar{t}|\bar{s}, a) = \prod_{i=1}^{N} \frac{1}{|j|(i,j) \in a|} \sum_{\forall (i,j) \in a} w(t_i|s_j) \tag{1.28}$$

If there are multiple alignments for a given phrase pair, the lexical weight is defined as the maximum weight for all alignments.

$$p_w(\bar{t}|\bar{s}) = \max_a p_w(\bar{t}|\bar{s}, a) \tag{1.29}$$

For example, consider one of the phrases pairs learned in Figure 1.2, $\langle$Mary did not, Marie ne$\rangle$. The lexical weight for this phrase pair in both directions for the alignment shown in Figure 1.2 are computed as follows.

$$\begin{aligned} p_w(\text{Mary did not}|\text{Marie ne}) &= w(\text{Mary}|\text{Marie}) \times w(\text{did}|\text{ne}) \times w(\text{not}|\text{ne}) \\ p_w(\text{Marie ne}|\text{Mary did not}) &= w(\text{Marie}|\text{Mary}) \times \frac{1}{2}\left(w(\text{ne}|\text{did}) + w(\text{ne}|\text{not})\right) \end{aligned}$$

1.3.4 MERT

The log-linear model described depends on the parameters λ_1^M. These parameters need to be tuned, so that the log-linear model produces good translations.

Minimum Error Rate Training (MERT) described in (Och, 2003), is an efficient algorithm to determine the optimum weight parameters λ_1^M for the log-linear model. This is a supervised learning algorithm, where we tune the model parameters on a given development set consisting of source language sentences and their corresponding translations. The algorithm can be summarized as follows.

1. *Initialization*: We assign some initial weights to the log-linear model. These can be random or based on some heuristics.

2. *Translate*: Using the log-linear model, we translate the source sentences of the development set to get an n-best list of translations.

3. *Score*: We compare the n-best list of translations obtained in the previous step, with the reference translations provided with the development set. In this work, we use the BLEU metric (discussed in section 1.4 to do the comparison.

4. *Re-estimate*: The parameters λ_1^M are now improved in the direction of the best candidate translation. To do this, we need an efficient way of computing the denominator of the log-linear model (shown in Equation 1.5), which is a sum over all possible translations. (Och, 2003) provides an approximation for doing this using n-best lists. The resulting parameters are used as the new weights for the log-linear model.

5. *Repeat until convergence*: We repeat the steps 2-4 until the weights converge.

The weights λ_1^M obtained this way are usually optimized for the domain of the training data. This forms the basis of our refinements discussed in Chapter 4.

1.4 Evaluation

In our experiments, we consider three different metrics for automatic evaluation of machine translation performance.

1.4.1 BLEU

The BLEU score (Papineni et al., 2002), computes the geometric mean of modified n-gram precisions (p_n) between a hypothesis translation and a set of reference translations.

The standard n-gram precision measurement is sometimes inaccurate. Consider the following example,

- Reference: The road less travelled.

- Candidate: The the the.

Here the standard unigram precision value will be $\frac{3}{3} = 1$. However, it is clearly a bad translation. To avoid this problem, we use modified n-gram precision where once an n-gram is matched it is consumed and cannot be matched again.

To compute the modified n-gram precision, we first count the number of times an n-gram occurs in a candidate sentence. This count is then *clipped*, so that it is at most equal to the number of times that n-gram occurs in the reference translation. This is done to avoid over-rewarding common n-grams like "the". A side effect of clipping the n-gram counts is that BLEU is hard to optimize (Pauls, DeNero, and Klein, 2009). The modified n-gram precision is then computed as follows.

$$p_n = \frac{\sum_{C \in Candidates} \sum_{n-gram \in C} count_{clip}(n-gram)}{\sum_{C' \in Candidates} \sum_{n-gram' \in C'} count(n-gram')} \tag{1.30}$$

So, in the example the modified unigram precision would be $\frac{1}{3}$.

The BLEU score is computed using the following equation.

$$BLEU = BP \cdot \exp(\sum_{n=1}^{N} \frac{\log p_n}{N}) \tag{1.31}$$

Here p_n is the modified n-gram precision. BP is the *brevity penalty* and it penalizes short sentences. Longer sentences are already penalized by the modified n-gram precisions. In this work, we use $N = 4$. It is important to note that BLEU measures the accuracy and not the number of errors. Thus good translations will have higher BLEU.

1.4.2 WER

WER (word error rate) is calculated as the minimum number of substitutions, insertions and deletions required to convert the output sentence into the reference sentence. It is also referred to as the *edit distance* or *Levenshtein distance*.

1.4.3 PER

In WER, the word order in a sentence is very important. This is misleading because a sentence with a different word order than the reference sentence would get a high WER score even though it is an acceptable translation. To overcome this problem a position-independent variant of WER is introduced. This is called position independent word error

rate (PER). Here the sentence is treated as a bag of words and the edit distance with the reference translation is calculated. The PER is always less than or equal to the WER.

Chapter 2

Model Adaptation

2.1 The Need for Model Adaptation

We saw in Chapter 1 that we can use parallel corpora (sentence aligned corpora in different languages) to train a log-linear model that can translate from one of the languages to the other. We do this by learning phrase translations, language models and other features from the parallel training data.

Where do we obtain this parallel training data from? Obtaining data like this manually is a very time consuming and expensive task, often prohibitively so. Fortunately, there are some texts that are naturally published in multiple languages. For example, the proceedings of the Canadian parliament are officially recorded in both English and French (Hansard corpus). Similarly the European Parliament proceedings are recorded in a number of European languages (Europarl corpus).

This data can now be used to learn the phrase translations and language models, based on which our log-linear model can perform translations between languages. Unfortunately, this parallel data is quite "domain-specific". The sentences occurring in the Hansard and Europarl corpora, for example, will be limited to legislative discourse.

This has the effect of overfitting the log-linear model to the domain of the training data. This is because the underlying statistical models closely approximate the distributions of the training data, which are not completely accurate for other domains. While the model performs well on sentences in that domain, it can produce sub-par translations in domains which have insufficient overlap with the training data.

While there is a scarcity of domains in which parallel data is available, there is no limit

on the domains in which our log-linear model may be required to operate. It is possible that we would need a good translation system in a domain that contains no parallel data. In this case, we would train our translation system in a different domain and then tweak its performance in the required domain. This process is called *Model Adaptation* (it is also referred to in the literature as *Domain Adaptation*).

In model adaptation, as explained above, the domain of the training data is different from that of the test data. We refer to data in the target domain as being *in-domain*, while data in a different domain is said to be *out-of-domain*.

2.1.1 Workshop on Machine Translation

Since 2005, annual Workshops on Statistical Machine Translation (WMT) have been held, with the goal to encourage work on SMT. These workshops focus on different aspects of SMT like training, evaluation etc. Different participants can compete in the *shared task*, which is some contest related to SMT. In order to facilitate comparison between different works, training and test data is provided for various European language pairs.

In 2008 and 2009, a *system combination* shared task was added. For this task, training data was provided from two domains; European parliamentary proceedings (Koehn, 2005) and News data. As a result, in recent years there has been a lot of interest in model adaptation. A majority of the work in model adaptation that we review in section 2.2.4, uses WMT shared task data.

In this thesis, we use parallel data from two domains provided in WMT 2008. In addition, we also use data obtained separately in a legal domain. Details of the data used are provided in Chapter 3.

2.2 Techniques in Model Adaptation

Model Adaptation is an active research area in Statistical Machine Translation. There have been several papers in recent years that outline various supervised, semi-supervised and unsupervised techniques to adapt the models to different domains. The approaches vary also in the amount and type of data used to perform adaptation.

Model adaptation usually boils down to Language Model or Translation Model adaptation. It is also common to adapt both the models to the new domain. We provide a high-level overview of techniques used in adapting the models.

2.2.1 Language Model adaptation

A language model is only built for the target language using monolingual data in that language. So depending on the type and amount of data available, there are several possible adaptation strategies.

- If there is no in-domain monolingual data, then there is not much one can do and we just stick to the out-of-domain monolingual data. A notable exception is the mixture-modeling approach in (Foster and Kuhn, 2007), which we discuss later.
- If a small amount of in-domain data exists, then using a language model built entirely from in-domain data is not the best option. Some of the options available to do adaptation are combining the data to build a single language model, build a language model for each domain and add both as separate components in the log-linear model and interpolate the two language models to form a language model with low perplexity. Usually, the latter two options give the best performance (Schwenk and Koehn, 2008).
- If a large amount of in-domain data is available, then using a single language model built from this data will give a huge performance improvement (Bertoldi and Federico, 2009). However, in the case of SMT, more data is usually better. Using interpolated or multiple language models would give better results even in this case.

2.2.2 Translation Model adaptation

Training translation models requires parallel data. Unfortunately, parallel data is much scarcer than monolingual data in all domains, and it is non-existent in some. Thus, techniques that utilize less parallel data are very widely applicable.

- If there is a large amount of parallel data available, then training a new translation model on this data will give immediate improvements. In addition, one could combine the in-domain and out-of-domain parallel data to train a combined translation model or use multiple translation models. As shown in (Koehn and Schroeder, 2007), using multiple models performs better. Unfortunately, there is no equivalent of interpolated language models here as we cannot tune for lowest perplexity.
- If there is little or no in-domain parallel data available, then directly adapting the translation model is difficult. One approach is to use the out-of-domain SMT system

to first generate in-domain parallel data by translating monolingual sources as shown in (Ueffing, Haffari, and Sarkar, 2007) and (Bertoldi and Federico, 2009).

2.2.3 Other approaches

- *Mixture modeling* (Hastie et al., 2005): An interesting technique to adapt the model is by partitioning the out-of-domain data based on "closeness" to in-domain data. This technique does not require parallel data; the "closeness" can be measured from monolingual data in the two domains.

  The partitions are then used to train different language or translation models. The final model is a weighted combination of these two, where each model is weighted according to the "closeness" metric discussed above. (Foster and Kuhn, 2007) and (Paul, Finch, and Sumita, 2009) get positive results using this approach.

- In this work, we provide a completely new approach to performing model adaptation, by tuning it in an *unsupervised* fashion on monolingual in-domain data. This approach is described in detail in Chapter 4.

2.2.4 Summary of previous work

In this section we give a summaries of related works in model adaptation.

In (Ueffing, Haffari, and Sarkar, 2007) a framework is presented, which uses monolingual source data to enhance translation performance. The log-linear model is iteratively improved by using the translations generated by the model in the previous iteration. This is done by incorporating information in the translations as additional data in the log-linear model for the next iteration. However, not all of the generated translations are good. So, the translations are first *filtered* to weed out bad translations. This *filtering* step is crucial and was extensively studied in that paper. Although there was no explicit domain-adaptation performed in the paper, the technique introduced there forms the basis of one group of experiments detailed in Chapter 3.

(Koehn and Schroeder, 2007) explore the performance differences in combining training data from different domains as compared to using separate translation models from each domain. In addition, they try combining, interpolating and using multiple language models (from different domains). They find that using separate translation models (in addition to multiple language models) gives best results. Combination of language models across

domains was also studied in (Schwenk and Koehn, 2008). They find that interpolated language models give similar performance to using multiple language models, but are more memory efficient. In addition, they find significant BLEU gains when re-ranking the n-best hypotheses with large continuous space language models.

In (Hildebrand et al., 2005), information retrieval techniques are used to select sentence pairs in the training data that are similar to the test sentences. These selected sentences are used to train language models and translation models to improve performance.

In (Foster and Kuhn, 2007), two approaches are considered; a supervised approach (called cross-domain adaptation) with a small in-domain development set and an unsupervised approach (called dynamic adaptation) where only the in-domain source sentences are available. They split up the training data into different domains based on the match with the in-domain source sentences. A mixture model is trained on the phrase tables and language models learned from the split components of the training data. In the case of cross-domain adaptation, the weights are tuned using the in-domain development set. In dynamic adaptation, the weights are set based on a "distance function" which indicates the similarity between the corpus components and the in-domain source text. A variation of this approach was also proposed in (Finch and Sumita, 2008).

(Bertoldi and Federico, 2009) use large monolingual sources to improve the performance across domains. They first synthesize parallel data by translating the data into another language (they only select the best translation). This parallel data is then used to learn a new translation model which *replaces* the translation model used to generate the parallel data. In addition, they use the large target domain data to create a language model that significantly improves the performance. They observe that using both translation models *simultaneously* reduces the performance of their system. This is in direct contrast with the results obtained in our experiments, as can be seen in Chapter 3.

In (Nakov and Ng, 2009), systematic improvements are made to the baseline model and the improvements at each step are noted. They use a large out-of-domain parallel corpus (Europarl) and a smaller in-domain parallel corpus (News-Commentary). They find that merging the out-of-domain and in-domain phrase tables yields best results. A particularly interesting improvement was provided by including "cognates" into the phrase tables. Cognates are words derived from a common root, but in different languages. They find cognates in the parallel data and boost their alignment (by adding them as extra parallel data).

In (Paul, Finch, and Sumita, 2009), the in-domain and out-of-domain data are used to

train separate models. These models are then weighted according to their fit with the in-domain dataset. They also attempt to model the translation of unknown words, which are usually copied over without translation. They train a *transliteration model* which is similar to a phrase translation model, except it works with sequences of characters. For training data, they collect aligned words from the parallel corpus, where the translated words are similar to the source words (using edit-distance). This *transliteration model* is then applied to the translations as a post processing step. They get positive results from both approaches tried in the Spanish-English task of WMT 2009.

2.2.5 Back-Translations

In this work, we make extensive use of "back-translations". We first use some SMT system to translate sentences into the target language. If we use a *reverse* SMT system to translate these sentences back into the source language, we call the final translations in the source language as "back-translations". In literature, they are also referred to as "round-trip translations". Figure 2.1 shows an example back-translation.

