Bilingual Language Models using Word Embeddings for Machine Translation

by

Jasneet Singh Sabharwal

B.Tech., Guru Gobind Singh Indraprastha University, 2010

Thesis Submitted in Partial Fulfillment of the
Requirements for the Degree of
Master of Science

in the
School of Computing Science
Faculty of Applied Sciences

SIMON FRASER UNIVERSITY
Summer 2016

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 - Big Data Systems 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) refers 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 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. We propose to use bilingual word embeddings to reduce the vocabulary of the bitokens. Bilingual word embeddings are a low dimensional representation of words in two languages such that words in either language that are similarly distributed are placed nearby in an abstract vector space. Our approach outperforms previous work using Bi-LMs with an increase of 1.4 BLEU points in our experiments.

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

Table of Contents

List of Tables

List of Figures

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 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 [Koehn et al., 2003] 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 [Lopez, 2008]:

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 [Niehues et al., 2011] 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, [Niehues et al., 2011] replaced words in the corpus with their part-of-speech tags and then created the bitokens. [Stewart et al., 2014] 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 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 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*.

Figure 1.1: A sample alignment between a Spanish sentence and English sentence [Koehn, 2009]

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 [Och and Ney, 2003], *(a) statistical alignment models*, and *(b) heuristic models*. In this thesis we only focus on *statistical alignment models* 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 $f_1^J = f_1, ..., f_j, ..., f_J$ and a target language string $e_1^I = e_1, ..., e_i, ..., e_I$. 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 [Och and Ney, 2003], the relation between translation model and statistical alignment model is

$$P(f_1^J|e_1^I) = \sum_{a_1^J} P(f_1^J, a_1^J|e_1^I) \tag{1.1}$$

To account for the unknown parameters θ learnt from training data, the statistical alignment model is represented as $p_\theta(f_1^J, a_1^J|e_1^I)$. For each sentence pair $(\mathbf{f_n}, \mathbf{e_n})$, for $n =$

$1, ..., N$, where N is the size of parallel corpus, the alignment is denoted as $\mathbf{a} = a_1^J$. We find the unknown parameters θ by maximizing the likelihood on the parallel corpus:

$$\hat{\theta} = \underset{\theta}{\operatorname{argmax}} \prod_{n=1}^{N} \sum_{\mathbf{a}} p_\theta(\mathbf{f_n}, \mathbf{a}|\mathbf{e_n}) \tag{1.2}$$

To perform the maximization, Expectation Maximization (EM) [Dempster et al., 1977] or some variant of it is used. After finding the unknown parameters, the best alignment for a pair of sentences can be calculated as:

$$\hat{a}_1^J = \underset{a_1^J}{\operatorname{argmax}}\, p_{\hat{\theta}}(f_1^J, a_1^J|e_1^I) \tag{1.3}$$

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 [Taskar et al., 2005], where nodes correspond to words in the two parallel sentences. Aside from graph matching algorithms, discriminative approaches also use perceptron algorithm [Moore et al., 2006], support vector machines [Cherry and Lin, 2006], conditional random fields [Blunsom and Cohn, 2006] or neural networks [Ayan et al., 2005].

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 translated. To get the translation of source phrase, a phrase table is learnt from the parallel corpora. [Koehn, 2009] 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 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 approach outlined in [Koehn et al., 2003] 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++ [Och and Ney, 2003] to get the bidirectional alignments.
- Using the bidirectional alignments from step 1, extract all posible phrase pairs which are consistent with the alignments [Och et al., 1999, Koehn et al., 2003]. 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 posible phrase pairs that are consistent with the alignments shown in Fig 1.1.
- Assign probabilities to phrase pairs using their relative frequencies:

$$\phi(\bar{f}|\bar{e}) = \frac{count(\bar{e}, \bar{f})}{\sum_{\bar{f}_i} count(\bar{e}, \bar{f}_i)} \tag{1.4}$$

  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

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 tell us that the probability of former sentence should be higher than the latter, that is, $p_{LM}(\textit{this is a house}) > p_{LM}(\textit{this a house is})$. From this example, we notice that language

Source Phrase	Target Phrase
Maria	Mary
no	did not
daba una bofeta	slap
a la	the
bruja	witch
verde	green
Maria no	Mary did not
no daba una bofetada	did not slap
daba una bofetada a la	slap the
bruja verde	green witch
Maria no daba una bofetada	Mary did not slap
no daba una bofetada a la	did not slap the
a la burja verde	the green witch
Maria no daba una bofetada a la	Mary did not slap the
daba una bofetada a la burja verde	slap the green witch
no daba una bofetada a la burja verde	did not slap the green witch
Maria no daba una bofetada a la burja verde	Mary did not slap the green witch

Table 1.1: All possible phrase pairs consistent with the alignments shown in Fig.1.1

Source Phrase	Target Phrase	Feature Scores	Alignments
in europa	in europe	0.829007 0.207955 0.801493 0.492402	0-0 1-1

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

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, $p_{LM}(\textit{I am going home}) > p_{LM}(\textit{I am going house})$.

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 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:

$$p(W) = p(w_1, w_2, ..., w_n) \tag{1.5}$$

Using chain rule, we can break this down:

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

Here, we have broken down the probability of a sentence into 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.

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

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:

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

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

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

Even though we use a large monolingual corpora to train the language model, we still cannot cover every word and it's usage. To tackle the problem of unseen words, the literature describes various smoothing techniques like *add-one smoothing*, *add-α smoothing*, *Good-Turing smoothing* [Good, 1953], *Witten-Bell smoothing* [Witten and Bell, 1991], *Kneser-Ney smoothing* [Kneser and Ney, 1995], etc.

1.1.4 Decoder

[Koehn et al., 2003] states the mathematical model for translation as $p(e|f)$. The job of the decoder is to find the translation e_{best} with the highest probability. This can be mathematically formulated as:

$$e_{best} = argmax_e p(e|f) \tag{1.10}$$

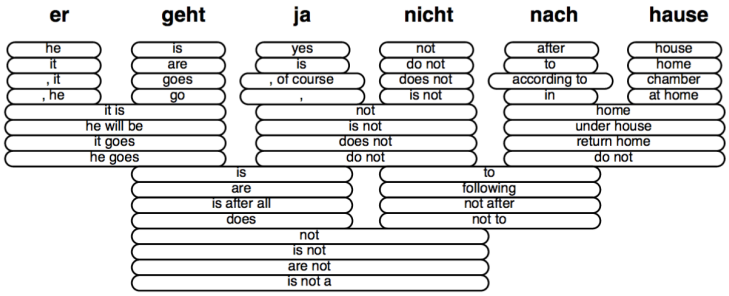

Figure 1.2: A sample German source sentence broken into possbile phrases and the top four translation options for each of the source phrase [Koehn, 2009]

When the decoder proposed by [Koehn et al., 2003] 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

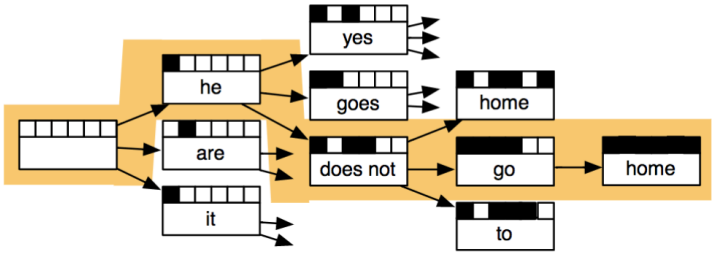

Figure 1.3: A sample German source sentence broken into possbile phrases and the top four translation options for each of the source phrase [Koehn, 2009]

A limitation of this approach is that for each source sentence, exponential number of hypothesis are generated. Searching through these hypothesis is an NP-complete problem [Knight, 1999]. To tackle this problem, [Koehn et al., 2003] proposed using the hypothesis recombination strategy as in [Och et al., 2001]. Along with this, hypothesis are also pruned by comparing their current score and the future score proposed by [Koehn et al., 2003]. Histogram pruning and threshold pruning proposed by [Koehn, 2004] 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 it's process. In addition to the feature scores from 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.

[Koehn et al., 2003] proposed a weighted model comprising of the phrase translation model ($\phi(\bar{f}|\bar{e})$), a reordering model (d), and language model ($p_{LM}(e)$), which is mathematically formulated as:

$$e_{best} = argmax_e \prod_{i=1}^{I} \phi(\bar{f}_i|\bar{e}_i)^{\lambda_\phi} d(a_i - b_{i_1} - 1)^{\lambda_d} \prod_{i=1}^{|e|} p_{LM}(\mathbf{e})^{\lambda_{LM}} \tag{1.11}$$

Here a_i and b_{i_1} 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. λ_ϕ, λ_d and λ_{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:

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

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:

$$\begin{aligned} p(e,a|f) = exp[&\lambda_\phi \sum_{i=1}^{I} \log \phi(\bar{f}_i|\bar{e}_i) \\ &+ \lambda_d \sum_{i=1}^{I} \log d(a_i - b_{i-1} - 1) \\ &+ \lambda_{LM} \sum_{i}^{|e|} \log p_{LM}(e_1|e_1 ... e_{i-1})] \end{aligned} \tag{1.13}$$

This formulation allows us to add more independent feature functions, that is, feature functions that are independent of other feature functions. In practice, Moses [Koehn et al., 2007], a popular SMT toolkit that we use in our work, uses 15 features which are as follows:

- Unknown word penalty
- 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 have a weight associated with them. 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 **Bi**lingual **E**valuation **U**nderstudy (BLEU) [Papineni et al., 2002]. BLEU compares the output translation with reference translations according to the equation:

$$BLEU_{score} = BPexp \sum_{i=1}^{n} w_i \log(precision_i) \tag{1.14}$$

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

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

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

system translation. The beauty of this precision based metric is that it allows the use of multiple reference translations. Note, reference translations are human generated translations for the source sentence under test.

