Continuous-Space based Statistical Machine Translation

ABSTRACT

Natural language, or human language, is understood as a cultural specific communication system in informal usage [1]. Natural Language Processing (NLP, which is also called computational linguistics) employs computational techniques for the purpose of learning, understanding, and producing human language content [2]. Statistical Machine Translation (SMT), which is one of the most popular aspects of NLP, is a machine translation paradigm where translations are generated on the basis of statistical models whose parameters are derived from the analysis of bilingual text corpora. It contrasts with the rule-based machine translation approaches.

The history of Machine Translation (MT) can date back to 1940s, which is almost immediately after the first computer ENIAC. At the beginning, MT plays a role as the translation memory tools for translators. One of the basic ideas behind MT is that a foreign language is considered as *encrypted English* [3]. So *noisy-channel model* is used for decoding foreign language into English. From 1980s, more and more bilingual corpora become available, and *data-Driven methods* are proposed. Example-based MT systems have been built especially in Japan since the 1980s. In 1990s, the emergence of SMT is groundbreaking, especially IBM proposes a series statistic models for SMT. Phrase-based SMT is widely considered as state-of-the-art system. Recently, continuous-space methods, especially Neural Network (NN) based methods become popular, as the upgrading of computer performance. NN methods are used to improve translation model, language model or directly integrated into end-to-end SMT systems.

Although continuous-space methods have been shown helpful in many SMT tasks, they also suffer some shortcomings: 1) Non-linear algorithms are applied to continuous-space based methods, which ensure high performances. Meanwhile, this will make the training and decoding speed much slower. 2) Since most of continuous-space based methods, especially NN based methods, learn features automatically. However some useful features, such as semantic information, are missing.

In this thesis, we focus on continuous-space SMT in the following two aspects: NN based methods and graph based methods. For NN based methods, 1) we propose a continuous-space language model conversion method, which can accelerate the decoding speed of NN language

model as well as maintain its accuracy. 2) Making use of the generation abilities of NN, we combine the connecting phrase methods and NN methods together, to enhance SMT adaptation and generation. For graph based method, we propose a novel bilingual sense unit Bilingual Contexonym Cliques (BCCs). BCC can describe senses of a word better, in comparison with simple document or sliding window information. Bilingual Graph-based Semantic Model (BGSM) is capable of effectively modeling word sense representation instead of word itself. The proposed model is applied to phrase pair translation probability estimation and generation for SMT.

We have empirically evaluated the proposed methods on IWSLT and NIST data sets, and compared with the existing state-of-the-art methods. Empirical results shows that the proposed methods outperform the existing methods in both performance and efficiency.

KEY WORDS: **Statistical Machine Translation, Continuous-Space Models, Neural Network Models, Bilingual Graph Semantic Models, Language Models**

Menu

Index of Figures

Index of Tables

Chapter 1

Introduction

1.1 Motivation

Statistical Machine Translation (SMT) was one of the first nonnumeric applications of computers and was studied intensively starting in the late 1940s. However, the hand-built grammar-based systems of early decades achieved very limited success. The field was transformed in the early 1990s when researchers at IBM acquired a large quantity of English and French sentences that were translations of each other (known as parallel text), produced as the proceedings of the bilingual Canadian Parliament [4–6]. These data allowed them to collect statistics of word translations and word sequences and to build a probabilistic model of SMT. Following a quiet period in the late 1990s, the new millennium brought the potent combination of ample online text, including considerable quantities of parallel text, much more abundant and inexpensive computing, and a new idea for building statistical phrase-based MT systems [7]. Traditionally, linear or log-linear models are widely applied to Statistical Machine Translation (SMT). Phrase-based SMT, which is a combination of a series of log-linear models, is state-of-the-art system until recently [8].

Continuous representations of words onto multi-dimensional vectors enhance Natural Language Processing (NLP), especially SMT, by measuring similarities of words using distances of corresponding vectors. One of the earliest successful distributed representations for words (concepts) can date back to 1980s [9, 10]. Most early work treated word representation as a cognitive processing task such as WordNet [11], EuroWordNet [12] and other work [13]. Typically, WordNet organized lexicon conceptually as a set of terms associated with a partition into Synsets[1], though words organized in this way were not conveniently represented as vectors.

Around the new millennium, researchers began to treat word representation as a *document-term* task. Many types of models were proposed for estimating continuous representations of words or semantic embedding using the relationships between document and word (term). These work mainly included Vector Space Models (VSMs), latent semantic analysis [14], latent dirich-

[1] Synset is a small group of synonyms labeled as a concept.

let allocation [15] and their applications in NLP [16]. Some of these word representations approaches were later applied to bilingual embedding in SMT [17, 18].

Finally, just in the past few year, we have seen the development of an extremely promising approach to SMT through the use of deep-learning–based sequence models [2]. The central idea of deep learning is that if we can train a model with several representational levels to optimize a final objective, such as translation quality, then the model can itself learn intermediate representations that are useful for the task at hand. This idea has been explored particularly for Neural Network (NN) models in which information is stored in real-valued vectors, with the mapping between vectors consisting of a matrix multiplication followed by a nonlinearity, such as a sigmoid function that maps the output values of the matrix multiplication onto [0, 1] [2]. Bengio *et al.* [19] proposed NN based Language Model (LM), where a feedforward NN with a linear projection layer and a non-linear hidden layer were used to jointly learn the word vector representation and statistical LM. It was followed by a series of NN based embedding [20–29], LMs [30–32], and translation models [33–43]. Building large models of this form is much more practical with the massive parallel computation that is now economically available via graphics processing units. For translation, research has focused on a particular version of recurrent neural networks, with enhanced Long Short-Term Memory (LSTM) computational units that can better maintain contextual information from early until late in a sentence [43, 44]. The distributed representations of NN are often very effective for capturing subtle semantic similarities, and NN based MT systems have already produced some state-of-the-art results [44–46]. Besides NN based embedding, there are also some other embedding methods [47–49].

Meanwhile, there are some drawbacks for traditional NN based SMT. 1) For NNLM, although it can predict the probability estimation more accurate, its decoding time is much slower than traditional Back-off N-gram LM (BNLM). 2) One of the advantages of NN methods is that they have generation abilities to predict the probabilities of phrases or phrase pairs outside corpus. However, few of the existing works make use of this ability. 3) Sense gives more exact meaning formulization than word itself. However, most of these existing NN embedding methods embed words as vectors using sliding window or document information directly, instead of sense information.

We propose several novel methods to overcome the above drawbacks of NN based methods.

1) We propose a Continuous-Space Language Model (CSLM) conversion method, which pre-computes the probabilities of n-grams using CSLM and store back the n-grams into BNLM. This makes that the converted CSLM run as fast as BNLM, and predict as accurate as CSLM.

2) We propose a linguistic based connecting phrase method. The connecting phrase method works together with NN based method for SMT adaptation and generation.

3) We present Bilingual Graph-based Semantic Model (BGSM). By means of maximum complete sub-graph (clique) for context selection, BGSM is capable of effectively modeling word sense representation instead of word itself. The proposed model is applied to phrase pair translation probability estimation and generation for SMT.

1.2 Structure of the Thesis

The main structure of this thesis is designed as follows:

Chapter 1 will introduce the motivation, structure and main contributions of this thesis.

Chapter 2 will introduce the overview of SMT, especially the recent popular continuous-space based methods.

Chapter 3 will describe the proposed NN based methods for SMT.

Chapter 4 will describe the proposed bilingual graph based semantic model for SMT.

Chapter 5 will conduct the experiments and analyse the results.

At last we will conclude this thesis.

1.3 Claims of the Thesis

We will propose three novel Neural Network (NN) based methods and one graph based bilingual embedding methods for SMT. The most important original contributions of this thesis are:

1) We propose an CSLM converting method. It can construct an efficient LM which can run as fast as a Back-off LM (BNLM) and predict probability as accurate as CSLM.

2) To solve the additional corpora SMT adaptation task, an NN based and a connecting phrase-based phrase adaptation methods are proposed for SMT. To solve the rare or special domain language SMT task, we generate translation model and LM for SMT by using of the generalization abilities of NN.

3) We propose Bilingual Contexonym Clique (BCC) as smallest sense unit and BGSM for bilingual dynamic representation, which can enhance phrase-based SMT performance.

1.4 Relation with Published Works

Much of the material contained in this thesis has been published previously, primarily in the form of conference and journal papers. The text of the published papers was used, in updated

and extended form, as the basis for various parts of the thesis. In particular, the individual chapters/sections are related to prior publications as follows:

In Chapter 3, the CSLM conversion methods (Section 3.1) were partially described in EMNLP-2013 [50]. The NN and Connecting phrase-based SMT adaptation methods (Section 3.2) were partially described in IEEE/ACM trans. on TASLP-2015 [51] and ACM trans. on TALLIP-2016 [52]. The NN and Connecting phrase-based SMT generation methods (Section 3.3) were partially described in EMNLP-2014 [53].

In Chapter 4, the bilingual graph semantic model is partially described in IJCAI-2016 [54].

Chapter 2

Overview of Statistical Machine Translation

2.1 Basis of SMT

As mention in introduction, the basis of SMT is considered as *noisy-channel model*. That is, given a source sentence S_F, the probability of translation target sentence S_E can be defined as,

$$P(S_E|S_F). \tag{2–1}$$

Because the source sentence is always given and fixed, so the probability can be estimated as,

$$P(S_E|S_F) \quad = \quad \frac{P(S_F|S_E) \times P(S_E)}{P(S_F)} \quad \propto \quad P(S_F|S_E) \times P(S_E). \tag{2–2}$$

The $P(S_E)$ indicates Language Model (LM) and $P(S_F|S_E)$ indicates translation model. There are several fragmentation methods for translation, such as word-level, phrase-level and sentence level. Until recently, phrase-based SMT is widely accepted as state-of-the-art system [8]. There are also some methods to enhance SMT by using hierarchical [55] or syntax [56] information, which can partially solve the re-ordering problem of phrase-based SMT. However these methods have been shown useful on some specific corpora and do not work well on several typical corpora [51]. After translation model and LM are conducted, a global decoding method will be used to select the best translation candidate using all of the models:

$$S_{E_{best}} = argmax P(S_F|S_E) \times P(S_E). \tag{2–3}$$

In this thesis, phrase-based SMT is selected as main baseline system. In Section 2.2, we will introduce LM. In Section 2.3, several components of translation model of phrase-based SMT will be introduced. A series methods of NN based translation models will also be introduced in comparison[1]. In Section 2.4, the decoding method of using various models will be introduced.

[1] In this thesis, we consider re-ordering as a part of translation model, so we do not introduce re-ordering model independently.

2.2 Language Modeling

LMs are important for various NLP tasks, especially in speech recognition and SMT. There are mainly two ways to improve the performance of LMs in SMT. One is to estimate the probabilities of n-grams better and the other is to use a larger corpus for building LM.

Back-off N-gram Language Models (BNLMs) [57–59] are widely used for probability estimation. CSLMs, especially NNLMs [19, 30, 32], begin to be applied to SMT for better probability estimation recently [60–64].

2.2.1 Back-off N-gram Language Model

Given preceding (n-1) words history, $h_i = w_{i-n+1}^{i-1}$, BNLM can predict the probability of the target word w_i . However, if the context, h_i, does not appear in the training data, the data sparseness problem will occur. To overcome this problem, *backing-off* to models with smaller histories is proposed. Taking interpolated Kneser-Ney (KN) smoothing [57, 58] as example, the probability of w_i given history h_i, $P_b(w_i|h_i)$ is defined as,

$$P_b(w_i|h_i) = \hat{P}_b(w_i|h_i) + \alpha(h_i)P_b(w_i|w_{i-n+2}^{i-1}), \tag{2–4}$$

where $\hat{P}_b(w_i|h_i)$ indicates discounted probability and $\alpha(h_i)$ indicates back-off weight.

2.2.2 Continuous-Space Language Model

Take a four-layer feedforward neural network based CSLM as example, it works in the following way: input layer projects word w_i and its context h_i onto the projection layer (first hidden layer). Then the second hidden layer and output layer use non-liner functions to estimate the probability $P(w_i|h_i)$ of output words by given context [30].

