Approval

Name: Jasneet Singh Sabharwal
Degree: Master of Science (Computing Science)
Title: *Bilingual Language Models using Word Embeddings for Machine Translation*

Examining Committee: Chair: Ryan Shea
University Research Associate
Simon Fraser University

Anoop Sarkar
Co-Senior Supervisor
Professor, School of Computing Science
Simon Fraser University

Fred Popowich
Co-Senior Supervisor
Professor, School of Computing Science
Simon Fraser University

Jiannan Wang
Internal Examiner
Assistant Professor, School of Computing Science
Simon Fraser University

Date Defended: 19 May 2016
Abstract

Bilingual language models (Bi-LMs) refer to language models that are modeled using both source and target words in a parallel corpus. While translating a source sentence to a target language, the decoder in phrase-based machine translation system breaks down the source sentence into phrases. It then translates each phrase into the target language. While decoding each phrase, the decoder has very little information about source words that are outside the current phrase in consideration. Bi-LMs have been used to provide more information about source words outside the current phrase. Bi-LMs are estimated by first creating bitoken sequences using a parallel corpus and the word alignments between the source and target words in that corpus. When creating the bitoken sequences, the vocabulary expands considerably and Bi-LMs suffer due to this huge vocabulary which in turn increases the sparsity of the language models. In previous work, bitokens were created by first replacing each word in the parallel corpus either by their part-of-speech tags or word classes after clustering using the Brown clustering algorithm. Both of these approaches only take into account words that are direct translations of each other as they only depend on word alignments between the source word and target word in the bitokens. In this thesis, we propose the use of bilingual word embeddings as a first step to reduce the vocabulary of the bitokens. Bilingual word embeddings are a low dimensional representation of words trained on a parallel corpus of aligned sentences in two languages. Using bilingual word embeddings to build Bi-LMs for machine translation is significantly better than the previous state of the art that uses Bi-LMs with an increase of 1.4 BLEU points in our experiments.

Keywords: Bilingual Language Models; Bilingual Word Embeddings; Machine Translation
Dedication

To Mom, Dad and Dr. Anoop Sarkar!
Acknowledgements

First, I would like to express my gratitude towards my supervisor and mentor Dr. Anoop Sarkar. This thesis would not have been possible without his guidance, support and brilliant insights. My achievements during the Master’s career would have never been possible without his understanding, encouragement and patience. I am very fortunate to have Dr. Fred Popowich as my supervisor. His advice and valuable comments not only made this thesis possible but also helped me in my professional adventures.

I am grateful to Dr. Hieu Hoang, Dr. Kenneth Heafield and Dr. Karl Mortiz Hermann for patiently answering my queries during the development work for this thesis. I would like to extend thanks to Dr. George Foster, Dr. Roland Kuhn, Darlene Stewart and Eric Joanis for their insights into their system which became a base for this thesis. I am also grateful to Te Bu for patiently looking at the bilingual word embeddings and helping me choose the best embeddings. Without Te, this thesis would have been very frustrating.

I wish to thank all my fellow Natural Language Lab colleagues, specially Bradley Ellert, Ramtin Meh dizadeh, Rohit Dholakia, Anahita Mansouri, Maryam Siahbani, Mark Schmidt, Hassan Sharavani, Milan Tofiloski, Bruce Krayenhoff, Lydia Odilinye and Ann Clifton. Discussions with them lead to the development of ideas which helped me in this thesis and other endeavours.

I have had an amazing time in Vancouver and I owe it to Gladys We, Carolyn Hanna, Diljot Grewal, Rejaa Dar and Amir Yaghoubi Shahir for their immense support and for making my life here exciting and colourful.

I specially thank Milica Videnović for being extremely patient, supportive and for standing by me through my thick and thin since I have known you. Also, special thanks to my friends Gagan Bindra and Tejdeep Bawa who have stood by me through the best and worst of times in my life.

Finally, I would like to express my gratitude towards my family. I am extremely fortunate for having been blessed with incredible parents who have always supported me with their unconditional love and have pushed me to embrace opportunities, to ask questions, to learn from others, to push harder when facing big challenges and that cannot is never an option.
# Table of Contents

Approval .............................................. ii
Abstract ........................................... iii
Dedication ......................................... iv
Acknowledgements ................................ v
Table of Contents ................................ vi
List of Tables ..................................... viii
List of Figures .................................... ix

1 Introduction ...................................... 1
   1.1 Statistical Machine Translation .................. 2
   1.1.1 Word Alignments ............................... 2
   1.1.2 Translation Model .............................. 4
   1.1.3 Language Model ................................ 5
   1.1.4 Decoder ........................................ 8
   1.1.5 Tuning ......................................... 10
   1.2 Bi-LMs and why do we need them? ................. 12
   1.3 Summary ......................................... 13

2 Word Embeddings ................................. 14
   2.1 What are word embeddings? ...................... 14
   2.2 Creating Word Embeddings ....................... 15
   2.3 Visualizing Word Embeddings .................... 16
   2.4 Summary ......................................... 20

3 Bilingual Language Models ..................... 21
   3.1 What are Bilingual Language Models? .......... 22
   3.2 Bi-LMs using Word Embeddings ................ 25
      3.2.1 Using Word Embeddings to create Bi-LMs .. 26
List of Tables

Table 1.1 All possible phrase pairs consistent with the alignments shown in Fig. 1.1 6
Table 1.2 A sample entry in the translation model (phrase table) . . . . . . . . . 6
Table 3.1 Number of target words vs. number of bitokens in our data . . . . . 23
Table 3.2 Feature combinations used in Baseline, Embed-Brown, Embed-Embed-
Reduced-Vocab and Embed-Embed-Full-Vocab SMT systems. . . . . . . . . 29
Table 4.1 Corpus Statistics: Chinese-English Parallel Corpus . . . . . . . . . . . . . 33
Table 4.2 5-gram language model for English and counts of each gram. . . . . . . . . 40
Table 4.3 Results comparing the baseline system 4.1 and three of our proposed
SMT systems 4.2 . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 40
# List of Figures

<p>| Figure 1.1 | A sample alignment between a Spanish sentence and English sentence [41] | 3 |
| Figure 1.2 | A sample German source sentence broken into possible phrases and the top four translation options for each of the source phrase [41, p. 159] | 8 |
| Figure 1.3 | A sample German source sentence broken into possible phrases and the top four translation options for each of the source phrase [41, p. 160] | 9 |
| Figure 1.4 | An example of modified n-gram precision. | 11 |
| Figure 2.1 | WordEmbeddingsViz upload screen: Here, the user can upload bilingual word embeddings, word list, training corpus and alignments (optional) | 18 |
| Figure 2.2 | WordEmbeddingsViz: A scatter plot of word embeddings of Zh-En parallel corpus. Orange squares represent English words and blue circles represent Chinese words. | 19 |
| Figure 2.3 | WordEmbeddingsViz: Alignments of English words <em>broadcast</em>, <em>broadcast</em>, <em>broadcasting</em> with their Chinese counterparts. | 19 |
| Figure 2.4 | WordEmbeddingsViz: Alignments of English words <em>clocks</em>, <em>timepiece</em> &amp; <em>chiming</em> with their Chinese counterparts. | 20 |
| Figure 3.1 | Source and target sentences with their alignments for creating bitokens [61] | 22 |
| Figure 3.2 | Bitokens created from the parallel sentences and their alignments shown in Fig. 3.1 | 22 |
| Figure 3.3 | Creating bitokens, word clusters and bitoken clusters [61] | 24 |
| Figure 3.4 | Creating <em>Brown Coarse LM 100</em> [61] | 25 |
| Figure 3.5 | Creating <em>Brown Coarse LM 1600</em> [61] | 25 |
| Figure 3.6 | Creating <em>Brown Coarse Bi-LM (400, 400)</em> [61] | 26 |
| Figure 3.7 | Creating <em>Brown-Brown Coarse Bi-LM (400, 400)</em>. It uses the bitoken sequences created in Fig 3.4 [61] | 26 |
| Figure 3.8 | Creating <em>Embed Coarse LM 100</em> and <em>Embed Coarse LM 1600</em> | 28 |</p>
<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>3.9</td>
<td>Creating Embed Coarse Bi-LM (400, 400)</td>
<td>28</td>
</tr>
<tr>
<td>3.10</td>
<td>Creating Embed-Brown Coarse Bi-LM 400(400, 400)</td>
<td>29</td>
</tr>
<tr>
<td>3.11</td>
<td>Creating Embed-Embed Coarse Bi-LM 400(400, 400)</td>
<td>29</td>
</tr>
<tr>
<td>3.12</td>
<td>Creating Embed-Embed Coarse Bi-LM 400(</td>
<td>V_f</td>
</tr>
</tbody>
</table>
Chapter 1

Introduction

Statistical Machine Translation (SMT) enables translation between various languages, such as French, German, Chinese, English, etc. With the advent of Google Translate and their support for translation between 81 languages, translation services have become available to the masses for day to day use. Current state of the art SMT utilize words, sub-phrases and phrases in the parallel text (corpora containing translations of the same text in two languages) to build fluent and accurate translation systems. In order to create such translation systems, machine learning methods are applied over the statistical information extracted from parallel text to develop models for translation.

Phrase-based SMT [43] uses contiguous sequence of words (phrases) as the unit of translation. In this, each source phrase is translated to a non-empty target phrase, where the source and target phrases can be of different lengths. The translation process of phrase-based SMT can be divided into three steps, as described in the survey by Adam Lopez [44]:

1. Split the sentence into phrases.
2. Translate each phrase
3. Permute over each translated phrase to get the final order.

When translating each source phrase to the target language, the SMT system only has information of source words in the current source phrase. Information from source words outside the current source phrase is incorporated only indirectly, via target words that are translations of these source words, if the relevant target words are close enough to the current target word to affect the language model probability scores. SMT systems use language models to determine how fluent is the translation. To add more information about source outside the current source phrase [51] introduced bilingual language models that use alignments between source words and target words to create bitokens. A language model was then estimated using the bitokens. When using bitokens, the vocabulary expands
significantly. To counter this, [51] replaced words in the corpus with their part-of-speech tags and then created the bitokens. [61] extended this work by clustering the words in the original corpus using a Brown clustering algorithm. They also clustered the bitokens before estimating the bilingual language models.