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 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:

$$p_n = \frac{\sum_{C \in SystemTranslation} \sum_{n-gram \in C} Count_{clip}(n-gram)}{\sum_{C\prime \in SystemTranslation} \sum_{n-gram\prime \in C\prime} Count(n-gram\prime)} \tag{1.16}$$

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) [Och and Ney, 2003] is the most widely used tuning algorithm. Lattice MERT [Macherey et al., 2008] is a variant of MERT that uses lattices instead of n-best list. Pairwise ranked optimization (PRO) [Hopkins and May, 2011] works by ranking learning the weight set that ranks the n-best list in the same order as BLEU. Batch MIRA [Cherry and Foster, 2012] 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 [Cherry and Foster, 2012] 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, has very little information from source words outside the current phrase pair. [Stewart et al., 2014] 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, [Niehues et al., 2011] introduced part-of-speech based *bilingual language models* (Bi-LMs) which was extended by [Stewart et al., 2014]. 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. [Niehues et al., 2011] 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, [Stewart et al., 2014] used MKCLS [Och, 1995] 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 corporas 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 [Brown et al., 1992].

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 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 [Harris, 1954]. 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 [Brown et al., 1992] and [Pereira et al., 1993] are the two most dominant approaches. Hidden Markov Models can also be used to induce clustering on words [Huang and Yates, 2009]. 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 [Deerwester et al., 1990], canonical correlation analysis [Dhillon et al., 2011], neural-networks [Collobert and Weston, 2008, Huang et al., 2012, Mikolov et al., 2013a, Mikolov et al., 2013c].

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 lan-

guages, 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 *lake* and Chinese word 潭 *(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 n-dimension space. And words which are semantically similar to 潭 *(deep pond)* and possibly direct translations of *lake* would also be close to each other in that n-dimension space. For word embeddings, we measure semantic similarity by measuring the cosine similarity between two word embeddings. It is formally defined as:

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

Here, $\mathbf{A}$ and $\mathbf{B}$ are two vectors of size n.

Bilingual word embeddings have been created by using various techniques like latent dirichlet allocation and latent semantic analysis [Boyd-Graber and Blei, 2012, Zhao and Xing, 2006], canonical correlation analysis [Faruqui and Dyer, 2014], neural-networks [Klementiev et al., 2012, Zou et al., 2013, Mikolov et al., 2013b, Hermann and Blunsom, 2014, Chandar A P et al., 2014]. In the next section we discuss the reasons for choosing the algorithms to create monolingual and bilingual word embeddings.

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 *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 [Dhillon et al., 2011, Faruqui and Dyer, 2014] and neural networks [Collobert and Weston, 2008, Huang et al., 2012, Klementiev et al., 2012, Mikolov et al., 2013a, Mikolov et al., 2013c, Mikolov et al., 2013b, Zou et al., 2013, Hermann and Blunsom, 2014, Chandar A P et al., 2014]. Neural network approaches to create word embeddings have been widely adopted due to the following advantages:

- Training the networks can be done using parallel processing and distributed processing.
- Graphical processing units (GPU) can be utilized for faster training.
- 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.

- 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 neural network based approach. For creating monolingual word embeddings we utilize **Word2Vec** [Mikolov et al., 2013a, Mikolov et al., 2013c, Mikolov et al., 2013b], 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, [Zou et al., 2013] utilize sentence aligned parallel corpora and their alignments to induce the embeddings whereas [Hermann and Blunsom, 2014] only utilizes a sentence aligned parallel corpora (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 [Berg-Kirkpatrick et al., 2010] for Chinese>English parallel corpus has an alignment error rate of 30%), using word embeddings that require alignments [Zou et al., 2013] 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 [Hermann and Blunsom, 2014] to create the bilingual word embeddings.

When 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 **WordEmbeddingsViz**[1], 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) [Van der Maaten and Hinton, 2008] 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 dimension 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

[1] WordEmbeddingsViz: Tool to visualize bilingual word embeddings `https://github.com/anoopsarkar/WordEmbeddingsViz`

probability. t-SNE defines a similar probability distribution over pairs of objects in lower dimension 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.

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)

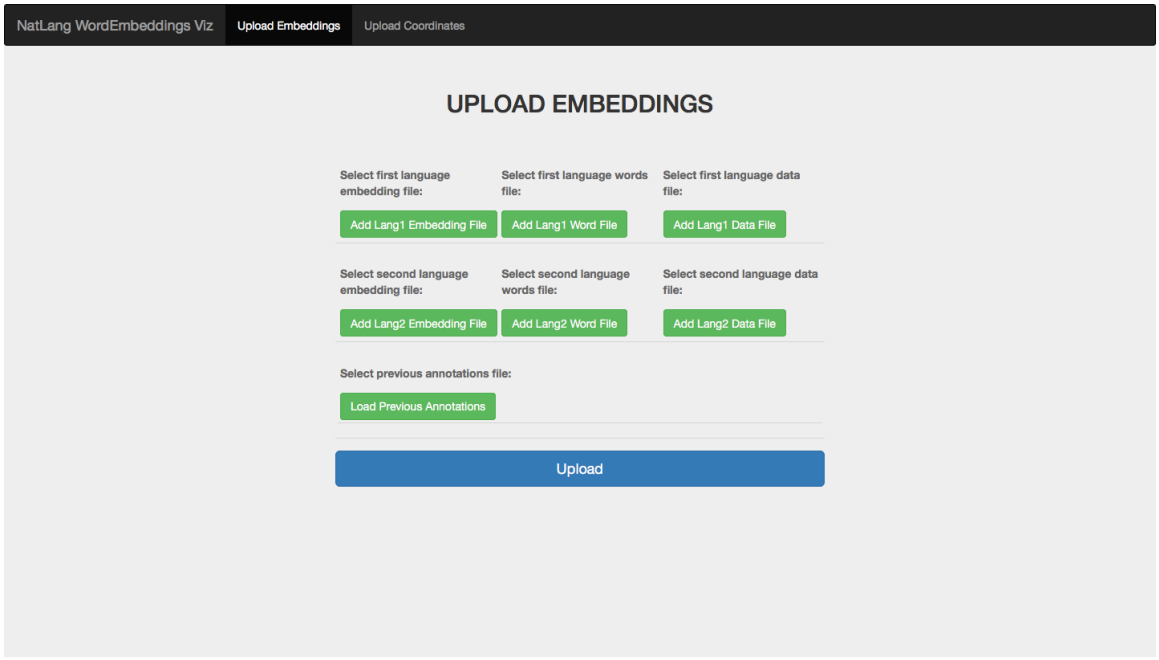

Figure 2.1: WordEmbeddingsViz upload screen: Here, the user can upload bilingual word embeddings, word list, training corpus and alignments (optional)

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 top 1000 words by their occurance count for *verb, noun, adjective & adverb* POS tags. For our work, we extracted POS tags for English data using Stanford POS tagger [Toutanova et al., 2003].

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 one or more Chinese words in the neighbourhood that are possible translation 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, braodcast & 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.

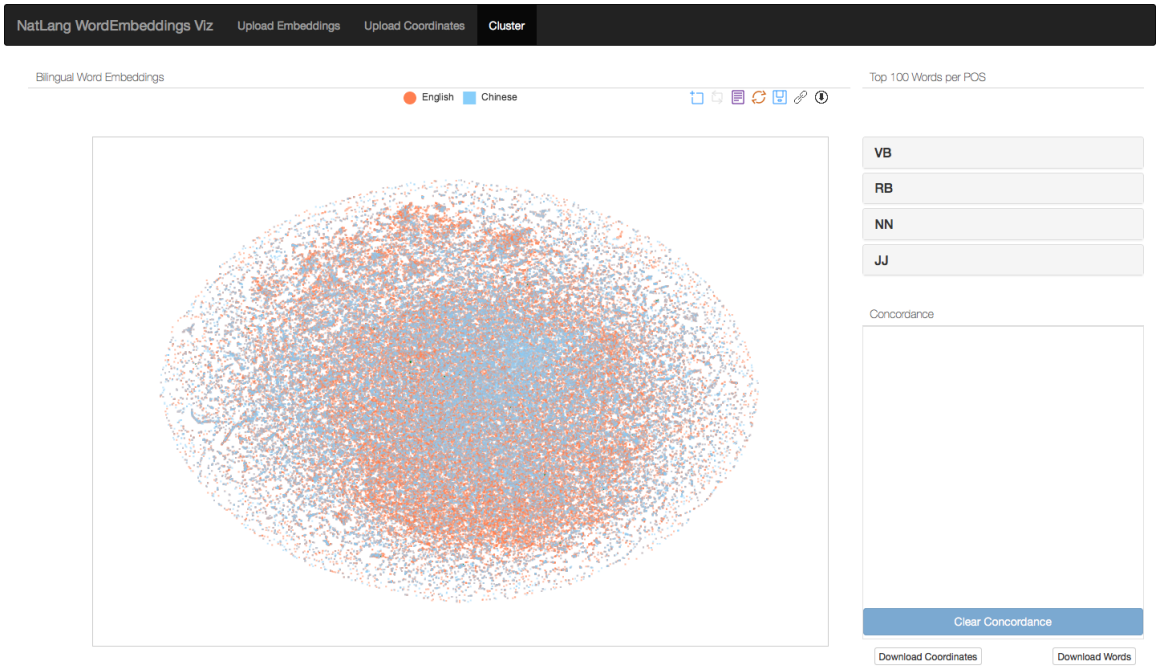

Figure 2.2: WordEmbeddingsViz: A scatter plot of word embeddings of Zh-En parallel corpus. Orange squares represent English words and blue circles represent Chinese words.

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.