All the probabilities of all the words in vocabulary can be calculated by CSLM, however, it will take too high computational time to calculate them at once. So CSLM is only applied to calculating the probabilities of a small subset in entire vocabulary, which is called *short-list*. Only the most frequent words in vocabulary are selected and stored in short-list. A specific neuron is assigned to calculate the probability sum of all words inside short-list for further normalization. BNLM is used to calculate the probabilities of other words outside short-list [30, 65].

A CSLM with a BNLM calculates the probability of current word w_i by given history h_i, $P(w_i|h_i)$, as follows:

$$P(w_i|h_i) = \begin{cases} \frac{P_c(w_i|h_i)}{\sum_{w \in V_0} P_c(w|h_i)} P_s(h_i) & \text{if } w_i \in V_0 \\ P_b(w_i|h_i) & \text{otherwise} \end{cases} \tag{2–5}$$

where V_0 is short-list, $P_c(\cdot)$ is the probability calculated by the CSLM, $\sum_{w \in V_0} P_c(w|h_i)$ is the sum of probabilities over all the words in the short-list in the output layer, $P_b(\cdot)$ is the probability calculated by using BNLM, and

$$P_s(h_i) = \sum_{v \in V_0} P_b(v|h_i). \tag{2–6}$$

This approach can be viewed as that BNLM summarize the probability mass of all the words in short-list, and CSLM redistributes this probability mass.

The structure of CSLM [30] can be illustrated as in Fig. 2–1.

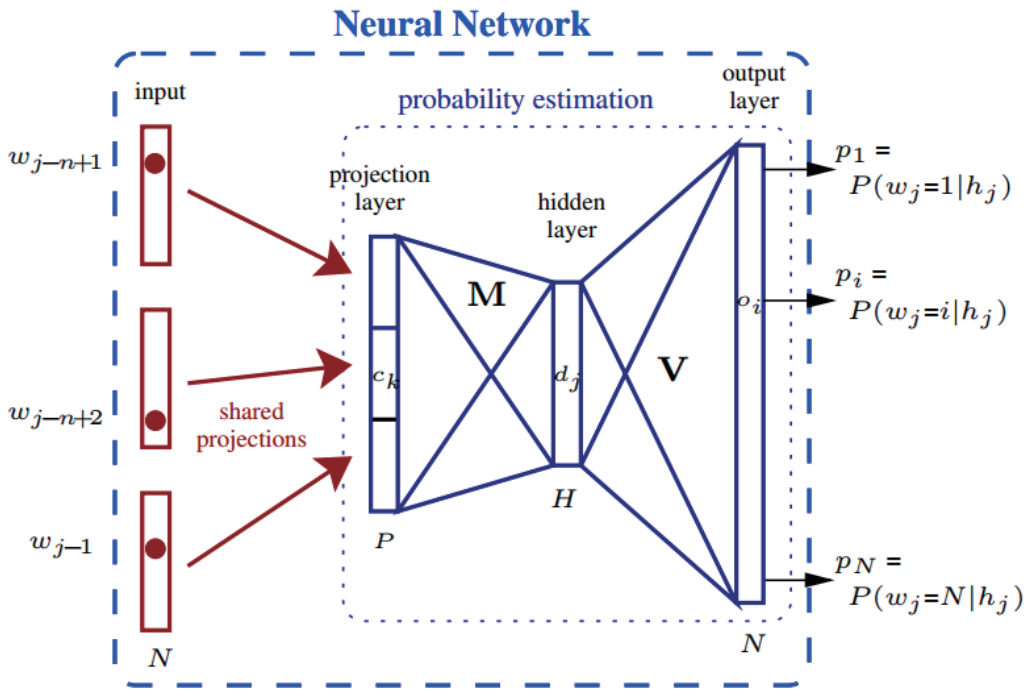

Figure 2–1 The structure of CSLM [30].

2.2.3 Other NNLMs

Besides Feedforward NNLM, there are also some other NNLM [30–32], such as Recurrent NNLM (RNNLM) [31]. RNNLM does not only consider the history of current time, but also

the previous histories. The output values from neurons in the hidden and output layers are computed as follows:

$$s(t) = f(Uw(t) + Ws(t-1)), \tag{2–7}$$

$$y(t) = g(Vs(t)). \tag{2–8}$$

where $f(z)$ and $g(z)$ are sigmoid and softmax activation functions (the softmax function in the output layer is used to ensure that the outputs form a valid probability distribution, *i.e.* all outputs are greater than 0 and their sum is 1).

The structure of RNNLM [31] can be illustrated as in Fig. 2–2.

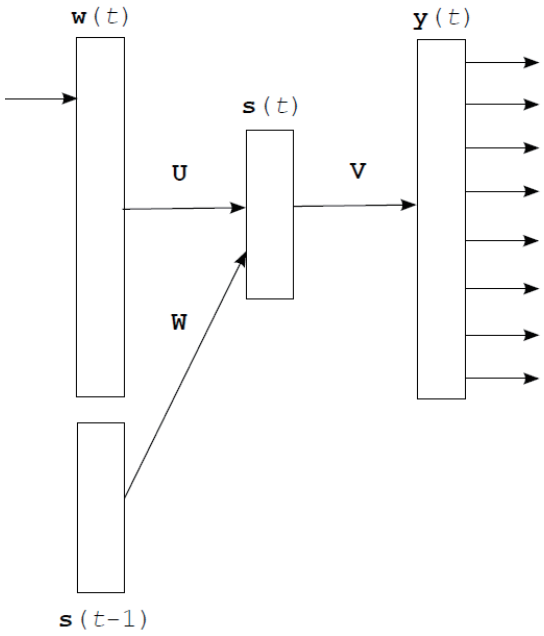

Figure 2–2 The structure of RNNLM [31].

In addition, some other technologies, such as hierarchical method [66] and Long Short-Term Memory (LSTM) [67] can be applied to NNLM.

2.3 Translation Modeling

2.3.1 Phrase-based Translation Model

In this section, phrase-based translation model will be introduced. Take IBM model-1 [4] as example, we will introduce lexical translation probability estimation and word alignment at first. Then phrase pairs will be extracted using lexical alignment. At last phrase table is constructed after phrase pair translation probability is estimated.

In the following contents, a source phrase $F = (f_1, f_2, f_3, ..., f_i, ..., f_{lf})$ and $E = (e_1, e_2, e_3, ..., e_j, ..., e_{le})$, where f_i and e_j are source and target words, respectively.

2.3.1.1 Lexical Translation

A lexical translation probability distribution will help us to answer a typical question [8]. What is the most likely English translation for a foreign word, such has German *Haus*? **Part of materials in this section, especially the examples, refer to Philipp Koehn's *Statistical Machine Translation* [8].**

The probability function from source word f into target word e defined as, $P(e|f)$. That is, given a foreign word f (here *Haus*), returns a probability, for each choice of English translation e, that indicates how likely that translation is.

The definition of probability distribution requires the function $P(e|f)$ to have two properties:

$$\sum_{e} P(e|f) = 1, \tag{2–9}$$

$$0 \leqslant P(e|f) \leqslant 1. \tag{2–10}$$

Take the German word *Haus* as example again, its translation distribution may be as follows:

$$P(e|f) = \begin{cases} 0.70 \text{ if } e = house \\ 0.22 \text{ if } e = building \\ 0.05 \text{ if } e = home \\ 0.025 \text{ if } e = household \\ 0.005 \text{ if } e = shell \end{cases} \tag{2–11}$$

For unseen events, some probability mass can also be reserved (unlike the examples shown above). There are many methods to build a model on given data. One popular type of estimation

is called maximum likelihood estimation, since it maximizes the estimation likelihood of the data. This method of obtaining a probability distribution from data is not only very intuitive, it also has a strong theoretical motivation.

2.3.1.2 Word Alignment

In sentence level, word alignment should also be considered. Take the German sentence *das Haus ist klein* as example. One solution is to translate the sentence word by word into English. One possible translation may be *the house is small*.

Implicit in this translation is an alignment, meaning a mapping from German words to English words. The German word *das* is translated to the English word *the*, *Haus* to *house* et. al. This alignment between input words and output words can be illustrated by a diagram:

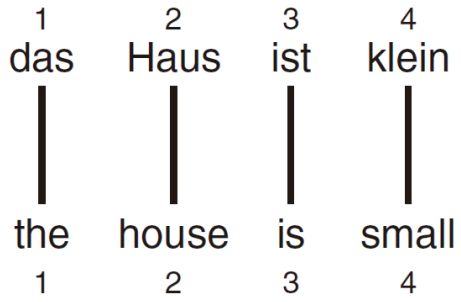

Figure 2–3 An alignment example [8].

The above example is simple, since there is no any distortion in it. An alignment can be formalized with an alignment function a. This function maps each English output word at position j to a German input word at position i:

$$a : i \leftarrow j. \tag{2–12}$$

2.3.1.3 Expectation Maximization Algorithm

Lexical translation probabilities and the notion of alignment allow us to define a model that generates a number of different translations for a sentence, each with a different probability. We use one of the most popular **IBM Model 1** [4] for illustration.

The translation probability for a foreign sentence $S_F = (f_1, f_2, f_3, ..., f_i, ..., f_{lf})$ of length l_f to an English sentence $S_E = (e_1, e_2, e_3, ..., e_j, ..., e_{le})$ of length l_e with the alignment function $a : i \leftarrow j$ is defined as follows:

$$P(S_E, a|S_F) = \frac{\epsilon}{(l_f+1)^{l_e}} \prod_{j=1}^{l_e} t(e_j|f_{a(j)}). \tag{2–13}$$

Now we are facing a typical problem for machine learning, i.e., to estimate our model from incomplete data. One aspect of our model is hidden from plain view: the alignment between words. For this reason, the alignment is considered a hidden variable in IBM models. It is a chicken and egg problem. If we had the word alignments marked up in our data, it would be trivial to estimate the lexical translation model. We simply collect counts and perform maximum likelihood estimation. On the other hand, if we had the model given to us, it would be possible to estimate the most likely alignment between words for each sentence pair. In other words: given the model, we could fill in the gap in our data. Given the complete data, we could estimate the model. Unfortunately, we have neither [8].

The expectation maximization algorithm, or EM algorithm, addresses the situation of incomplete data. It is an iterative learning method that fills in the gaps in the data and trains a model in alternating steps [68].

The EM algorithm works as follows:

1. Initialize the model, typically with uniform distributions.
2. Apply the model to the data (expectation step).
3. Learn the model from the data (maximization step).
4. Iterate steps 2 and 3 until convergence.

2.3.1.4 Other IBM Models

Besides IBM Model-1, there are also a series of works propose several methods to improve IBM models [4–6]. We summarize the features of IBM Models 1-5 in Table 2–1.

Models	Features
IBM Model-1	lexical translation
IBM Model-2	adds absolute reordering model
IBM Model-3	adds fertility model
IBM Model-4	relative reordering model
IBM Model-5	fixes deficiency

Table 2–1 Features of IBM models.

2.3.1.5 Phrase Alignment and Extraction

The previous subsections introduced word based SMT. But words may not be the best candidates for the smallest sense units for SMT. Sometimes one word in a foreign language translates into two English words, or vice versa. Sometimes Word-based models often break down in these cases.

Consider Figure 2–4 as example, which illustrates how phrase-based models work. The German input sentence is first segmented into so-called phrases (not necessarily linguistically motivated). Then, each phrase is translated into an English phrase. Finally, phrases may be reordered. The six German words and eight English words are mapped as five phrase pairs.

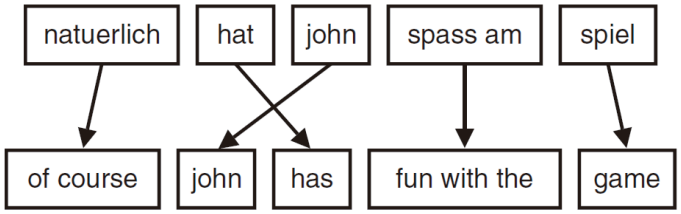

Figure 2–4 Phrase-based alignment [8].

Using word alignment information, phrase can be extracted accordingly. Figure 2–5 illustrates a phrase extraction example.The English phrase *assumes that* and the German phrase *geht davon aus, dass* are aligned, because their words are aligned to each other.