But, in these approaches, bilingual information is available only through word alignments. And, state of the art word alignment algorithms have a high error rate. In order to compensate for the alignment errors and to add more information from source words which are not only direct translations of target words but are also semantically similar to the target words, we introduce a new approach to train bilingual language models using monolingual and bilingual word embeddings. In this thesis, we propose to use the bilingual word embeddings as a first step to reduce the vocabulary of the bitokens. Using bilingual word embeddings to build Bi-LMs for machine translation is significantly better than the previous state of the art that uses Bi-LMs with an increase of 1.4 BLEU points in our experiments.

In the next section we give an introduction to statistical machine translation and the steps involved in building an SMT system.

1.1 Statistical Machine Translation

The process of building a phrase-based SMT system using a parallel corpus can be broadly divided into five modules:

1. Learn bi-directional alignments of words.

2. Extract phrase pairs from the alignments and calculate probability of each translation pair, called the translation model.

3. Estimate language models.

4. Tune the parameters for features used in the system.

5. Using the language model and translation model, decode the translation of a new source language sentence into a target language sentence.

In this thesis, we focus on Module 3 and 5, that are, estimating language models and using language models while decoding a new source language sentence. In the next subsections, we give a brief overview on each of the modules in an SMT system.

1.1.1 Word Alignments

Word alignment is the task of identifying translation relationships between words of sentence aligned parallel corpora. By sentence aligned parallel corpora, we mean a parallel corpus in which each sentence in the source language is aligned with the same sentence in the target
language. Word alignments do not have to be a one-to-one mapping. Words in one language can be aligned to more than one words or no words at all in the other language. Figure 1.1 shows alignment matrix between a Spanish sentence and English sentence. As shown in the example, the English word *slap* is aligned to three Spanish words *daba una bofetada*.

![Alignment Matrix](image)

Figure 1.1: A sample alignment between a Spanish sentence and English sentence [41]

It is not easy to find accurate alignments between words of two languages. Specially, for some function words which may or may not have an equivalent word in the other language. Also, it is important that content words in source language are aligned to the corresponding content word in the target language.

Approaches for learning word alignments can be classified into two general categories, as described by [53], (a) statistical alignment models, and (b) heuristic models. As, statistical alignment models are currently the state of the art, we only focus on them and look at various methods in this category.

In statistical alignment models, we collect statistics from the sentence aligned parallel corpus to generate word alignment models. We are given a source language string $f_1^J = f_1, ..., f_J$ and a target language string $e_1^I = e_1, ..., e_I$. In SMT, we have a translation model $P(f_1^J|e_1^I)$, which is the translation probability describing the relationship between a source language string and target language string. In this translation model, we introduce a hidden alignment $a_1^J$ which describes the mapping between word $f_j$ and $e_i$. This gives us the statistical alignment model as $P(f_1^J, a_1^J|e_1^I)$. As shown by [53], the relation between translation model and statistical alignment model is

$$P(f_1^J|e_1^I) = \sum_{a_1^J} P(f_1^J, a_1^J|e_1^I)$$ (1.1)
To account for the unknown parameters $\theta$ learnt from training data, the statistical alignment model is represented as $p_\theta(f^I_1, a^I_1|e^I_1)$. For each sentence pair $(f^I_n, e^I_n)$, for $n = 1, ..., N$, where $N$ is the size of parallel corpus, the alignment is denoted as $a = a^I_1$. We find the unknown parameters $\theta$ by maximizing the likelihood on the parallel corpus:

$$\hat{\theta} = \arg\max_{\theta} \prod_{n=1}^{N} \sum_{a} p_\theta(f^I_n, a|e^I_n)$$ (1.2)

To perform the maximization, Expectation Maximization (EM) \[19\] or some variant of it is used. After finding the unknown parameters, the best alignment for a pair of sentences can be calculated as:

$$\hat{a}^I_1 = \arg\max_{a^I_1} p_\hat{\theta}(f^I_1, a^I_1|e^I_1)$$ (1.3)

Such statistical alignment models are called generative models and are generally created using unsupervised learning techniques. A major drawback of generative models is that incorporating arbitrary features is difficult. For example, if we want to include orthographic similarity between two words, presence of the pair in some dictionary, etc. Another drawback of unsupervised generative models based on EM is that they require large amount of data and processing to converge to a good solution. Discriminative models on the other hand allow us to have arbitrary features. In discriminative models the features do not have to adhere to the independence assumption, that is, features can be dependent on other features. Whereas, generally in generative EM based algorithms, we assume that the features are independent of each other.

There are various ways to create discriminative models for getting word alignments. Word alignment problem can also be thought of as a maximum weighted matching problem where each pair of words in parallel sentences would be assigned a score depending on how likely they are to be aligned. The word alignment problem can also be considered as a maximum weight bipartite matching problem [63], where nodes correspond to words in the two parallel sentences. Aside from graph matching algorithms, discriminative approaches also use perceptron algorithm [50], support vector machines [14], conditional random fields [6] or neural networks [3].

### 1.1.2 Translation Model

In phrase-based SMT, continuous sequence of words (phrases) are the atomic units of translation. The source sentence is first broken down into phrases and each phrase is then trans-
iated. To get the translation of source phrase, a phrase table is learnt from the parallel corpora. [41] states the following advantages of using phrases as atomic units:

- Many-to-many translation can handle non-compositional phrases.
- Use of local context in translation.
- Longer phrases can be learnt if more data is available.

The learning of the phrase table can be broken down into three steps:

- Get alignments (as described in Section 1.1.1) of words from parallel corpora in both directions (bi-directional alignments). To combine the alignments from two runs, there are various heuristics in the literature, but for our work, we use the most widely used approach outlined in [43] called grow-diag-final-and. This heuristic has several steps. In the first step, all intersection alignment points are selected. In the grow-diag step, neighbouring and diagonally neighbouring alignment points which are in the union but not in the intersection of the two runs are selected. And, in final-and step, alignment points which are unaligned and present in the intersection are selected. For our work, we use GIZA++ [53] to get the bidirectional alignments.

- Using the bidirectional alignments from step 1, extract all possible phrase pairs which are consistent with the alignments [54, 43]. A phrase pair is consistent with the alignments if words within the source phrase are only aligned to words in the target phrase. Table 1.1 shows all the possible phrase pairs that are consistent with the alignments shown in Fig 1.1.

- Assign probabilities to phrase pairs using their relative frequencies:

\[
\phi(\bar{f}|\bar{e}) = \frac{\text{count}(\bar{e}, \bar{f})}{\sum_{\bar{f}_i} \text{count}(\bar{e}, \bar{f}_i)}
\]  

(1.4)

Here, \(\bar{f}\) is a source phrase and \(\bar{e}\) is the target phrase. Along with \(\phi(\bar{f}|\bar{e})\), other features like inverse lexical weighting, direct phrase translation probability and direct lexical weighting are also calculated.

At the end of these steps, we get a phrase table containing the bilingual phrase pairs, the alignments within those phrase pairs and feature scores as shown in table 1.2.

1.1.3 Language Model

The language model is an integral part of an SMT system. The job of a language model is to measure how likely a string of words (sentence) in a language would be uttered by a human speaker, that is, how fluent is the sentence. For example, we have two sentences in English, "this is a house" and "this a house is". The language model should
Table 1.1: All possible phrase pairs consistent with the alignments shown in Fig. 1.1

<table>
<thead>
<tr>
<th>Source Phrase</th>
<th>Target Phrase</th>
</tr>
</thead>
<tbody>
<tr>
<td>María no daba una bofetada a la bruja verde</td>
<td>Mary did not slap the green witch</td>
</tr>
<tr>
<td>María no daba una bofetada a la bruja verde</td>
<td>Mary did not slap the green witch</td>
</tr>
<tr>
<td>María no daba una bofetada a la bruja verde</td>
<td>Mary did not slap the green witch</td>
</tr>
<tr>
<td>María no daba una bofetada a la bruja verde</td>
<td>Mary did not slap the green witch</td>
</tr>
</tbody>
</table>

Table 1.2: A sample entry in the translation model (phrase table)

<table>
<thead>
<tr>
<th>Source Phrase</th>
<th>Target Phrase</th>
<th>Feature Scores</th>
<th>Alignments</th>
</tr>
</thead>
<tbody>
<tr>
<td>in europa</td>
<td>in europe</td>
<td>0.829007 0.207955 0.801493 0.492402</td>
<td>0-0 1-1</td>
</tr>
</tbody>
</table>
tell us that the probability of former sentence should be higher than the latter, that is, 
$p_{LM}(this\ is\ a\ house) > p_{LM}(this\ a\ house\ is)$. From this example, we notice that language
models along with telling how fluent a sentence is, they also help in deciding the right
order of words. They also help in choosing the right words for translation. For example,
$p_{LM}(I\ am\ going\ home) > p_{LM}(I\ am\ going\ house)$.

For an SMT system, the language model is trained on large monolingual corpora of
the target language. This is because we want to aid the translation system in deciding a
good translation for a source sentence. Due to the abundance of data available in a single
language, the amount of training data used in training the language model is generally
orders of magnitude more than the parallel corpora used to train the translation model.

The state of the art method to train a language model is $n$-gram language modelling. In
$n$-gram language models, we compute the probability of a sentence $W = w_1, w_2, w_3, ..., w_n$.
The probability of sentence $p(W)$ can be represented as a joint probability of words in the
sentence:

$$p(W) = p(w_1, w_2, ..., w_n) \quad (1.5)$$

Using the chain rule, we can break this down:

$$p(W) = p(w_1)p(w_2|w_1)p(w_3|w_1, w_2)...p(w_n|w_1, w_2, ..., w_{n-1}) \quad (1.6)$$

Here, we have broken down the probability of a sentence into the probability of words
depending on the preceeding words. To be able to calculate these probabilities easily, we
limit the history of each word to $m$ words.

$$p(W) = p(w_1)p(w_2|w_1)p(w_3|w_1, w_2)...p(w_n|w_{n-m}, ..., w_{n-2}, w_{n-1}) \quad (1.7)$$

This model in which we step through a sequence of words and consider a limited history for
each transition is called a Markov chain. Here $m$ is the order of the model. For example,
a 3 gram language model would be:

$$p(W) = p(w_1)p(w_2|w_1)p(w_3|w_1, w_2)p(w_4|w_2, w_3)...p(w_n|w_{n-2}, w_{n-1}) \quad (1.8)$$

To estimate the probability of the n-grams, we collect the required counts from the
monolingual corpora and use maximum likelihood estimation:

$$p(w_3|w_1, w_2) = \frac{\text{count}(w_1, w_2, w_3)}{\sum_w \text{count}(w_1, w_2, w)} \quad (1.9)$$

Even though we use a large monolingual corpora to train the language model, we still
cannot cover every word and its usage. To tackle the problem of unseen words, the literature
describes various smoothing techniques like add-one smoothing, add-α smoothing, Good-Turing smoothing [26], Witten-Bell smoothing [69], Kneser-Ney smoothing [38], etc.