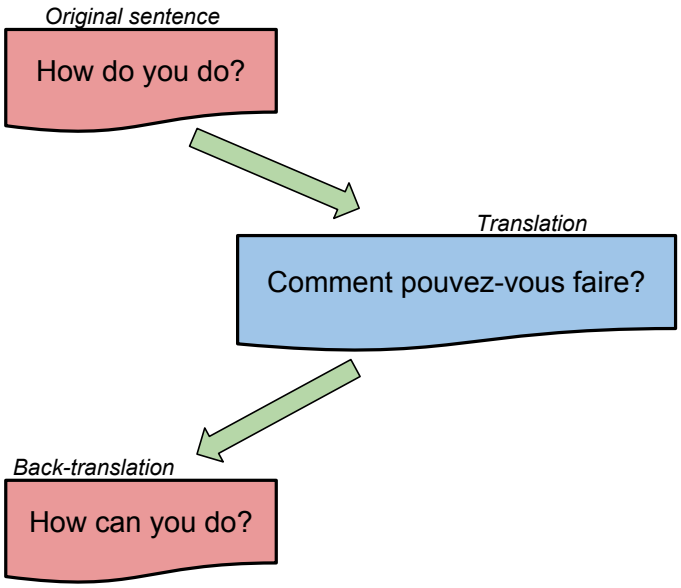

Figure 2.1: Example "back-translation". The intermediate translation is in French. The remaining sentences are in English.

The idea of "back-translations" is not new, and their utility has been debated for a while. In (Huang, 1990), back-translations were discussed as a technique for evaluation of obtained translations. The author notes that while this technique seems the most "natural", it has some complexities and may not be reliable. They note that even though the back-translations may closely match the original, there is no guarantee that the intermediate translations (in the target language) are good. In addition, we could fault the wrong SMT system for the poor performance since we have no way of knowing whether the forward or reverse translation systems is the source of errors.

In (Koehn, 2005) and (Somers, 2005), the authors were pessimistic about the idea. In experiments, they found that back-translations are not an accurate indicator of the performance of the intermediate systems. In particular, they note that we can get perfect accuracy for SMT systems which do nothing, i.e. in both translation directions the text is left unchanged. In both works, the authors use automatic evaluation metrics to arrive at their conclusions.

On the other hand, in (Rapp, 2009) the authors provides experimental evidence in favour of back-translations. A variant of BLEU called *OrthoBLEU* is introduced, which measures the similarity between translations based on sequences of characters. They find that back-translations correlate highly with human judgement and can therefore be used to evaluate SMT performance.

Although back-translations have been discussed for a while in literature, they have never been used in a model adaptation setting, to the best of our knowledge. We discuss some techniques for using these in the next chapters and find that back-translations do have some merit, at least in the field of model adaptation.

Chapter 3

Transductive Model Adaptation

3.1 Motivation

In Chapter 2 , we looked at the problem of *Model Adaptation* in Statistical Machine Translation (SMT) and discussed its importance. SMT models are tuned to the domain they are trained in. They do not perform to full potential in a different domain. As discussed in Chapter 2 , several approaches have been tried to adapt SMT models to work better in different domains.

This work is largely influenced by (Ueffing, Haffari, and Sarkar, 2007). The interesting thing about their approach is that it only requires source language data from the domain of interest, to improve the translation quality in that domain. Due to the paucity of parallel data in most domains, this approach has wide applicability.

In (Ueffing, Haffari, and Sarkar, 2007), the log-linear model is iteratively improved by using the translations generated by the model in the previous iteration, to improve performance. This is done by incorporating information in the translations as additional data in the log-linear model for the next iteration. However, not all of the generated translations are good. So, the translations are first *filtered* to weed out bad translations. This *filtering* step is crucial and was extensively studied in that paper.

In this work, we look at a novel way of filtering the generated translations. Essentially, we create another log-linear model which translates in the reverse direction (from target language to source language) and use the "back-translations" to determine the quality of the translation. In addition, we also explore new features which can be learned from the generated translations.

Our approach is *semi-supervised transductive* model adaptation. It is semi-supervised because we use labeled out-of-domain data in conjunction with unlabeled in-domain data to improve the model. By transductive, we mean that we use our model generated translations from the source sentences, to learn features which are then used to boost the model's performance. In particular, we learn new features from the *test set*. It is important to clarify that we use only the source part of the test set, and not the reference translations. The technique is described completely in Section 3.2.

3.2 Description

3.2.1 Algorithm

We denote the source language as S and the target language as T. Suppose we have an existing translation system in a some domain for both directions, denoted by $M_{S\rightarrow T}$ for the forward direction and $M_{T\rightarrow S}$ for the reverse direction. We would like to improve the translation performance of $M_{S\rightarrow T}$ in a different domain. Unfortunately there is very limited data in this domain, so the traditional approach of learning features from the in-domain data is infeasible.

Algorithm 3.1 describes the method used to adapt the model $M_{S\rightarrow T}$ to the new domain. The technique of transductive model adaptation is inspired by (Ueffing, Haffari, and Sarkar, 2007). A crucial difference is that we perform just one iteration of forward translation. It was observed in (Ueffing, Haffari, and Sarkar, 2007) that a significant amount of the boost in performance was observed during the first iteration. Instead of running multiple iterations, we focus on the extracted features and variations therein which might prove useful.

For transductive model adaptation, we require a relatively small amount of parallel data as a *dev set*. The relative sizes of different datasets can be seen in table 3.2. As shown in algorithm 3.1, we first obtain the n-best translations of the source sentences in the dev set S_{dev}, using $M_{S\rightarrow T}$. Now, our in-domain generated bitext G_{dev} consists of the source sentences and the n-best translations. We translate the n-best translations using the reverse translator $M_{T\rightarrow S}$ to get "back-translations" in the source language, denoted by S^b_{dev}. These back-translations are our measure of how accurate the translation was. Intuitively, if a sentence in S^b_{dev} closely matches the corresponding sentence in S_{dev}, then the intermediate translated sentence in T'_{dev} is a good translation and vice versa. We now extract features N_{dev} (in the form of addition phrase translation models) from the generated

Algorithm 3.1: Transductive Model Adaptation

Input: Existing Decoder $M_{S \to T}$ that translates from S to T (forward direction)
Input: Existing Decoder $M_{T \to S}$ that translates from T to S (reverse direction)
Input: Development set of untranslated sentences $\langle S_{dev}, T_{dev} \rangle$
Input: Source sentences of test dataset S_{test}
Output: Translations of S_{test} using domain-adapted model

`// First learn the new features and tune the model on the dev set`

1 Translate S_{dev} using $M_{S \to T}$ to get n-best output T'_{dev}
2 Replicate sentences in S_{dev} to correspond to the n-best output T'_{dev} and get the generated dataset $G_{dev} = \langle S_{dev}, T'_{dev} \rangle$
3 Translate T'_{dev} using $M_{T \to S}$ to get back-translations S^b_{dev}
4 $N_{dev} = Extract(M_{S \to T}, G_{dev}, S^b_{dev})$; `// Extract new features`
5 Add N_{dev} to $M_{S \to T}$ and ***train the new model*** $M^{dev}_{S \to T}$ on $\langle S_{dev}, T_{dev} \rangle$

`// Obtain corresponding models for the test set and transfer learned weights`

6 Translate S_{test} using $M_{S \to T}$ to get n-best output T'_{test}
7 Replicate sentences in S_{test} to correspond to the n-best output T'_{test} and get the generated dataset $G_{test} = \langle S_{test}, T'_{test} \rangle$
8 Translate T'_{test} using $M_{T \to S}$ to get back-translations S^b_{test}
9 $N_{test} = Extract(M_{S \to T}, G_{test}, S^b_{test})$; `// Extract new features`
10 Add N_{test} to $M_{S \to T}$ and get the new model $M^{test}_{S \to T}$ by ***copying over learned weights*** from $M^{dev}_{S \to T}$

`// Translate the test sentences using the adapted model`

11 Translate S_{test} using $M^{test}_{S \to T}$ and return the output

bitext G_{dev}, using the information provided by the back-translations. These features are added to the original model $M_{S \to T}$ as new components in the log-linear model. Figure 3.1 shows a graphical description of feature extraction from some example data.

The resulting model is tuned on the dev set, the same dev set from which G_{dev} was extracted, using MERT (Och, 2003) to obtain the tuned model $M^{dev}_{S \to T}$.

To test the performance of this model, we perform identical steps on the source language part of the test set S_{test} to generate G_{test}. Using the same extraction function as before, we extract features N_{test} from the generated bitext. Now, we already have the tuned model weights for the dev set. We simply copy over the weights to this new model, to get the relative importance of each feature in the log-linear model. We make a reasonable assumption here that, in a given domain, the tuned weights will be identical for different data sets. So the final model uses the newly generated features for the test set, with the weights learned for

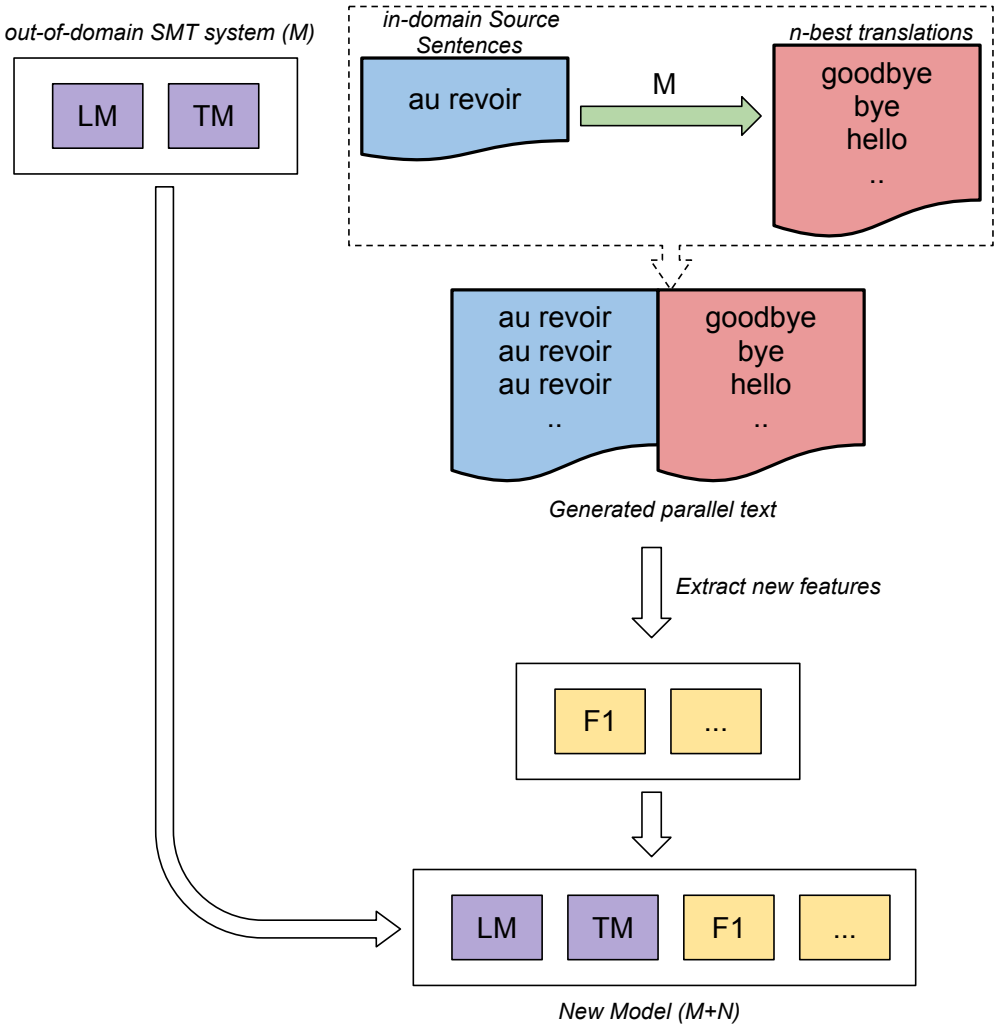

Figure 3.1: Process of feature extraction shown for example data. The out-of-domain SMT system M (corresponding to $M_{S \to T}$ in Algorithm 3.1) is first used to generate in-domain parallel data. (F1, ...) indicate the new features learned. The new model is denoted by $(M + N)$.

the dev set and is denoted by $M^{test}_{S \to T}$. Figure 3.2 shows graphically the process of learning the weights for the new features.

3.2.2 Feature Extraction

The crucial step in algorithm 3.1 is the extraction of features from the generated bitext (G_{dev} or G_{test}). In our experiments, the features extracted refer to additional phrase translation models that are learned from the generated bitext.

In (Ueffing, Haffari, and Sarkar, 2007), the generated bitext G is filtered based on some criteria. These filtered sentences are then used to create a phrase table that is used as an additional component in the log-linear model. In our experiments, the filtering is done based on the quality of the "back-translations". We measure this quality using the BLEU, WER and PER metrics (by comparing the back-translations S^b to the original source sentences S). BLEU is a score indicating the quality of a translation while WER and PER are error measures. Good translations are indicated by higher BLEU scores and lower WER or PER values. Only those sentences with scores greater than some threshold in the case of BLEU, and less than some threshold in the case of WER and PER, are selected. These form the "Good" phrase tables, indicating that these refer to the good translations.

In addition, we also experiment with including the "Bad" sentences as a separate model, i.e. those sentences that were discarded during the filtering step. The motivation for this is that there could be some useful phrases in the bad translations, which given the correct weight would improve the overall accuracy. Thus we run a set of experiments where we include the "Bad" translations as a separate model. We show a simplified example of this process in Figure 3.3.