2.3.2 Neural Network Translation Models

Beside linear translation models, NN based translation models are wisely used in SMT rencently. As mentioned above, the probability of a phrase-pair is estimated as,

$$P(E|F) = P(e_1, ..., e_q|f_1, ..., f_p), \tag{2–14}$$

where $f_i, i \in [1, p]$ and $e_j, j \in [1, q]$ are source and target words, respectively. Originally,

$$P(e_1, ..., e_q|f_1, ..., f_p) = \prod_{k=1}^{q} P(e_k|e_{k+1}, ..., e_q, f_1, ...f_p). \tag{2–15}$$

There are several translation model methods [33–42]. Some of the structures are shown in Fig. 2–6. In this example, the lengths of source and target phrases are limited to three, due to limited space.

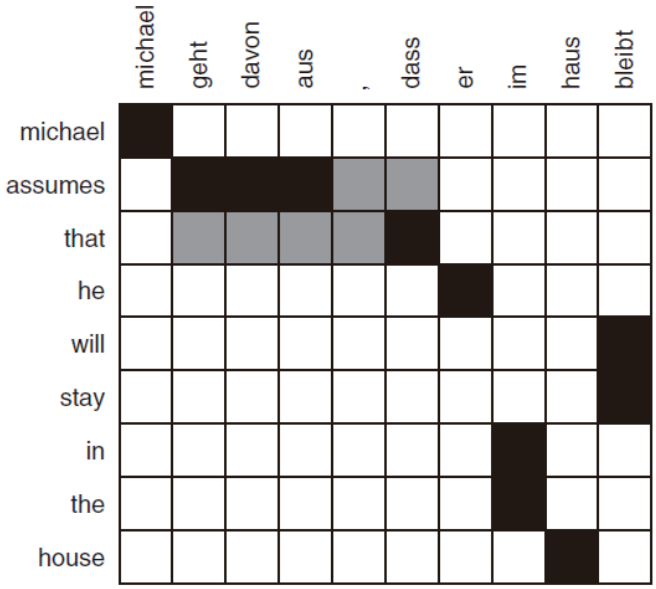

Figure 2–5 Extracting aphrase from a word alignment [8].

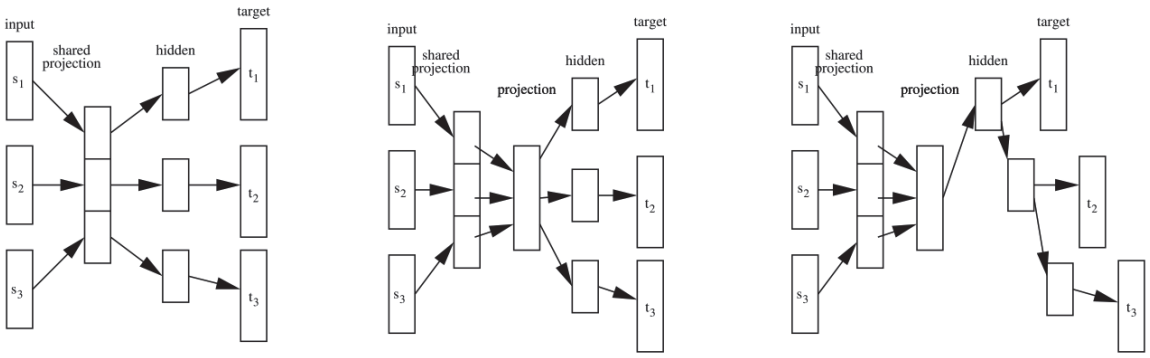

Figure 2–6 The left structure is a simple extension of CSLM. The middle structure uses an additional hidden layer in order to introduce a dependence between target words. The right structure adopts hierarchical/recurrent dependency [69].

2.4 Decoding

2.4.1 Linear Model Decoding

The task of decoding in SMT is to find the best scoring translation according to various features. Meanwhile, the number of hypotheses will increase exponentially, for each specific source sentence. Phrase-based SMT system has been shown NP-complete [70]. In other words, exhaustively examining all possible translations, scoring them, and picking the best is computationally too expensive for an input sentence of even modest length.

In this section, we will introduce the techniques that make it possible to efficiently carry out the search for the best translation. These methods are called *heuristic search* methods. This means that there is no guarantee that they will find the best translation, but we do hope to find it often enough, or at least a translation that is very close to it. **Parts of materials and examples in this section refer to Philipp Koehn's *Statistical Machine Translation* [8].**

There are two types of error that may prevent decoding from find the best translation:

1) The first type search error is the failure to find the best translation according to the model, in other words, the highest-probability translation. It is a consequence of the heuristic nature of the decoding method, which is unable to explore the entire search space (the set of possible translations). It is search space that presents methods that lead to a low search error.

2) The other type of error is the model error. That is, the translation, which have the highest-probability given by the model, may even not be a proper translation at all. In this thesis, we are not concerned with this type of error.

Now the most popular phrase-based SMT decoding method is Beam search [71]. Beam search belongs to a heuristic search algorithm. It expands the most promising node in a limited set to explore the whole graph, which is considered as an optimization of best-first search because it can reduce memory requirements. Only a pre-determined number of best current solutions are select as candidates in Beam search.

2.4.1.1 Translation Options and Hypothesis Expansion

For a given input sentence, such as *er geht ja nicht nach hause* of German, we can consult our phrase table and look up all translation options, *i.e.*, the phrase translations that apply to this input sentence. Figure 2–7 [8] displays the top four choices for all phrases in the input, using an actual phrase table (learnt from the Europarl corpus[1]) .

[1] `http://www.statmt.org/europarl/`

Armed with the notion of a translation process that builds the output sentence sequentially and a set of translation options to choose from, we can now appreciate the computer's decoding challenge. In the search heuristic, we also want to build the output sentence from left to right in sequence by adding one phrase translation at a time. During the search, we are constructing partial translations that we call hypotheses [71].

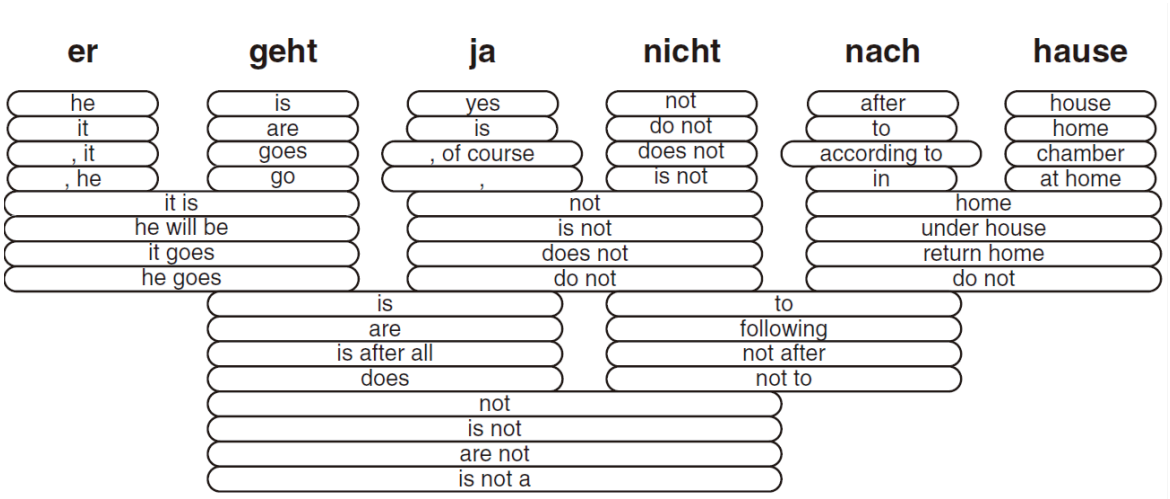

Figure 2–7 Translation options and hypothesis expansion [8].

2.4.1.2 Hypothesis Recombination

Hypothesis recombination takes advantage of the fact that matching hypotheses can be reached by different paths. Given the translation options, we may translate the first two German words separately. But it is also possible to get this translation with a single translation option that translates the two German words as a phrase.

Note that the two resulting hypotheses are not identical: although they have identical coverage of already translated German words and identical output of English words, they do differ in their probability scores. Using two phrase translations results in a different score than using one phrase translation.

It should also be noted that different model components place different constraints on the possibility of hypothesis recombination.

2.4.1.3 Future Cost Estimation

Pruning out hypotheses is risky, and a key to minimizing search errors is to base on the pruning decision on the right measure [8]. In Fig. 2–8, the hypothesis that translates *the first time* has a better score (−4.11) than the correct hypothesis that starts with the hard words *the tourism initiative* at the beginning of the sentence. Both translated three words, and the worse-scoring (−5.88) but correct hypothesis may be pruned out. Scores in Figure 2–8 are scaled log-probabilities from an English–German model trained on Europarl data (tm: translation model, lm: language model, d: reordering model).

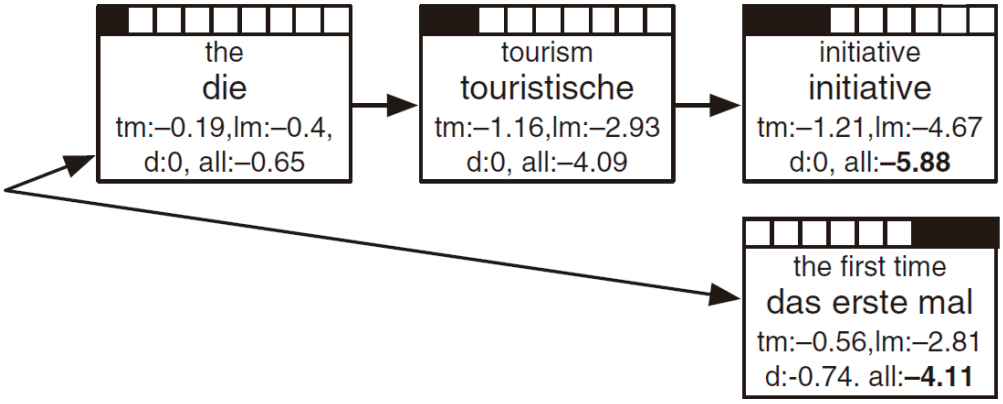

Figure 2–8 Future cost estimation: translating the easy part first [8].

In summary, Beam search will try to find the best translation candidates with lowest overall scores.

2.4.2 End-to-end NN decoding

Some of the NN based models are integrated into SMT as an additional features, such as [30–33, 35]. The decoding for these models are the same as linear models decoding. Recently, end-to-end NN methods are proposed and show helpful for SMT performances [44, 45]. These methods usually mainly use RNN structures. That is, the target word to be translated is predicted by both source and target translated words recurrently. Take Sutskever *et. al.*'s method [44] as example. Fig. 2–9 illustrates its integrated decoding structure.

Their model reads an input sentence *ABC* and produces *WXYZ* as the output sentence. The model stops making predictions after outputting the end-of-sentence token. Note that the LSTM

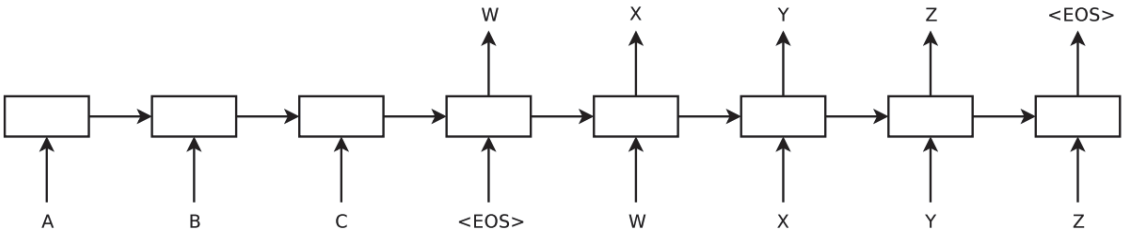

Figure 2–9 End-to-end NN decoding [44].

reads the input sentence in reverse, because doing so introduces many short term dependencies in the data that make the optimization problem much easier.

2.5 SMT Performance Evaluation

Quality is considered to be the correspondence between a machine's output and that of a human: the closer a machine translation is to a professional human translation, the better it is. The most popular SMT performance evaluation is BiLingual Evaluation Understudy (BLEU), which been reported as correlating well with human judgement[72]. Scores are calculated for individual translated segments, typically as n-grams, by comparing them with a set of good quality reference translations. Those scores are then averaged over the whole corpus to reach an estimate of the translation's overall quality. Meanwhile, intelligibility or grammatical correctness are not taken into account.