1.1.4 Decoder

[43] states the mathematical model for translation as \( p(e|f) \). The job of the decoder is to find the translation \( e_{\text{best}} \) with the highest probability. This can be mathematically formulated as:

\[
e_{\text{best}} = \arg \max_e p(e|f)
\]  

(1.10)

Figure 1.2: A sample German source sentence broken into possible phrases and the top four translation options for each of the source phrase [41, p. 159]

When the decoder proposed by [43] has to translate a source sentence, it first breaks the sentence down into the atomic units of phrase-based SMT, that is, phrases as shown in Fig 1.2. The target sentence is then generated left to right in the form of partial translations called hypothesis and it employs a beam search algorithm. The decoder starts with an initial empty hypothesis. A new hypothesis is expanded from an existing hypothesis by selecting the next untranslated source phrase, finding it’s possible target phrase from the translation model. The target phrase is appended to the existing target sentence. The hypothesis is then scored using weighted combination of scores from certain feature functions and the source phrase is marked as translated. The final hypothesis in the search tree which has the highest probability is chosen as the best translation for the source sentence as shown in Fig 1.3.

A limitation of this approach is that for each source sentence, an exponential number of hypothesis are generated. Searching through these hypothesis is an NP-complete problem [39]. To tackle this problem, [43] proposed using the hypothesis recombination strategy as in [55]. Along with this, hypotheses are also pruned by comparing their current score and the future score proposed by [43]. Histogram pruning and threshold pruning proposed by [40] are also used to prune the search tree.
We mentioned above that the decoder scores each hypothesis using a weighted combination of scores from various feature functions. We also mentioned above that the decoder uses translation model in its process. In addition to the feature scores from the translation model, the decoder also uses the language model we described earlier, a reordering model which is created using the alignments extracted earlier and various feature functions.

[43] proposed a weighted model comprising the phrase translation model \( \phi(\bar{f}|\bar{e}) \), a reordering model \( d \), and language model \( p_{LM}(e) \), which is mathematically formulated as:

\[
e_{best} = \arg\max_{e} \prod_{i=1}^{I} \phi(\bar{f}_i|\bar{e}_i)^{\lambda_\phi} d(a_i - b_i - 1)^{\lambda_d} \prod_{i=1}^{|e|} p_{LM}(e)^{\lambda_{LM}}
\] (1.11)

Here \( a_i \) and \( b_i \) are the starting and ending position of the source phrase that was translated to the \( i^{th} \) target phrase and \( i - 1^{th} \) target phrase. \( \lambda_\phi, \lambda_d \) and \( \lambda_{LM} \) are the weights for translation model, reordering model and language model respectively. These scores are calculated incrementally for each hypothesis.

Such a weighted model is actually a log-linear model of the form:

\[
p(x) = \exp \sum_{i=1}^{n} \lambda_i h_i(x)
\] (1.12)

When working with probabilities, it is easier to deal with log values to avoid floating point underflow problems. We can rewrite equation 1.11 as:

\[
p(e, a|f) = \exp \left[ \lambda_\phi \sum_{i=1}^{I} \log \phi(\bar{f}_i|\bar{e}_i) \right. \\
+ \left. \lambda_d \sum_{i=1}^{I} \log d(a_i - b_{i-1} - 1) \right.
\] \\
+ \left. \lambda_{LM} \sum_{i=1}^{|e|} \log p_{LM}(e_i|e_1...e_{i-1}) \right]
\] (1.13)
This formulation allows us to add more independent feature functions, that is, feature functions that are independent of other feature functions. In practice, Moses [42], a popular SMT toolkit that we use in our work, uses 15 features which are as follows:

- Unknown word penalty (1 feature)
- Word penalty (1 feature)
- Phrase penalty (1 feature)
- Translation model (4 features)
- Lexical reordering (6 features)
- Distortion (1 feature)
- Language model (1 feature)

Each of these features have a weight associated with them and it is the job of a tuning algorithm which we will look at in the next section to optimize them.

### 1.1.5 Tuning

A simple SMT system utilizes number of features during its decoding stage. Each of these features has a weight associated with them, and a default value for each of these weights. To understand which of these features are better indicators of a good translation and vice-versa, we need to tune these weights (also called parameters). While tuning, we need to understand the affect of the parameters on translation performance. A popular metric that is used for this is Bilingual Evaluation Understudy (BLEU) [56]. BLEU compares the output translation with reference translations according to the equation:

$$BLEU_{score} = BP \cdot \exp \left( \sum_{i=1}^{n} w_i \log (precision_i) \right)$$ (1.14)

Here, BP is called brevity penalty and is formulated as:

$$BP = \min(1, \frac{output - length}{reference - length})$$ (1.15)

$w_i$ are the weights associated with different n-gram precisions. These weights are generally set to 1. Brevity penalty is used to penalize phrases that are much shorter compared to the reference translation. A thing to note is that BLEU score is 0 if any of the n-gram precisions is 0. To calculate precision, one simply counts the number of n-grams of system translation which occur in reference translations divided by the total number of n-grams in the system translation. The beauty of this precision based metric is that it allows the

MT systems can easily over-generate reasonable words, which would result in high precision for sentences like the one in example 1.4. To counter this issue, modified n-gram precision exhausts a reference n-gram once it is matched, that is, a reference n-gram once matched cannot be matched again. Fig. 1.4 also shows the output of modified n-gram precision.

| System Translation: the the the the the the. |
| Reference 1: The cat is on the mat. |
| Reference 2: There is a cat on the mat. |
| Modified unigram precision: \( \frac{2}{7} \).

Figure 1.4: An example of modified n-gram precision.

Modified n-gram precision is given as follows:

\[
p_n = \frac{\sum_{C \in \text{Candidates}} \sum_{n\text{-gram} \in C} \text{Count}_{\text{clip}}(n\text{-gram})}{\sum_{C' \in \text{Candidates}} \sum_{n\text{-gram}' \in C'} \text{Count}(n\text{-gram}')} \tag{1.16}
\]

Candidates refers to the target set of sentences. Where, \( \text{Count}_{\text{clip}} = \min(\text{Count}, \text{Max}_\text{Ref}_\text{Count}) \), that is, do not exceed the largest count of the n-gram in any single reference.

When tuning the parameters of the feature functions, we always use a small parallel corpora that was not used during the training of the models. This small parallel corpora is called the tuning set or dev set. Tuning algorithms can be divided into two main classes:

- **Batch tuning algorithms**: In batch tuning algorithms, the complete tuning set is decoded with some initial weights. Generally an n-best list of decoded output is generated. The tuning algorithm then updates the weights based on the decoder output. The tuning set is again decoded based on the updated weights. This procedure is repeated to optimize the weights until we reach convergence or up to a certain number of iterations. Various such tuning algorithms have been described in the literature, Minimum Error Rate Training (MERT) [53] is the most widely used tuning algorithm. Lattice MERT [45] is a variant of MERT that uses lattices instead of n-best list. Pairwise ranked optimization (PRO) [32] works by ranking learning the weight set that ranks the n-best list in the same order as BLEU. Batch MIRA [13] is a type of margin based classification algorithm that works in the batch tuning setting.

- **Online tuning algorithms**: Online algorithms work together tightly with the decoder. After decoding each sentence, the tuning algorithm updates the weights before the
next sentence is decoded. The MIRA tuning algorithm [13] is the most widely used tuning algorithm in this setting.

1.2 Bi-LMs and why do we need them?

In phrase-based SMT, during the decoding process, the decoder decodes a partial hypothesis containing a phrase from the source sentence into the target language. During this process, it has very little information from source words outside the current phrase pair. [61] states that information from source words outside the current phrase pair is incorporated only indirectly, via target words that are translations of these source words, if the relevant target words are close enough to the current target word to affect the language model scores. To add more information about the source words, [51] introduced part-of-speech based bilingual language models (Bi-LMs) which were extended by [61]. Bilingual language models are generated by aligning each target word in the parallel training corpus with source words to create bitokens. These bitokens are then used to estimate an n-gram language model. Coarse Bi-LMs are Bi-LMs which are estimated by first clustering the bitokens and then estimating the language model. Similarly, coarse LMs are also language models which are estimated by clustering the words and then estimating the language model based on the clustered data. [51] generated the Bi-LMs by first replacing the words in the parallel corpus with part-of-speech tags. Using this augmented corpus and alignments, the bitokens were created to estimate the Bi-LMs. Similarly, [61] used MKCLS [52] to create word classes.

In this thesis, we propose a new method of generating Bi-LMs. We create word embeddings and bilingual word embeddings (Chapter 2 will give an introduction to word embeddings and bilingual word embeddings) of words in our training data. These embeddings are clustered using a spectral clustering algorithm. This allows us to group together words which are semantically similar. These clusters are used to augment the original corpus, hence reducing the vocabulary of the original parallel training corpus. The augmented corpora are used to training Coarse LMs and Bi-LMs (Chapter 3 explains in detail the steps to create Coarse LMs and Bi-LMs). We call these LMs & Bi-LMs coarse because they are estimated using data whose vocabulary has been reduced by using certain clusters. In the literature, work has been done to use part-of-speech tags or monolingual clusters of words using Brown clustering algorithm [9].