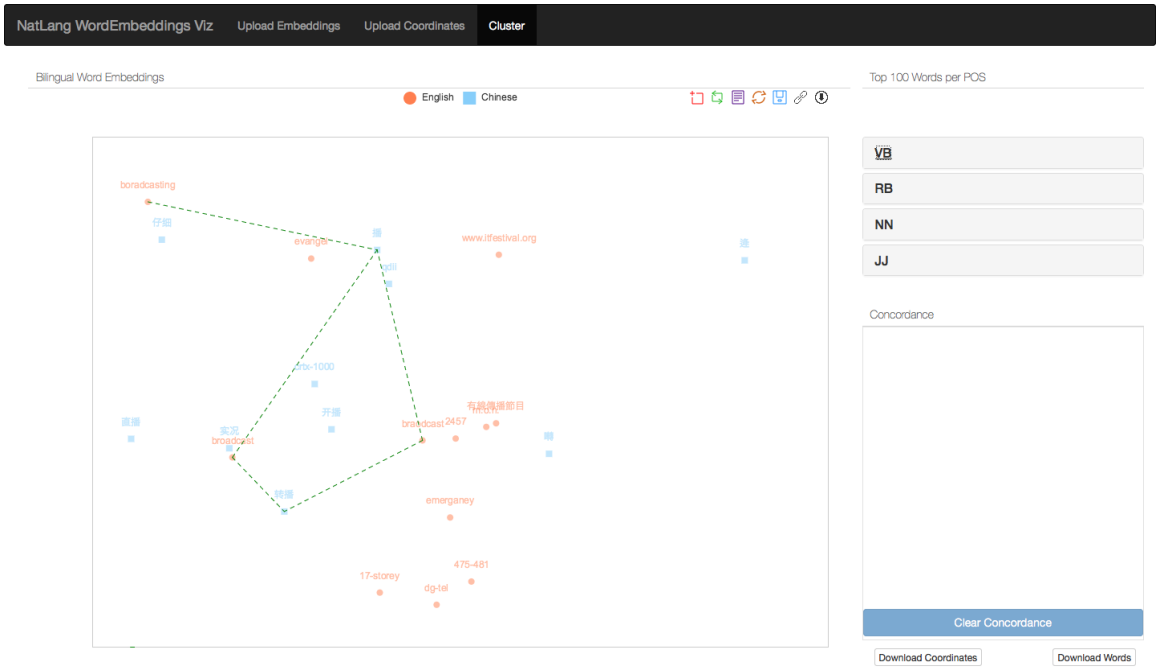

Figure 2.3: WordEmbeddingsViz: Alignments of English words *broadcast, braodcast, broadcasting* with their Chinese counterparts.

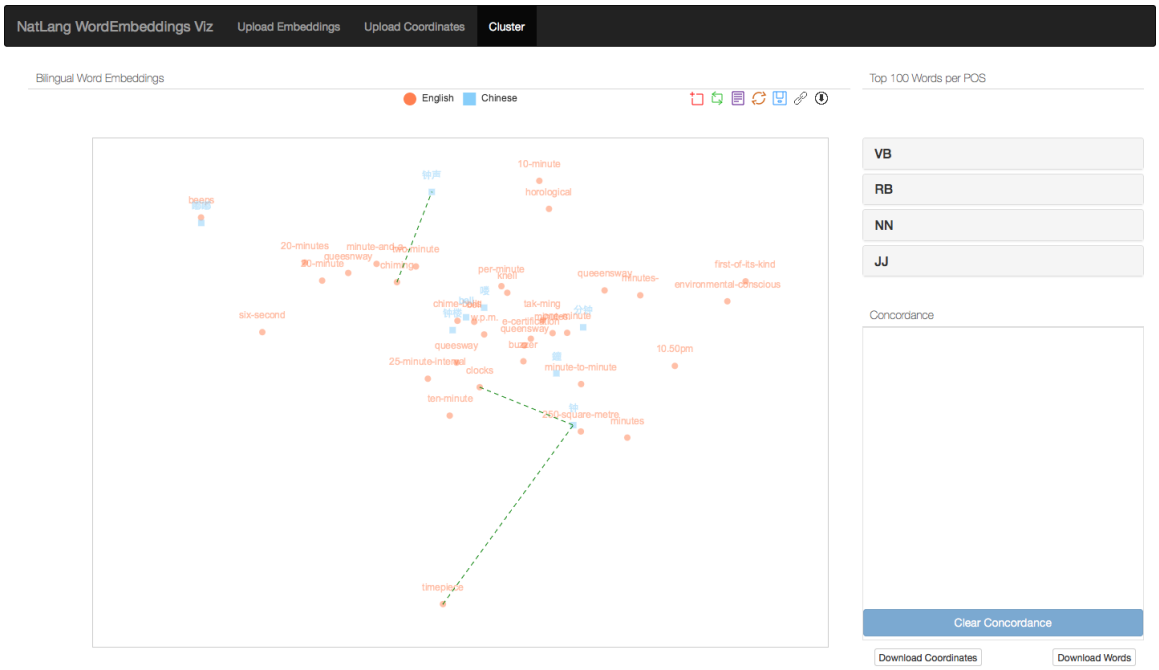

Figure 2.4: WordEmbeddingsViz: Alignments of English words *clocks, timepiece & chiming* with their Chinese counterparts.

2.4 Summary

In this chapter we introduced word embeddings and bilingual word embeddings. We also release 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.

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, 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 following example:

Maria no daba una bofetada a la burja 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, considerable amount of work has been done in the past. In this thesis, we extend the work of [Niehues et al., 2011] and [Stewart et al., 2014] 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:

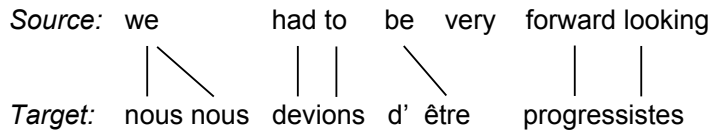

Figure 3.1: Source and target sentences with their alignments for creating bitokens [Stewart et al., 2014]

Using the parallel sentences and their alignments shown above we will create a bitoken sequence. When creating hte bitokens, we want to make sure that all the target words are used. In the exmample 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:

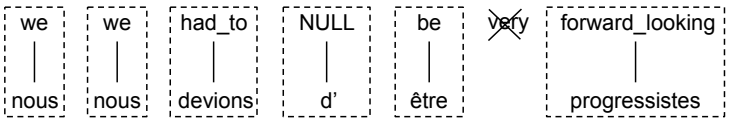

Figure 3.2: Bitokens created from the parallel sentences and their alignments shown in Fig. 3.1

More formaly, given a pair of sentences $e_1^I = e_1...e_I$, $f_1^J = f_1...f_J$ and alignments $A = \{(i, j)\}$, the bitokens are:

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

Number of Target Words	Number of Bitokens
152318	3827728

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

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 bilingual language model, formalized as follows:

$$p(e_1^I, f_1^J, A) = \prod_{j=1}^{J} P(b_j|b_{j-1}...b_{j-n}) \tag{3.2}$$

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 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 [Stewart et al., 2014], 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, [Niehues et al., 2011] 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, [Niehues et al., 2011] replaced both the source and target words in their Arabic>English SMT with the corresponding Part-of-Speech tags. [Stewart et al., 2014] extended this idea and replaced both source and target words with cluster ids generated using **mkcls** [Och, 1995]. [Stewart et al., 2014] 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 bitoken sequences for 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.

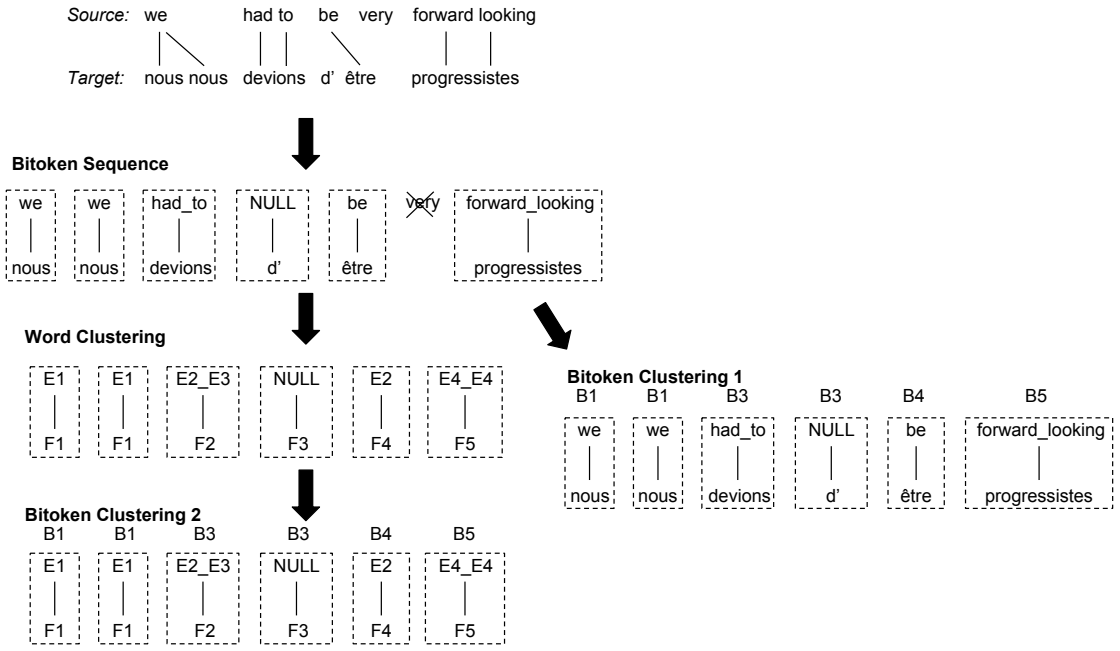

Figure 3.3: Creating bitokens, word clusters and bitoken clusters [Stewart et al., 2014]

[Ammar et al., 2013, Bisazza and Monz, 2014] showed that coarse LMs along with the standard LM are particularly effective for morpholigically rich languages. Motivated by this, [Stewart et al., 2014] used a combination of coarse LMs and coarse Bi-LMs in their experiments, which are as follows:

- Coarse LM created by augmenting the target language corpus with word cluster ids of size 100. (Coarse LM 100_{tgt})
- Coarse LM created by augmenting the target language corpus with word cluster ids of size 100. (Coarse LM 1600_{tgt})
- Augment the source and target language corpus with word cluster ids of size 400 and 400 respectively. This augmented corpus along with their alignments was used to create bitoken sequences. These bitoken sequences were used to create the following two coarse Bi-LMs:

- Use the bitoken sequences to train a coarse Bi-LM. (Coarse Bi-LM (400_{src}, 400_{tgt}))
- Cluster the bitoken sequences to get bitoken cluster ids of size 400. Augment the corpus using these bitoken cluster ids to create clustered bitoken sequences. Use these sequences to train a coarse Bi-LM. (Coarse Bi-LM $400_{bi}(400_{src}, 400_{tgt})$).