Besides BLEU, there are also some other automatic metrics, such as Translation Error Rate (TER) [73] and Metric for Evaluation of Translation with Explicit Ordering (METEOR) [74, 75] are also been applied to SMT evaluation. However, they are not so popular as BLEU. Therefore BLEU is adopted as the main evaluation metric in this thesis.

Chapter 3

Neural Network based Statistical Machine Translation

3.1 Converting CSLMS into *N*-gram Language Models

In recent years, LMs in real application systems, such as Amazon Web Services, are increasing in size, because the SMT performance of LM is largely determined by the size of corpus, and available corpora have become very large [76]. BNLMs with cheaper computational cost can be trained from much larger corpora than CSLMs. So improving a BNLM, by using a CSLM trained from a smaller corpus, is a more cheap and realistic way. Actually, a CSLM from a smaller corpus has been shown to enhance SMT [72] reranking [65, 77].

With the same sized training corpus, CSLMs have been shown to outperform BNLMs in SMT performance. However, CSLMs have not been widely used in SMT in practice, because too high computational cost is needed. Various methods have been proposed to reduce the training cost [61, 62, 78], though few of them try to reduce the decoding cost. High computational cost makes it difficult to apply CSLM to SMT decoding directly. So a simple approach to solve this problem is a two pass way: the first pass decodes to produce an n-best list by using a BNLM. In the second pass, those n-best translations are reranked by using CSLM [60–63]. Another approach to reduce decoding cost is to use Restricted Boltzmann Machine (RBM) [64], instead of using multi-layer NNS [19, 30, 32]. Since the probability of an RBM can be calculated efficiently [64], the RBM LM can be used in SMT decoding, but only in a small scale task because of high training cost.

Vaswani *et. al.* [79] extend the Neural Probabilistic Language Model (NPLM) [19] from two angles: 1) the rectified linear unit plays as the activation function that computes much more efficient than Sigmoid or hyperbolic tangent units [80], and 2) Noise-Contrastive Estimation (NCE) [66, 81] is adopted for model training, where the repeated summations over the vocabulary are not necessary. These two extensions enable building large-scale NPLMs efficiently. The main contribution of their work is to reduce training cost and apply CSLM into SMT decoder directly. However, their method is still much slower that n-gram LM. In this thesis, we propose a novel method to speed up decoding. The main idea of using NNLM efficiently behind

our method is to store the pre-calculated probabilities from CSLM into n-gram format, which is quite different from that of [79]. We will compare the decoding efficiency in Section 5.1.2.5. There are also some related researches that implement NNLMs or NN translation models for SMT [25, 26, 28, 33–36, 82–86]. Especially, Devlin *et. al.* [35] propose a fast joint model of LM and translation model using self-normalized NN and pre-computing the hidden layer. However, to integrate these techniques into SMT, almost all these existing methods need more or less modifications over the SMT decoder, while our method can be directly integrated into SMT without any modification.

Since CSLM outperforms BNLM in probability estimation accuracy and BNLM outperforms CSLM in computational time, our key idea of converting CSLM into BNLM is to use the probabilities of n-grams calculated by CSLM to re-write the probabilities of n-grams of BNLM. That is, n-grams from BNLM are used as the input of CSLM, and the output probabilities of CSLM together with the corresponding n-grams of BNLM constitute Converted CSLM (CONV).

The n-grams in the CONV can be viewed as a subset of the CSLM. In another words, this CONV can only represent part of the CSLM. To construct a larger CONV to simulate the CSLM and eventually improve the LM, the n-grams outside the corpus must be considered[1]. The n-grams can be divided into artificial ones and natural ones.

The artificial converting approaches [87–89] mainly focus on speech recognition. [87] use a RNNLM [31] to sample words from the probability distributions and generate texts. Then they train a large BNLM from these texts using interpolated KN smoothing. Arsoy *et.al.* [88] convert NNLMs into BNLMs using *artificial* higher-order n-grams constructed from lower-order n-grams. That is, all the words in short-list are added to the tail of i-grams in the corpus in order to produce (i+1)-grams which may not be in the corpus. These (i+1)-grams are called *artificial n-grams*. However, most of the artificial n-grams do not have linguistic sense. For example, if a trigram is '*would like to*', then the extended 4-grams will be '*would like to* *', where '*' stands for any word in the short-list. A lot of meaningless 4-grams, such as '*would like to would*' or '*would like to cat*', will be generated. Although some of them can be pruned using a modified entropy-based method [89], a large part of meaningless 4-grams will still remain in the converted CSLM. In addition, a huge intermediate LM will be produced, which is commonly thousands times larger than the original LM before pruning[2], and the probabilities of its n-grams will have

[1]Both of the larger CSLM and CONV can improve the performance of LM. Because constructing a larger CSLM is very time consuming, so constructing a larger CONV is a better choice.

[2]We are aware that this intermediate LM can be pruned parallel during converting. However, it still costs more space

to be calculated using the CSLM. Consequently, the huge intermediate LM should be pruned into a specified size. Because Arsoy method is very time and space consuming, it can only be applied to small corpora. For our proposed method, n-grams in the corpus are used as the input n-grams of CSLM. We therefore call them *natural n-grams*. Since no any intermediate LM will be constructed during converting, our method can be applied to very large corpora. Two methods [53] and [51] propose a series of bilingual LM growing methods, which can be viewed as LM adaptation methods because they focus on special domain corpora.

In short, our proposed converting method uses natural n-grams that actually occur in corpus, while Arsoy method uses artificial n-grams. For a small corpus such as 42M words/1M sentences, which is commonly used for SMT, Arsoy method may perform well, because a large CONV can be generated without extra corpus. However, Arsoy method is hard to scale up, because the computational cost will increase dramatically as facing a very large corpus. In contrast, our method can work on an extremely large corpus with billions of words by selecting the most useful n-grams in a large LMs for CSLM converting. That is, we can use our bilingual pruning method specially designed for SMT at first, and then apply our CSLM converting method.

3.1.1 Using Artificial *N*-grams for CSLM Conversion

Since the CSLM can calculate the probabilities of n-grams outside training corpus, one way to construct a large Converted CSLM (CONV) is to generate n-grams based on the training corpus. The existing methods use short-list to construct those *new* n-grams.

To construct the high ordered (i+1)-grams, Arsoy adds all the words in short-list to the tail of i-grams. Then the probabilities of the (i+1)-grams are calculated using the (i+1)-CSLM[1]. As a result, a very large converted (i+1)-gram LM will be generated, and then this large intermediate LM is pruned into smaller size using a modified entropy-based method [90][2]. The (i+2)-grams are grown using (i+1)-grams, recursively. At last the CONV is interpolated with the original BNLM. Figure 3–1 illustrates Arsoy method. The 'BLM' indicates the Background LM. After calculating the probabilities of the bigrams ending with the words in the short-list, Arsoy *et. al.* [88, 89] apply entropy-based pruning, producing a bigram pruned back-off CONV. The size of this model is determined by a pruning threshold. To create trigram background LM, they

(depending on the threshold set for pruning) than the original one.

[1]The (i+1)-CSLM indicates the (i+1)-order CSLM with i history words as input, which can calculate the probabilities of (i+1)-grams.

[2]They use the $P_b(w|h)$ of the background BNLM in Eq. (2–6) to approximate the whole unpruned CONV, and this will save a lot of computational time.

append trigrams from the trigram BNLM to the bigram pruned back-off CONV and renormalize the model. The recursive steps keep going on until the *4*-gram CONV is conducted.

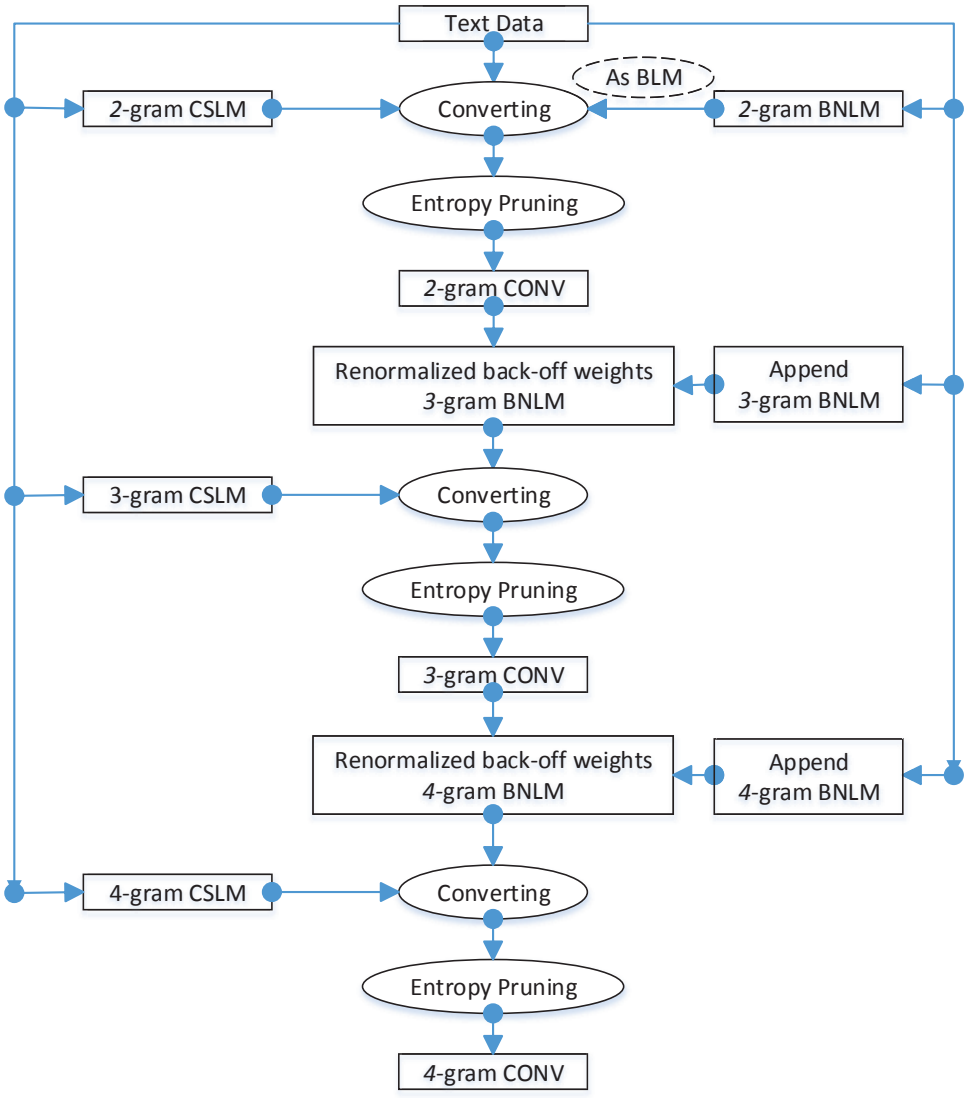

Figure 3–1 Converting CSLM into BNLM using artificial n-grams [88, 89].

3.1.2 Using Natural N-grams for CSLM Conversion

A drawback of Arsoy method is over-generation of poor-quality artificial n-grams, such as '*would like to would*', '*would like to cat*' and so on. Although part of these n-grams can be pruned by using an entropy-based strategy [89], a lot of useless n-grams will still remain in the converted CSLM. Instead of using artificial n-grams, we put forward to generating natural

n-grams from a larger in-domain corpus as the original training corpus.

The pipeline of ***Natural N-Grams Converting (NNGC)*** is given in Figure 3–2. Our new approach can be summarized as the following four steps:

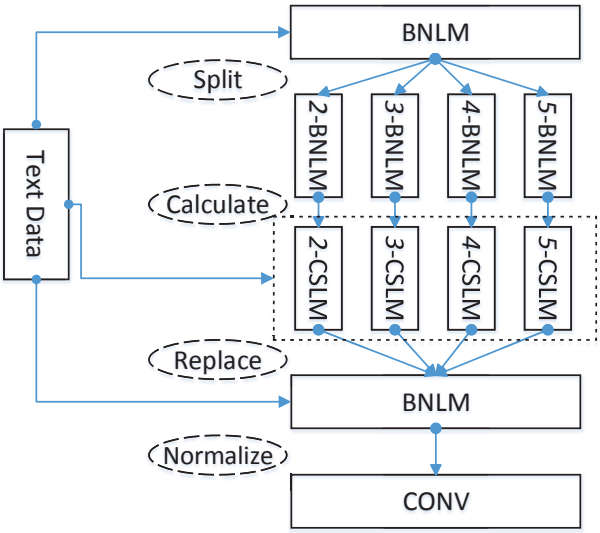

Figure 3–2 Illustration of NNGC CSLM converting.