In our work we propose three new approaches of creating and using coarse LMs and Bi-LMs to improve statistical machine translation task. We show that our best approach which was based on our original hypothesis of using bilingual word embeddings and monolingual word embeddings achieves +1.4 BLEU points in the Chinese-English SMT task and two of our approaches achieve an increment in BLEU score by 0.1 and 0.4.
1.3 Summary

In this chapter we introduce the individual steps in training a statistical machine translation system. We then give an introduction to bilingual language models and how they can be helpful in statistical machine translation. In the end we introduce our idea of learning bilingual language models using word embeddings. In Chapter 2 we will discuss about word embeddings, bilingual word embeddings and a method to judge the best bilingual word embeddings. In Chapter 3, we will discuss in detail about bilingual language models. We will also introduce our baseline system and our approaches to develop bilingual language models. Later, in Chapter 4 we describe our experimental setup and results from our approaches. Finally, in Chapter 5 we conclude this thesis and introduce ideas that would be natural extensions of our work which we would like to do in the future.
Chapter 2

Word Embeddings

2.1 What are word embeddings?

Semantic relations between words denote how two words are related or how close their meanings are. One way to represent this relation is by representing each word as a vector (also called word embeddings) such that, words which are similar, their vectors would lie closer to each other in some $n$ dimension space. Whereas, vectors of dissimilar words would lie far apart. When creating the word embeddings, we assume that words are characterized by the words that surround it, that is the company that the word keeps [28]. The relation between two vectors (words) is represented by using a displacement vector, that is, a vector between two vectors. The displacement vector can help us find relations like $queen : king :: woman : man$, which would mathematically be denoted as $v_{queen} - v_{king} = v_{woman} - v_{man}$. Here, $v_i$ means an $n$-dimension vector of the word $i$.

Learning word embeddings broadly falls into two categories. *Clustering based representations*, often use hierarchical clustering methods to group similar words based on their contexts. Brown Clustering [9] and [57] are the two most dominant approaches. Hidden Markov Models can also be used to induce clustering on words [36]. The problem with clustering approach is that the representations generated are sparse vectors. The reason they are sparse is because the vectors generated would generally be one-hot vectors. Such vectors are contains binary values 0 or 1 where 1 indicates the cluster number to which the word belongs. To reduce the sparsity issues, the other approach is to generate *dense representations* of words. These representations are low dimensional real valued dense vectors. These embeddings can be generated by using latent semantic analysis [18], canonical correlation analysis [21], neural-networks [16, 35, 47, 49, 27].

As estimating Bi-LMs required parallel corpora of two languages, it is natural to utilize bilingual word embeddings that denote semantic relations among words across two languages, that is words which are semantically similar in either of the languages are close to each other in some $n$-dimension space. This enables us to understand how close a word in
one language would be to another word in the second language. For example, the English word \textit{lake} and Chinese word \textit{深（deep pond)}, even though they are not direct translations of each other, but due to their semantic similarity, they would be close to each other in some \textit{n}-dimension space. And words which are semantically similar to \textit{深（deep pond)} and possibly direct translations of \textit{lake} would also be close to each other in that \textit{n}-dimension space. For word embeddings, we measure semantic similarity by measuring the cosine similarity between two word embeddings. It is formally defined as:

\[
similarity = \cos(\theta) = \frac{A \cdot B}{||A|| ||B||} = \frac{\sum_{i=1}^{n} A_i B_i}{\sqrt{\sum_{i=1}^{n} A_i^2} \sqrt{\sum_{i=1}^{n} B_i^2}} \tag{2.1}
\]

Here, \(A\) and \(B\) are two vectors of size \(n\).

Bilingual word embeddings have been created by using various techniques like latent dirichlet allocation and latent semantic analysis \[8, 71\], canonical correlation analysis \[24\], neural-networks \[37, 73, 48, 31, 12\]. In the next section we discuss the reasons for choosing the algorithms to create monolingual and bilingual word embeddings.

\subsection*{2.2 Creating Word Embeddings}

As stated in Section 2.1, the underlying idea of most of the methods is based on the concept that the meaning of a word can be determined by the \textit{company that it keeps}. This idea is the underlying method for most of the work done to create monolingual and bilingual word embeddings. For both the embeddings, most of the popular approaches are based on using either canonical correlation analysis \[21, 24\] and neural networks \[16, 35, 37, 47, 49, 48, 73, 31, 12\]. Neural network approaches to create word embeddings have been widely adopted due to the following advantages:

\begin{itemize}
  \item Training the networks can be done using parallel processing and distributed processing.\textsuperscript{1}
  \item Graphical processing units (GPU) can be utilized for faster training.\textsuperscript{2}
  \item If new data is available for training, the weights of the network can be updated by only using the new data and not the previously used training data. That is, the network does not need to be retrained by using all the previous and new training data. Matrix factorization methods like canonical correlation analysis and latent semantic analysis would require retraining of models using all the data.\textsuperscript{3}
\end{itemize}

\textsuperscript{1}Gensim toolkit\[58\] has implementation of word2vec\[47\] that allows training of word embeddings using parallel processing. It also allows updating of network weights with new training data that was not used to train the network previously.

\textsuperscript{2}CUDA implementation of word2vec: \url{https://github.com/whatupbiatch/cuda-word2vec}

15
They are currently state of the art methods in producing good quality word embeddings.

Due to their speed of training and being the state of the art algorithms for training embeddings, we decided to use a neural network based approach. For creating monolingual word embeddings we utilize Word2Vec [47, 49, 48], as it is currently state of the art toolkit for creating monolingual word embeddings. We will explain the usage of monolingual embeddings in Chapter 3.

For creating bilingual word embeddings, [73] utilize sentence aligned parallel corpus and their alignments to induce the embeddings whereas [31] only utilizes a sentence aligned parallel corpus (they state that there is no theoretical dependence on sentence aligned parallel corpora and technically it could also be used with document aligned parallel corpora). As creating alignments is not perfect and they have a small margin of error (the state of the art method to create alignments [4] for Chinese-English parallel corpus has an alignment error rate of 30%), using word embeddings that require alignments [73] would increase the chances of propagating errors. Hence, to keep the possibility of errors in creating alignments and creating bilingual word embedding independent of each other, we use the work of [31] to create the bilingual word embeddings.

When training bilingual word embeddings we need to manually choose multiple hyper-parameters for the algorithms. Varying the hyper-parameters changes the embeddings that are generated. To understand the effects of the hyper-parameters and to choose the ones which give good embeddings we introduce WordEmbeddingsViz⁴, a tool to visualize bilingual word embeddings and to study the effects of different values of hyper-parameters.

In the next section we explain how one can use WordEmbeddingsViz to choose the best embedding parameters.

### 2.3 Visualizing Word Embeddings

WordEmbeddingsViz enables a user to visualize bilingual word embeddings. The tool uses t-Distributed Stochastic Neighbor Embedding (t-SNE) [67] to project the embeddings into two dimension space. t-SNE is a non linear dimensionality reduction technique. t-SNE constructs a probability distribution over pairs of objects in high dimensions such that similar objects (that is, objects which are close to each other) have a high probability whereas dissimilar objects (objects that are far apart) have a low probability. t-SNE defines a similar probability distribution over pairs of objects in lower dimensions and minimizes the Kullback-Leibler divergence between the two distributions. In the higher dimension space, it uses Gaussian distribution to measure the similarity between objects, whereas in lower dimension space, it uses a Student’s t-distribution to measure the similarity. This is because

Student’s t-distribution has a long tail and it allows dissimilar objects to be modeled far apart. [67] showed that due to t-SNE’s non-linear dimensionality reduction and optimization function, it handles the modeling of curved manifolds better than other techniques like Classical Scaling [65] and Sammon Mapping [60]. t-SNE produces better visualization than other techniques by reducing the tendency to crowd points together towards the center of the map. Techniques like Isomap [64] and Locally Linear Embedding [59] are prone to this problem.

To use visualize the embeddings, a user will upload the following for each language:

- Word Embeddings
- Words
- Training Corpus (with part-of-speech for one language)
- Alignments (optional)

Figure 2.1 shows the upload screen where a user can upload the required data.

We require part-of-speech (POS) for one of the languages as this will be used by the tool to show a list of the top 1000 words by their occurrence count for verb, noun, adjective & adverb POS tags. For our work, we extracted POS tags for English data using the Stanford POS tagger [66].

WordEmbeddingsViz will perform non-linear dimensionality reduction using tSNE on the word embeddings. The dimensions would be reduced to two dimensions. The value of these dimensions for each word would be treated as x and y coordinates to visualize them as a scatter plot. Figure 2.2 shows the scatter plot for bilingual word embeddings of Chinese(Zh)-English(En) parallel corpus. The user can then zoom into the scatter plot to look at the word embeddings. The user can also select one of the English words from the sidebar (sidebar shows a list of top 1000 for each of the verbs, nouns, adjectives and adverbs). On selection of a word from the sidebar, that word will be highlighted and the user can then zoom in to look at the neighbouring embeddings. If for any English word, there are one or more Chinese words in the neighbourhood that are possible translations of that English word, then the user can align them using the alignment option built into the tool. Figure 2.3 and Figure 2.4 shows examples of alignments between English words broadcast, broadcast (an incorrect spelling of broadcast in the corpus) & broadcasting, and, clocks, timepiece & chiming along with their Chinese counterparts. The alignments can also be downloaded which can then be utilized for various usecases, such as, using the annotated alignments as information in word alignment algorithms.

Using WordEmbeddingsViz, a human annotator can look at bilingual word embeddings generated with different parameters. If the embeddings generated are of good quality then semantically similar words in two languages would lie close to each other in the projected space.
Figure 2.1: WordEmbeddingsViz upload screen: Here, the user can upload bilingual word embeddings, word list, training corpus and alignments (optional)
Figure 2.2: WordEmbeddingsViz: A scatter plot of word embeddings of Zh-En parallel corpus. Orange squares represent English words and blue circles represent Chinese words.

Figure 2.3: WordEmbeddingsViz: Alignments of English words broadcast, braodcast, broad-casting with their Chinese counterparts.
2.4 Summary

In this chapter we introduced word embeddings and bilingual word embeddings. We also described a tool WordEmbeddingsViz, which we developed to judge the bilingual word embeddings. In the next chapter, we will provide an in depth description of bilingual language models and our approach of using word embeddings to model bilingual language models.