Another interesting observation is that the generated phrase tables contain phrases that have already been seen in $M_{S \to T}$, which is the out-of-domain phrase table. It seems unfair that both seen and unseen phrases should get the same log-linear weight. This suggests splitting up the phrase tables based on whether they have been seen before. We perform experiments in which this information is used as a separate feature. A simplified example of splitting phrase tables into seen and unseen components is shown in Figure 3.4.

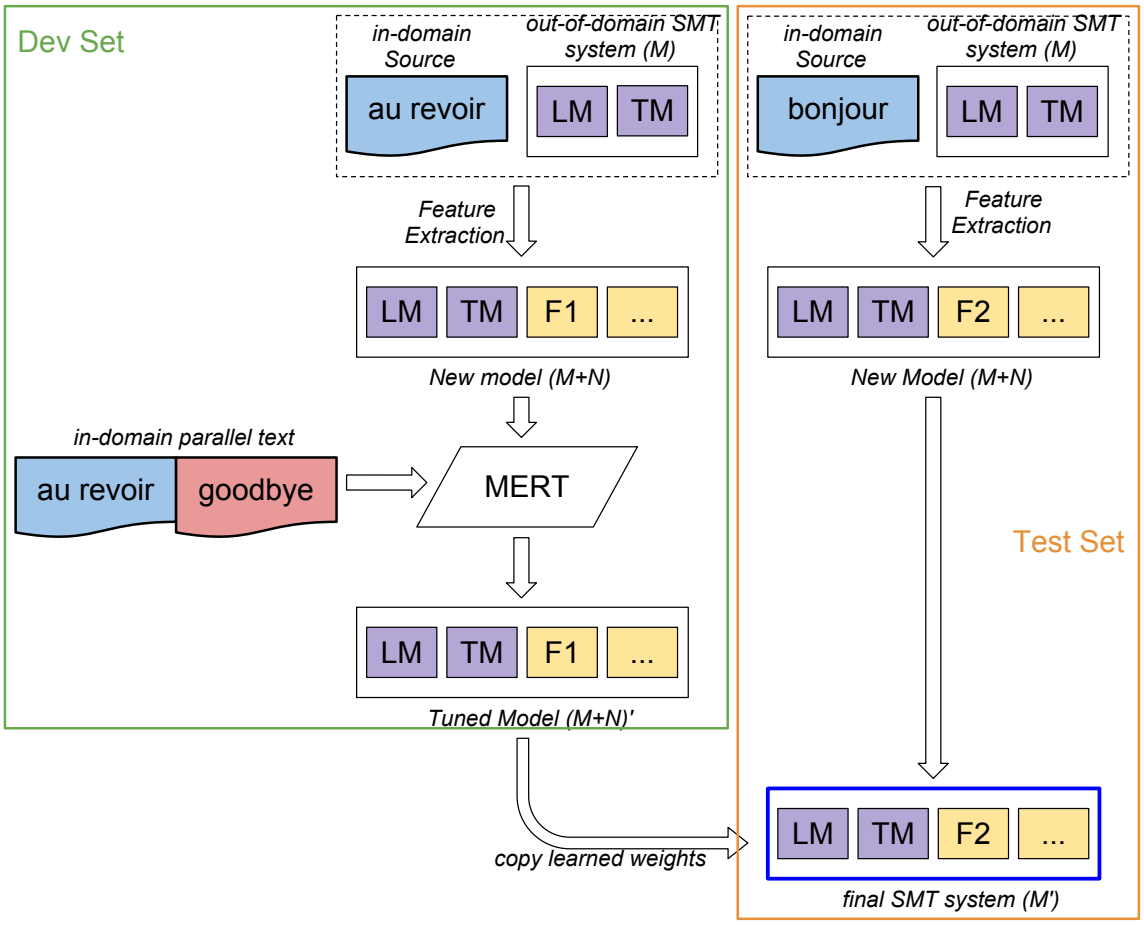

Figure 3.2: Process of learning the weights for the new features. Note that we extract separate features (F2, ...) from the test set and discard the features learnt from the dev set. The weights learned on the dev set are just copied over to the model learned from the test set. The final model M' corresponds to $M_{S \to T}^{test}$ referred to in Algorithm 3.1.

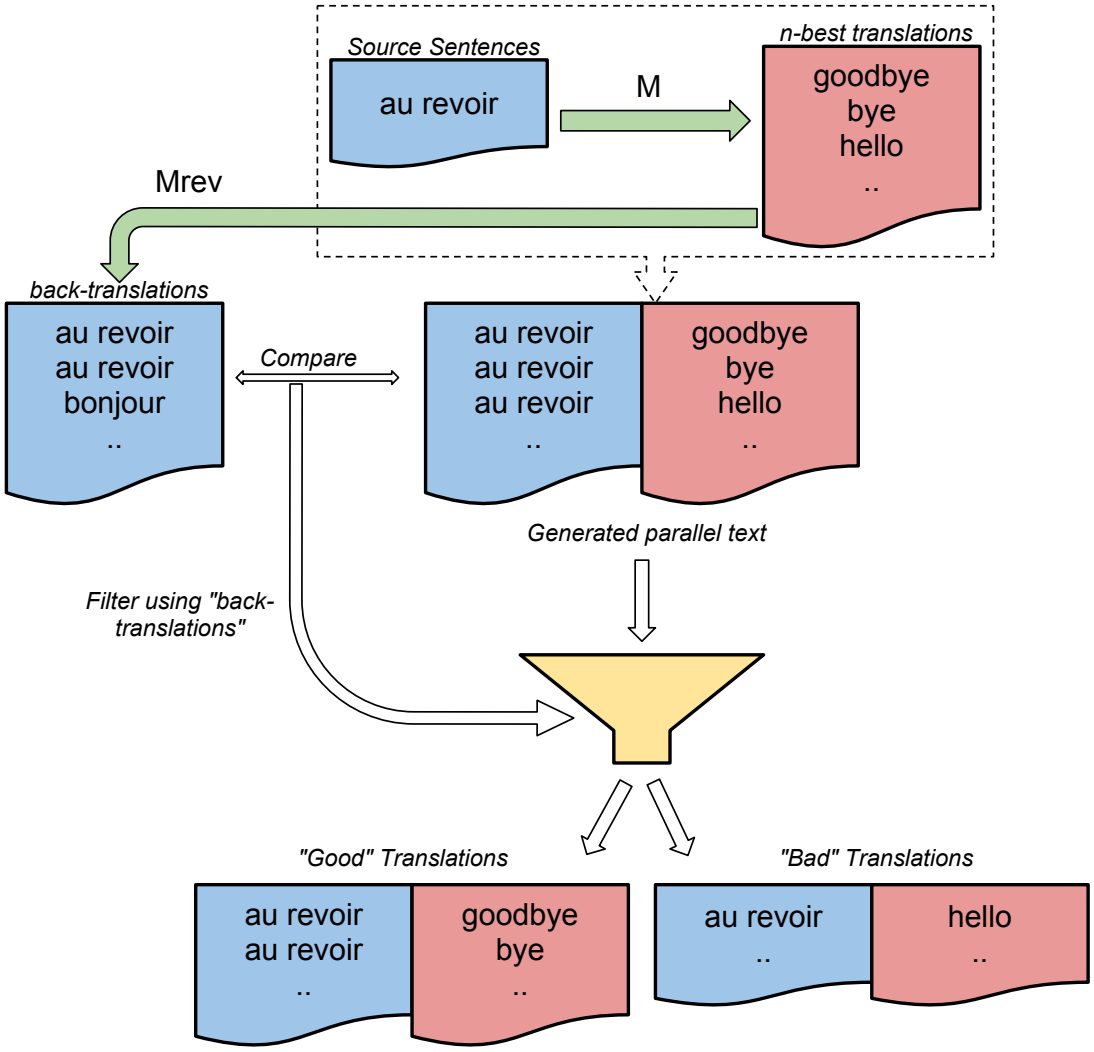

Figure 3.3: Example indicating the process of filtering out "Good" and "Bad" translations. M corresponds to $M_{S \to T}$ while $Mrev$ corresponds to $M_{T \to S}$ in Algorithm 3.1. In this trivial example, one can see how the bad translation has been singled out. In experiments, the comparison is based on BLEU, WER and PER scores.

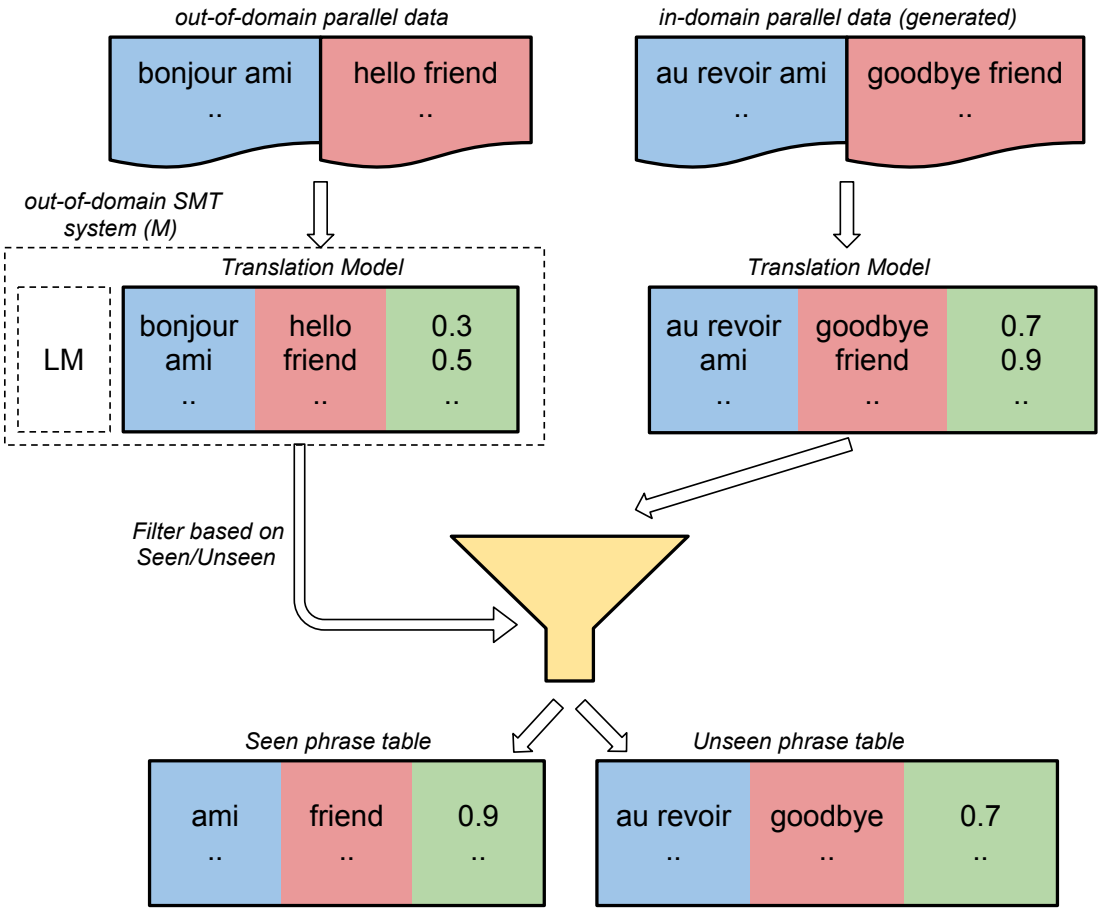

Figure 3.4: Example showing the process of filtering a phrase table into "Seen" and "Unseen" components. If a phrase has been seen in out-of-domain data, then its put into the "Seen" phrase table, and likewise for unseen phrases.

3.3 Experiments

3.3.1 Setup

We use data from three different domains in our experiments, which are listed below.

1. Europarl, (Koehn, 2005), mainly consists of European parliamentary proceedings which are legislative in nature. This consists of a large amount of parallel data and is used as the out-of-domain data. We used the Europarl corpus provided by the shared task in the *Third Workshop on Statistical Machine Translation* (WMT 2008).

2. News-Commentary is less formal and consists of commentary on news articles. This is a smaller data set and was also obtained from the WMT 2008 shared task.

3. JRC-Acquis, (Steinberger et al., 2006) is a legal corpus that consists of aligned documents of the European union. This was not part of the WMT 2008 shared task and was obtained separately.

All the domains are significantly different from each other. This fact is reflected in the difference in the machine translation performance between the three domains. In table 3.1, we display an example sentence from each of the three domains used in this thesis. These sentences were selected at random from a pre-processed corpus (tokenized and lower-cased). Hopefully, these sentences give a feel for the domain differences that exist between the three domains.

We use the Moses decoder (Koehn et al., 2007) as the phrase based SMT system. Please refer to Appendix B for more information on Moses. All of the experiments were performed on French as source and English as the target language. The Europarl corpus from WMT 2008 was selected as the out-of-domain corpus and used to generate the phrase tables for both $M_{S \to T}$ and $M_{T \to S}$. In addition, the Europarl corpus is also used to create a language model with KN-discounting which is then used in the translations. Based on results by (Goodman, 2001), we limit ourselves to a 5-gram language model. Using higher order language models does not improve performance by much. This is the language model used in all the experiments. Two sets of experiments were performed with different in-domain data. In the first set, we used the news commentary corpus from WMT 2008. In the second, the JRC legal dataset was used as the in-domain data. The sizes of the datasets are shown in table 3.2.

Domain	Example Sentence (French)	Reference Translation (English)
Europarl	le parlement , dans sa sagesse , est totalement d' accord avec vous .	parliament , in its wisdom , completely shares your view .
News-Commentary	je crois que l' allemagne s' en sortirait mieux , dans dix ans , avec davantage de politiques néolibérales .	i think that germany would be better off in a decade under more neo-liberal policies .
JRC	a) une copie certifiée conforme de l autorisation accordée par l état membre de référence ;	(a) a certified copy of the authorisation granted by the reference member state ;

Table 3.1: Example sentences in the Europarl, News-Commentary and JRC domains used in this thesis. These sentences have been pre-processed to be tokenized and in lower case.