1) **Split** the BNLM into different order n-grams (n=2,3,4,5), and use the unigrams in BNLM as they are.

2) Take n-grams (n=2,3,4,5) in the BNLM as input of corresponding order (n=2,3,4,5) CSLM[1] and **calculate** the probabilities of the n-grams by using Eqs. (2–5) and (2–6).

3) **Replace** the probabilities of n-grams (n=2,3,4,5) in the BNLM with the probabilities calculated by the corresponding CSLMs. Note that only the probabilities of n-grams ending with a word in the short-list needs to be re-written. When a BNLM trained from a larger corpus is re-written, the n-grams in the BNLM often contain unknown words. In that case, the probabilities of unknown words remain unchanged as they are.

4) **Re-normalize** the probabilities and back-off weights $\alpha(h_i)$ in accordance with Eqs. (2–4) and (3–1) using SRILM's *-renorm* option [59, 91]. This is because the probabilities of n-grams in the BNLM have been changed, so the corresponding back-off weights should also be changed [57, 58]. The re-normalized $\alpha(h_i)$ can be calculated as,

$$\alpha(h_i) = \frac{P_{\text{left-over}}}{\sum P_b(w_i|w_{i-n+2}^{i-1})}, \tag{3–1}$$

[1]Note that the n-gram CSLM means that the length of its history is n-1.

where the $P_{\text{left-over}}$ indicates the left-over probability mass of the (n-1)-gram and $P_b(w_i|w_{i-n+2}^{i-1})$ is the same as in Eq. (2–4).

3.1.3 Comparison of Computational Complexity

3.1.3.1 Computational Complexity of CSLM Converting Methods

The time complexity of the CSLM[1] is given as [30],

$$(n-1) \times P \times H + H + (H \times N) + N, \tag{3–2}$$

where n is the order of n-grams, P is size of projection layer, H is the size of hidden layer and N is the size of output layer. The original N is equal to the size of the vocabulary ($|V|$). To reduce the time complexity, the short-list V_0 in Eq. (2–5), which is a subset of the whole vocabulary, is used as the output layer N. The probabilities of other words outside short-list in the output layer will be calculated using the background BNLM.

Arsoy *et. al.* have applied their method to 4-gram LM for speech recognition [88, 89]. Arsoy method needs to add every word in the short-list after the tail word of i-gram to construct the (i+1)-gram. The short-list usually includes thousands of words, so the generated (i+1)-grams will be thousands more larger than the i-grams before they are pruned. Because the higher order n-grams commonly contain more n-grams, computational cost significantly grows as the order of n-gram increases. The limited size of vocabulary and low order n-gram make Arsoy method applicable to speech recognition. However, SMT needs a higher order BNLM (5-gram LM as common setting for SMT, compared with 4-gram LM for speech recognition). Thus, Arsoy method is not feasible for large BNLMs because the converting time for construction is basically prohibitive.

Since we rewrite the n-grams from the original BNLM, we only need to calculate the same number of n-grams as the original BNLM, instead of thousands times of the original BNLM as that in [88, 89]. NNGC method thus requires much lower computational cost, which makes NNGC have good scalability on large corpora.

[1] We are aware that more than one hidden layers can be applied to CSLM, however, empirical results show that they cannot improve CSLM performance. Therefore only one hidden layer is adopted in this thesis.

3.1.3.2 Computational Complexity of CSLM Methods in Decoding

To improve the efficiency of CSLM in SMT decoding, NPLM [79] use rectified linear units [80], where its activations are cheaper to compute in comparison with sigmoid or hyperbolic tangent units. They also use NCE [66, 81] in output layer, instead of softmax which requires a summation over all the units in the output layer in training, to produce "approximately normalized probabilities" and "simply ignore normalization" in decoding [79]. Arsoy converting method and the proposed NNGC method both store the n-grams and their pre-calculated probabilities from CSLM into n-gram LM format. So computation complexity is similar with n-gram LM. In summary, Table 3–1 shows the computation complexity of all the methods mentioned above[1].

LMs	Time Complexity
BNLM	$n \times \mathbb{C}(mapping)$[a]
CSLM	$(n-1) \times P \times H + H \times \mathbb{C}(tanh) + H \times N + N \times \mathbb{C}(softmax)$[b]
NPLM	$(n-1) \times P \times H + H \times \mathbb{C}(RLU)$[c] $+ H \times N + N$
Arosy	$n \times \mathbb{C}(mapping)$
CONV	$n \times \mathbb{C}(mapping)$

Table 3–1 LM Time Complexity Comparison.

$\mathbb{C}(\cdot)$ stands for time of calculating functions or methods, and $mapping$ stands for searching n-grams in BNLM. n is order of n-grams, P is size of projection layer, H is size of hidden layer and N is size of output layer. Because N is much larger than P or H, $N \times \mathbb{C}(softmax)$ is the most time-consuming part. RLU stands for rectified linear units.

3.1.3.3 Combining LM Conversion and Adaptation

To apply the proposed converting method to a extreme large corpora with billions of words, the time cost of CSLM converting is still an obstacle. Therefore, to prune LMs before converting LMs, without LM quality loss, is important for CSLM converting. A commonly used method for pruning LMs is *cutoff*, which simply discards n-grams whose frequencies are below certain threshold. However, [92] have shown that the cutoff method may lead to significant performance loss in SMT. Another well-known method is entropy-based pruning [90]. According to our best

[1] Devlin *et. al.* [35] propose a joint model of LM and translation model using self-normalized NN and pre-computing the hidden layer. Because their method is a joint model rather than LM only, we did not conduct comparison experiments.

knowledge, this is the *state-of-the-art* pruning method. A shortcoming of this method is that it only uses monolingual information from target language. Motivated by this, we will propose a bilingual based LM pruning/adaptation method in the next subsection.

3.2 NN and Connecting Phrase based SMT Adaptation

Large corpora are important for Statistical Machine Translation (SMT) training. However only the relevant additional corpora, which are also called in-domain or related-domain corpora, can enhance the performance of SMT directly. Otherwise the irrelevant additional corpora, which are also called out-domain corpora, can not enhance or even reduce the performance of SMT [93].

SMT adaptation means selecting useful data from mix-domain (mixture of in-domain and out-domain) data, and applying it to SMT. Existing work focus on various granularity levels. Most of them first select sentences, and then train models by using selected sentences [94–98]. There is a problem for sentence level adaptation: the fragment of a sentence and the whole sentence itself may hold different domain. That is, although a sentence is out-domain, part of it can be in-domain. Some work build lexicon, translation models (Phrase Tables, PTs), reordering models or Language Models (LMs) at first and then select the fragments or adapt the models [99–110]. Some work show that linguistic information can be applied to LM pruning/growing [52, 53], which is similar with domain adaptation.

Recently, NN based methods are widely used in SMT, such as NNLM, translation model, re-ordering model or integrated Neural Machine Translation. Most of these methods focus on better probability estimation instead of better data selection. Du et. al. [96] show that NN can be used for sentence selection and then selected sentences are used to build SMT systems. NN joint models are also proposed for domain adaptation [111, 112]. Most of these methods train two NN models (one in-domain and one out-domain) and penalize the sentences/phrases similar to the out-domain corpora.

Instead of focusing on one method only, we propose two novel phrase-based SMT adaptation methods from the following two aspects: 1) NN aspect: in comparison with the existing NNLMs based method for sentence selection [96] or NN joint models for model adaptation [111, 112], an NN translation model based adaptation method for phrase pair selection is proposed. 2) Linguistic aspect: the translation hypotheses of a phrase-based SMT system are a concatenation of the phrases from PT. Based on this, we propose a connecting phrase-based method to select phrase pairs for translation model adaptation and n-grams for LM adaptation.

Certain phrases (pairs) are selected using these two methods. We will introduce these methods and investigate how these selected phrases can enhance SMT in the following sections.

3.2.1 Hypothesis

Two PTs, one from out-domain corpus and the other from in-domain corpus, are built at first. The hypothesis is that adding all of the out-domain phrase pairs into in-domain PT will decrease the performance of in-domain SMT. However some of these out-domain phrase pairs are useful and can enhance the performance. So we select phrase pairs from the PT made from an out-of-domain corpus, in order to improve the performance of in-domain SMT.

3.2.2 NN based Translation Model Adaptation

The hypothesis behind NN based translation model (or PT) adaptation is: for an in-domain phrase pair (F, E), the translation probabilities $P_{in}(E|F)$ calculated by translation models trained from in-domain corpora should be high and $P_{out}(E|F)$ calculated by translation models trained from out-domain corpora should be low. Conversely, for an out-domain phrase pair (F, E), $P_{in}(E|F)$ should be low and $P_{out}(E|F)$ should be high.

The probability of a phrase-pair is estimated as,

$$P(E|F) = P(e_1, ..., e_q|f_1, ..., f_p), \tag{3–3}$$

where f_i, $i \in [1, p]$ and e_j $j \in [1, q]$ are source and target words, respectively. Originally,

$$P(e_1, ..., e_q|f_1, ..., f_p) = \prod_{k=1}^{q} P(e_k|e_1, ..., e_{k-1}, f_1, ...f_p). \tag{3–4}$$

For the purpose of translation model adaptation , the dependence between target words are dropped[1] and the probabilities of different length target phrase should be normalized[2]. So the final probability estimation is,

$$P(E|F) \approx \frac{\prod_{k=1}^{q} P(e_k|f_1, ...f_p)}{q}. \tag{3–5}$$

Finally, the minus $D_{minus}(E|F)$ between $P_{in}(E|F)$ and $P_{out}(E|F)$ are used to selected the related-domain phrase pair,

[1]This is similar with continuous space translation model [69]. We have also empirically compared the performance of using NN with target words dependence and the results show not so well (not shown for limited space).

[2]If the source phrase contains less than seven words, the projections of missing words are set as zero.

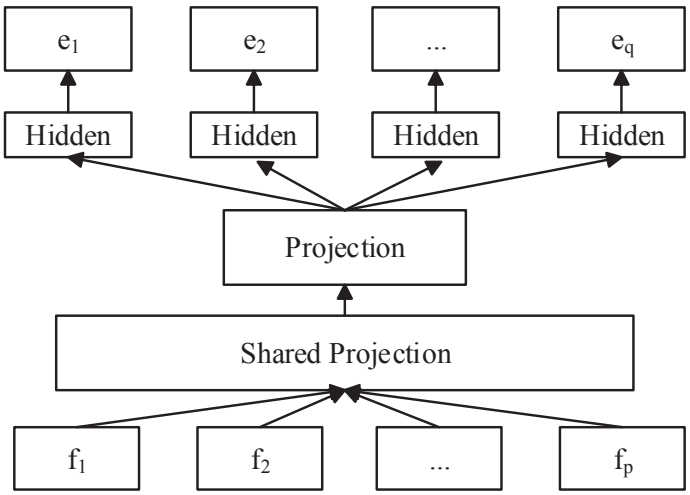

Figure 3–3 NN based translation model

$$D_{minus}(E|F) = P_{in}(E|F) - P_{out}(E|F). \tag{3–6}$$

The threshold of $D_{minus}(E|F)$ for phrase pair selection is tuned using development data.

3.2.3 Connecting Phrases based SMT Adaptation

Suppose that two phrases '*would like to learn*' and '*Chinese as second language*' are in in-domain PT. In decoding, these two phrases may be connected together as '*would like to learn Chinese as second language*". The phrases '*would like to learn Chinese*' or '*learn Chinese as second language*' may not be in in-domain PT, but they may be in out-domain PT. If we can add these connecting phrases, which may potentially occur in decoding, into in-domain bilingual PT or monolingual LM, they may help enhance SMT performance. Connecting phrases can occur in decoding by combining two phrases from in-domain PT. However their translation probabilities are only calculated by the combination of probabilities from in-domain PT. For the proposed methods, the translation probabilities of connecting phrases from out-of-domain corpus are estimated by real corpus. Note that connecting phrases are generated from in-domain PT, then we check if these in-domain connecting phrases actually occur in out-of-domain PT.