Figure 2.4: WordEmbeddingsViz: Alignments of English words *clocks, timepiece & chiming* with their Chinese counterparts.
Chapter 3

Bilingual Language Models

In phrase-based statistical machine translation (SMT), the decoder (Section 1.1.4) breaks down a source sentence into phrases and translates one source phrase at a time. For each source phrase, the decoder uses a translation model (Section 1.1.2) to get the corresponding target phrase. To model the target language fluency, it also uses a language model (Section 1.1.3). A log-linear combination of these models along with additional features are used to score each hypothesis. The decoder then searches for a path in the search tree which gives the highest hypothesis score for the final translation.

As stated in section 1.2, during the decoding process, information from source words outside the current phrase pair in consideration is available indirectly through target words which are translations of these source words, if those target words are close enough to affect the language model scores. Due to this, the translation of each source phrase is performed in isolation without significant information from other source words in the sentence. The effect can be seen in the sentence *Maria no daba una bofetada a la bruja verde*. For this sentence we would get the following phrase segmentations: *Maria no, daba una bofetada a la, bruja verde*. Here, the translation of *Maria no* is not affected by the source words *daba* or *bofetada* or *bruja*. The other possible segmentation could be as shown in Table 1.2. The translation of words *Maria no daba una bofetada a la* can be done using the phrases *Maria no, daba una bofetada a la* or *Maria, no, daba una bofetada, a la*. The decoder cannot make use of the fact that both these options lead to the same translation *Mary did not slap the*. If the first option is chosen, the translation of *no* is affected by *Maria*, but in the second option, *no* is only affected by *Maria* via the language model.

To introduce the effect of source words outside the current phrase pair in consideration, a considerable amount of work has been done in the past. In this thesis, we extend the work of [61] to create bilingual language models (*Bi-LMs*) that will be used as additional features to the decoder.
3.1 What are Bilingual Language Models?

Bi-LMs are n-gram language models which are trained on bitokens instead of simple word tokens as done for standard language models (Section 1.1.3). Bitokens are generated using the source and target sentences from the parallel corpora and their alignments. To understand what bitokens are, let us look at two parallel sentences shown in Fig. 3.1:

![Figure 3.1: Source and target sentences with their alignments for creating bitokens](image)

Using the parallel sentences and their alignments shown in Fig. 3.1 we will create a bitoken sequence. When creating the bitokens, we want to make sure that all the target words are used. In the example above, the source word *we* is aligned to target words *nous* *nous*. We will replicate *we* twice and align both *nous* with both the *we* to create the bitokens *we-nous* and *we-nous*. Next we have the source words *had to* aligned to *devions*. We will join *had* and *to* to create a single token *had_to* and align that to *devions* to get *had_to-devions*. Since, the target word *d’* is not aligned to any source word, we align it to *NULL* and create a bitoken *NULL-d’*. For the source word *very*, as it is not aligned to any target word, it is dropped. Similarly, we also get the bitoken *forward_looking-progressistes*. The final bitokens are shown in Fig. 3.2:

![Figure 3.2: Bitokens created from the parallel sentences and their alignments shown in Fig. 3.1](image)

More formally, given a pair of sentences $e^I_1 = e_1...e_I$, $f^J_1 = f_1...f_J$ and alignments $A = \{(i,j)\}$, the bitokens are:

$$b_j = \{f_j\} \cup \{e_i | (i,j) \in A\}$$  \hspace{1cm} (3.1)

This makes sure that the number of bitokens $b_j$ are equal to the number of target words. These bitoken sequences can then be used to create a language model called a bilingual language model, formalized as follows:
<table>
<thead>
<tr>
<th>Number of Target Word Types</th>
<th>Number of Bitoken Types</th>
</tr>
</thead>
<tbody>
<tr>
<td>152318</td>
<td>3827728</td>
</tr>
</tbody>
</table>

Table 3.1: Number of target words vs. number of bitokens in our data

\[ p(e^I_1, f^I_1, A) = \prod_{j=1}^{J} P(b_j | b_{j-1}...b_{j-n}) \] (3.2)

The advantage of using Bi-LMs is that they can be used in phrase-based SMT as additional features in the log linear model. When the decoder scores each hypothesis using scores from translation and language model, it can easily incorporate the probability from Eqn. 3.2. Even though, Bi-LMs are language models, but they act more as translation models as they do not model the fluency of the target language but model the translation of source words.

In Bi-LMs the bitoken vocabulary size increases by many folds compared to the vocabulary size of target words. For example, as shown by [61], the target word _être_ might be split into multiple bitokens: _be-être_, _being-être_ and _to_be-être_. This large vocabulary also leads to an increase in the sparsity in data for language modelling. Table 3.1 shows the number of bitokens compared to the number of target words in our corpus. We will explain in detail about our data and alignments used to create the bitokens in Chapter 4.

To tackle the problem of large vocabulary and sparsity, [51] introduced coarse Bi-LMs. When training LMs and Bi-LMs, if the words are replaced by some word class to reduce the vocabulary size, they are then called Coarse LMs/Bi-LMs. When creating Bi-LMs, [51] replaced both the source and target words in their Arabic-English SMT with the corresponding Part-of-Speech tags. [61] extended this idea and replaced both source and target words with cluster ids generated using mkcls [52]. [61] not only clustered the initial source and target words, but also experimented with clustering the bitokens too. Figure 3.3 shows the 3 ways of creating coarse bitoken sequences for Coarse Bi-LMs:

- Word Clustering: Create bitoken sequences with only source and target word cluster ids. The bitokens are then used to create Bi-LMs.
- Bitoken Clustering 1: Create bitokens without clustering source and target words. Cluster the bitokens and then use the bitoken sequences augmented with bitoken cluster ids to create Bi-LMs.
- Bitoken Clustering 2: Create bitoken sequences with source and target word cluster ids. Cluster the bitoken sequences and then use the bitoken sequences with bitoken cluster ids to create Bi-LMs.
[2, 5] showed that coarse LMs along with the standard LM are particularly effective for morphologically rich languages. Motivated by this, [61] used a combination of coarse LMs and coarse Bi-LMs in their experiments. They created the following four feature functions:

- **Brown Coarse LM 100**: Using mkcls (Brown clustering), the target corpus is clustered into 100 clusters. The words in the original target corpus are replaced by their respective cluster ids. This step brings down the vocabulary of the original target corpus to 100. Hence, the new augmented corpus is called *coarse target corpus*. Using this coarse corpus, a language model is then trained called *Brown Coarse LM 100*, where 100 denotes the vocabulary size of the corpus used to train the language model. (See Fig. 3.6)

- **Brown Coarse LM 1600**: Similar to *Brown Coarse LM 100*, the target corpus is clustered into 1600 clusters using mkcls. The original corpus is then augmented with the new cluster ids of the respective words to create a coarse target corpus. This coarse target corpus is used to train the language model *Brown Coarse LM 1600*. (See Fig. 3.5)

- **Brown Coarse Bi-LM (400, 400)**: Using mkcls first cluster the source and target parallel corpus into clusters of size 400 and 400 respectively. The words in the original parallel corpus are then replaced with their corresponding cluster ids to create coarse
source and target corpora. Using the coarse corpora and the alignments between the words in the original parallel corpus, bitokens sequences are generated using the process shown in Fig. 3.2. The bitoken sequences are then used to estimate a language model Brown Coarse Bi-LM (400, 400). (See Fig. 3.4)

- **Brown-Brown Coarse Bi-LM (400, 400):** To further reduce the vocabulary of bitokens created in the previous step, they are clustered into 400 clusters using mkcls. The bitokens in the bitoken sequences are replaced with their corresponding cluster ids to create coarse bitoken sequences. The coarse bitoken sequences have a reduced vocabulary of 400 and they are used to estimate the language model Brown-Brown Coarse Bi-LM 400(400, 400). (See Fig. 3.7)

Apart from the above mentioned cluster sizes, [61] also experimented with other cluster sizes and combinations but they showed that the above combination performs well for multiple language pairs.

In this thesis we extend the work of [61] to improve coarse LMs and Bi-LMs. In the next section, we describe in detail the contribution of this thesis, that is, using word embeddings to improve coarse LMs and Bi-LMs.

### 3.2 Bi-LMs using Word Embeddings

**mkcls** [52] is one of the most widely used word clustering toolkits. Motivated by Brown Clustering [9], mkcls\(^1\) implements an ensemble of optimizers and merges their results to cluster words into the provided number of classes. **mkcls** can only cluster monolingual corpus and it performs strongly in that aspect [7].

\(^1\)Understanding **mkcls** by Dr. Chris Dyer: [http://statmt.blogspot.ca/2014/07/understanding-mkcls.html](http://statmt.blogspot.ca/2014/07/understanding-mkcls.html)
The main goal of Bi-LMs is to add more information about source words that are not in the current phrase pair. But current state of the art coarse Bi-LMs only depend on alignments and monolingual word clusters to add more information about source words. For example, the English word lake and Chinese word 池 (deep pond), which are not direct translations of each other, won’t be captured by alignments and hence won’t be influencing the coarse Bi-LMs until unless mkcls assigns 池 and another Chinese word which is a direct translation of lake to the same cluster. To increase the probability that words in Chinese which are semantically similar to lake and possibly direct translations get clubbed together in a cluster, we utilize bilingual word embeddings to create the coarse Bi-LMs.

3.2.1 Using Word Embeddings to create Bi-LMs

In Chapter 2, we mention that we utilize [31] to create bilingual word embeddings for the words in our sentence aligned parallel corpus. To create coarse LMs and Bi-LMs, as done by [61], we need to cluster these embeddings. As mentioned earlier, in our baseline system, we used mkcls to cluster the words in our parallel corpus. To cluster word embeddings, we used greedo [62], a bottom-up hierarchical clustering algorithm for clustering low-dimensional representation of words under the [9] model. [62] show that the clusters created by greedo recover clusters which are of comparable quality to the algorithm of [9]. Using greedo gives us the opportunity to compare our approach to the baseline system without any modifications to the number of clusters because mkcls also creates Brown clusters.
To use the embeddings and their clusters for creating coarse LMs and coarse Bi-LMs, we propose the following six new feature functions:

- **Embed Coarse LM 100 and Embed Coarse LM 1600** The target word embeddings from the bilingual word embeddings are clustered into clusters of size 100 and 1600. We create two copies of the target corpus. In the first copy, we replace the target words with their cluster ids from 100 clusters to create *coarse target corpus 100*. Similarly, in the other copy of the target corpus, we replace the target words with their corresponding cluster ids from 1600 clusters to create *coarse target corpus 1600*. The two coarse target corpora are then used to estimate coarse language models Embed Coarse LM 100 and Embed Coarse LM 1600.