Figure 3.4 shows in detail the steps taken by [Stewart et al., 2014] to create coarse LM and coarse Bi-LMs. The four coarse LMs and Bi-LMs are then used as additional features in phrase-based SMT. We use this system as a baseline to compare with our results. For our work, we used Moses decoder [Koehn et al., 2007] and added additional stateful feature functions to use these coarse LMs and Bi-LMs. Also, to create the LMs and Bi-LMs we used **SRILM** [Stolcke et al., 2011].

In this thesis we extend the work of [Stewart et al., 2014] 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 [Och, 1995] is one of the most widely used word clustering toolkit. Motivated by Brown Clustering [Brown et al., 1992], **mkcls**[1] implements an ensemble of optimzers and merges their results to cluster words into the provided number of classes. **mkcls** can only cluster monolingual corpus and it performes strongly in that aspect [Blunsom and Cohn, 2011].

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 潭 *(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 潭 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 [Hermann and Blunsom, 2014] to create bilingual word embeddings for the words in our sentence aligned parallel corpus. To create coarse LMs and Bi-LMs, as done by [Stewart et al., 2014], we need to cluster these embeddings. As mentioned earlier, in our baseline system, we used **mkcls** to cluster the words in our

[1]Understanding **mkcls** by Dr. Chris Dyer: `http://statmt.blogspot.ca/2014/07/understanding-mkcls.html`

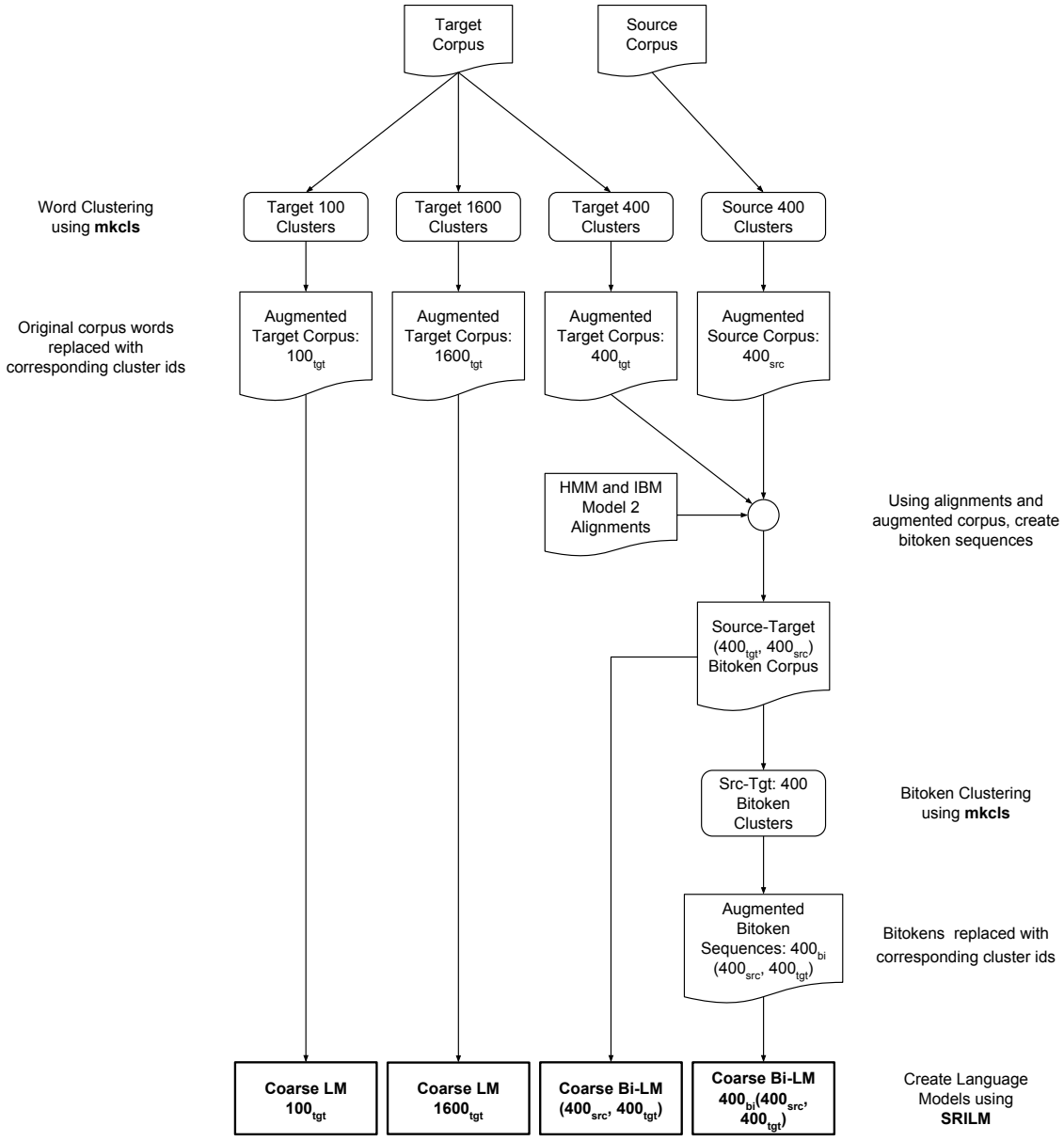

Figure 3.4: Creating coarse LMs and coarse Bi-LMs [Stewart et al., 2014]

parallel corpus. To cluster word embeddings, we used **greedo** [Stratos et al., 2014], a bottom-up hierarchical clustering algorithm for clustering low-dimensional representation of words under the [Brown et al., 1992] model. [Stratos et al., 2014] show that the clusters created by **greedo** recovers clusters which are of comparable quality to the algorithm of [Brown et al., 1992]. Using **greedo** gives us the opportunity to compare our approach to the baseline system without any modifications to the number of clusters because **mkcls** is also creates Brown clusters.

To use the embeddings and their clusters for creating coarse LMs and coarse Bi-LMs, we propose three approaches.

Approach 1

Once we create the bilingual word embeddings, we can then use them to create the LMs. Figure 3.5 shows the steps that we took to create our coarse LMs and Bi-LMs. As we did in the baseline, we had clustered the corpora using **mkcls**, but now we cluster the bilingual word embeddings using **greedo** to create the following clusters:

- Cluster the target language embeddings into clusters of size 100, 400 and 1600.
- Cluster the source language embeddings into cluster of size 400.

Using the target language embedding clusters of size 100 and 1600, we augment the target language corpus by replacing the words with their cluster ids. This augmented data is then used to create coarse LMs *Coarse LM* 100_{tgt} and *Coarse LM* 1600_{tgt} correspondingly using the language modelling toolkit **SRILM**.

Using the clusters of size 400 for Target and Source, we augment the parallel corpus by replacing the words with their cluster ids. Using the alignments between the source and target words on the original parallel corpus, we create bitoken sequences. *Coarse Bi-LM* $(400_{src},\ 400_{tgt})$ is estimated using the bitoken sequences. The bitoken sequences are further clustered to reduce the vocabulary using **mkcls**. The size of the cluster in this case is also 400 as done in the baseline system. The bitokens in bitoken sequences are replaced with the new cluster ids. This augmented bitoken sequences is finally used to estimate *Coarse Bi-LM* $400_{bi}(400_{src},\ 400_{tgt})$.

Approach 2

Figure 3.6 shows our second idea to create coarse LMs and Bi-LMs. Based on Approach 1 3.2.1, we first create bilingual word embeddings using [Hermann and Blunsom, 2014] for the sentence aligned parallel corpus used for training the SMT system. The bilingual word embeddings are clustered using **greedo** to create the same clusters as in Approach 1, which are:

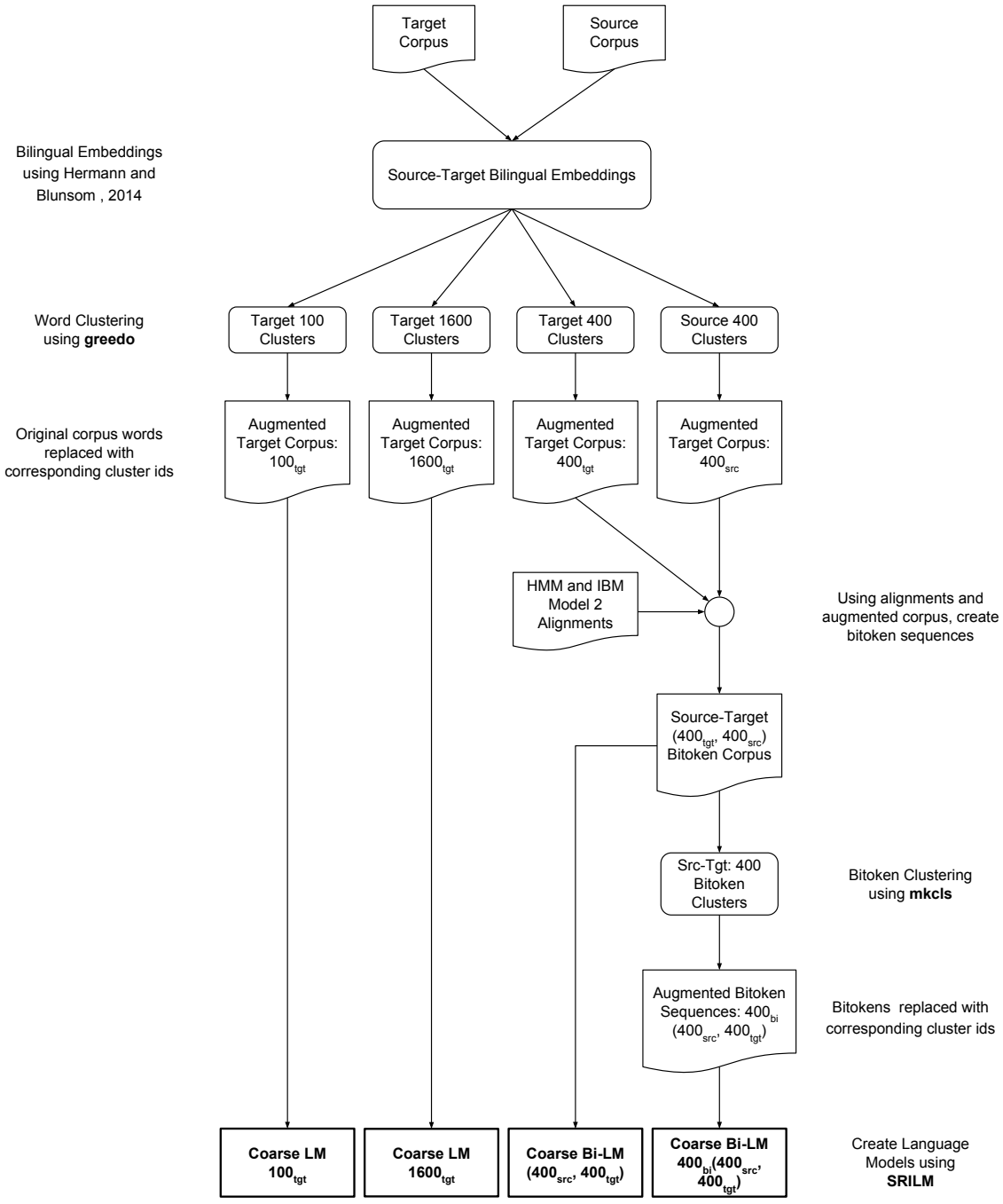

Figure 3.5: Approach 1 for creating Coarse LMs and Coarse Bi-LMs

- Target language word embedding clusters of size 100, 400 and 1600.
- Source language word embeddings cluster of size 400.

The clusters are then used to augment the source and target language corpus by replacing the words with their cluster ids. Augmented target language corpus 100_{tgt} and 1600_{tgt} are used to estimate coarse LMs *Coarse LM* 100_{tgt} and *Coarse LM* 1600_{tgt}. Augmented corporas 400_{tgt} and 400_{src} along with bidirectional word alignments of the original corpora are used to create bitoken sequences *(*400_{src}, 400_{tgt}*)*. *Coarse Bi-LM* (400_{src}, 400_{tgt}) is estimated using the bitoken sequences.

Using **word2vec** [Mikolov et al., 2013a], we create bitoken embeddings for bitoken sequence corpus ($400_{src}, 400_{tgt}$). As in our baseline system (Figure 3.4) and approach 1 (Figure 3.5) we had clustered the bitoken sequences using **mkcls**, we again create clusters of size *400* by clustering the bitoken embeddings using **greedo**. The bitokens in bitoken sequence corpus (400_{src}, 400_{tgt}) are replaced with their cluster ids from **greedo**. The augmented bitoken sequences $400_{bi}(400_{src}, 400_{tgt})$ are used to estimate coarse Bi-LM *Coarse Bi-LM* $400_{bi}(400_{src}, 400_{tgt})$ using **SRILM**.

Approach 3

In approach 3 (shown in Figure 3.7), we estimate the coarse LMs *Coarse LM* 100_{tgt} and *Coarse LM* 1600_{tgt} using the same procedure as in approach 1 (3.2.1) and approach 2 (3.2.1).

To create coarse Bi-LM, we first create bitoken sequences using the original sentence aligned parallel corpus along with their alignments. Note, we did not cluster the data before creating the bitoken sequences. Using **word2vec**, we create bitoken embeddings from the bitoken sequences. The bitoken embeddings are clustered using **greedo**, with the number of clusters being 400, as done in previous approaches. The bitokens in bitoken sequences are replaced with their corresponding cluster ids to create an augmented corpus $400_{bi}(|V|_{src}, |V|_{tgt})$. Here $|V|_{tgt}$ denotes the vocabulary size of target language corpus and $|V|_{src}$ is the vocabulary size of source language corpus. **SRILM** is then utilized to estimate *Coarse Bi-LM* $400_{bi}(|V|_{src}, |V|_{tgt})$ using the augmented bitoken corpus $400_{bi}(|V|_{src}, |V|_{tgt})$.

As compared to baseline, approach 1 and approach 2, approach 3 only has one coarse Bi-LM while others have two.

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 about other approaches in the literature for introducing information about source words.