3.2.3.1 Connecting Phrases Method

Let w_a^b be a phrase starting from the a-th word and ending with the b-th word, and $\gamma w_a^b \beta$ be a phrase including w_a^b as a part of it, where γ and β represent any word sequence or none. An

i-gram phrase $w_1^k w_{k+1}^i$ ($1 \leq k \leq i-1$) is a connecting phrase[1], if

1) w_1^k is the right (rear) part of one phrase γw_1^k in PT, and

2) w_{k+1}^i is the left (front) part of one phrase $w_{k+1}^i \beta$ in PT.

For a 4-gram phrase '*a b c d*', it is a connecting phrase if at least one of the following conditions holds,

1) phrases 'β *a*' and '*b c d* γ' are in the phrase table, or

2) phrases 'β *a b*' and '*c d* γ' are in the phrase table, or

3) phrases 'β *a b c*' and '*d* γ' are in the phrase table.

3.2.3.2 Language Model Adaptation

LM adaptation is similar with LM pruning. The difference is mainly that LM adaptation focus on selecting n-grams from out-of-domain data, and LM pruning focus on selecting n-grams from in-domain data. In this thesis connecting phrase-based LM adaptation for in-domain data is also called ***Bilingual LM Pruning (BLMP)***.

Research on LM pruning has been focused on the development of the pruning criterion, in order to evaluate the performance loss or gain of pruned model [113, 114]. There are mainly four kinds of methods [113] for LM pruning:

1) The basic traditional method is the cutoff method [115]. It uses the absolute frequency of n-grams in the corpus as the pruning criterion;

2) The Perplexity (PPL) based method assumes that LM performance change can be measured by a weighted logarithmic probability difference before and after n-grams pruning [113];

3) The ranking criterion based method removes all the n-grams which change rank by less than certain threshold before and after n-grams pruning [113];

[1]We are aware that connecting phrases can be applied to three or more phrases. However, experimental results show that more connecting phrases can not improve the performance, so only two connecting phrases are applied in this thesis.

4) The entropy-based method removes all the n-grams which change corresponding entropy by less than certain threshold before and after n-grams pruning [90].

In this thesis, we choose the most commonly used cutoff and entropy-based pruning methods as our baselines.

a) Cutoff Pruning

The definition of the cutoff method is as follows: Set the minimal count k_i of n-grams of order i that will be included in LM, and all n-grams with a frequency lower than k_i will not be included in LM.

b) Entropy-based Pruning

Entropy-based pruning is first proposed by [90]. It is a criterion for pruning n-grams from LM, which is based on the relative entropy between the original and pruned models.

Let $p(a|b)$ denote the conditional probability assigned by original LM, and $p'(a|b)$ denote the probability assigned by pruned LM. Then, the relative entropy between these two models is defined as,

$$D(p||p') = \sum_{w_i,h_j} p(w_i|h_j)[\log p(w_i|h_j) - \log p'(w_i|h_j)]. \tag{3–7}$$

The PPL of the original LM is as,

$$PP = e^{-\sum_{h,w} p(h,w) \log p(w|h)}. \tag{3–8}$$

The PPL of the pruned LM in the original distribution is defined as,

$$PP' = e^{-\sum_{h,w} p(h,w) \log p'(w|h)}. \tag{3–9}$$

The relative change in model PPL after one n-gram pruned is defined as,

$$\frac{PP' - PP}{PP} = e^{D(p||p')} - 1. \tag{3–10}$$

There are mainly three steps for the entropy-based LM pruning:

1) Select a threshold θ.

2) Compute the relative PPL increase due to pruning each n-gram individually.

3) Remove all n-grams that raise the PPL by less than θ and recompute the back-off weights.

As stated above, all of the existing pruning methods only consider the monolingual information. However, both of the parallel information can be applied to SMT decoding, so they may discard potentially useful information for SMT.

Bilingual (Connecting Phrase) based LM Pruning

Since LM only focus on target side monolingual data, the connecting phrase method are only applied to target side. Using connecting phrases, a large LM can be pruned into a smaller one. During this pruning process, we may determine the size of the pruned LM. Thus the *more useful* connecting phrases should be selected by ranking the occurrence probabilities of the connecting phrases in SMT decoding.

Suppose that $P(e|f)$ is the translation probability from f (indicates the source phrase) to e (indicates the target phrase), which is estimated by bilingual training data. The probability of target phrase e occurring in SMT decoding can be estimated as,

$$P_{\text{target}}(e) = \sum_{f} P_{\text{source}}(f) \times P(e|f), \tag{3–11}$$

where $P_{\text{source}}(f)$ is the occurrence probability of source phrase f, which is estimated by source LMs using monolingual training data. This $P_{\text{target}}(e)$ will absorb bilingual information rather than only using monolingual target LMs[1]. If a target phrase e is long, its aligned source phrase f will also be long and has a low occurrence probability $P_{\text{source}}(f)$. According to Eq. (3–14), $P_{\text{target}}(e)$ will also be low.

The occurrence probabilities of connecting phrases are calculated, to determine which connecting phrases should be kept and which should be discarded. Taking an i-gram connecting phrase $w_1^k w_{k+1}^i$ as example, its probability can be roughly estimated as[2]

[1] We can prune the connecting phrases by directly discarding the target phrases e with low occurrence probabilities $P_{\text{target}}(e)$. In detail, we first prune the phrase table according to the target phrase probabilities, then the connecting phrases will be pruned accordingly. In the experiments (not shown), the performance of this method is not as good as the method in this thesis.

[2] In practice, we have also tried to measure the occurrence probability of target phrase using all the features in phrase table, such as phrase translation probabilities, word lexical weights and phrase length penalty. However the performance will decrease (BLEU decrease around 0.3 in comparison with using Eq. (3–15)). So we apply the current method, although it is straightforward.

$$P_{\text{connect}}(w_1^k w_{k+1}^i) = \sum_{k=1}^{i-1}(\sum_{\gamma} P_{\text{target}}(\gamma w_1^k) \times \sum_{\beta} P_{\text{target}}(w_{k+1}^i \beta)), \tag{3–12}$$

where w_1^k is part of γw_1^k, w_{k+1}^i is part of $w_{k+1}^i \beta$, γw_1^k and $w_{k+1}^i \beta$ are phrases in the phrase table. At last, a threshold for occurrence probabilities is set, and only the connecting phrases with $P_{\text{connect}}(w_1^k w_{k+1}^i)$ higher than the threshold are retained. In this way, the sizes of the pruned LMs are tuned to smaller proper ones.

The thresholds for each order n-grams are different. The distribution ratios of different order n-grams (n = 2, 3, 4, 5) in pruned LMs follow the small LM. It should be noted that the unigrams are not pruned because we use the vocabulary of small corpus for all LMs. In practice, the n-grams in different orders are ranked independently. The top n-grams in different orders of large LMs, such as KN5B, are selected with the same distribution ratio of small LMs, such as around 46: 148: 244: 295 of KN42M in NTCIR task (please refer to Section 5.1.1 for LMs setting up).

3.2.3.3 Translation Model Adaptation

For phrase pair (F, E), there are four kinds of situations: a) Both of F and E are connecting phrases; b) either F or E is connecting phrases; c) only F is connecting phrase; d) only E is connecting phrase. We empirically evaluate the performance of these four methods and the results show that a) gains the best BLEU, so it is adopted.

The probability of a phrase w_1^i being a connecting phrase is as follows (using target phrase as example),

$$\begin{aligned} P_{\text{con}}(F, E) = P_{\text{con}}(F) \times P_{\text{con}}(E) = & \sum_{k=1}^{p-1}(\sum_{\beta} P_s(\beta f_1^k) \times \sum_{\gamma} P_s(f_{k+1}^p \gamma)) \times \\ & \sum_{k=1}^{q-1}(\sum_{\beta} P_t(\beta e_1^k) \times \sum_{\gamma} P_t(e_{k+1}^q \gamma)). \end{aligned} \tag{3–13}$$

where the P_s (for source phrase) or P_t (for target phrase) is calculated using source of target monolingual LM trained from in-domain corpus.

where P_s (for source phrase) or P_t (for target phrase) is calculated using source or target monolingual LM trained from in-domain corpus.

The threshold of $P_{\text{con}}(F, E)$ for phrase pair selection is also tuned using development data.

3.2.3.4 Integration into SMT

The adapted LM has the same format as the original BNLM, so it can be directly used in SMT.

For translation model, the selected phrase pairs are added into the in-domain PT. Although these additional phrases are related to the in-domain ones, they are not as useful as the in-domain ones. So a penalty score is set as an additional score for them. That is, for in-domain phrase pairs, the penalty is set as one; for the out-domain ones the penalty is set as e (approximately 2.718)[1].

The thresholds for NN and connecting phrases based selected phrases are tuned using development data. Namely, we first determine the right order of magnitude (such as 10^{-1}, 10^{-2}, 10^{-3}, 10^{-4}...) empirically, and then finely tune its values (not shown for limited space).

The selected phrase pairs using NN based method and connecting phrase-based method can be used independently or jointed together. There are several methods to combine them together, such linear PT interpolation or using two PTs together. We choose linear PT interpolation after several empirical experiments (not shown for limited space).

The phrase pairs in re-ordering model are selected using the same way as PT. The selected monolingual n-grams are added to the original LM, and the probabilities are re-normalized by SRILM [59, 91].

3.2.3.5 Computational Complexity of Adaptation Methods

For connecting phrase adaptation methods:

1) the time complexity: entropy-based pruning method needs to calculate all the related probabilities and back-off weights for the n-grams with the same history h and back-off history h'. That is, the time complexity of entropy-based LM pruning is $O(|N|)$, where $|N|$ is the size of n-grams. In comparison, our proposed BLMP only takes the n-gram itself into account, which allows parallel computation speedup[2]. That is, the time complexity of BLMP is $O(1)$.

2) the space complexity: the entropy-based pruning method needs to load all the related n-

[1]The weights of penalty will be further tuned by MERT. The penalty setting is similar with [102].

[2]Arsoy *et. al.* proposed a modified pruning method specially for CSLM converting, which can run parallel specially in the CSLM converting method with a short-list [88, 89]. Arsoy method can only work recursively parallel, not fully parallel. That is, they can only prune i-grams parallel, and (i+1)-grams parallelly one after another, but not prune both n-grams simultaneously. BLMP can prune n-grams in different orders parallel. In addition their method can not be used in the common LM pruning, where there is no short-list for CSLM converting. This is because the neuron in the CSLM of calculating the probabilities of n-grams inside or outside short-list takes part in their pruning method, however for a common LM pruning, there is no this neuron for the short-list.

grams of the LM into the memory, which is hard to be implemented if a very huge LM (such as larger than 1T) needs to be pruned. In contrast, BLMP only needs to take the n-grams used for pruning, rather than loading any other n-gram.

For NN based adaptation methods, the complexity is similar as CSTM.

3.3 NN and Connecting phrase-based SMT Generation

For some rare languages or specific-domain languages, it is very difficult or even impossible to obtain large enough corpora for SMT training. Instead, only a very small corpus is available. As mentioned above, NN has the ability to calculate the probabilities of phrase pair and n-gram outside the original corpus. Therefore, we propose LM and translation model generation method by using NN in this section.

3.3.1 Language Model Generation

Nowadays, many studies focus on constructing larger BNLMs [76, 92, 116], for a better PPL. These larger LMs have been successfully applied to Statistical Machine Translation (SMT) [76] and help bring better BLEU [72]. Meanwhile, larger in-domain corpora used in LM training in SMT[1] are necessary in most of the existing approaches. How to select the in-domain corpora is also a critical problem in large LM constructing and adaptation, because additional corpora from different domains with original one may not result in better LMs [99] or translation performance [94, 96, 117, 118]. Meanwhile, it will be very difficult or even impossible to collect an extra large corpus for some rare languages or in some special domains, such as the Technology Entertainment Design (TED) corpus [119]. Therefore, it is an important topic on how to improve the LM performance in SMT without assistance of extra corpus.

Language Model Growing (LMG) refers to generating additional *n*-grams and their probabilities and adding them into original LM. LMG is useful because it can improve LM through adding more and more useful *n*-grams from a small training corpus. In the past decades, various methods [114, 120–122] have been developed for selecting *n*-grams from corpus measure by various criteria. However, none of these approaches can conduct the *n*-grams outside the corpus.