- **Embed Coarse Bi-LM (400, 400)** The target word embeddings from the bilingual word embeddings are clustered into 400 clusters and similarly the source word embeddings are clustered into 400 clusters. The target words in the target corpus are then replaced with the cluster ids from 400 target word clusters to create a *coarse target corpus* and similarly the source words in the source corpus are replaced with their corresponding cluster ids to create *coarse source corpus*. The two coarse corpora along with bidirectional word alignments between the words in the original parallel corpus are used to create bitoken sequences using the process shown in Fig. 3.2. These bitoken sequences are then used to estimate a coarse Bi-LM Embed Coarse Bi-LM (400, 400).

- **Embed-Brown Coarse Bi-LM 400(400, 400)** In this feature function we use the bitoken sequences created in the previous step. The bitokens are clustered into 400 clusters using *mkcls*. The bitokens in the bitoken sequences are then replaced with their corresponding cluster ids to create *coarse bitoken sequences*. These coarse bitoken sequences are used to estimate coarse Bi-LM Embed-Brown Coarse Bi-LM 400(400, 400).

- **Embed-Embed Coarse Bi-LM 400(400, 400)** Similar to the previous step, we use the bitoken sequences that we had created earlier. Instead of clustering them using *mkcls*, we create bitoken embeddings of the bitokens using *word2vec*. These bitoken embeddings are clustered into 400 clusters using *greedo*. The bitokens in the bitoken sequences are then replaced with their corresponding cluster ids to create *coarse bitoken sequences*. These coarse bitokens sequences are then used to estimate the coarse Bi-LM Embed-Embed Coarse Bi-LM 400(400, 400).

- **Embed-Embed Coarse Bi-LM 400(\(|V_f|, |V_e|\))** In this feature function, instead of first reducing the vocabulary of the parallel corpus, we will use the full vocabulary to create the bitokens. To do this, we take the parallel corpus and the bidirectional
alignments to create the bitoken sequences using the process shown in Fig. 3.2. Using word2vec bitoken embeddings are generated for the bitokens. These bitokens embeddings are clustered into 400 clusters using greedo. The bitokens in the bitoken sequences are replaced with their corresponding cluster ids to generate coarse bitokens sequences. These coarse bitoken sequences are further used to estimate coarse Bi-LM $Embed$-$Embed$ Coarse Bi-LM 400($|V_f|$, $|V_e|$). Here, $V_f$ denotes the full vocabulary of the source corpus and $V_e$ denotes the full vocabulary of the target corpus.

We propose three new SMT systems $Embed$-$Brown$, $Embed$-$Embed$-$Reduced$-$Vocab$ and $Embed$-$Embed$-$Full$-$Vocab$ that use a combination of the newly proposed feature functions. Table 3.2 shows the combination of feature functions used in the baseline and the three new systems. All the three systems that have been emphasized in the table utilize bilingual word embeddings when creating Coarse LMs and Bi-LMs.
Figure 3.10: Creating Embed-Brown Coarse Bi-LM 400 (400, 400)

Figure 3.11: Creating Embed-Embed Coarse Bi-LM 400 (400, 400)

<table>
<thead>
<tr>
<th>SMT System</th>
<th>Feature Function 1</th>
<th>Feature Function 2</th>
<th>Feature Function 3</th>
<th>Feature Function 4</th>
</tr>
</thead>
<tbody>
<tr>
<td>Baseline</td>
<td>Brown Coarse LM 100</td>
<td>Brown Coarse LM 1600</td>
<td>Brown Coarse Bi-LM (400, 400)</td>
<td>Brown-Brown Coarse Bi-LM 400(400, 400)</td>
</tr>
<tr>
<td>Embed-Brown</td>
<td>Embed Coarse LM 100</td>
<td>Embed Coarse LM 1600</td>
<td>Embed Coarse Bi-LM (400, 400)</td>
<td>Embed-Brown Coarse Bi-LM 400(400, 400)</td>
</tr>
<tr>
<td>Embed-Embed-Reduced-Vocab</td>
<td>Embed Coarse LM 100</td>
<td>Embed Coarse LM 1600</td>
<td>Embed Coarse Bi-LM (400, 400)</td>
<td>Embed-Embed Coarse Bi-LM 400(400, 400)</td>
</tr>
<tr>
<td>Embed-Embed-Full-Vocab</td>
<td>Embed Coarse LM 100</td>
<td>Embed Coarse LM 1600</td>
<td>-</td>
<td>Embed-Embed Coarse Bi-LM 400(</td>
</tr>
</tbody>
</table>

Table 3.2: Feature combinations used in Baseline, Embed-Brown, Embed-Embed-Reduced-Vocab and Embed-Embed-Full-Vocab SMT systems.
The feature functions in each of the system in Table 3.2 are added to the log linear model (Eqn. 1.13) as follows:

\[
p(e, a | f) = \exp[\lambda_\phi \sum_{i=1}^I \log \phi(\bar{f}_i | \bar{e}_i) + \lambda_d \sum_{i=1}^I \log d(a_i - b_{i-1} - 1) + \lambda_{LM} \sum_{i} \log p_{LM}(e_i | e_1 ... e_{i-1})] \\
+ \lambda_{FeatureFunction1} \sum_{i} \log p_{FeatureFunction1}(c_i | c_1 ... c_{i-1}) \\
+ \lambda_{FeatureFunction2} \sum_{i} \log p_{FeatureFunction2}(c_i | c_1 ... c_{i-1}) \\
+ \lambda_{FeatureFunction3} \sum_{i} \log p_{FeatureFunction3}(b_i | b_1 ... b_{i-1}) \\
+ \lambda_{FeatureFunction4} \sum_{i} \log p_{FeatureFunction4}(b_i | b_1 ... b_{i-1})
\] (3.3)
c is a coarse target word (a target word that has been replaced by a cluster id) and b is a bitoken. Adding to the log-linear model allows us to score each possible target phrase pair with our coarse models and influence the translation of each source sentence by providing more information from source words not in the source phrase being translated.

In Chapter 4 we will discuss about the setup and how the three approaches were tested and compare the performance of the three approaches to the baseline system. In the next section we discuss other approaches in the literature for introducing information about source words.

### 3.3 Previous Work

In the previous section we introduced Bi-LMs [51, 61] and three new approaches to estimate coarse LMs and Bi-LMs. Apart from Bi-LMs, there are other approaches for introducing source side contextual information in SMT. [11] proposed to use stochastic finite state transducers based on bilingual n-grams. This approach was extended by [46, 17, 70]. [1] successfully applied the implementation of [46] on their French-English SMT task. In this approach, the translation model is implemented as a stochastic finite state transducer trained as an n-gram language model of (source, target) pairs. When this model is trained, the source sentences are first reordered to match the order of target words using a finite state reordering model. The reordering model uses part-of-speech information to generalize reordering patterns.

[72] proposed spectral bilingual clustering for HMM (hidden markov model) based SMT [53]. This model adds information of both source and target languages to the HMM model. [29] introduced a lexical trigger model for SMT in which they used triplets incorporating long distance dependencies that go beyond the local context of phrases and n-gram based language models. [25] proposed factored markov backoff models along with a robust smoothing strategy that helps to generalize well. [23, 22] proposed operational sequence models (OSD) in which they generate a sequence of source and target words and perform reordering by integrating both translation and reordering models into a single generative story. In this approach, translation decisions can influence and get impacted by reordering decisions and vice versa. This approach can be viewed as an extension to [11, 46].

Neural network based language models have also been used to introduce source side information. Bilingual Neural Language Model [20] which is based on Neural Probabilistic Language Model [68] uses target-side history as well as source-side context to incorporate not only information about target words, but also source words.
3.4 Summary

In this chapter we introduced coarse language models and bilingual language models. We gave an in depth explanation of how bilingual language models are generated using parallel corpus and the alignments between the source and target words. We introduced the work of [51] in which he introduced part-of-speech based coarse Bi-LMs which was extended by [61] to introduce word class based coarse LMs and coarse Bi-LMs. Motivated by these approaches, we introduce propose three new SMT systems to create coarse LMs and Bi-LMs using monolingual embeddings from word2vec [47] and bilingual word embeddings [31]. As [61] had used mkcls to cluster the source and target parallel corpus, we utilized greedo [62] to cluster the word embeddings. Since, both mkcls and greedo are based on the [9] model of hierarchical clustering of words, this allows us to compare our approaches more coherently. In the next chapter we discuss about the setup and how we tested the three approaches and compared them to our baseline implementation of [61].
Chapter 4

Experiments and Results

In Chapter 3 we introduced coarse LMs and coarse Bi-LMs. We also introduced bilingual word embeddings and clustering of embeddings using **greedo** [62]. We described our baseline system, which is an implementation of [61]. We also introduced three new SMT systems (Table 3.2) in which we utilize bilingual word embeddings [31] to create coarse LMs and Bi-LMs. In this chapter we will explain the steps that we took to test and compare our approaches to the baseline system.

For our experiments we use a Chinese(Zh)-English(En) parallel corpus. The data is separated into three parts:

- The training dataset is used to train the phrase-based SMT system and bilingual word embeddings.

- The tuning dataset is used to tune the weights of features used in Moses decoder [42].

- We report our results on the test dataset. This is a blind dataset that was not used during the training and tuning step.

Table 4.1 shows the details of our data.

### 4.1 Baseline SMT System

Our baseline system is the system developed by [61]. As shown in Table 3.2, the baseline system uses the four feature functions.