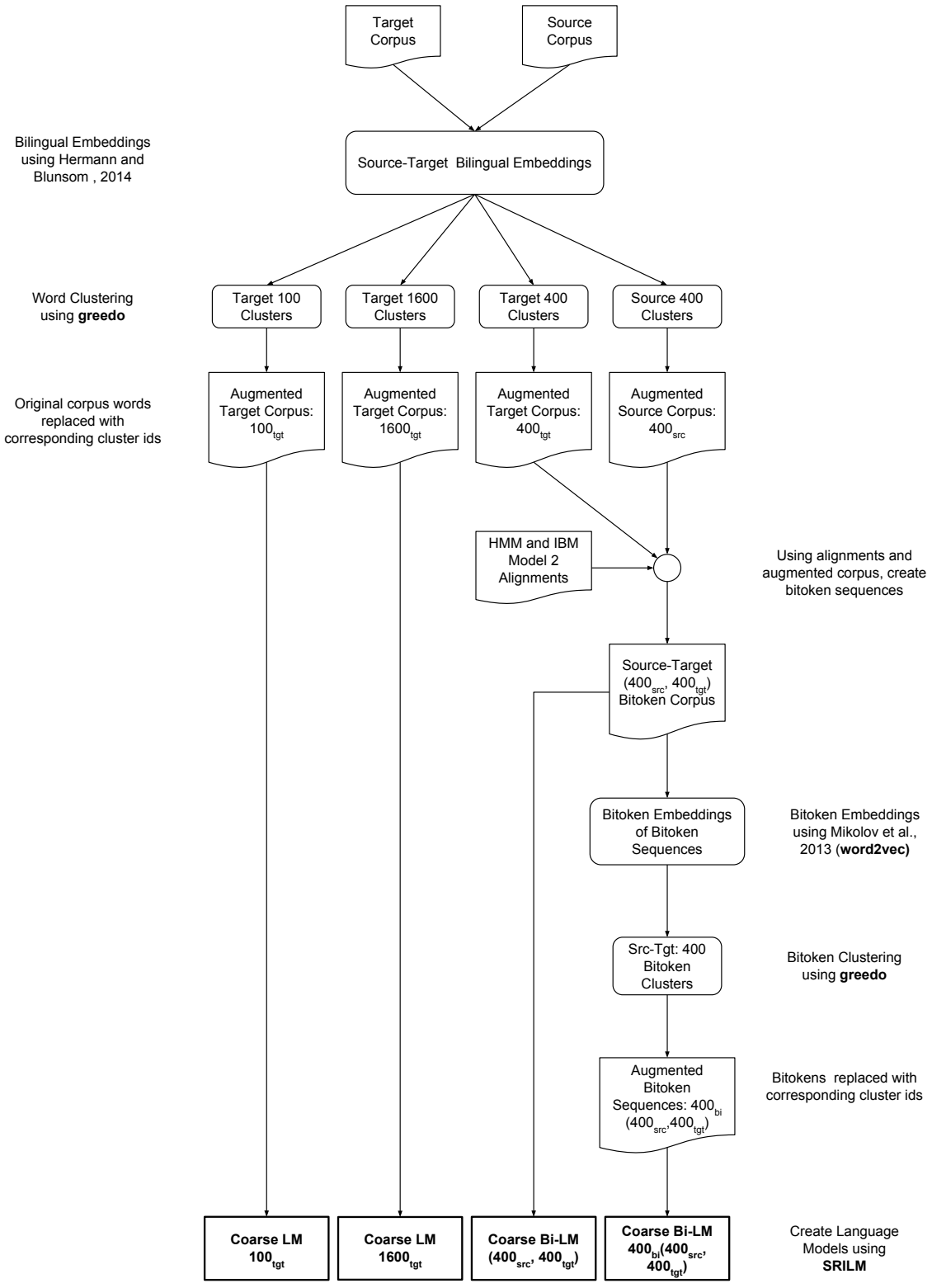

Figure 3.6: Approach 2 for creating Coarse LMs and Coarse Bi-LMs

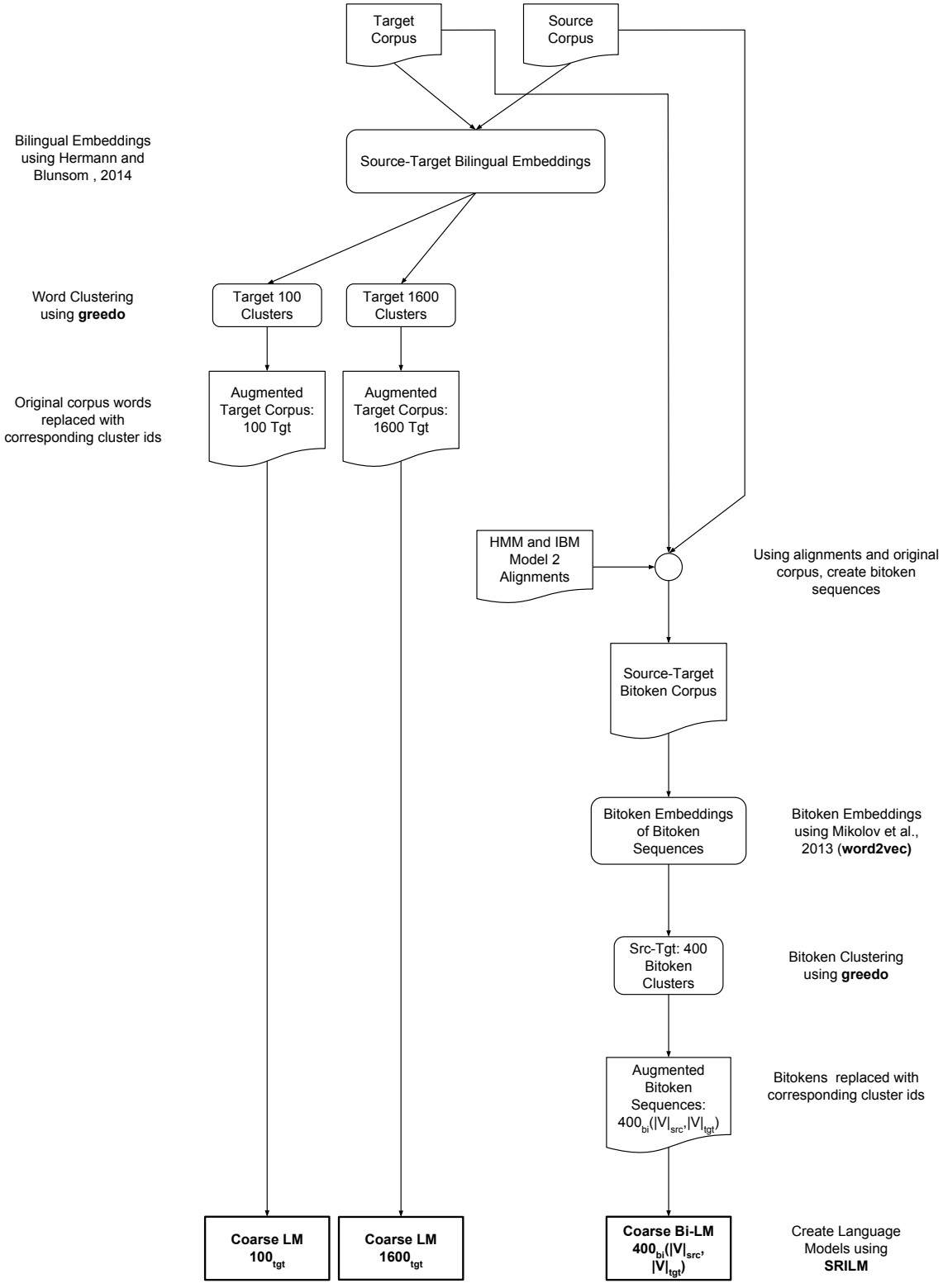

Figure 3.7: Approach 3 for creating Coarse LMs and Coarse Bi-LMs

3.3 Previous Work

In the previous section we introduced Bi-LMs [Niehues et al., 2011, Stewart et al., 2014] 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. [Casacuberta and Vidal, 2004] proposed to use stochastic finite state transducers based on bilingual n-grams. This approach was extended by [Marino et al., 2006, Crego and Yvon, 2010, Zhang et al., 2013]. [Allauzen et al., 2010] successfully applied the implementation of [Marino et al., 2006] 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.

[Zhao et al., 2005] proposed spectral bilingual clustering for HMM (hidden markov model) based SMT [Och and Ney, 2003]. This model adds information of both source and target languages to the HMM model. [Hasan et al., 2008] 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. [Feng et al., 2014] proposed factored markov backoff models along with a robust smoothing strategy that helps to generalize well. [Durrani et al., 2011, Durrani et al., 2014] 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 [Casacuberta and Vidal, 2004, Marino et al., 2006].

3.4 Summary

In this chapter we introduced coarse language models and bilingual language models. We gave an in depth explaination of how bilingual language models are generated using parallel corpus and the alignments between the source and target words. We introduced the work of [Niehues et al., 2011] in which he introduced part-of-speech based coarse Bi-LMs which was extended by [Stewart et al., 2014] to introduce word class based coarse LMs and coarse Bi-LMs. Motivated by these approaches, we introduced our three approaches to create coarse LMs and Bi-LMs using monolingual embeddings from **word2vec** [Mikolov et al., 2013a] and bilingual word embeddings [Hermann and Blunsom, 2014]. As [Stewart et al., 2014] had used **mkcls** to cluster the source and target parallel corpus, we utilized **greedo** [Stratos et al., 2014] to cluster the word embeddings. Sine, both **mkcls** and **greedo** are based on the [Brown et al., 1992] 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 [Stewart et al., 2014].

Chapter 4

Experiments and Results

In Chapter 2 we introduced coarse LMs and coarse Bi-LMs. We also introduced biligual word embeddings and clustering of embeddings using **greedo** [Stratos et al., 2014]. We described our baseline system, which is am implementation of [Stewart et al., 2014]. We also introduced three new approaches (Section 3.2.1) in which we utilize bilingual word embeddings [Hermann and Blunsom, 2014] to create coarse LMs and BiLMs. 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 [Koehn et al., 2007].
- We report our results on the test dataset. This is a bling dataset that was not used during the training and tuning step.

Table 4.1 shows the details of our data.

Dataset	**Corpus**	**Size**	**Number of References**
Training	HK + GALE Phase 1	2,352,888	N/A
Tuning	MTC Parts 1 & 3	1927	4
Testing	MTC Part 4	919	4

Table 4.1: Corpus Statistics: Chinese-English Parallel Corpus

4.1 Baseline System

Our baseline system is the system developed by [Stewart et al., 2014]. As shown in Figure 3.4, we first use **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 augment corporas 100_{en}, 1600_{en} and 400_{en}. Similarly, we augment the Chinese corpus using the Chinese clusters to create 400_{zh}.

Using **SRILM**, we estimate coarse LMs *Coarse LM* 100_{en} and *Coarse LM* 1600_{en}.

To create coarse Bi-LMs, we first need to create bitoken sequences using augmented corporas 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++[Och and Ney, 2003], we create the following biderectional alignments:

- IBM Model 2 [Brown et al., 1993] alignments.
- Hidden Markov Model (HMM) [Och and Ney, 2003] alignments.

For both the alignments, *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 augmented corporas 400_{zh} and 400_{en}, the bitoken sequences are created, called (400_{zh}, 400_{en}) bitoken corpus.

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

When estimating the coarse LMs and Bi-LMs, we use *Witten-Bell smoothing* [Witten and Bell, 1991]. Coarse LMs and Bi-LMs create counts of counts that the SRILM implementation of Kneser-Ney smoothing cannot cope up with. [Stewart et al., 2014] 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 decoder in a later section. In the next section we explain our steps to create the coarse models as described in our appraoches.

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 [Hermann and Blunsom, 2014]. 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. We experimented with different values of parameters and different dimensions, but these parameters gave us the best embeddings. To judge how good the embeddings are, we use **WordEmbeddingsViz** (Section 2.3). Using **WordEmbeddingsViz**, a human annotator looked at bilingual word embeddings generated with different parameters. 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. Based on observations of the human annotator, the above defined parameters for **BiCVM** were chosen. The above defined parameters were chosen because the English and Chinese word embeddings were close to each other after being projected to two dimensions using **tSNE**.

[1] BiCVM by Karl Mortiz Hermann: `https://github.com/karlmoritz/bicvm`

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).

Approach 1

As shown in Figure 3.5, we first cluster the Zh-En bilingual word embeddings using **greedo** [Stratos et al., 2014]. 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 corpus (as described in 4.1) to get augmented corporas 100_{en}, 1600_{en}, 400_{en} and 400_{zh}. Similar to our baseline system, we estimate *8-gram* coarse LMs *Coarse LM* 100_{en} and *Coarse LM* 1600_{en} using **SRILM** with *Witten-Bell smoothing*. Similarly, we use HMM and IBM Model 2 alignments to create a bitoken corpus (400_{zh}, 400_{en}) from augmented corporas 400_{en} and 400_{zh}.

The bitoken corpus (400_{zh}, 400_{en}) is clusterd using **mkcls** with cluster size set to 400. Using these clusters and bitoken corpus, augmented corpus $400_{bi}(400_{zh},\ 400_{en})$ is generated. Bitoken corpus (400_{zh}, 400_{en}) and **SRILM** are used to estimate *8-gram* coarse Bi-LM *Coarse Bi-LM* (400_{zh}, 400_{en}). Similarly, *Coarse Bi-LM* $400_{bi}(400_{zh},\ 400_{en})$ is estimated from augmented corpus $400_{bi}(400_{zh},\ 400_{en})$.

Approach 2

In *approach 2* (Figure 3.6), we follow the same steps as in *approach 1* to estimate *Coarse LM* 100_{en}, *Coarse LM* 1600_{en} and *Coarse Bi-LM* (400_{zh}, 400_{en}). Using **word2vec**, we create bitoken embeddings for bitoken corpus (400_{zh}, 400_{en}). For **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 **greedo** with cluster size set to 400. Utilizing the clusters, the bitoken corpus is transformed to create augmented bitoken corpus $400_{bi}(400_{zh},\ 400_{en})$. From this augmented bitoken corpus, we estimate an 8-gram coarse Bi-LM *Coarse Bi-LM* $400_{bi}(400_{zh},\ 400_{en})$.