Recently, CSLMs (NNLMs) [19, 30–32], are actively used in SMT [60, 62–64]. These models have demonstrated that CSLMs can improve BLEU scores of SMT over *n*-gram LMs with

[1] It is common to use larger monolingual corpus in SMT, in comparison to the small bilingual parallel corpus.

the same sized corpus for LM training. An attractive feature of CSLM is that it can overcome the *data sparsity* problem and estimate the probabilities of *n*-grams outside training corpus more directly and accurately.

Similar as mentioned in Section 3.1, it is difficult to apply CSLMs to SMT decoding directly, due to too high computational cost. So a simple approach is a two-pass procedure [60–63]. Another method is using RBMs [64] in SMT, instead of multi-layer NNs [19, 30, 32]. Vaswani *et. al.* propose several technologies to reduce the training cost and integrate CSLM into SMT decoder directly [79]. However, their LM cannot run as fast as BNLM. Some other studies try to implement neural network LM or translation model for SMT [25, 26, 28, 33–36, 82, 86]. But until now, *n*-gram LM is still the state-of-the-art method in decoding speed.

To integrate CSLM more efficiently into decoding, some existing approaches calculate the probabilities of the *n*-grams before decoding and store them [50, 53, 88, 89]. The '*converted CSLM*', which is mentioned in previous sections, is directly used in SMT, and its decoding speed is as fast as the *n*-gram LM. Actually, more *n*-grams which are outside the training corpus can be generated by using these '*converting*' methods. Unfortunately, all of these existing approaches can only construct an LM with the similar size of the original *n*-gram LM.

These CSLM methods mentioned above can calculate the probabilities of the *n*-grams outside training corpus more accurately. However, it is difficult to decide which *n*-grams should be grown by using monolingual target language information for SMT, because the *n*-grams appearing in phrase-based SMT should be selected from the bilingual phrase table. As we know, every translation candidate phrase for phrase-based SMT decoding is from the bilingual phrase table. Therefore, additional *n*-grams from a larger LMs, which are outside the phrase table, will never be actually used in SMT. These *n*-grams are useless because they do nothing but waste computing time and storage space in LM constructing.

For phrase-based SMT system, the translated phrases are mainly from either of the following two cases: 1) the phrase is already included in a phrase in the phrase table, or 2) the phrase is the result of concatenating two or more phrases in the phrase table. These phases are called '*connecting phases*'. Based on this observation, the probabilities of the connecting phrases, which are not all in the training corpus, can be calculated by CSLM. In this section, a novel NN based bilingual LM growing method will be introduced, to make use of the connecting phrases without assistance of extra corpus.

3.3.1.1 Related Work

Most of the existing LM growing methods need extra larger monolingual corpus and focus on how to select more useful *n*-grams from the corpus by different criteria.

Ristad and Thomas describe an algorithm for growing *n*-gram LM [120]. They use a greedy search for finding the individual candidate *n*-grams to be added to the LM by using '*Minimum Description Length (MDL)*'-based cost function. They obtain significant improvement over their baseline *n*-gram LM, but their baseline model performs worse as longer contexts are used according to [114]. This means that the baseline model they used [120] is not well optimized.

Niesler and Woodland present a method for backing-off from standard *n*-gram LMs to cluster LMs [121]. They present an approach to grow a class *n*-gram LM, which estimates the probability of a cluster given the possible word clusters of the context. They also use greedy search for finding the candidates to be added to the LM similar with the technique used by [120]. The difference is that they add conditional word distributions for *n*-gram contexts and prune away unnecessary *n*-grams.

Siu and Ostendorf construct *n*-gram LM as a tree structure and show how to combine the tree nodes in several different ways [122]. Each node of the tree represents an *n*-gram context and the conditional *n*-gram distribution for the context. Their experiments indicate that most gain can be achieved by choosing an appropriate context length separately for each word distribution, and the size of LMs can be halved with no significant loss in performance.

Siivola *et. al*. present a method for estimating variable-length *n*-gram LM incrementally while maintaining some aspects of Kneser Ney smoothing [114], which is popular applied to several natural language processing tasks. The growing algorithm is similar to that of [121], whereas their method uses a MDL-based cost criterion. The MDL criterion is defined in a simpler manner than in the algorithm of [120], where a more compact and more theoretical criterion is developed.

It should be noted that these four kinds of methods mentioned above do not consider to grow the *n*-grams outside the corpus. As a result, they actually belong to LM pruning or adaptation methods from the point of view that they all construct a smaller LM from a large corpus.

As CSLM or NNLM makes it possible to calculate the probabilities of the *n*-gram outside the training corpus more accurately, various studies try to implement NNLMs or translation models to SMT [28, 33–35, 69, 79, 83]. However, their decoding speed is still not as fast as the *n*-gram LM. Because directly using CSLM in SMT or speech recognition is very time consuming, some studies focus on converting CSLM into *n*-gram LM.

In our previous sections, we propose a method for converting CSLM into *n*-gram LM [50]. The probabilities of the *n*-grams in training corpus using CSLM are calculated, and then stored together with the *n*-grams in the format of *n*-gram LMs. The converted LM can be directly used in SMT decoding with the same speed as the original *n*-gram LM. The results show that this conversion method can obtain better BLEU in SMT. However, it can not generate the *n*-grams outside the corpus. We select this method as a baseline in this study.

Arsoy *et. al.* present another CSLM converting method [88, 89]. The basic idea behind their method is that a very large LM is generated by adding all the words in the '*short-list*' of CSLM after the '*tail (end) word*' of every *n*-gram of the original *n*-gram LM[1]. The LMs are pruned into the same size of the original *n*-gram LM using entropy-based LM pruning method [90]. Their converted CSLM is applied to speech recognition. In fact, a larger LM has been actually grown, but this method needs to spend additional time and space on the large converted LM[2]. To our best knowledge, the method developed by Arsoy *et. al.* [88, 89] is the only exiting LM growing approach that starts from an original small corpus. We also add their method into the baselines for comparison.

In addition, all the existing LM growing methods only exploit monolingual information from corpus and do not take bilingual information into account. The performance of these grown LMs are mostly measured by PPL or word error rate in speech recognition, which is beyond the SMT topic of this thesis.

3.3.1.2 Bilingual LM Growing

This section describes the proposed method, '*bilingual CSLM growing method*' or '*connecting phrase-based CSLM growing method*'.

Following the discussion in Section 4.1, the translation output of phrase-based SMT system can be viewed as a concatenation of phrases[3] from phrase table. This means that an *n*-gram that appears in a translation output satisfies either one of the following two conditions: 1) it is included in a phrase in the phrase table or 2) it is the result of concatenating two or more phrases in the phrase table.

Based on the above observations, we propose the following procedure for constructing connecting phrases:

[1]The definition of *short-list*, *tail (end) word* and other details of CSLM will be given in Section 5.1.2.

[2]We are aware that this intermediate LM can be pruned parallel during converting, however it still costs more space than the original one depending on the threshold set for pruning.

[3]Except unknown words.

Step 1. At first, the *n*-grams included in phrase table should be maintained;

Step 2. The connecting phrases defined in Section 3.2.3.1 should be generated.

After the probabilities of generated *n*-grams are calculated using CSLM (in Section 5.1.2), the *n*-grams in the phrase table from Step 1 and the connecting phrases from Step 2 are combined, and the combined LM is re-normalized. Finally, the connecting phrase-based grown LM is built up.

3.3.1.3 Ranking the Connecting Phrases

Using connecting phrase LM growing method, the *n*-grams outside the corpus can be generated, and a larger LM can be constructed. However, all of the connecting phrases is too huge to be added into LM, because their sizes are usually more than one Terabyte (TB). Therefore, it is necessary to determine and measure the usefulness of connecting phrases which are more likely to occur in SMT. The connecting phrases can be ranked by their appearing probabilities in SMT decoding. In this way, the size of the grown LM will be tuned by selecting connecting phrases with high appearing probabilities.

Similar as Section 3.2.3.2, the appearing probability of a target phrase e in SMT decoding can be estimated as,

$$P_{target}(e) = \sum_{f} P_{source}(f) \times P(e|f). \tag{3–14}$$

In comparison with using monolingual target LMs only, the bilingual based $P_{target}(e)$ obtain bilingual information. The probability of the connecting phrases appearing in SMT decoding can be roughly estimated as,

$$P_{connecting}(w_1^k w_{k+1}^i) = \sum_{k=1}^{i-1}(\sum_{\beta} P_{target}(\beta w_1^k) \times \sum_{\gamma} P_{target}(w_{k+1}^i \gamma)), \tag{3–15}$$

and then the value of threshold for $P_{connecting}(w_1^k w_{k+1}^i)$ is set. Finally, only the connecting phrases, whose appearing probabilities in SMT decoding higher than certain threshold, is selected as *n*-gram to be further added into the original LM. These generated *n*-grams are called '*grown n-grams*'. This proposed method is called ***Bilingual LM Growing (BLMG)***.

3.3.1.4 Calculating the Probabilities of Grown *N*-grams

For BLMG, *5*-gram BNLM and *n*-gram (n=2,3,4,5) CSLMs are built from the target monolingual corpus. Phrase table is built from the bilingual parallel corpus.

The probabilities of unigrams will be maintained as in the original BNLM. Using BLMG method, the *n*-grams from the bilingual phrase table will be generated. The distributions of *n*-grams with different orders (n=2,3,4,5) of the original BNLM and the grown LMs are set all the same. 2,3,4,5-CSLMs is used to calculate the probabilities of grown *n*-grams (n=2,3,4,5), respectively. That is, the grown *n*-grams will be the input into CSLMs, and the calculated probabilities as the output. It should be noted that the $P_b(\cdot)$ in Eq. (2–5) will be adopted, if the tail word of a grown *n*-gram is not in short-list.

The grown *n*-grams (*n*=2,3,4,5) are combined together, and the probabilities and back-off weights of the *n*-gram LM are re-normalized using the SRILM's '*-renorm*' option [59, 91]. Finally the grown LM and the original BNLM are interpolated[1] using the default setting of SRILM. Figure 3–4 illustrates the entire process.

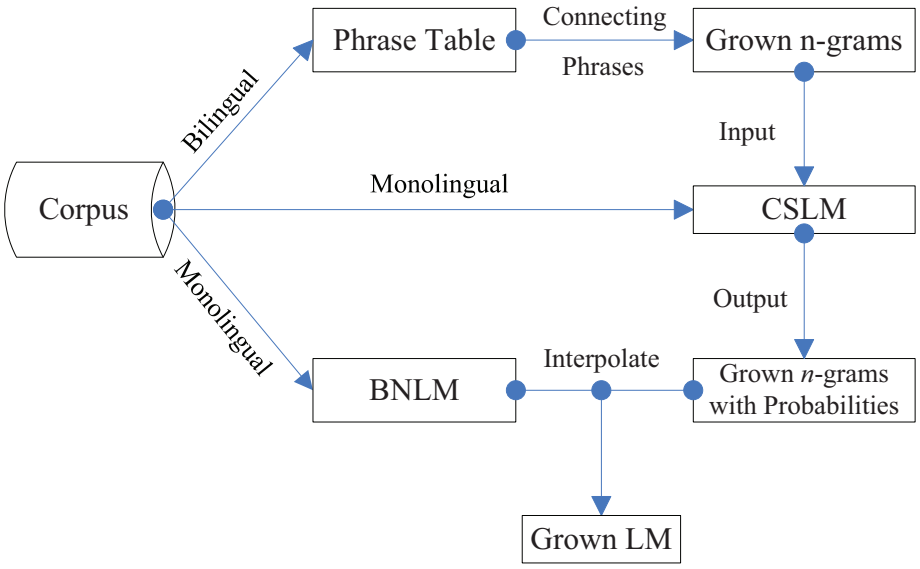

Figure 3–4 Process of bilingual CSLM growing method.

[1]The interpolation is setup for fair comparisons with Wang *et. al* [50]. and Arsoy *et. al*.'s methods [88, 89], because they both use interpolation.

3.3.2 Translation Model Generation

Similar with LM generation for monolingual additional n-grams, translation model generation focuses on bilingual phrase pair generation.