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Corpus</th>
<th>Size</th>
<th>Number of References</th>
</tr>
</thead>
<tbody>
<tr>
<td>Training</td>
<td>HK + GALE Phase 1</td>
<td>2,352,888</td>
<td>N/A</td>
</tr>
<tr>
<td>Tuning</td>
<td>MTC Parts 1 &amp; 3</td>
<td>1927</td>
<td>4</td>
</tr>
<tr>
<td>Testing</td>
<td>MTC Part 4</td>
<td>919</td>
<td>4</td>
</tr>
</tbody>
</table>

Table 4.1: Corpus Statistics: Chinese-English Parallel Corpus
• Brown Coarse LM 100
• Brown Coarse LM 1600
• Brown Coarse Bi-LM (400, 400)
• Brown-Brown Coarse Bi-LM 400(400, 400)

We first use \texttt{mkcls} to cluster the English corpus into clusters of size 100, 400 and 1600. We also cluster the Chinese corpus into cluster of size 400. Using the English clusters, the words in English corpus are replaced with the cluster ids to create coarse corpora $100_{en}$, $1600_{en}$ and $400_{en}$. Similarly, we augment the Chinese corpus using the Chinese clusters to create coarse Chinese corpus $400_{zh}$.

Using \texttt{SRILM}, we estimate coarse LMs \textit{Brown Coarse LM 100} and \textit{Brown Coarse LM 1600}.

To create coarse Bi-LMs, we first need to create bitoken sequences using coarse corpora $400_{zh}$ and $400_{en}$. To create the bitoken sequences, as shown in Figure 3.3, we first need to create bidirectional alignments using the Zh-En parallel corpus. Using GIZA++[53], we create the following bidirectional alignments:

• IBM Model 2 [10] alignments.
• Hidden Markov Model (HMM) [53] alignments.

For both the alignments, \textit{grow-diag-final-and} heuristic (outlined in Section 1.1.2) is used. The alignments between the source and target words in Zh-En parallel corpus from both the alignment models are concatenated together. This is done to increase the different types of bitokens. Using the alignments, and coarse corpora $400_{zh}$ and $400_{en}$, the bitoken sequences are created, called \((400_{zh}, 400_{en})\) bitoken sequences.

A copy of the \((400_{zh}, 400_{en})\) bitoken sequences are used to estimate coarse Bi-LM \textit{Brown Coarse Bi-LM (400, 400)}. The second copy of bitoken sequences are clustered using \texttt{mkcls} with cluster size set to 400. The bitokens in the bitoken sequences are replaced with the new cluster ids to create coarse bitoken sequences $400_{zh}(400_{zh}, 400_{en})$. Using \texttt{SRILM}, coarse Bi-LM \textit{Brown-Brown Coarse Bi-LM 400(400, 400)} is estimated.

When estimating the coarse LMs and Bi-LMs, we use \textit{Witten-Bell smoothing} [69]. Coarse LMs and Bi-LMs create counts of counts that the SRILM implementation of Kneser-Ney smoothing cannot cope up with. [61] states that Witten-Bell smoothing outperforms Good-Turing smoothing. They also state that 8-gram coarse models outperformed lower n-gram coarse models. Hence, all our coarse LMs and Bi-LMs are 8-gram models.

The four coarse LMs and Bi-LMs are used in a stateful feature function in Moses decoder. We will talk about the decoder in a later section. In the next section we explain our steps to create the coarse models required in the three new systems that we are proposing.
4.2 Bi-LMs using Word Embeddings

In this thesis we propose three approaches to create coarse LMs and Bi-LMs (Section 3.2.1). The first step in creating coarse LMs and Bi-LMs using word embeddings is to create bilingual word embeddings for Zh-En parallel corpus.

4.2.1 Creating Bilingual Word Embeddings

In order to create bilingual word embeddings for Zh-En parallel corpus, we use BiCVM \(^1\), which has the implementation of [31]. We use the following parameters to train the embeddings:

- Tree type: **plain**
- Type of model: **additive**
- Training method: **adagrad**
- Word vector dimensions: **word-width**: 300
- Hinge loss margin: 300
- Number of noise samples per positive training example: 50
- Step size during gradient descent: 0.05
- L2 regularization for embeddings: 2
- Consider bi error for language 1: **true**
- Consider bi error for language 2: **true**
- Number of training iterations: 500
- Number of batches for adagrad: 50

This will create bilingual word embeddings with 300 dimensions. The above mentioned parameter values gave us the best embeddings, but we had experimented with the following values:

- Word vector dimensions: 150–500
- Hinge loss margin: 150–500
- Number of noise samples per positive training example: 10–100
- Step size during gradient descent: 0.05

\(^1\)BiCVM by Karl Mortiz Hermann: https://github.com/karlmoritz/bicvm
• L2 regularization for embeddings: 1-2
• Number of batches for adagrad: 50-100

To judge how good the embeddings are, we use WordEmbeddingsViz (Section 2.3). Using WordEmbeddingsViz, a human annotator\(^2\) looked at bilingual word embeddings generated with different parameter values. The annotator looked at the embeddings and tried to align an English word with a Chinese word if the two words are a translation of each other using the alignment tool in WordEmbeddingsViz. As the idea of word embeddings is that similar words should be close to each other and dissimilar words should be far apart from each other, hence the parameter values which allowed the annotator to align words in English and Chinese at a faster rate were deemed better. Based on these observations by the human annotator, the above defined parameters for BiCVM were chosen.

4.2.2 Creating Coarse LMs and Bi-LMs using Word Embeddings

In this subsection, we describe in detail the steps to create coarse LMs and Bi-LMs using the Zh-En bilingual embeddings (Subsection 4.2.1).

Embed-Brown SMT System

As shown in Table 3.2, the Embed-Brown SMT System uses the following four feature functions in addition to the standard feature functions in Moses:

• Embed Coarse LM 100
• Embed Coarse LM 1600
• Embed Coarse Bi-LM (400, 400)
• Embed-Brown Coarse Bi-LM 400(400, 400)

To create the models used in these feature functions we first cluster the Zh-En bilingual word embeddings using greedy\[^62\]. The following clusters are generated:

• Create clusters of size 100, 400 and 1600 for English embeddings.
• Create cluster of size 400 for Chinese embeddings.

Using the cluster ids for each word, we augment the original corpora (as described in 4.1) to get coarse corpora 100\(_{en}\), 1600\(_{en}\), 400\(_{en}\) and 400\(_{zh}\). Similar to our baseline system, we estimate 8-gram coarse LMs Embed Coarse LM 100 and Embed Coarse LM 1600 using

\(^2\)The human annotator was a native Chinese speaker who was not involved in development of our approaches for developing Coarse LMs and Bi-LMs.
SRILM with Witten-Bell smoothing and, corpora coarse 100\text{en} and coarse 1600\text{en} respectively. Similarly, we use HMM and IBM Model 2 alignments to create bitoken sequences \((400_{zh}, 400_{en})\) from coarse corpora 400\text{en} and 400\text{zh}.

The bitoken sequences \((400_{zh}, 400_{en})\) and SRILM are used to estimate \textit{8-gram} coarse Bi-LM \textit{Coarse Bi-LM} \((400_{zh}, 400_{en})\). The bitoken sequences \((400_{zh}, 400_{en})\) are further clustered using \texttt{mkcls} with cluster size set to 400. Using these clusters and bitoken sequences, coarse bitoken sequences \(400_{bi}(400_{zh}, 400_{en})\) are generated. Using the coarse bitoken sequences \(400_{bi}(400_{zh}, 400_{en})\) and SRILM, an \textit{8-gram} coarse Bi-LM Embed-Brown \textit{Coarse Bi-LM} \((400_{zh}, 400_{en})\) is estimated.

Embed-Embed-Reduced-Vocab SMT System

The Embed-Embed-Reduced-Vocab SMT System uses the following four features as shown in Table 3.2:

- Embed Coarse LM 100
- Embed Coarse LM 1600
- Embed Coarse Bi-LM (400, 400)
- Embed-Embed Coarse Bi-LM 400(400, 400)

We follow the same steps as in Embed-Brown SMT System to estimate Embed Coarse LM 100, Embed Coarse LM 1600 and Embed Coarse Bi-LM (400, 400). Using \texttt{word2vec}, we create bitoken embeddings for bitoken sequences \((400_{zh}, 400_{en})\). For \texttt{word2vec} we use continuous bag of words (CBOW) learning algorithm and the following parameters:

- Initial learning rate: 0.05
- Word vector dimensions: 300
- Threshold for configuring which higher-frequency words are randomly downsampled: \(1e^{-4}\)
- Negative sampling will be used and the value determines the number of noise words to be drawn: 5
- Number of training iterations: 15
- Maximum distance between the current and predicted word within a sentence: 8

The bitoken embeddings are clustered using \texttt{greedo} with cluster size set to 400. Utilizing the clusters, the bitoken corpus is transformed to create coarse bitoken sequences \(400_{bi}(400_{zh}, 400_{en})\). From these coarse bitoken sequences, we estimate an \textit{8-gram} coarse Bi-LM Embed-Embed Coarse Bi-LM 400(400, 400).
Embed-Embed-Full-Vocab SMT System

In *Embed-Embed-Full-Vocab SMT System* we proposed to use only three feature functions (Table 3.2 as follows:

- Embed Coarse LM 100
- Embed Coarse LM 1600
- Embed-Embed Coarse Bi-LM 400($|V_f|$, $|V_e|$)

We utilize the coarse LMs *Embed Coarse LM 100* and *Embed Coarse LM 1600* estimated for *Embed-Brown SMT System*.

Using Zh-En parallel corpus, HMM alignments and IBM Model 2 alignments, we create bitoken sequences ($|V|_{zh}$, $|V|_{en}$). Here, $|V|$ denotes that we use the full vocabulary instead of first clustering the parallel corpus and then creating bitokens. Utilizing *word2vec* again, we create bitoken embeddings with the same parameters as used in *Embed-Embed-Reduced-Vocab SMT System*. The bitoken embeddings are clustered into 400 clusters using *greedo*. The clusters are then utilized to augment the bitoken sequences to create coarse bitoken sequences $400_{bh}(|V|_{zh}$, $|V|_{en}$). Using the coarse bitoken sequences and *SRILM*, we estimate the 8-gram coarse Bi-LM *Embed-Embed Coarse Bi-LM 400*($|V\_zh|$, $|V\_en|$) with *Witten-Bell smoothing*.

We use the coarse LMs and Bi-LMs estimated by the three approaches in Moses decoder. We describe this process in the next section.