Data	Use	Domain	Size
Europarl (dev)	phrase tables + language model	out-of-domain	948507
Europarl (dev1)	tune $M_{S\to T}$ and $M_{T\to S}$	out-of-domain	2000
news-commentary (dev2)	learn new features and tune	in-domain	1057
news-commentary (test)	learn new features and test	in-domain	1064
JRC (dev2)	learn new features and tune	in-domain	2000
JRC (test)	learn new features and test	in-domain	2000

Table 3.2: Description and size of the data sets. The sizes are shown in number of sentences.

For each of the target domains, we perform the following experiments. In all the experiments except "Baseline 1", we tune the log-linear model on the in-domain dev set, dev2. In all cases, we learn new in-domain translation models (phrase tables) and add them as new features in the log-linear model, in addition to the out-of-domain phrase table that is already present.

1. Baseline 1: The translator consists of the out-of-domain models (phrase table + language model), trained on out-of-domain dev sets. This is our existing translation system $M_{S\to T}$ that we want to adapt to the new domain. Since it is optimized for out-of-domain translation, the performance in the target domain will be sub-optimal.

2. Baseline 2: The translator consists of the out-of-domain models, trained on the in-domain dev set. This optimizes the weights of the log-linear model for that domain.

3. Keep All: We use the "Baseline 1" translation system $M_{S\to T}$ to get 100-best translations for the in-domain dev set (dev2). The dev2 source language sentences and the

Domain	Translation Model	Number of Phrases
Out-of-domain	Europarl	64,558,603
News-Commentary	Keep All	526,154
	Seen	58,642
	Unseen	467,512
JRC	Keep All	868,614
	Seen	107,304
	Unseen	761,310

Table 3.3: Number of phrases in some of the translation models that are learned in the experiments. Please note that the Europarl translation model is included as an additional model in all the experiments.

generated 100-best target language sentences form our in-domain corpus G_{dev}, that we use to generate the in-domain phrase table. This phrase-table is added as an additional component to the log-linear model. The resulting model is tuned on the same in-domain dev set, dev2. Since we use the entire set of translations, we refer to this experiment as "Keep All".

4. Seen Unseen: In this group of experiments, we split up the phrase table generated in the "Keep All" experiment into two phrase tables, one contains the phrases that are common to the out-of-domain phrase table, and the other contains phrases that weren't seen out-of-domain. We perform three experiments; using just the seen phrases as a new model, using just unseen ones and using both as two additional models. In all cases, the models are used as additional components in the log-linear model, along with the out-of-domain phrase table. The resulting models are tuned on the in-domain dev set.

5. Keep All *: This experiment is inspired by the seen/unseen experiments discussed above. Instead of splitting the generated phrase table, we add two new features to the phrase tables corresponding to their seen/unseen property.

   If a phrase in the generated phrase table exists in the out-of-domain phrase table, then the seen feature is set to 1 and the unseen feature to 0. This is reversed if the phrase is new. These features are also added to the out-of-domain phrase table, to give a total of four new features. The resulting model is tuned on the in-domain dev set.

6. Good: In this group of experiments, we threshold the generated corpus G_{dev} we saw in

"Keep All" and keep only the sentences that are better translations than the threshold value. The threshold used is based on back-translations. We back-translate the 100-best generated sentences in the target language, back to the source language using a reverse direction translator similar to the forward direction translator in "Baseline 1". Now we compare the back-translated sentences, to the original source sentences and determine "good" sentences based on a metric. In our experiments, we use BLEU, WER and PER as thresholding metrics.

7. Good Bad: This group of experiments is similar to the "Good" experiments. Instead of using just the "good" sentences, we also use the "bad" sentences by adding them in as a separate model (by creating a separate phrase table). This model is then trained on the in-domain dev set to obtain the optimal weights.

8. Good Bad Seen Unseen(GBSU): In this group of experiments, we further split up the "good" and "bad" phrase tables into seen and unseen, based on whether that phrase combination was seen in the original out-of-domain phrase table. So we get four in-domain phrase tables in total; good-seen, good-unseen, bad-seen and bad-unseen, and also one out-of-domain phrase table. These phrase tables are tuned on the in-domain dev set.

3.3.2 Results

Figure 3.5 shows the variation in the size of the filtered "Good" dataset with the selected threshold. It is important to note that BLEU is a score of how good a translation is, while WER and PER are error measures. Therefore good translations have higher BLEU scores but lower WER and PER values. This trend is also reflected in the Figure 3.5. It is also interesting that the WER measure filters out more sentences than the PER score. This is probably what makes thresholding by WER more effective than that by PER, as can be seen in the results. In particular we note that in the best performing experiments ("Good" with high thresholds), the new features added are extracted from approximately 1000 sentences.

Table 3.3 shows the number of phrases that were learned in the different phrase translation models, and we can understand the relative sizes of the phrase tables. We note that the complete generated phrase table (used in "Keep All") is a small fraction in size compared to the out-of-domain phrase tables (about 0.8% for News-Commentary and about 1.3% for

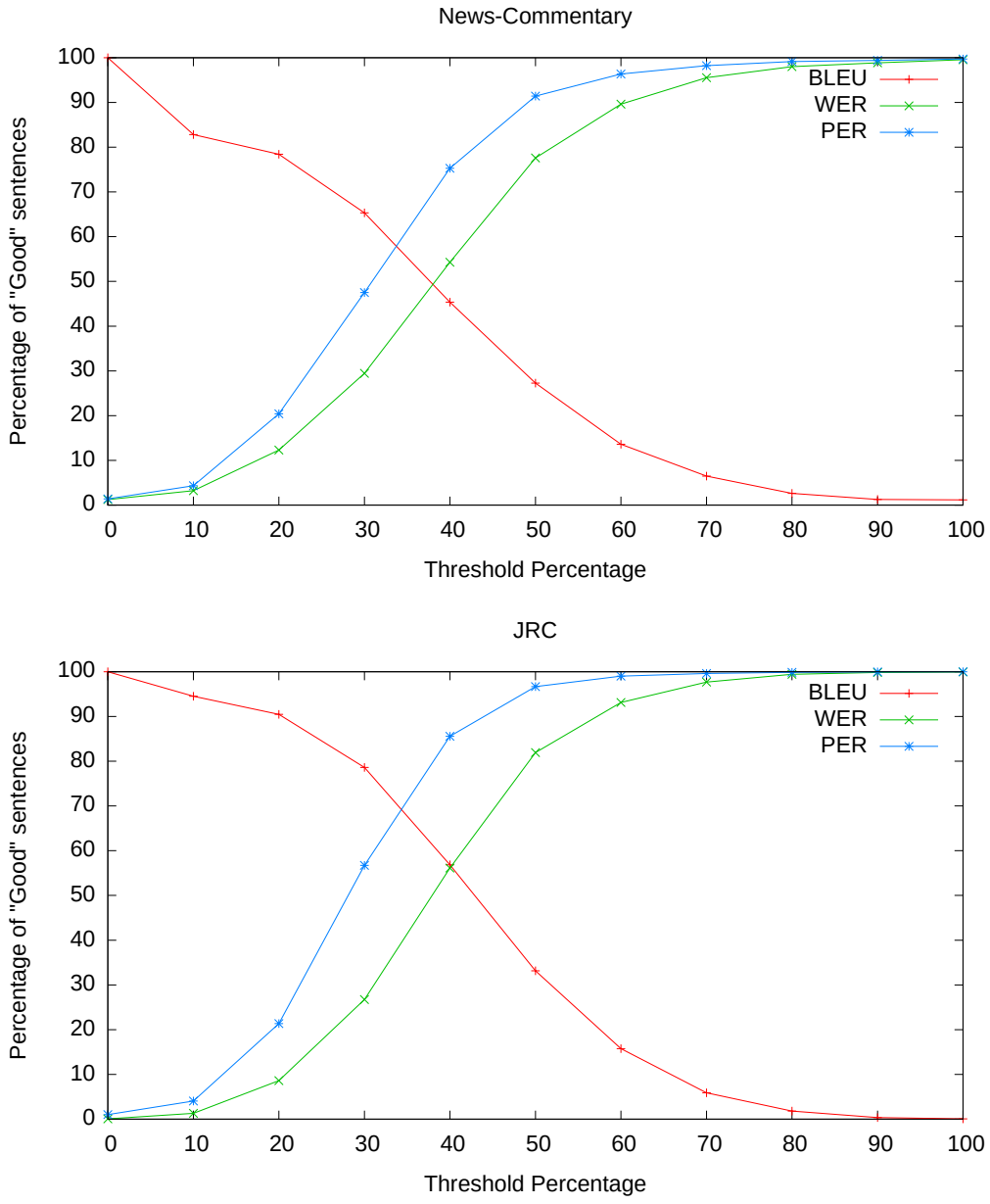

Figure 3.5: Cumulative histograms showing the percentage of sentence pairs in the generated dataset G_{dev} that are "Good" for a given threshold. The remaining sentences are classified as "Bad". The graphs are shown for the data generated from the development set G_{dev} only. The results for the test set G_{test} are very similar.

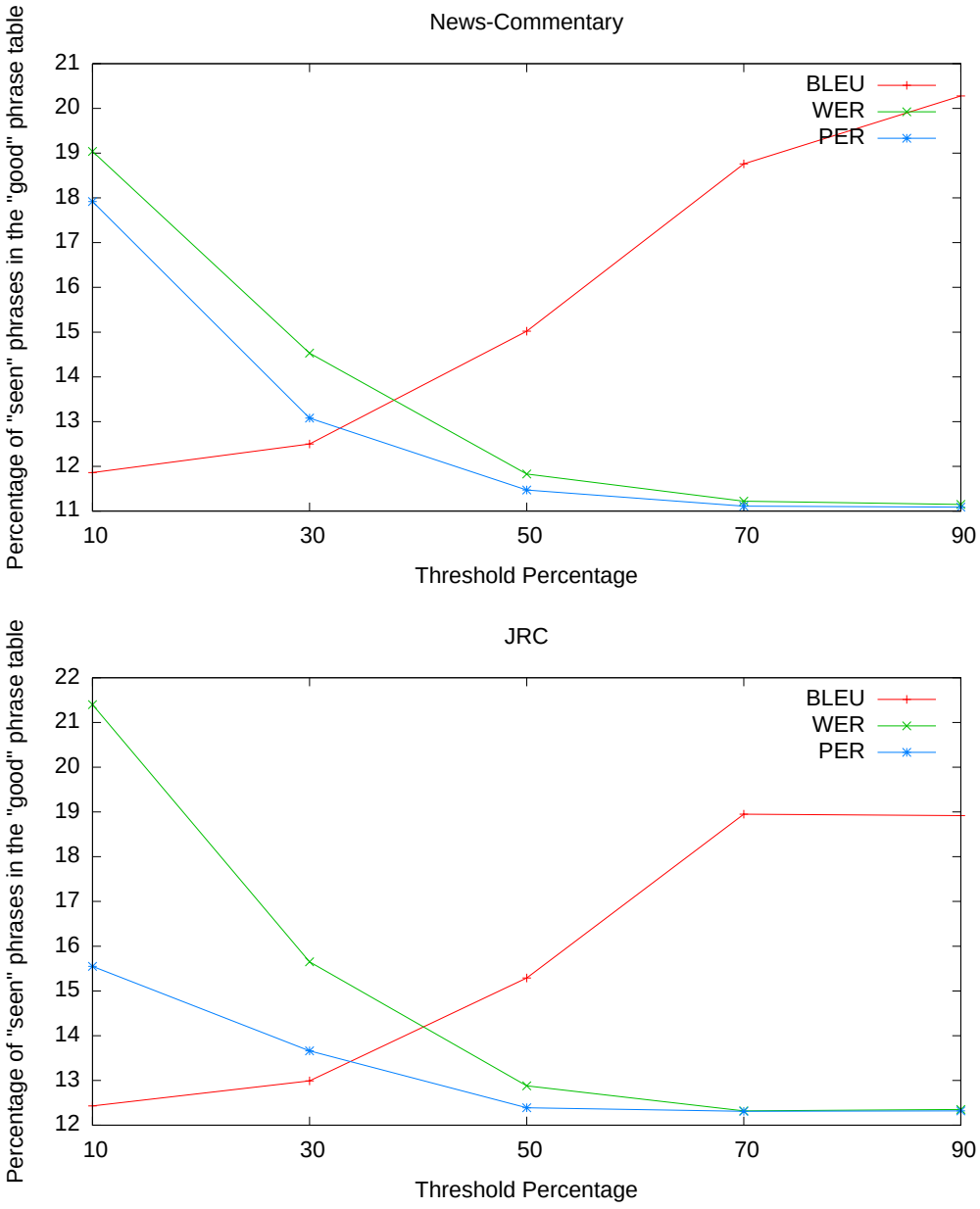

Figure 3.6: This figure shows the variation in percentage of "seen" phrases in the "good" phrase table with thresholding, for both domains. The clear trend is that better translations have more seen phrases.