Approach 3

In *approach 3*, we utilize the coarse LMs *Coarse LM* 100_{en} and *Coarse LM* 1600_{en} estimated for *approach 2*. As shown in Figure **??**, in this appraoch we only use one coarse Bi-LM model instead of two. To

Using Zh-En parallel corpus, HMM alignments and IBM Model 2 alignments, we create a bitoken corpus $(|V|_{zh},\ |V|_{en})$. 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 *approach 2*. The bitoken embeddings are clustered into 400 clusters using **greedo**. The clusters are then utilizied to augment the bitoken corpus to create an augmented bitoken corpus $400_{bi}(|V|_{zh},\ |V|_{en})$. Using the augmented bitoken corpus and **SRILM**, we estimate the 8-gram coarse Bi-LM *Coarse Bi-LM* $400_{bi}(|V|_{zh},\ |V|_{en})$ 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. 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 [Heafield et al., 2013] wrapper integrated with Moses. **SRILM** stores the estimated language models in ARPA format[2]. 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 create 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 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:

[2]More details about ARPA format: `http://www.speech.sri.com/projects/srilm/manpages/ngram-format.5.html`

- Extract target phrase from hypotheis.
- As each coarse LM was estimated for an augmented corpus, augmented 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 augmented 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 bitoken optionaly, we replace the words in phrase pair with the cluster ids, depending if we are calculating for *approaches 1& 2 or 3*. The bitokens are then optionally replaced by their cluster ids (they are replaced by cluster ids when estimating the score from *Coarse Bi-LM* $400_{bi}(|V|_{src},\ |V|_{tgt})$ and are not replaced in case of *Coarse Bi-LM* $(400_{src},\ 400_{tgt})$). 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 describe the results of the baseline system and our approaches.

4.4 Results

For our experiments as mentioned earlier we use Moses decoder [Koehn et al., 2007]. We implemented a stateful feature (Section 4.3) function[3] 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 Zh-En training dataset (Table 4.1):

- Train alignments using GIZA++[Och and Ney, 2003]. 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*.
- 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.

[3]Moses Feature Functions: `http://www.statmt.org/moses/?n=Moses.FeatureFunctions`

Corpus	Counts				
	1-gram	**2-gram**	**3-gram**	**4-gram**	**5-gram**
English Gigaword	3,621,795	15,004,955	31,570,877	43,974,562	521,798,40

Table 4.2: 5-gram language model for English and counts of each gram.

For all our experiments, we tune the feature weights using *Minimum Error Rate Training* (MERT) [Och and Ney, 2003] 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. *Approach 2* 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** [Clark et al., 2011] to calculate the BLEU score, TER score and p-value. Since, when looking at BLEU score *approach 2* has statistically significant results, we deem it as the winning candidate out of all three of our approaches.

Metric	**System**	**Score**	***p*-value**
BLEU ↑	Baseline	23.0	-
	Approach 1	23.4	0.33
	Approach 2	**24.4**	**0.00**
	Approach 3	23.1	0.82
TER ↓	baseline	77.5	-
	Approach 1	**71.0**	**0.00**
	Approach 2	71.9	0.00
	Approach 3	73.0	0.00

Table 4.3: Results comparing the baseline system 4.1 and three of our approaches 4.2

4.5 Summary

In this chapter we described the steps we took to implement the baseline system [Stewart et al., 2014] 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 **approach 2** outperforms the baseline by **1.4 BLEU points** and other approaches 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 where we would like to further take this research.

Chapter 5

Conclusion & Future Work

5.1 Conclusion

We started the dissertation 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 that try 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, [Niehues et al., 2011] introduced bilingual language models. They achieved statistically significant gains by replacing words with part-of-speech tags and then creating their bilingual language models. [Stewart et al., 2014] extended their work and showed that significant gains can be achieved by using a combination of coarse language models and coarse bilingual language models. [Stewart et al., 2014] made their language models coarse by clustering their corpora using **mkcls**, 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 dissertation 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 [Hermann and Blunsom, 2014] and cluster these embeddings using **greedo** [Stratos et al., 2014]. The clusters are used to augment the Chinese-English parallel corpus. These augmented corporas are used to create coarse language models in all three of our systems. In two our systems we use

augmented corporas to create bitokens, whereas in the third system we use the Chinese-English parallel corpus to create the bitokens. In the first two systems, we create coarse bilingual language models using the bitokens itself. In all three systems, we further cluster the bitokens. We experiment with clustering the bitokens using **mkcls** and also by creating bitoken embeddings using **word2vec**. The bitoken embeddings are clustered again using **greedo**. The clusters are then used to augment the bitokens and this augmented bitoken corpus is used to create coarse bilingual language models.

In our experiments we compare our systems to a baseline system, which is an implementation of [Stewart et al., 2014]. 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 creating bitoken embeddings performs the best and when looking at TER score, the first system which uses **mkcls** to cluster the bitokens performs the best. The second system has a p-value of 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 **greedo** [Stratos et al., 2014]. **greedo** creates a hierarchical cluster of the embeddings by measuring the euclidean distances. As **greedo** and **mkcls** are based on [Brown et al., 1992] 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 [Hornik et al., 2012] 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, but 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

[Allauzen et al., 2010] Allauzen, A., Crego, J. M., El-Kahlout, I. D., and Yvon, F. (2010). Limsi's statistical translation systems for wmt'10. In *Proceedings of the Joint Fifth Workshop on Statistical Machine Translation and MetricsMATR*, WMT '10, pages 54–59, Stroudsburg, PA, USA. Association for Computational Linguistics.

[Ammar et al., 2013] Ammar, W., Chahuneau, V., Denkowski, M., Hanneman, G., Ling, W., Matthews, A., Murray, K., Segall, N., Lavie, A., and Dyer, C. (2013). The CMU machine translation systems at WMT 2013: Syntax, synthetic translation options, and pseudo-references. In *Proceedings of the Eighth Workshop on Statistical Machine Translation*, pages 70–77, Sofia, Bulgaria. Association for Computational Linguistics.

[Ayan et al., 2005] Ayan, N. F., Dorr, B. J., and Monz, C. (2005). NeurAlign: Combining word alignments using neural networks. In *Proceedings of Human Language Technology Conference and Conference on Empirical Methods in Natural Language Processing*, pages 65–72, Vancouver, British Columbia, Canada. Association for Computational Linguistics.

[Berg-Kirkpatrick et al., 2010] Berg-Kirkpatrick, T., Bouchard-Côté, A., DeNero, J., and Klein, D. (2010). Painless unsupervised learning with features. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 582–590, Los Angeles, California. Association for Computational Linguistics.

[Bisazza and Monz, 2014] Bisazza, A. and Monz, C. (2014). Class-based language modeling for translating into morphologically rich languages. In *Proceedings of COLING 2014, the 25th International Conference on Computational Linguistics: Technical Papers*, pages 1918–1927, Dublin, Ireland. Dublin City University and Association for Computational Linguistics.

[Blunsom and Cohn, 2006] Blunsom, P. and Cohn, T. (2006). Discriminative word alignment with conditional random fields. In *Proceedings of the 21st International Conference on Computational Linguistics and the 44th Annual Meeting of the Association for Computational Linguistics*, ACL-44, pages 65–72, Stroudsburg, PA, USA. Association for Computational Linguistics.

[Blunsom and Cohn, 2011] Blunsom, P. and Cohn, T. (2011). A hierarchical pitman-yor process hmm for unsupervised part of speech induction. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies - Volume 1*, HLT '11, pages 865–874, Stroudsburg, PA, USA. Association for Computational Linguistics.

[Boyd-Graber and Blei, 2012] Boyd-Graber, J. L. and Blei, D. M. (2012). Multilingual topic models for unaligned text. *CoRR*, abs/1205.2657.

[Brown et al., 1992] Brown, P. F., deSouza, P. V., Mercer, R. L., Pietra, V. J. D., and Lai, J. C. (1992). Class-based n-gram models of natural language. *Comput. Linguist.*, 18(4):467–479.

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

[Casacuberta and Vidal, 2004] Casacuberta, F. and Vidal, E. (2004). Machine translation with inferred stochastic finite-state transducers. *Computational Linguistics*, 30(2):205–225.

[Chandar A P et al., 2014] Chandar A P, S., Lauly, S., Larochelle, H., Khapra, M., Ravindran, B., Raykar, V. C., and Saha, A. (2014). An autoencoder approach to learning bilingual word representations. In Ghahramani, Z., Welling, M., Cortes, C., Lawrence, N., and Weinberger, K., editors, *Advances in Neural Information Processing Systems 27*, pages 1853–1861. Curran Associates, Inc.

[Cherry and Foster, 2012] Cherry, C. and Foster, G. (2012). Batch tuning strategies for statistical machine translation. In *Proceedings of the 2012 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 427–436. Association for Computational Linguistics.

[Cherry and Lin, 2006] Cherry, C. and Lin, D. (2006). Soft syntactic constraints for word alignment through discriminative training. In *Proceedings of the COLING/ACL 2006 Main Conference Poster Sessions*, pages 105–112, Sydney, Australia. Association for Computational Linguistics.

[Clark et al., 2011] Clark, J. H., Dyer, C., Lavie, A., and Smith, N. A. (2011). Better hypothesis testing for statistical machine translation: Controlling for optimizer instability. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies: short papers-Volume 2*, pages 176–181. Association for Computational Linguistics.

[Collobert and Weston, 2008] Collobert, R. and Weston, J. (2008). A unified architecture for natural language processing: Deep neural networks with multitask learning. In *Proceedings of the 25th International Conference on Machine Learning*, ICML '08, pages 160–167, New York, NY, USA. ACM.

[Crego and Yvon, 2010] Crego, J. M. and Yvon, F. (2010). Factored bilingual n-gram language models for statistical machine translation. *Machine Translation*, 24(2):159–175.

[Deerwester et al., 1990] Deerwester, S., Dumais, S. T., Furnas, G. W., Landauer, T. K., and Harshman, R. (1990). Indexing by latent semantic analysis. *JOURNAL OF THE AMERICAN SOCIETY FOR INFORMATION SCIENCE*, 41(6):391–407.

[Dempster et al., 1977] Dempster, A. P., Laird, N. M., and Rubin, D. B. (1977). Maximum likelihood from incomplete data via the em algorithm. *Journal of the royal statistical society. Series B (methodological)*, pages 1–38.