### 4.3 Integration with Decoder

In Section 4.1 and Section 4.2, we described in detail the steps to create coarse LMs and Bi-LMs. In phrase-based SMT, these models would be used as extra features in the log linear model described in Section 1.1.4 and as shown in Eqn. 3.3. In Moses decoder, these features can be added by creating stateful feature functions. In the stateful feature function that we created, we use these models as language model using the KenLM [30] wrapper integrated with Moses. *SRILM* stores the estimated language models in ARPA format\(^3\). This format is a standard format which can be read by most of the popular language modelling toolkits and specially *KenLM*.

Using coarse LMs in a stateful feature function is straightforward. When the decoder is translating a source sentence, it creates a partial hypothesis for each of the possible phrases in the source sentence and their translations from the phrase table. For each partial hypothesis, all the defined feature functions are called and each of those feature functions would generate a score for the partial hypothesis. In our case, it would be log probability

\(^3\)More details about ARPA format: [http://www.speech.sri.com/projects/srilm/manpages/ngram-format.5.html](http://www.speech.sri.com/projects/srilm/manpages/ngram-format.5.html)
score by our language models. For coarse LMs, whenever the feature function is called, for each of the coarse LMs, we perform the following steps:

- Extract target phrase from hypothesis.

- As each coarse LM was estimated for a coarse corpus, created by replacing words with corresponding cluster ids, we use the cluster mapping for that coarse LM to replace the words in the target phrase with the corresponding cluster id.

- Score the coarse target phrase using the required coarse LM.

The feature function would return the score that is calculated. This score is then used by the decoder in choosing the best possible hypothesis path while translating the sentence.

To score partial hypothesis using coarse Bi-LMs, the feature function not only uses the bilingual phrase pair in the partial hypothesis, but also uses the alignments within those phrase pairs which are available from the phrase table. Using the alignments and bilingual phrase pairs, we create the bitokens. Before creating the bitokens, optionally, we replace the words in phrase pair with the cluster ids, depending if we are calculating for Embed-Brown & Embed-Embed-Reduced-Vocab SMT System or Embed-Embed-Full-Vocab SMT System. The bitokens are then optionally replaced by their cluster ids (they are replaced by cluster ids when estimating the score from Embed-Brown Coarse Bi-LM 400(400, 400), Embed-Embed Coarse Bi-LM 400(400, 400) & Embed-Embed Coarse Bi-LM 400(|V_f|, |V_e|), and are not replaced in case of Embed Coarse Bi-LM (400, 400)). Once we have the set of required phrase of tokens, we can then score them with our coarse Bi-LMs.

In the next section we can describe the results of the baseline system and our approaches.

### 4.4 Results

For our experiments as mentioned earlier we use the Moses decoder [42]. We implemented a stateful feature (Section 4.3) function\(^4\) to score each partial hypothesis with the coarse LMs and Bi-LMs in the baseline system and our approaches. For all our experiments, we use a 5-gram English language model. Table 4.2 shows the statistics of our language model. For training the translation table, we use Moses to perform the following step on the Zh-En training dataset (Table 4.1):

- Train alignments using GIZA++[53]. By default Moses will train IBM Model 4 alignments with `grow-diag-final-and` as the merging heuristic.

- Perform phrase extraction and scoring of features. For our experiments, we set the `max-phrase-length` setting to 7 and `distortion-limit` as -1 (We borrowed these settings from [61]).

• Create lexicalised reordering model using the heuristic `msd-bidirectional`.

This will create `moses.ini` which contains the settings of all the default features of Moses. We modify the `moses.ini` file and add information about coarse LM and coarse Bi-LMs.

For all our experiments, we tune the feature weights using Pairwise Ranked Optimization (PRO) [33] and BLEU score as the metric for optimization.

Table 4.3 shows BLEU scores and Translation Error Rate (TER) for the four systems. All three of our approaches consistently outperform the baseline system. Embed-Embed-Reduced-Vocab SMT System achieves an increase of 1.4 BLEU points over the baseline system. In the table we also report the p-value of our results. The p-value shows how statistically significant our results are. To be statistically significant, the p-value should be < 0.05, and approach 2 achieves a p-value of 0.00. We used `multeval` [15] to calculate the BLEU score, TER score and p-value. When looking at TER, Embed-Brown SMT System achieves a reduction of 6.5% and Embed-Embed-Reduced-Vocab SMT System achieves a 5.6% reduction in TER. Since, when looking at BLEU score Embed-Embed-Reduced-Vocab SMT System has statistically significant results and the difference in TER reduction between Embed-Brown SMT System and Embed-Embed-Reduced-Vocab SMT System is only 0.9%, we deem Embed-Embed-Reduced-Vocab SMT System as the winning candidate out of all three of our approaches and it is based on the main hypothesis of this thesis that Bi-LMs estimated from bilingual word embeddings trained from parallel data are useful for SMT.

<table>
<thead>
<tr>
<th>Metric</th>
<th>SMT System</th>
<th>Score</th>
<th>p-value</th>
</tr>
</thead>
<tbody>
<tr>
<td>BLEU ↑</td>
<td>Baseline</td>
<td>23.0</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td>Embed-Brown</td>
<td>23.4</td>
<td>0.33</td>
</tr>
<tr>
<td></td>
<td><strong>Embed-Embed-Reduced-Vocab</strong></td>
<td><strong>24.4</strong></td>
<td><strong>0.00</strong></td>
</tr>
<tr>
<td></td>
<td>Embed-Embed-Full-Vocab</td>
<td>23.1</td>
<td>0.82</td>
</tr>
<tr>
<td>TER ↓</td>
<td>baseline</td>
<td>77.5</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td><strong>Embed-Brown</strong></td>
<td><strong>71.0</strong></td>
<td><strong>0.00</strong></td>
</tr>
<tr>
<td></td>
<td>Embed-Embed-Reduced-Vocab</td>
<td>71.9</td>
<td>0.00</td>
</tr>
<tr>
<td></td>
<td>Embed-Embed-Full-Vocab</td>
<td>73.0</td>
<td>0.00</td>
</tr>
</tbody>
</table>

Table 4.3: Results comparing the baseline system 4.1 and three of our proposed SMT systems 4.2
4.5 Summary

In this chapter we described the steps we took to implement the baseline system [61] and our three approaches. We also describe in detail how we chose the parameters when creating the bilingual word embeddings and bitoken embeddings. Finally we show that Embed-Reduced-Vocab SMT System outperforms the baseline by 1.4 BLEU points and other proposed SMT systems by almost 0.1 to 0.4 BLEU points. We also show that our results are statistically significant. In the next chapter we will conclude our thesis and describe the future work that could possibly be done to extend the work done in this thesis.
Chapter 5

Conclusion & Future Work

5.1 Conclusion

We started the thesis by introducing statistical machine translation and we gave a brief overview about the steps involved in training a phrase-based statistical machine translation system using a parallel corpus. When decoding a source sentence to translate it into a target language, we show that the decoder has very little information about source words outside the current phrase pair in consideration. Little work has been done in the literature to incorporate information about source words outside the current phrase pair in consideration. In the quest of providing the decoder more information from words in the source sentence, [51] introduced bilingual language models. They achieved statistically significant gains by replacing words with part-of-speech tags and then creating their bilingual language models. [61] extended their work and showed that significant gains can be achieved by using a combination of coarse language models and coarse bilingual language models. [61] made their language models coarse by clustering their corpora using mkcls (Brown clustering), a popular monolingual word clustering algorithm. Their approach depends on using word alignments and monolingual clusters to create the bitokens. This approach will not be able to capture the information provided by words which are not direct translations of each other as captured by word alignments. In order to include information from words which are not direct translations of each other, we proposed a novel approach of using word embeddings and bilingual word embeddings to create coarse language models and bilingual language models.

In this thesis we present three new systems of using word embeddings and bilingual word embeddings to create coarse language models and bilingual language models. In all three systems we create bilingual word embeddings using BiCVM [31] and cluster these embeddings using greedo [62]. The clusters are used to augment the Chinese-English parallel corpus by replacing the words with their corresponding cluster ids. These coarse corpora are used to create coarse language models in all three of our systems. In two of
our systems we use coarse corpora and the alignments between the words to create bitoken sequences, whereas in the third system we use the Chinese-English parallel corpus and the alignments to create the bitoken sequences. In the first two systems, we create coarse bilingual language models using the bitoken sequences themselves. In all three systems, we further cluster the bitokens. We experiment with clustering the bitokens using \texttt{mkcls} and also by creating bitoken embeddings using \texttt{word2vec}. The bitoken embeddings are clustered again using \texttt{greedo}. The clusters are then used to augment the bitoken sequences with the cluster ids of the bitokens to create coarse bitoken sequences. The coarse bitoken sequences are used to create coarse bilingual language models.

In our experiments we compare our systems to a baseline system, which is an implementation of [61]. We show that all three of our systems outperform the baseline system. When looking at the BLEU score, the second system which uses coarse bilingual language models by utilizing bilingual word embeddings and bitoken embeddings performs the best and when looking at TER score, the first system which uses \texttt{mkcls} to cluster the bitokens performs the best. The second system has a \(\text{p-value} = 0.00\) when looking at BLEU score, that is the improvements are statistically significant, hence we deem it as the winning system out of all three of our systems. Overall, we achieve an improvement of 1.4 BLEU points compared to our baseline system.

5.2 Future Work

5.2.1 Clustering of Embeddings

In our systems, we cluster the embeddings using \texttt{greedo} [62]. \texttt{greedo} creates a hierarchical cluster of the embeddings by measuring the euclidean distances. As \texttt{greedo} and \texttt{mkcls} are based on the [9] model, it made it easier to compare our systems to the baseline system.

Word embeddings show unique properties when we measure their similarity using cosine similarity. Word embeddings which are have a high cosine similarity tend to be semantically similar. Based on this idea, we would like to experiment with clustering algorithms that use cosine similarity as their distance measure. Specifically, we would like to experiment with using spherical k-means clustering [34] as it uses cosine similarity as its distance measure.

5.2.2 Extending Bi-LMs to Translation Model

Even though Bi-LMs are language models, they act more as translation models as they do not model the fluency of target language but model the translation of source words. Based on this idea, we would like to extend the idea of using word embeddings in translation model. In phrase-based SMT, the translation model consists of phrase pairs. One way to modify the translation model to include embeddings would be to have a translation model that contains phrase embedding pairs instead of words in phrases. We would like to test
this new embeddings based translation model as a standalone translation model and also as an additional translation model that complements the standard word based translation model.
Bibliography