Threshold	News-Commentary			JRC		
	BLEU[%] ±0.9%	WER[%] ±1.1%	PER[%] ±1.1%	BLEU[%] ±0.9%	WER[%] ±0.85%	PER[%] ±0.7%
BLEU > 10	24.15	60.24	47.06	40.31	48.04	33.80
BLEU > 30	24.37	60.22	46.86	40.56	47.79	33.40
BLEU > 50	23.70	61.71	48.36	39.75	48.45	33.92
BLEU > 70	24.75	59.37	**46.17**	39.21	48.84	34.20
BLEU > 90	***25.12***	***59.09***	46.24	**41.07**	***47.12***	***32.94***
PER < 10	**24.72**	**59.81**	**46.45**	38.53	49.62	34.58
PER < 30	24.13	60.79	47.30	40.29	48.00	33.54
PER < 50	24.49	60.27	47.10	40.39	**47.80**	**33.34**
PER < 70	24.11	60.56	47.08	40.20	48.04	33.75
PER < 90	24.48	60.29	46.98	**40.45**	47.89	33.39
WER < 10	**25.08**	**59.31**	***46.13***	***41.24***	**47.16**	**32.95**
WER < 30	23.78	61.29	48.16	39.91	48.33	33.85
WER < 50	24.05	60.78	47.28	40.40	47.73	33.36
WER < 70	24.31	60.43	46.95	40.59	47.61	33.18
WER < 90	24.51	60.47	47.11	40.40	47.82	33.33

Table 3.4: Results for "Good" experiments for both domains. The numbers in **bold** font indicate the best performing threshold for the metric in that block. The numbers in ***bold slanted*** font indicate the best performing threshold for all the metrics.

Threshold	News-Commentary			JRC		
	BLEU[%] ±0.9%	WER[%] ±1.1%	PER[%] ±1.1%	BLEU[%] ±0.9%	WER[%] ±0.85%	PER[%] ±0.7%
BLEU > 10	22.04	64.34	50.79	33.36	53.66	37.53
BLEU > 30	22.90	61.66	47.99	**37.08**	50.82	**35.05**
BLEU > 50	**24.02**	**61.03**	**47.84**	37.03	**50.51**	35.30
BLEU > 70	23.89	61.50	48.23	35.95	52.02	36.63
BLEU > 90	23.69	61.75	48.41	31.65	55.24	39.59
PER < 10	24.15	60.49	47.41	33.97	53.08	36.93
PER < 30	23.73	61.27	47.57	***38.41***	***49.37***	***34.36***
PER < 50	24.08	61.19	47.78	35.28	51.41	36.03
PER < 70	**24.28**	**60.63**	**47.36**	29.86	60.73	44.17
PER < 90	24.04	61.05	47.68	26.75	66.72	49.28
WER < 10	24.19	***60.56***	***47.29***	34.40	52.87	36.53
WER < 30	***24.45***	60.62	47.75	**37.28**	**50.45**	**35.33**
WER < 50	22.67	62.25	48.55	36.10	51.72	35.90
WER < 70	24.26	60.61	47.43	35.35	52.05	36.02
WER < 90	23.59	61.22	47.94	21.58	88.22	71.49

Table 3.5: Results for "Good+Bad" experiments for both domains. The numbers in **bold** font indicate the best performing threshold for the metric in that block. The numbers in ***bold slanted*** font indicate the best performing threshold for all the metrics.

Threshold	News-Commentary			JRC		
	BLEU[%] ±0.9%	WER[%] ±1.1%	PER[%] ±1.1%	BLEU[%] ±0.9%	WER[%] ±0.85%	PER[%] ±0.7%
BLEU > 10	**24.13**	61.02	**47.48**	39.35	48.89	33.91
BLEU > 30	23.94	60.94	47.50	39.60	48.79	33.76
BLEU > 50	23.82	61.45	48.16	39.38	48.93	33.76
BLEU > 70	24.09	61.04	47.88	39.63	48.97	33.74
BLEU > 90	**24.13**	**60.79**	47.61	**39.72**	**48.48**	**33.68**
PER < 10	24.18	60.86	47.79	39.45	48.89	***33.62***
PER < 30	24.18	61.15	47.84	39.29	48.95	33.83
PER < 50	23.95	61.29	47.78	39.23	49.07	33.74
PER < 70	***24.26***	***60.72***	***47.37***	39.63	48.72	34.03
PER < 90	24.17	60.86	47.78	***39.87***	***48.46***	33.91
WER < 10	24.14	60.96	47.66	39.49	48.76	**33.65**
WER < 30	24.15	60.84	47.71	39.32	48.96	33.73
WER < 50	24.16	**60.74**	**47.64**	**39.57**	**48.70**	33.66
WER < 70	**24.17**	60.92	**47.64**	39.51	48.73	34.05
WER < 90	24.05	61.10	47.93	39.30	49.13	33.94

Table 3.6: Results for "GBSU" experiments for both domains. The numbers in **bold** font indicate the best performing threshold for the metric in that block. The numbers in ***bold slanted*** font indicate the best performing threshold for all the metrics.

Experiment		BLEU[%] ±0.9%	WER[%] ±1.1%	PER[%] ±1.1%
Baseline 1		23.53	63.39	50.17
Baseline 2		24.81	59.36	46.37
Keep All		24.45	60.02	46.7
Seen		25.1	59.39	46.28
Unseen		24.53	60.49	47.36
Seen+Unseen		24.37	60.48	47.37
Keep All *		24.28	60.27	46.84
Good	BLEU > 90	***25.12***	***59.09***	46.24
	WER < 10	25.08	59.31	***46.13***
Good+Bad	WER < 10	24.19	60.56	47.29
	WER < 30	24.45	60.62	47.75
GBSU	PER < 70	24.26	60.72	47.37

Table 3.7: All Results for *News-Commentary* domain. Only the best performing results from "Good", "Good+Bad" and "GBSU" are displayed. For exhaustive results please refer to Tables 3.4, 3.5 and 3.6. The numbers in ***bold slanted*** font indicate the best performing experiments. Note the exceptional performance of the "Good" thresholding experiments. The "Seen" experiment also does well.

Experiment		BLEU[%] ±0.9%	WER[%] ±0.85%	PER[%] ±0.7%
Baseline 1		39.39	49.47	34.16
Baseline 2		***41.39***	***46.96***	32.99
Keep All		39.61	48.86	34.35
Seen		41.10	47.21	33.15
Unseen		39.57	48.70	34.03
Seen+Unseen		39.06	49.08	33.80
Keep All *		40.53	47.84	33.50
Good	BLEU > 90	41.07	47.12	***32.94***
	WER < 10	41.24	47.16	32.95
Good+Bad	PER < 30	38.41	49.37	34.36
GBSU	PER < 10	39.45	48.89	33.62
	PER < 90	39.87	48.46	33.91

Table 3.8: All Results for the *JRC* domain. Only the best performing results from "Good", "Good+Bad" and "GBSU" are displayed. For exhaustive results please refer to Tables 3.4, 3.5 and 3.6. The numbers in ***bold slanted*** font indicate the best performing experiments. Observe the "Good" thresholding experiments and the "Seen" experiments come close to "Baseline 2", which performs the best.

JRC). Also, about a tenth of the phrases generated are already seen in the out-of-domain data (11.1% for News-Commentary and 12.4% for JRC).

In Figure 3.6, we can see the percentage of "seen" phrases in the "good" phrase tables for the different threshold criteria. We note that as we get more selective and choose only the best performing sentence pairs, the phrases learned contain more "seen" phrases. This indicates the importance of seen phrases and suggests that they are an important factor to consider while doing model adaptation.

The results for "Good" experiments are shown in Table 3.4. We can observe the trend that being more selective generally gives better results. This clearly shows that filtering the generated data has a significant impact on performance.

The results for "Good+Bad" and "GBSU" are shown in Table 3.5 and Table 3.6 respectively. Unfortunately, for these experiments no clear trend emerges. They do not perform as well as the "Good" experiments. In fact, in some cases the scores are lower than the baselines.

Table 3.7 shows the results for all the experiments. For the "Good", "Good+Bad" and "GBSU" experiments, it only shows the best results. We can make the following

observations:

1. The effectiveness of model adaptation depends to some extent on the domains involved. We get clearly better results for the News-Commentary domain. While we see similar trends in the JRC domain, the model adaptation is less effective.

2. "Baseline 1" is easily beat by most of the experiments in the News-Commentary domain. For JRC, "Baseline 1" is comparatively harder to beat. The best performing model adaptation experiments, still perform comfortably better than this baseline.

3. "Baseline 2" is harder to beat in both domains. In fact, though some of the experiments give better scores in the News-Commentary domain, they do not beat the scores by more than the 95% confidence interval. In the JRC domain, "Baseline 2" is the best performing experiment for the BLEU and WER metrics.

4. The "Keep All" experiment performs moderately well. It beats "Baseline 1" in both domains, but falls short of "Baseline 2".

5. In the seen/unseen group of experiments, the "Seen" experiment performs the best. This indicates that reinforcing already seen phrases has a positive impact on model adaptation. In the News-commentary domain, the score obtained is quite close to the best and beats "Baseline 2". Similarly, in the JRC domain, the score obtained by the "Seen" experiment is close to the best obtained. Unfortunately, here "Baseline 2" is not beaten, but it comes close (within the 95% confidence interval).

6. The "Unseen" experiment does not match up to the performance of the "Seen" experiment, indicating that reinforcing seen phrases is more important than using unseen phrases from the generated data.

7. The "Seen+Unseen" experiment performs worse than both the "Seen" and the "Unseen" experiments. The reason for this is that MERT does not find the global optimum. For example, setting the weights on the "Unseen" model to zero, would make the model identical to the log-linear model in the "Seen" experiment.

8. The "Keep All *" experiment performs moderately well in both domains. In both cases, it beats "Baseline 1" comfortably but falls shy of "Baseline 2". The log-linear weight learned by MERT for the "seen" feature is much higher than that for the "unseen" feature. This is more evidence of the importance of "seen" phrases.

9. The "Good" experiments with a high thresholds give the best scores for all the experiments. This is also consistent across the domains. This reiterates the importance of *filtering* as noted in (Ueffing, Haffari, and Sarkar, 2007), but cast in the light of model adaptation. In fact, the best performance is obtained when we select the top 1% of the sentences generated. Another important observation is that the technique of filtering using "back-translations" is quite effective and is a viable alternative to the options studied in (Ueffing, Haffari, and Sarkar, 2007).

10. The "Good+Bad" and "GBSU" experiments do not perform very well in either domain. They only marginally beat "Baseline 1" in the News-Commentary domain. The "Good+Bad" experiments are especially unstable in the JRC domain as can be seen in Table 3.5. The reason for this is unclear. The results of the "GBSU" experiments, on the other hand, have little variation with the threshold values.

    A reason for the poor performance could again be the inability of MERT to find the global optimum. The "Good+Bad" experiments would be identical to "Good" experiments with the log-linear weights for the "bad" phrase translation models set to zero.

3.3.3 Are we overfitting?

In Algorithm 3.1, we tune the newly obtained log-linear model on the same development set $\langle S_{dev}, T_{dev} \rangle$ that was utilized to add features N_{dev} into the log-linear model. This is done so that the way features are used during testing corresponds to the way features are trained.

It is possible that during tuning, some overfitting occurs as MERT may overestimate the utility of the in-domain phrases. This could have an adverse effect on the model's performance on the test set.

A simple way to check this is to tune the new log-linear model $M^{dev}_{S \to T}$ on a different development set. We perform some of the experiments to gauge the effect of overfitting. The results are shown in tables 3.9 and 3.10. Comparing these to the results in tables 3.7 and 3.8, we see that we get mixed results.

Keep All performs better in News-Commentary when only one dev set is used. However, in the JRC domain this is reversed. Here, using separate a dev set for tuning gives better results.

Seen experiments perform marginally worse when we use a separate dev set for tuning.

Experiment	BLEU[%] ±0.9%	WER[%] ±1.1%	PER[%] ±1.1%
Baseline 1	23.53	63.39	50.17
Baseline 2	24.81	59.36	46.37
Keep All	23.74	61.34	47.87
Seen	25.07	59.50	46.35
Unseen	***25.11***	***59.26***	***46.08***
Seen+Unseen	23.67	60.47	46.85

Table 3.9: Results for *News-Commentary* domain when tuned on a different dev set. The numbers in ***bold slanted*** font indicate the best performing experiments.

This indicates that there might be some overfitting happening for the seen phrases. In the end, this overfitting is beneficial because we get a higher performance on the test set (as we transfer the weights to the features learnt from the test set itself). It may be the case that in a non-transductive setting, where we do not learn features from the test set, the weights learned here would be less than optimal.

For the *Unseen* experiments, we again get mixed results. Using a separate dev set for tuning gives better results for the News-Commentary domain but worse results for JRC domain. Surprisingly for News-Commentary, this performance boost lets the *Unseen* experiments beat the *Seen* experiments.

Finally, for the *Seen+Unseen* experiments, we get worse results in both domains when using a separate dev set for tuning.

From the results, we can observe that although we get many mixed results, more experiments perform worse when using a separate dev set for tuning. This indicates that using the same dev set for learning new features and tuning is quite effective in a transductive setting. We don't observe any consistent negative effects associated with overfitting in the sample experiments shown in this Section. In fact, in most cases compensating for this by using a separate dev set leads to worse results.

3.3.4 Examples

Tables 3.11 and 3.12 contain hand-picked example sentences and their reference translations in the News-Commentary and JRC domains respectively. We show the reference translations, baseline outputs and the outputs from the best performing model adaptation experiments. In the News-Commentary domain, we can see the clear improvement over

Experiment	BLEU[%] ±0.9%	WER[%] ±0.85%	PER[%] ±0.7%
Baseline 1	39.39	49.47	34.16
Baseline 2	***41.39***	***46.96***	***32.99***
Keep All	39.76	47.97	33.52
Seen	40.76	47.41	33.31
Unseen	38.73	49.86	34.75
Seen+Unseen	38.57	49.69	34.27

Table 3.10: Results for *JRC* domain when tuned on a different dev set. The numbers in ***bold slanted*** font indicate the best performing experiments.

"Baseline 1".

The examples in the JRC domains show similar trends. The output of "Baseline 1" seems incomplete and is the worst of the lot. The rest of the shown experiments display sensible output and "Baseline 2" appears to have the best output, although the differences are minor.

3.4 Conclusions

We see several interesting results in the domain adaptation experiments.