[Dhillon et al., 2011] Dhillon, P. S., Foster, D., and Ungar, L. (2011). Multi-view learning of word embeddings via cca. In *Advances in Neural Information Processing Systems (NIPS)*, volume 24.

[Durrani et al., 2014] Durrani, N., Koehn, P., Schmid, H., and Fraser, A. M. (2014). Investigating the usefulness of generalized word representations in smt. In *COLING*, pages 421–432.

[Durrani et al., 2011] Durrani, N., Schmid, H., and Fraser, A. (2011). A joint sequence translation model with integrated reordering. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies - Volume 1*, HLT '11, pages 1045–1054, Stroudsburg, PA, USA. Association for Computational Linguistics.

[Faruqui and Dyer, 2014] Faruqui, M. and Dyer, C. (2014). Improving vector space word representations using multilingual correlation. In *Proceedings of EACL*, volume 2014.

[Feng et al., 2014] Feng, Y., Cohn, T., and Du, X. (2014). Factored markov translation with robust modeling. In *CoNLL*, pages 151–159.

[Good, 1953] Good, I. J. (1953). The population frequencies of species and the estimation of population parameters. *Biometrika*, 40(3-4):237–264.

[Harris, 1954] Harris, Z. S. (1954). Distributional structure. *Word*.

[Hasan et al., 2008] Hasan, S., Ganitkevitch, J., Ney, H., and Andrés-Ferrer, J. (2008). Triplet lexicon models for statistical machine translation. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, EMNLP '08, pages 372–381, Stroudsburg, PA, USA. Association for Computational Linguistics.

[Heafield et al., 2013] Heafield, K., Pouzyrevsky, I., Clark, J. H., and Koehn, P. (2013). Scalable modified Kneser-Ney language model estimation. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics*, pages 690–696, Sofia, Bulgaria.

[Hermann and Blunsom, 2014] Hermann, K. M. and Blunsom, P. (2014). Multilingual Distributed Representations without Word Alignment. In *Proceedings of ICLR*.

[Hopkins and May, 2011] Hopkins, M. and May, J. (2011). Tuning as ranking. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 1352–1362. Association for Computational Linguistics.

[Hornik et al., 2012] Hornik, K., Feinerer, I., Kober, M., and Buchta, C. (2012). Spherical k-means clustering. *Journal of Statistical Software*, 50(10):1–22.

[Huang et al., 2012] Huang, E. H., Socher, R., Manning, C. D., and Ng, A. Y. (2012). Improving word representations via global context and multiple word prototypes. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Long Papers - Volume 1*, ACL '12, pages 873–882, Stroudsburg, PA, USA. Association for Computational Linguistics.

[Huang and Yates, 2009] Huang, F. and Yates, A. (2009). Distributional representations for handling sparsity in supervised sequence-labeling. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP: Volume 1 - Volume 1*, ACL '09, pages 495–503, Stroudsburg, PA, USA. Association for Computational Linguistics.

[Klementiev et al., 2012] Klementiev, A., Titov, I., and Bhattarai, B. (2012). Inducing crosslingual distributed representations of words. In *Proceedings of the International Conference on Computational Linguistics (COLING)*, Bombay, India.

[Kneser and Ney, 1995] Kneser, R. and Ney, H. (1995). Improved backing-off for m-gram language modeling. In *Acoustics, Speech, and Signal Processing, 1995. ICASSP-95., 1995 International Conference on*, volume 1, pages 181–184 vol.1.

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

[Koehn, 2004] Koehn, P. (2004). Pharaoh: a beam search decoder for phrase-based statistical machine translation models. In *Machine translation: From real users to research*, pages 115–124. Springer.

[Koehn, 2009] Koehn, P. (2009). *Statistical machine translation*. Cambridge University Press.

[Koehn et al., 2007] Koehn, P., Hoang, H., Birch, A., Callison-Burch, C., Federico, M., Bertoldi, N., Cowan, B., Shen, W., Moran, C., Zens, R., et al. (2007). Moses: Open source toolkit for statistical machine translation. In *Proceedings of the 45th annual meeting of the ACL on interactive poster and demonstration sessions*, pages 177–180. Association for Computational Linguistics.

[Koehn et al., 2003] Koehn, P., Och, F. J., and Marcu, D. (2003). Statistical phrase-based translation. In *Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology - Volume 1*, NAACL '03, pages 48–54, Stroudsburg, PA, USA. Association for Computational Linguistics.

[Lopez, 2008] Lopez, A. (2008). Statistical machine translation. *ACM Comput. Surv.*, 40(3):8:1–8:49.

[Macherey et al., 2008] Macherey, W., Och, F. J., Thayer, I., and Uszkoreit, J. (2008). Lattice-based minimum error rate training for statistical machine translation. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 725–734. Association for Computational Linguistics.

[Marino et al., 2006] Marino, J. B., Banchs, R. E., Crego, J. M., de Gispert, A., Lambert, P., Fonollosa, J. A., and Costa-Jussà, M. R. (2006). N-gram-based machine translation. *Computational Linguistics*, 32(4):527–549.

[Mikolov et al., 2013a] Mikolov, T., Chen, K., Corrado, G., and Dean, J. (2013a). Efficient estimation of word representations in vector space. *CoRR*, abs/1301.3781.

[Mikolov et al., 2013b] Mikolov, T., Le, Q. V., and Sutskever, I. (2013b). Exploiting similarities among languages for machine translation. *CoRR*, abs/1309.4168.

[Mikolov et al., 2013c] Mikolov, T., Sutskever, I., Chen, K., Corrado, G. S., and Dean, J. (2013c). Distributed representations of words and phrases and their compositionality. In Burges, C., Bottou, L., Welling, M., Ghahramani, Z., and Weinberger, K., editors, *Advances in Neural Information Processing Systems 26*, pages 3111–3119. Curran Associates, Inc.

[Moore et al., 2006] Moore, R. C., Yih, W.-t., and Bode, A. (2006). Improved discriminative bilingual word alignment. In *Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the Association for Computational Linguistics*, pages 513–520, Sydney, Australia. Association for Computational Linguistics.

[Niehues et al., 2011] Niehues, J., Herrmann, T., Vogel, S., and Waibel, A. (2011). Wider context by using bilingual language models in machine translation. In *Proceedings of the Sixth Workshop on Statistical Machine Translation*, pages 198–206. Association for Computational Linguistics.

[Och, 1995] Och, F. J. (1995). Maximum-likelihood-schätzung von wortkategorien mit verfahren der kombinatorischen optimierung. *Studienarbeit, Friedrich-Alexander-Universität, Erlangen-Nürnberg, Germany.*

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

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

[Och et al., 2001] Och, F. J., Ueffing, N., and Ney, H. (2001). An efficient a* search algorithm for statistical machine translation. In *Proceedings of the workshop on Data-driven methods in machine translation-Volume 14*, pages 1–8. Association for Computational Linguistics.

[Papineni et al., 2002] Papineni, K., Roukos, S., Ward, T., and Zhu, W.-J. (2002). Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th annual meeting on association for computational linguistics*, pages 311–318. Association for Computational Linguistics.

[Pereira et al., 1993] Pereira, F., Tishby, N., and Lee, L. (1993). Distributional clustering of English words. In *Proceedings of the ACL*, pages 183–190.

[Stewart et al., 2014] Stewart, D., Kuhn, R., Joanis, E., and Foster, G. (2014). Coarse "split and lump"bilingual language models for richer source information in smt.

[Stolcke et al., 2011] Stolcke, A., Zheng, J., Wang, W., and Abrash, V. (2011). Srilm at sixteen: Update and outlook. In *Proceedings of IEEE Automatic Speech Recognition and Understanding Workshop*, page 5.

[Stratos et al., 2014] Stratos, K., Kim, D.-k., Collins, M., and Hsu, D. (2014). A spectral algorithm for learning class-based n-gram models of natural language. *Proceedings of the Association for Uncertainty in Artificial Intelligence*.

[Taskar et al., 2005] Taskar, B., Lacoste-Julien, S., and Klein, D. (2005). A discriminative matching approach to word alignment. In *Proceedings of the Conference on Human Language Technology and Empirical Methods in Natural Language Processing*, HLT '05, pages 73–80, Stroudsburg, PA, USA. Association for Computational Linguistics.

[Toutanova et al., 2003] Toutanova, K., Klein, D., Manning, C. D., and Singer, Y. (2003). Feature-rich part-of-speech tagging with a cyclic dependency network. In *Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology-Volume 1*, pages 173–180. Association for Computational Linguistics.

[Van der Maaten and Hinton, 2008] Van der Maaten, L. and Hinton, G. (2008). Visualizing data using t-sne. *Journal of Machine Learning Research*, 9(2579-2605):85.

[Witten and Bell, 1991] Witten, I. H. and Bell, T. C. (1991). The zero-frequency problem: estimating the probabilities of novel events in adaptive text compression. *IEEE Transactions on Information Theory*, 37(4):1085–1094.

[Zhang et al., 2013] Zhang, H., Toutanova, K., Quirk, C., and Gao, J. (2013). Beyond left-to-right: Multiple decomposition structures for smt. In *HLT-NAACL*, pages 12–21.

[Zhao and Xing, 2006] Zhao, B. and Xing, E. P. (2006). Bitam: Bilingual topic admixture models for word alignment. In *In Proceedings of the 44th Annual Meeting of the Association for Computational Linguistics (ACL'06*.

[Zhao et al., 2005] Zhao, B., Xing, E. P., and Waibel, A. (2005). Bilingual word spectral clustering for statistical machine translation. In *Proceedings of the ACL Workshop on Building and Using Parallel Texts*, ParaText '05, pages 25–32, Stroudsburg, PA, USA. Association for Computational Linguistics.

[Zou et al., 2013] Zou, W. Y., Socher, R., Cer, D., and Manning, C. D. (2013). Bilingual word embeddings for phrase-based machine translation. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing (EMNLP 2013)*.