We used just a small development set of in-domain data. The target language sentences were only used by MERT during tuning, and were not used in the log-linear model itself. In spite of the frugal use of in-domain data, we are able to boost the performance of the log-linear model by about 2 BLEU points (from "Baseline 1"), in both domains studied.

To the best of our knowledge, this is the first attempt at using "back-translations" to improve model adaptation performance. In the framework provided by (Ueffing, Haffari, and Sarkar, 2007), these provide a strong contender for filtering the generated data. In fact, the best performing model adaptation experiments use only about 1% of the generated data. This filtering was performed using back-translations.

We provide several pieces of evidence indicating the importance of boosting "seen" phrases during model adaptation. The "Seen" experiments are consistently among the best performers for the domains studied. High quality "back-translations" have a greater percentage of "seen" phrases. Also, since the "Good" experiments with high thresholds contain a lot more "seen" phrases, their consistent high performance is another indication of the importance of these "seen" phrases.

Splitting translation models into smaller components has its drawbacks. MERT is unable to find the global optimum which results in poor performance in some of the experiments. This means that adding relatively bad data as a separate component in the log-linear model, has an adverse effect on overall performance, as seen in the experiments.

Original sentence in French	bien que cela puisse être vrai , la maturité possède son propre inconvénient .
Reference Translation in English	while this may be true , *there is a downside to maturity* .
Baseline 1	although it may be true , *has its own maturity* .
Baseline 2	although it may be true , *the maturity has its own disadvantage* .
BLEU > 90	although it may be true , *maturity has his own disadvantage* .
Seen	although it may be true , *the maturity has its own disadvantage* .
Original sentence in French	les partisans d' un déficit ininterrompu le dépeignent souvent comme un mal nécessaire pour rétablir une économie dans laquelle la confiance s' effrite .
Reference Translation in English	those who advocate running deficits often portray them as necessary to fix an economy in which confidence is draining away .
Baseline 1	the supporters of a continuous deficit *makes* the often as a necessary evil to *restore* an economy *where the confidence crumbles* .
Baseline 2	the supporters of a continuous deficit *from* the often as a necessary evil to *re-establish* an economy *where the confidence crumbles* .
BLEU > 90	the supporters of a continuous deficit *from* the often as a necessary evil to *restore* an economy *where confidence crumbles* .
Seen	the supporters of a continuing deficit *from* the often as a necessary evil to *restore* an economy *where the confidence crumbles* .
Original sentence in French	le président n' avait pas pris une décision politique : il avait accordé une faveur personnelle .
Reference Translation in English	the president had not made a political decision ; he had bestowed a personal favor .
Baseline 1	president had not taken a political decision : *it had given a personal support* .
Baseline 2	president had not taken a political decision : *it was given a personal favour* .
BLEU > 90	president had not taken a political decision : *it had given a personal favour* .
Seen	president had not taken a political decision : *it had given a favour* .

Table 3.11: Example outputs from different experiments in the News-Commentary domain. The differences are *highlighted*. In addition to the baselines, we show the output from the best performing adaptation experiments.

Original sentence in French	l' application de ces critéres conduit à la répartition indicative des ressources globales figurant à l' annexe i.
Reference Translation in English	the indicative allocation of the total resources resulting from the application of those criteria is set out in annex i.
Baseline 1	the application of these criteria *, which led to the indicative distribution of global resources* in annex i.
Baseline 2	the application of these criteria *led to the indicative distribution of global resources* in annex i.
WER < 10	the application of these criteria *led to the indicative distribution of overall resources contained* in annex i.
Seen	the application of these criteria *led to the indicative distribution of total funding contained* in annex i.
Original sentence in French	4 . les informations communiquées par les différents états membres sont traitées de manière confidentielle .
Reference Translation in English	4 . the information communicated by individual member states shall be treated as confidential . ' ;
Baseline 1	4 . information provided by the *different* member states are treated as confidential . '
Baseline 2	4 . *the* information provided by the *different* member states are treated as confidential . '
WER < 10	4 *; the* information provided by the *various* member states are treated as confidential . '
Seen	4 . *the* information provided by the *different* member states are treated as confidential . '
Original sentence in French	2 . le notifiant fournit , à la demande des autorités compétentes concernées , une ou plusieurs traductions agréées dans une langue acceptable pour elles .
Reference Translation in English	2 . the notifier shall provide the competent authorities concerned with authorised translation (s) into a language which is acceptable to them , should they so request .
Baseline 1	*. the notifiant 2* provides , at the request of the authorities concerned , one or several translations authorised in a language *that is* acceptable to them .
Baseline 2	*2 . the notifiant* provides , at the request of the authorities concerned , one or several translations authorised in a language acceptable to them .
WER < 10	*2 . the notifiant* provides , at the request of the *competent* authorities concerned , one or several translations authorised in a language acceptable to them .
Seen	*2 . the notifiant* provides , at the request of the *competent* authorities concerned , one or several translations authorised in a language acceptable to them .

Table 3.12: Example outputs from different experiments in the JRC domain. The differences are *highlighted*. In addition to the baselines, we show the output from the best performing adaptation experiments.

Chapter 4

Unsupervised MERT

4.1 Background

In Chapter 1, we discussed the architecture of log-linear models in Statistical Machine Translation (SMT). We saw how log-linear models are easy to adapt, by adding in additional features. This property was utilized in Chapter 3 to adapt SMT systems trained in one domain to perform better in a different domain.

In a log-linear model, the best translation is determined from Equation 1.6, which is repeated here.

$$\hat{t}_1^I = \underset{t_1^I}{\operatorname{argmax}} \sum_{m=1}^{M} \lambda_m h_m(t_1^I, s_1^J) \tag{4.1}$$

$h_m(t_1^I, s_1^J), m = 1, \cdots, M$ are feature functions and λ_m are the corresponding weights.

The log-linear model parameters λ_1^M are clearly critical to the performance of this model and need to be tuned. As discussed in Chapter 1, this is done using the Minimum Error Rate Training (MERT) procedure described in (Och, 2003). It is an efficient line optimization method, which works as follows. We are given a development set of source sentences and their reference translations in the target language. First, we assign some values to λ_1^M to get a working model. In each iteration of the algorithm, we find the n-best candidate translations using the log-linear model from the previous iteration. These candidate translations are scored based on the reference translations. Then λ_1^M is improved in the direction of the best candidate translation and we start a new iteration. To do this, we need an efficient way of computing the denominator of the log-linear model (shown in Equation 1.5), which is a sum over all possible translations. (Och, 2003) provides an approximation for doing this using

n-best lists. This process continues until convergence and we get tuned λ_1^M.

Now, MERT is a *supervised* learning procedure and requires a development set of sentences and their reference translations. We improve the parameters λ_1^M by comparing our model translations to the provided references. The results of this tuning are more reliable if the development set is larger. However, in real world situations, such development sets are not available in some domains. While there is a large amount of parallel data (i.e. source language sentences and their corresponding reference translations) in some domains, it is non-existent in others. This makes tuning the log-linear model in those domains impossible.

In this chapter, we explore an *unsupervised* MERT procedure. In particular, this method does not require parallel data to tune the log-linear model. Instead, we require a model that translates in the reverse direction, from the target language to the source language, *possibly in a different domain*. In addition, we would use some monolingual source language data which is in the required domain. Note that we do not require parallel data. While parallel data may be rare, there is usually a sufficient amount of monolingual data in any given domain. Thus we would be able to use this technique to train the log-linear models in domains that lack parallel data.

4.1.1 Related Work

There are existing techniques to do unsupervised learning in log-linear models, though none have yet been explored for SMT. For example, the EM algorithm can be used to tune log-linear models, by iteratively improving estimates for the parameters λ_1^M. Unfortunately, EM has some drawbacks. It is especially difficult to efficiently compute the exponential (or possibly infinite) sum in the denominator used for normalization of log-likelihood of the log-linear model. Some approaches have been suggested to overcome this shortcoming.

In (Smith and Eisner, 2005), a contrastive estimation technique that exploits negative evidence in training data is used to tune log-linear models used in sequence labeling problems like part-of-speech tagging. Another example from the part-of-speech tagging literature is in (Haghighi and Klein, 2006) who describe a technique for learning Markov random fields using EM. Here they assume certain bounds on sentence length which make it easier to compute the denominator. In (Johnson et al., 1999), a pseudo-likelihood estimator was suggested for learning log-linear models using EM, and this technique was applied to stochastic parsing.

The unsupervised MERT procedure is different from the above approaches in the sense that it is more "discriminative" than these approaches. While the previous approaches

directly try to maximize the likelihood, we try to minimize the "errors" in translation by looking at back-translations as explained in Section 4.2.

4.2 Description

We denote the source language by S and the target language by T. Suppose we have a reverse translation system $M_{T\rightarrow S}$ that translates from T to S. $M_{T\rightarrow S}$ is tuned using supervised MERT on out-of-domain data. We would like to learn the log-linear weights for the forward translation system $M_{S\rightarrow T}$. Traditionally, the weights are tuned on a dev set consisting of source language sentences and their reference translations in the target language. We present a way to do learn these weights in an unsupervised setting, where we only use sentences in the source language.

The basic MERT procedure to learn the weights in each iteration is described in (Och, 2003). However, since we do not have the reference translations we make an adjustment to the training procedure as described in 4.1. Initially we assign random weights to the log-linear model and obtain $M_{S\rightarrow T}$. In each iteration we use the weights obtained in the previous iteration, to translate the source sentences S_{test} to get n-best output. Now, for each sentence in S_{test}, we choose one of the n-best output as the reference translation. Thus in each iteration, we have a different set of reference translations, by choosing the best sentences from the n-best output.

In each iteration, we choose the best translations as follows. We reverse translate the entire n-best output using $M_{T\rightarrow S}$ to get back-translations which are in the source language. We now compare these sentences with the original source sentences S_{test}. The idea is that back-translations that are close to the original sentences in S_{test} are likely to have good intermediate translations in the target language. We select the translation that gives the back-translation with the best BLEU score.

Algorithm 4.1: Reverse MERT

Input: Existing Decoder $M_{S \to T}$ that translates from S to T (forward direction)

Input: Existing Decoder $M_{T \to S}$ that translates from T to S (reverse direction)

Input: Source sentences of test dataset S_{test}

Output: Tuned decoder $M^*_{S \to T}$ for S_{test}

$i \leftarrow 1$

$M^1_{S \to T} \leftarrow M_{S \to T}$

while *true* **do**

  Translate S_{test} using $M^i_{S \to T}$ to get n-best output T^i_{test}

  Translate T^i_{test} using $M_{T \to S}$ to get back-translations B^i_{test}

  Compare B^i_{test} with S_{test} to get the BLEU score of each sentence in B^i_{test}

  $T^i_{ref} \leftarrow \emptyset$

  foreach $s_j \in S_{test}$ **do**

    Add to T^i_{ref} the output sentence in T^i_{test} corresponding to s_j for which the score obtained by the back-translation in B^i_{test} is maximum

  Compute new model $M^{i+1}_{S \to T}$ using the MERT procedure described in (Och, 2003) using T^i_{ref} as the reference set for this iteration

  if *weights in* $M^{i+1}_{S \to T}$ *have not changed significantly* **then**

    break

  $i \leftarrow i + 1$

$M^*_{S \to T} \leftarrow M^i_{S \to T}$

return $M^*_{S \to T}$

Thus in each iteration, we would have a different reference set of sentences. We continue iterating until there is no significant change in the weights from the previous iteration, indicating that we have attained a local optimum. The weights found in the last iteration are returned as the weights of the tuned log-linear model.

4.2.1 Convergence

As discussed above, we have a different set of reference sentences in each iteration of unsupervised MERT. The global optimum in each iteration would be different. It is therefore possible that the algorithm could get stuck in a loop, where we get the same sequence of

reference sentences over and over again. It would not terminate because at each step the model parameters change significantly. Also, while the supervised MERT procedure can get stuck in a local optimum, this effect may be worse due to the lack of proper references in unsupervised MERT.

These are serious drawbacks to this method. In this work, we have not theoretically analysed the convergence of this approach and cannot make strong statements about its convergence. However, in all the experiments performed using this algorithm, we found that the algorithm converges. In fact, the supervised version of MERT usually takes more iterations to converge than the unsupervised variant explored in this chapter.

4.3 Experiments

4.3.1 Setup

We use the same data that was used for the experiments in Chapter 3 and discussed in Section 3.3.1. We repeat some of the information here. We use data from the following domains.

1. Europarl, (Koehn, 2005), mainly consists of European parliamentary proceedings which are legislative in nature. This consists of a large amount of parallel data and is used as the out-of-domain data. We used the Europarl corpus provided by the shared task in the *Third Workshop on Statistical Machine Translation* (WMT 2008).

2. News-Commentary is less formal and consists of commentary on news articles. This is a smaller data set and was also obtained from the WMT 2008 shared task.

3. JRC-Acquis, (Steinberger et al., 2006) is a legal corpus that consists of aligned documents of the European union. This was not part of the WMT 2008 shared task and was obtained separately.

We use the Moses decoder (Koehn et al., 2007) as the phrase based SMT system. Please refer to Appendix B for more information on Moses. We performed the experiments with French as the source language and English as the target language.

The Europarl corpus from WMT 2008 was used to train the phrase tables for both forward and reverse translation directions. In addition, the Europarl corpus is also used to

Run	BLEU[±0.9%]	WER[±1.1%]	PER[±1.1%]
1	26.91	56.15	44.46
2	31.74	***55.87***	***40.88***
3	***31.89***	56.28	40.93

Table 4.1: Performance after each iteration for Europarl. The numbers in ***bold slanted*** font indicate the best performance for that metric.

Run	BLEU[±0.9%]	WER[±1.1%]	PER[±1.1%]
1	20.53	61.38	49.07
2	22.97	60.69	47.57
3	***23.93***	***60.62***	***47.21***

Table 4.2: Performance after each iteration for News-Commentary. The numbers in ***bold slanted*** font indicate the best performance for that metric.

create a 5-gram language model with KN-discounting which is then used in the translations. The log-linear model for reverse translation is then tuned on a Europarl dev set (using supervised MERT) from the WMT 2008 data, to give $M_{T \to S}$.

We perform three unsupervised MERT runs for each of the three domains, Europarl, News-Commentary and JRC. In all experiments, the reverse translator is tuned on a Europarl development set. Thus in the first experiment, we are tuning the model in the same domain as $M_{T \to S}$ (both Europarl). In the second and third experiment, we are tuning on data in the News-Commentary and JRC domain respectively, both of which are different from the domain used by $M_{T \to S}$. Thus, these experiments are model adaptation experiments.

In each iteration, we generate 100-best lists of translations and then use $M_{T \to S}$ to find out the reference sentences as described in Algorithm 4.1.

Our baseline for Europarl is the traditional supervised MERT using a dev set from the Europarl domain. For News-Commentary and JRC, we have two baselines. The first is similar to that in Europarl; it is just a supervised MERT run using a dev set from the News-Commentary domain. For the second baseline, we tune the out-of-domain model on in-domain data using *supervised* MERT and test its performance. This baseline uses in-domain parallel data, while unsupervised MERT only uses in-domain monolingual data. Therefore we expect this baseline to outperform our unsupervised technique.

Run	BLEU[±0.9%]	WER[±0.85%]	PER[±0.7%]
1	33.16	50.57	39.05
2	37.88	48.46	35.05
3	38.95	47.98	34.32
4	39.53	47.87	33.94
5	39.97	***47.83***	33.72
6	40.32	***47.83***	***33.53***
7	***40.47***	47.88	33.54
8	40.15	48.18	33.69
9	40.07	48.34	33.77
10	39.95	48.50	33.87
11	39.85	48.68	34.02
12	39.54	48.91	34.21

Table 4.3: Performance after each iteration for JRC. The numbers in ***bold slanted*** font indicate the best performance for that metric.

Experiment	BLEU[±0.9%]	WER[±1.1%]	PER[±1.1%]
Europarl (sup)	32.89	56.36	40.99
Europarl (unsup)	31.89	56.28	40.93
NC (sup on Europarl)	23.53	63.39	50.18
NC (sup on NC)	24.81	59.36	46.37
NC (unsup)	23.93	60.62	47.20
JRC (sup on Europarl)	39.39	49.47	34.16
JRC (sup on JRC)	41.39	46.96	32.99
JRC (unsup)	39.54	48.91	34.21

Table 4.4: Supervised vs unsupervised MERT on the three domains. NC stands for News-Commentary. The unsupervised MERT experiment performs better than supervised MERT on out of domain data (sup on Europarl).

4.3.2 Results

Tables 4.1, 4.2 and 4.3 show the performance of the model in each iteration of the unsupervised MERT procedure, for the Europarl, News-Commentary and JRC experiments respectively. An interesting observation is that unsupervised MERT converges in just three iterations in the first two domains. For JRC however, it requires twelve iterations. In supervised MERT, we found that usually more than ten iterations are required. This does not imply that unsupervised MERT could be faster though, because of the additional overhead of back-translating the n-best translations.

The final results are shown in Table 4.4. For the Europarl domain, we see that the unsupervised MERT procedure produces a model that performs very similarly to the supervised MERT procedure. Of course, this is done using the reverse translation model $M_{T\rightarrow S}$ which has been tuned using supervised MERT and acts as a kind of oracle.

For our out-of-domain data, we first note the difference in the baselines. When using a model that has been tuned (using supervised MERT) on a dev set in Europarl domain, the performance is significantly less than when it is tuned on an in-domain dev set. This is expected due to the differences between the domains.

In the News-Commentary domain, we observe that the model trained using unsupervised MERT for News-Commentary lies between the two baselines for that domain. It clearly does not perform as well as the in-domain supervised MERT procedure, as can be expected. It can be drawn from the Europarl experiments, that using a reverse translator ($M_{T\rightarrow S}$) that is tuned in the News-Commentary domain might give better results.

However, it does do significantly better than the model trained using out-of-domain supervised MERT. This indicates that there is some domain adaptation going on. This is useful because we are using no knowledge (except the source sentences) from the News-Commentary domain; even the reverse translation model ($M_{T\rightarrow S}$) has been tuned on the Europarl domain.

In the JRC domain, we find that the results are not as convincing as in the case of News-Commentary. Firstly, from table 4.3, we see that the optimum values are obtained at iteration 7 (for BLEU). After this, the performance of the model decreases. This happens because we are maximising the BLEU score of the back-translations and not the BLEU score of the intermediate translations with their corresponding references. Fortunately, this does not happen for the other two domains.

The performance of the final model is better than the first baseline, but the difference is not statistically significant. In fact, unsupervised MERT is marginally outperformed by the first baseline in the PER metric. This could be attributed to two factors. First, the JRC domain itself seems to be more challenging to adapt to. This was also observed in Chapter 3. Secondly, unlike in News-Commentary, the unsupervised MERT procedure did not converge at the optimum model.

Table 4.5 shows a few hand-picked examples from the News-Commentary domain. We show the original French sentence, the reference English translation and the translations obtained by the different models in the News-Commentary domain. We can see that the

model trained by the unsupervised MERT procedure provides a more accurate translation than that found by using the model trained on out-of-domain (Europarl) corpus. Using supervised MERT to train the model on in-domain data, gives a better translation too.

From the results, we see that there is clearly some merit in the idea of unsupervised MERT. If $M_{T\to S}$ has been tuned in the same domain as the test set, we get similar results to using supervised MERT. In addition, if there is very little data in the domain of interest, unsupervised MERT could provide an effective way to do domain adaptation and obtain better performance.

4.4 Conclusions

In this chapter, we have introduced, for the first time, an unsupervised technique to tune log-linear model weights. This unsupervised variant of MERT relies on a log-linear SMT system in the reverse direction, $M_{T\to S}$. Although we do not provide a thorough analysis of the convergence of unsupervised MERT, the empirical results indicate that it converges quickly. This could be a coincidence and needs to be further investigated.

We find that unsupervised MERT performs almost as well as the supervised version. Within the same domain, unsupervised MERT comes within 1 BLEU point of the supervised version, while the WER and PER scores are almost identical.

The performance of unsupervised MERT in the model adaptation experiments varies with the domain. JRC seems to be more challenging than News-Commentary. We find statistically significant improvements of about 3% in the WER and PER scores, and a 0.4 BLEU point improvement from the baseline for News-Commentary. For JRC, we get a 0.15 BLEU point improvement and 0.56% WER improvement, while the PER score drops by 0.05 %. None of the changes in JRC are statistically significant.

Within the same domain, the usefulness of unsupervised MERT is quite minimal. If one can train a reverse translator $M_{T\to S}$ in the domain using supervised MERT, then a forward translator $M_{S\to T}$ could just as easily be trained. The experiment indicating the performance of unsupervised MERT in the same domain is merely useful as a sanity test of the technique.

In the problem of domain adaptation though, unsupervised MERT could be useful in cases where parallel data is unavailable; even the relatively small amount needed for a development set. In such domains, the most significant results are obtained by constructing

in-domain language models (Koehn and Schroeder, 2007). Using unsupervised MERT to tune the log-linear model to the new domain without parallel data could provide some additional improvement as noted by the experiments. This is contingent on the assumption that unsupervised MERT can either be proven to converge effectively, or a workaround for this is found.

Original sentence in French	bien que cela puisse être vrai , la maturité possède son propre inconvénient .
Reference Translation in English	while this may be true , there is a downside to maturity .
NC (sup on Europarl)	although it may be true , *has its own maturity* .
NC (sup on NC)	although it may be true , *the maturity has its own disadvantage* .
NC (unsup)	although this may be true , *the maturity has its own problems* .
Original sentence in French	les visiteurs nocturnes de ebeneezer scrooge furent capables de le convaincre qu' il avait tort et de lui ouvrir les yeux sur ses erreurs .
Reference Translation in English	ebeneezer scrooge 's nocturnal visitors were able to convince him of the errors of his ways .
NC (sup on Europarl)	*the visitors ebeneezer* night scrooge were able to *persuade the council* that it was wrong and to *open our eyes* on its own mistakes .
NC (sup on NC)	*visitors night ebeneezer* scrooge were able to *convince him* that it was wrong and *open our eyes* on its mistakes .
NC (unsup)	*visitors night ebeneezer* scrooge were able to *persuade the council* that it was wrong and to *look at* its own mistakes .
Original sentence in French	les politiciens sont prêts à se donner beaucoup de mal pour satisfaire leurs partisans et remporter les élections .
Reference Translation in English	politicians are willing to go to great lengths to please their supporters and win elections .
NC (sup on Europarl)	politicians are prepared to *provide itself with a great deal of trouble* to satisfy their supporters and win the elections .
NC (sup on NC)	politicians are prepared to *much difficulty* to satisfy their supporters and win the elections .
NC (unsup)	politicians are prepared to *give a lot of trouble* to satisfy their supporters and win the elections .

Table 4.5: Example translations in News-Commentary domain for each of the experiments. NC stands for News-Commentary.

Chapter 5

Conclusions and Future Work

5.1 Results and Contributions in Model Adaptation

We obtained interesting results in both the semi-supervised and unsupervised versions of model adaptation experiments tried in this thesis. In all our experiments, we utilize very limited in-domain data. Thus our techniques have great practical use.

In this thesis, we have studied in good extent, the utility of "back-translations" in model adaptation, in both the semi-supervised and unsupervised settings. To the best of our knowledge, this is the first such study in the field.

In the semi-supervised approach, described in Chapter 3, we gained approximately 2 BLEU points over the un-adapted model, in both domains. We have provided an effective technique for *filtering* generated data using "back-translations", in the framework provided by (Ueffing, Haffari, and Sarkar, 2007). We find the best results by using about just 1% of the data.

We have also studied various combinations of features that could be learned to adapt the translation model. We find that boosting "seen" phrases in the generated data is one of the better ways to perform model adaptation with limited data.

In Chapter 4, we introduce a novel variant of MERT which tunes the log-linear model, in an *unsupervised* fashion. Thus, we do not need reference translations to be able to tune the log-linear model. The performance of this variant, while understandable lesser, is very close to that of supervised MERT with in-domain data.

Model adaptation is usually done by adapting the language model or translation model in some way. For the first time, we successfully adapt a log-linear model *by just tuning the*

weights in an unsupervised way. This technique, though a departure from tradition, gives statistically significant improvements in SMT performance in the target domain.

As noted in Section 4.4, unsupervised MERT is only practical for model adaptation to domains that have little or no parallel data. For domains in which the reverse translation model required for unsupervised MERT can be constructed, the forward translation model can also be trivially derived. Another limitation that we reiterate here, is that we provide no convergence guarantee for unsupervised MERT. The quick convergence observed in experiments may just be a coincidence.

5.2 Directions for future work

Due to the limited time available, not all avenues were explored in this work. Some possible directions for future work are described below.

1. One could try using the results learned in Chapter 3 to *non-transductive* model adaptation. By this we mean that instead of discarding the features N_{dev} learned during training and replacing them with features learned from the test set, N_{test}, we just use the model that was optimized on the dev set $M_{S\rightarrow T}^{dev}$ directly on the test set. Different approaches are possible depending on the amount of in-domain data available. We assume that there is always sufficient out-of-domain data.

    - If a large amount of parallel in-domain data is available, we can just learn a translation model on this data. We can alter this model based on the "seen" phrases, either by adding it as a separate model or by boosting their probabilities in some way. Here filtering the model based on back-translations does not make much sense, as all the data is going to be good.
    - If parallel data is unavailable but we have some in-domain monolingual data in the source language, then we can perform experiments with the filtering and feature extraction techniques that performed well in a non-transductive setting. If the amount of monolingual data is large, then we can limit ourselves to generate only 1-best translations, as in (Bertoldi and Federico, 2009).

2. A theoretical analysis of the convergence of unsupervised MERT would be invaluable in assessing the utility of this technique. Current experiments show quick convergence,

but this could be a coincidence.

3. It would be interesting to test the techniques in this thesis to other language pairs and more domains, as this would provide more exhaustive evidence of their utility.

4. The mixture model approach tried in (Foster and Kuhn, 2007) in the unsupervised setting, could be combined with unsupervised MERT. Since the techniques focus on different parts of the SMT system, combining them may result in a system that has the strengths of both, and performs better than either. Since this hybrid training method would not require any parallel data at all, it could be used to improve SMT performance in virtually any domain.

Appendix A

Example: Learning Word Alignments

The process of learning word alignments can be made clearer with a simple example, inspired by (Knight, 1999a). Please note that for purposes of demonstration, we only describe the technique for learning Model 1 alignment. The SMT systems used in this work use alignments generated by Giza++, which generates Model 6 alignments (Och and Ney, 2003). Please refer to section 1.3.3 for details. Appendix B describes Giza++.

Consider that we have only the following two sentence pairs in our corpus. The source language is French and the target language is English.

1. ⟨ maison bleue, blue house ⟩
2. ⟨ maison, house ⟩

To keep the example simple, we ignore the alignments from the $NULL$ source word and make the further assumption that each source word can generate only one target word. The possible alignments are shown in Figure A.1. From the figure, we can see that we would like a_2 to get a higher probability than a_1 as it makes more sense. In addition, it should be much more likely that "bleue" translates to "blue" than to "maison". This knowledge can be learned from the EM algorithm which proceeds as follows.

1. *Initialization*: We first set up the word translation probabilities to be uniformly distributed. Since there are only two words in the lexicon and all translations are equally

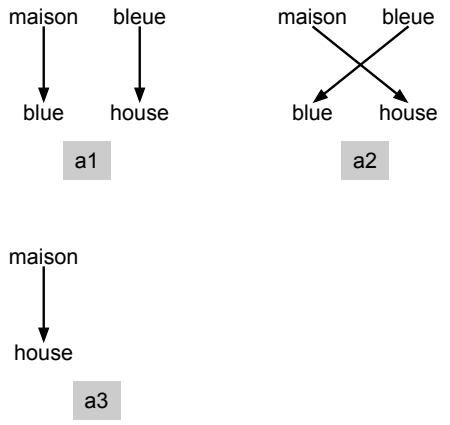

Figure A.1: Possible alignments for an example corpus consisting of just two sentences. We omit alignments that contain the *NULL* word, and assume that each source word can be aligned to only one target word. For the first sentence, we can see that a_2 is clearly a better choice than a_1.

likely, we have

$$\begin{aligned} Pr(\text{blue}|\text{bleue}) &= \frac{1}{2} \\ Pr(\text{blue}|\text{maison}) &= \frac{1}{2} \\ Pr(\text{house}|\text{bleue}) &= \frac{1}{2} \\ Pr(\text{house}|\text{maison}) &= \frac{1}{2} \end{aligned}$$

2. *Compute alignment probabilities*: In Figure A.1, we can compute the probability $Pr(a, t_1^I|s_1^J)$ for each of the alignments as follows.

$$\begin{aligned} Pr(a_1, t_1^I|s_1^J) &= \frac{1}{2} * \frac{1}{2} = \frac{1}{4} \\ Pr(a_2, t_1^I|s_1^J) &= \frac{1}{2} * \frac{1}{2} = \frac{1}{4} \\ Pr(a_3, t_1^I|s_1^J) &= \frac{1}{2} \end{aligned}$$

3. *Normalize alignment probabilities*: We now normalize $Pr(a, t_1^I|s_1^J)$ to get the $Pr(a|t_1^I, s_1^J)$ values.

$$
\begin{aligned}
Pr(a_1|t_1^I, s_1^J) &= \frac{1}{4}/\frac{2}{4} = \frac{1}{2} \\
Pr(a_2|t_1^I, s_1^J) &= \frac{1}{4}/\frac{2}{4} = \frac{1}{2} \\
Pr(a_3|t_1^I, s_1^J) &= \frac{1}{4}/\frac{1}{2} = 1
\end{aligned}
$$

Since there is only one alignment possible in the second sentence, $Pr(a_3|t_1^I, s_1^J)$ will always be equal to 1.

4. *Revise the word translation probabilities*: We do this by first obtained the fractional counts for all the word translation probabilities. From Figure A.1 and the alignment probabilities $Pr(a|t_1^I, s_1^J)$ calculated in the previous step, we get the fractional counts as follows

$$
\begin{aligned}
count(\text{blue}, \text{bleue}) &= \frac{1}{2} \\
count(\text{blue}, \text{maison}) &= \frac{1}{2} \\
count(\text{house}, \text{bleue}) &= \frac{1}{2} \\
count(\text{house}, \text{maison}) &= \frac{1}{2} + 1 = \frac{3}{2}
\end{aligned}
$$

We now normalize these counts to get the revised word translation probabilities.

$$
\begin{aligned}
Pr(\text{blue}|\text{bleue}) &= \frac{1}{2}/1 = \frac{1}{2} \\
Pr(\text{blue}|\text{maison}) &= \frac{1}{2}/\frac{4}{2} = \frac{1}{4} \\
Pr(\text{house}|\text{bleue}) &= \frac{1}{2}/1 = \frac{1}{2} \\
Pr(\text{house}|\text{maison}) &= \frac{3}{2}/\frac{4}{2} = \frac{3}{4}
\end{aligned}
$$

5. *Repeat until convergence*: We can see that in just one iteration, the translation probabilities have improved. We repeat steps 2 - 4 until the probabilities converge. Eventually, we get the following word translation probabilities.

$$
\begin{aligned}
Pr(\text{blue}|\text{bleue}) &= 0.9999 \\
Pr(\text{blue}|\text{maison}) &= 0.0001 \\
Pr(\text{house}|\text{bleue}) &= 0.0001 \\
Pr(\text{house}|\text{maison}) &= 0.9999
\end{aligned}
$$

Thus we can see how EM learned the correct word translations in this example. Using large parallel corpora, we can learn the word translations probabilities in this fashion.

Appendix B

Software

During the course of the experiments in this thesis, we used the open source software listed in this Appendix.

B.1 SRILM

We used the SRILM language modeling toolkit (Stolcke, 2002) to build our language models. It does this by gathering statistics from monolingual corpora. It provides tools to use a number of smoothing techniques, including modified Kneser-Ney discounting used in this work. SRILM also supports the language model interpolation techniques discussed in Chapter 2, which aid in model adaptation.

B.2 Giza++

Giza++ (Och and Ney, 2003) (Och, 2000) is a tool that creates translation models from aligned bilingual corpora. It learns word and phrase alignments as discussed in Section 1.3.3. It then extracts the aligned phrases and computes the phrase translation probabilities and lexical weights. The output is a *phrase table* that is a translation model for the given parallel corpus.

B.3 Moses

Moses (Koehn et al., 2007) is a phrase based Statistical Machine Translation (SMT) system. It uses log-linear models discussed in Chapter 1 and also supports more advanced options like factored models. It allows the generation of n-best lists of translations and provides implementations for tuning the log-linear models using MERT. Several modifications were made to the scripts provided in this system during the course of this thesis.

In addition Moses also provides several scripts that assist in the training on phrase pairs using Giza++.

Bibliography

Baum, L.E. 1972. An inequality and associated maximization technique in statistical estimation for probabilistic functions of Markov processes. *Inequalities*, 3(1):1–8.

Berger, A.L., V.J.D. Pietra, and S.A.D. Pietra. 1996. A maximum entropy approach to natural language processing. *Computational linguistics*, 22(1):39–71.

Bertoldi, N. and M. Federico. 2009. Domain adaptation for statistical machine translation with monolingual resources. In *Proceedings of the Fourth Workshop on Statistical Machine Translation*, pages 182–189, Athens, Greece, March. Association for Computational Linguistics.

Brown, P.F., J. Cocke, S.A.D. Pietra, V.J.D. Pietra, F. Jelinek, J.D. Lafferty, R.L. Mercer, and P.S. Roossin. 1990. A statistical approach to machine translation. *Computational Linguistics*, 16(2):79–85.

Brown, P.F., V.J. Della Pietra, S.A. Della Pietra, and R.L. Mercer. 1993. The mathematics of statistical machine translation: Parameter estimation. *Computational linguistics*, 19(2):263–311.

Finch, A. and E. Sumita. 2008. Dynamic model interpolation for statistical machine translation. In *Proceedings of the Third Workshop on Statistical Machine Translation*, pages 208–215, Morristown, NJ, USA. Association for Computational Linguistics.

Foster, G. and R. Kuhn. 2007. Mixture-model adaptation for smt. In *StatMT '07: Proceedings of the Second Workshop on Statistical Machine Translation*, pages 128–135, Morristown, NJ, USA. Association for Computational Linguistics.

Goodman, J.T. 2001. A bit of progress in language modeling. *Computer Speech and Language*, 15:403–434.

Haghighi, A. and D. Klein. 2006. Prototype-driven learning for sequence models. In *Proceedings of the main conference on Human Language Technology Conference of the North American Chapter of the Association of Computational Linguistics*, pages 320–327. Association for Computational Linguistics Morristown, NJ, USA.

Hastie, T., R. Tibshirani, J. Friedman, and J. Franklin. 2005. The elements of statistical learning: data mining, inference and prediction. *The Mathematical Intelligencer*, 27(2):83–85.

Hildebrand, A.S., M. Eck, S. Vogel, and A. Waibel. 2005. Adaptation of the translation model for statistical machine translation based on information retrieval. In *Proceedings of EAMT*, volume 2005, pages 133–142.

Huang, X. 1990. A machine translation system for the target language inexpert. In *Proceedings of the 13th conference on Computational linguistics*, pages 364–367, Morristown, NJ, USA. Association for Computational Linguistics.

Johnson, M., S. Geman, S. Canon, Z. Chi, and S. Riezler. 1999. Estimators for stochastic unification-based grammars. In *Proceedings of the 37th annual meeting of the Association for Computational Linguistics on Computational Linguistics*, pages 535–541, Morristown, NJ, USA. Association for Computational Linguistics.

Kneser, R. and H. Ney. 1995. Improved backing-off for m-gram language modeling. In *International Conference on Acoustics, Speech and Signal Processing*, volume 1, pages 181–184.

Knight, K. 1999a. A statistical MT tutorial workbook. `http://www.isi.edu/natural-language/mt/wkbk.rtf`.

Knight, K. 1999b. Decoding complexity in word-replacement translation models. *Computational Linguistics*, 25(4):607–615.

Koehn, P. 2005. Europarl: A parallel corpus for statistical machine translation. In *MT summit*, volume 5.

Koehn, P., H. Hoang, A. Birch, C. Callison-Burch, M. Federico, N. Bertoldi, B. Cowan, W. Shen, C. Moran, R. Zens, et al. 2007. Moses: open source toolkit for statistical machine translation. In *ACL '07: Proceedings of the 45th Annual Meeting of the ACL on Interactive Poster and Demonstration Sessions*, pages 177–180, Morristown, NJ, USA. Association for Computational Linguistics.

Koehn, P., F.J. Och, and D. Marcu. 2003. Statistical phrase-based translation. In *NAACL '03: Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology*, pages 48–54, Morristown, NJ, USA. Association for Computational Linguistics.

Koehn, P. and J. Schroeder. 2007. Experiments in domain adaptation for statistical machine translation. In *StatMT '07: Proceedings of the Second Workshop on Statistical Machine Translation*, pages 224–227, Morristown, NJ, USA. Association for Computational Linguistics.

Nakov, P. and H. T. Ng. 2009. NUS at WMT09: Domain adaptation experiments for English-Spanish machine translation of news commentary text. In *StatMT '09: Proceedings of the Fourth Workshop on Statistical Machine Translation*, pages 75–79, Morristown, NJ, USA. Association for Computational Linguistics.

Och, F. J. 2000. Giza++ tools for training statistical translation models. `http://www.fjoch.com/GIZA++.html`.

Och, F. J. 2003. Minimum error rate training in statistical machine translation. In *ACL '03: Proceedings of the 41st Annual Meeting on Association for Computational Linguistics*, pages 160–167, Morristown, NJ, USA. Association for Computational Linguistics.

Och, F. J. and H. Ney. 2003. A systematic comparison of various statistical alignment models. *Computational Linguistics*, 29(1):19–51.

Och, F.J. and H. Ney. 2002. Discriminative training and maximum entropy models for statistical machine translation. In *ACL '02: Proceedings of the 40th Annual Meeting on Association for Computational Linguistics*, pages 295–302, Morristown, NJ, USA. Association for Computational Linguistics.

Och, F.J., C. Tillmann, H. Ney, et al. 1999. Improved alignment models for statistical machine translation. In *Proceedings of the Joint SIGDAT Conference on Empirical Methods in Natural Language Processing and Very Large Corpora*, pages 20–28.

Papineni, K.A., S. Roukos, and T.R. Ward. 1997. Feature-based language understanding. In *Fifth European Conference on Speech Communication and Technology*, pages 1435–1438.

Papineni, K.A., S. Roukos, T.R. Ward, and W. Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *ACL '02: Proceedings of the 40th Annual Meeting on Association for Computational Linguistics*, pages 311–318, Morristown, NJ, USA. Association for Computational Linguistics.

Paul, M., A. Finch, and E. Sumita. 2009. NICT@WMT09: Model adaptation and transliteration for Spanish-English SMT. In *StatMT '09: Proceedings of the Fourth Workshop on Statistical Machine Translation*, pages 105–109, Morristown, NJ, USA. Association for Computational Linguistics.

Pauls, A., J. DeNero, and D. Klein. 2009. Consensus training for consensus decoding in machine translation. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing*, pages 1418–1427, Singapore, August. Association for Computational Linguistics.

Rapp, R. 2009. The backtranslation score: Automatic mt evalution at the sentence level without reference translations. In *Proceedings of the ACL-IJCNLP 2009 Conference Short Papers*, pages 133–136, Suntec, Singapore, August. Association for Computational Linguistics.

Schwenk, H. and P. Koehn. 2008. Large and diverse language models for statistical machine translation. In *IJCNLP '08: Proceedings of The Third International Joint Conference on Natural Language Processing*, pages 661–666.

Smith, N.A. and J. Eisner. 2005. Contrastive estimation: Training log-linear models on unlabeled data. In *ACL '05: Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics*, pages 354–362, Morristown, NJ, USA. Association for Computational Linguistics.

Somers, H. 2005. Round-trip translation: What is it good for. In *Proceedings of the Australasian Language Technology Workshop*, pages 127–133.

Steinberger, R., B. Pouliquen, A. Widiger, C. Ignat, T. Erjavec, D. Tufis, and D. Varga. 2006. The JRC-Acquis: A multilingual aligned parallel corpus with 20+ languages. In *Proceedings of the 5th International Conference on Language Resources and Evaluation*, pages 24–26.

Stolcke, A. 2002. SRILM-an extensible language modeling toolkit. In *Seventh International Conference on Spoken Language Processing*, volume 3, pages 901–904.

Ueffing, N., G. Haffari, and A. Sarkar. 2007. Transductive learning for statistical machine translation. In *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics*, pages 25–32, Prague, Czech Republic, June. Association for Computational Linguistics.

Vogel, S., H. Ney, and C. Tillmann. 1996. HMM-based word alignment in statistical translation. In *Proceedings of the 16th conference on Computational linguistics*, pages 836–841, Morristown, NJ, USA. Association for Computational Linguistics.