As mentioned in Section 3.2.2, the translation probabilities from F (indicates the source phrase) to E (indicates the target phrase) are estimated by Eq.3–5 using NN based translation models.

CSTM can be used to measure the $P(E|F)$. The w'_{ej} whose $P(E'|F)$similar with w_{fi} in distance is selected as the new translated candidate word. The w_{ej} in original phrase $E(w_{e_1}, w_{e_2}, ..., w_{e_j}, ..., w_{e_l})$ is replaced with w'_{ej} and the generated target candidate phrase $E'(w_{e_1}, w_{e_2}, ..., w_{e'_j}, ..., w_{e_l})$ for source phrase F is formed consequently. For a target phrase $E'(w_{e_1}, w_{e_2}, ..., w_{e_j}, ..., w_{e_l})$, only one word w_{e_j} is replaced with w'_{e_j} and other words maintain as they are[1].

Using the same pipeline, the generated source candidate phrase $F'(w_{f_1}, w_{f_2}, ..., w_{f'_i}, ..., w_{f_k})$ for target phrase E can also be formed. We generate both of source and target phrases, because some phrases in test data may also not be in phrase-table but have similar meaning with them.

The size tuning of generated phrase pairs is ranked using a threshold as follows, which is tuned using development data:

$$\frac{P(E'|F)}{P(E|F)}, \tag{3–16}$$

Similar as the proposed adaptation methods, the generated LM and translation model have the same format as the original LM and translation model, and can be directly integrated into phrase-based SMT system. That is, their probabilities are estimated by NN models, which makes them more accurate and their storage format is the same as n-gram LM and phrase table, which makes them more efficient.

[1]We are aware that two or more words can be replaced, however it may lead to serious sense bias, and experimental results indicated that replacing more than two words does not improve the performance (not shown).

Chapter 4

Bilingual Graph Semantic Model for SMT

4.1 Introduction

Continuous representations of words onto multi-dimensional vectors enhance natural language processing, especially Statistical Machine Translation (SMT), by measuring similarities of words using distances of corresponding vectors. Most of early works are derived from cognitive processing such as WordNet [11], in which lexicon is organized conceptually as a set of terms associated with a partition into Synsets[1], though words organized in this way are not conveniently represented as vectors.

Word embedding for vector representation is usually built in two-steps. The first step is about determining the detailed context related to a given word. The second step is to summary the relationship between word and its context into lower dimensions.

For context determination, three categories can be identified. 1) The first category is to exact the word or word relation information from the entire text, which is usually regarded as document level processing, such as bag-of-word, latent semantic analysis, latent dirichlet allocation and so on. 2) The second category is to use sliding window, such as n-grams, skip-grams or other local co-occurrence relation [22, 25, 123, 124]. 3) The third category, that has been seldom considered, uses much more sophisticated graph style context. Ploux and Ji [125] describe a graph based semantic matching model using bilingual lexicons and monolingual synonyms[2]. They later represent words using individual monolingual co-occurrences [126]. Saluja *et al.* [127] propose a graph based method to generate translation candidates using monolingual co-occurrences.

For relationship summarising, NNs are very popular for word embeddings and SMT recently [22, 25–28, 35, 44, 128]. There are also some works which use matrix factorization [125, 129, 130] and canonical correlation analysis [131, 132] for monolingual or bilingual word embedding.

[1] Synset is a small group of synonyms labeled as a concept.

[2] `http://dico.isc.cnrs.fr`

Sense gives more exact meaning formulization than word itself. However, most of these existing methods embed words as vectors using sliding window or document information directly, instead of sense information. Motivated by this, we propose ***Bilingual Contexonym Cliques (BCCs)***, which are extracted from bilingual Point-wise Mutual Information (PMI) based word co-occurrence graph. BCC plays a role of minimal unit for bilingual sense representation. Correspondence Analysis (CA) is used for summarizing BCC-word matrix into lower dimensions vectors for word representation. This work extends previous monolingual method [125], which need bilingual lexicons or synonyms for bilingual mapping and have never been applied to SMT.

The remaining of this chapter is organized as follows: the proposed ***Bilingual Graph-based Semantic Model (BGSM)*** will be introduced in Section 4.2. The BGSM will be applied to phrase translation probability estimation in Section 4.3 and phrase pair generation in Section 4.4 to enhance SMT.

4.2 Bilingual Graph-based Semantic Model

4.2.1 Graph Constructing

Formally, words are considered as nodes (vertices) and co-occurrence relationships of words are considered as the edges of graph. An edge-weighted graph derived from a bilingual corpus is defined as,

$$G = \{W, E\}, \tag{4–1}$$

where W is bilingual node set and E is edge set which is weighted by co-occurrence relationship, whose definitions are given as follows.

For a given bilingual parallel corpus, each source sentence $S_F = (w_{f_1}, w_{f_2}, ..., w_{f_k})$ and its corresponding target sentence $S_E = (w_{e_1}, w_{e_2}, ..., w_{e_l})$ are combined together to construct a *Bilingual Sentence* $(BS) = (w_{f_1}, w_{f_2}, ..., w_{f_k}, w_{e_1}, w_{e_2}, ..., w_{e_l})$. For words (either source or target word) w_i and w_j, if they are in the same BS, they are called co-occurrences for each other and marked as n_i and n_j in the graph G. **It should be noted that word w_i are always marked as node n_i of graph, so word w_i and node n_i have the same meaning in this thesis**. The *Edge Weight* (EW) connecting nodes n_i and n_j is defined by a modified PMI measure,

$$EW = \frac{Co(n_i, n_j)}{fr(n_i) \times fr(n_j)}, \tag{4–2}$$

where $Co(n_i, n_j)$ is the co-occurrence counting of n_i and n_j and $fr(n)$ stands for how many times a word (node) n occurs in the corpus.

For nearly all languages, stop words such as *of, a, the* in English or *de, une, la* in French have a wide distribution, and this results in that most nodes in the graph are unnecessarily connected with these stop-word nodes. A filter is thus set to prune these poorly connected edges [133] whose EW less than a threshold of γ. γ is tuned using the development data to let the resulted graph keeps the more useful edges based on the empirical results according to tasks (not shown for limited space).

4.2.2 Context-Dependent Clique Extraction

A clique defined in graph theory means a maximum, complete sub-graph [134]. For a subset of nodes with edges in the whole graph, if every two nodes in the subset are connected to each other by an edge, this subset of nodes form a clique. Suppose that both N_1 and N_2 are subsets of N in the graph G. If $N_1 \subset N_2$, N_1 cannot be a clique (maximality).

Graph problems are mostly associated with high computational complexity, such as finding all cliques from a graph (*Clique Problem*). The *Clique Problem* related to our model has been shown *NP-complete* [135], and it is time consuming or even impossible to find the cliques from the whole graph built from a very large corpus (such as millions of sentences) without any pruning. In addition, not all of the nodes are useful for word representations, because some nodes do not have any connection with input contextual words. These nodes actually have not a direct impact over the clique extraction. For a word n and its contextual words $\{n_1, n_2, ..., n_i, ...n_t\}$ as input[1], only the co-occurrence nodes n_{ij} of each n_i (including n itself) are defined as useful nodes. The set of nodes $\{n_{ij}\}$ with their weighted edges form an extracted graph $G_{extracted}$ for further cliques extraction.

So the number of nodes in the extracted graph $G_{extracted}$, $|N_{extracted}|$, is computed by

$$|N_{extracted}| = \left| \bigcup_{\forall i,j} \{n_{ij}\} \right| . \tag{4–3}$$

In practice, the $|N_{extracted}|$ is much smaller than $|V|$ (vocabulary size of bilingual corpus). For a typical corpus (IWSLT FR-EN in Section 5.4.2), $|N_{extracted}|$ is around 371.2 and $|V|$ is 162.3K. Thus the clique extraction in practice is quite efficient as it works over a quite small

[1]For SMT task, the words in aligned phrases are used as contextual words, please refer to Section 4.3 for details.

sized graph[1].

Clique extraction follows a standard routine in [134]. As the clique in this thesis is to represent a fine grained bilingual sense of a word given a set of its contextual words, it is called ***Bilingual Contexonym Clique (BCC)***. Similar but more fine grained than synset (a small group of synonyms labeled as concept) defined in WordNet [11], the BCC plays a role of minimal unit for bilingual meaning representation. Taking the word *work_e* and *readers_e* as an example (without context), some of the BCCs are as follows in Table 4–1:

Words	BCCs
work_e	{*employees_e*, *travail_f* (work), *unemployed_e*, *work_e* } {*heures_f* (hours), *travaillent_f* (to work, third-person plural form), *travailler_f* (work), *week_e*, *work_e* } {*readers_e*, *work_e* }...
readers_e	{*journaux_f* (newspapers), *lire_f* (read), *newspaper_e*, *presse_f* (press), *readers_e*, *reading_e*} {*informations_f* (information), *journaux_f*(newspapers), *online_e*, *readers_e*} {*readers_e*, *work_e*}...

Table 4–1 Some example of BCCs.

Suffixes '*_e*' and '*_f*' are used to indicate English or French, respectively. The English words in parentheses are the translations of corresponding French words.

Please note that edges pruning is static, and nodes selection is dynamic depending on input words sequence. The proposed node selection and clique extraction follows [125], except that we use co-occurrence graph rather than synonym or hypo(hypero)nym graphs. BCCs can be regarded as *noisy* synsets, because there are some useless information in them. To reduce noise and get semantic vector representations, a dimension reduction method proposed as follows.

4.2.3 Semantic Spatial Representation

Correspondence Analysis (CA) is conceptually similar to Principal Component Analysis (PCA), but applies to categorical rather than continuous data [136, 137]. It assesses the extent of match-

[1]Please refer to Section 5.4.4.3 for efficiency comparison in details.

ing between two variables and determines the first n factors of a system of orthogonal axes that capture the greatest amount of variance in the matrix. The first axis (or factor) captures the largest variations, the second axis captures the second largest, and so on. CA has been applied to some related semantic tasks [125, 126].

To project words onto lower dimensional semantic space, CA is conducted over the clique-word matrix constructed from the relation between BCCs and words. An initial correspondence matrix $M = \{m_{ij}\}$ is built, where m_{ij} = 1 if the BCC in row i contains the word in column j, and 0 if not. Normalized correspondence matrix $\mathcal{P} = \{p_{ij}\}$ is directly derived from M, where $p_{ij} = m_{ij}/N_M$, and N_M is grand total of all the elements in M. Let the row and column marginal totals of $\mathcal{P}$ be r and c which are the vectors of row and column masses, respectively, and D_r and D_c be the diagonal matrices of row and column masses. Coordinates of the row and column profiles with respect to principal axes are computed by using the Singular Value Decomposition (SVD) as follows.

Principal coordinates of rows $\mathcal{F}$ and columns $\mathcal{G}$:

$$\mathcal{F} = {D_r}^{-\frac{1}{2}} U\Sigma, \;\; \mathcal{G} = {D_c}^{-\frac{1}{2}} V\Sigma, \tag{4–4}$$

where U, V and Σ (diagonal matrix of singular values in descending order) are from the matrix of standardized residuals S and the SVD,

$$S = U\Sigma V^* = {D_r}^{-\frac{1}{2}} (P - rc^*) {D_c}^{-\frac{1}{2}}, \tag{4–5}$$

where $*$ denotes conjugate transpose and $U^*U = V^*V = I$.

By above processes, CA projects BCCs ($\mathcal{F}$) and words ($\mathcal{G}$) onto semantic geometric coordinates as vectors. Inertia χ^2/N_M is used to measure semantic variations of principal axes for $\mathcal{F}$ and $\mathcal{G}$:

$$\chi^2/N_M = \sum_i \sum_j \frac{(p_{ij} - r_i c_j)^2}{r_i c_j}. \tag{4–6}$$

Following standard setting of CA [137], top six principal dimensions (axes)[1] of vectors are chosen for word and clique representation. ***Bilingual Graph-based Semantic Model (BGSM)*** is constructed from these principal dimensions. In short, a word with its context are used as input of BGSM, and vectors of the word[2] and its bilingual co-occurrences are output.

[1] Experimental results show that six dimensions obtain the best performance on development data (not shown for limited space).

[2] In fact both BCCs and words can be represented as vectors.

The whole pipeline of constructing and using BGSM is illustrated in Figure 4–1.

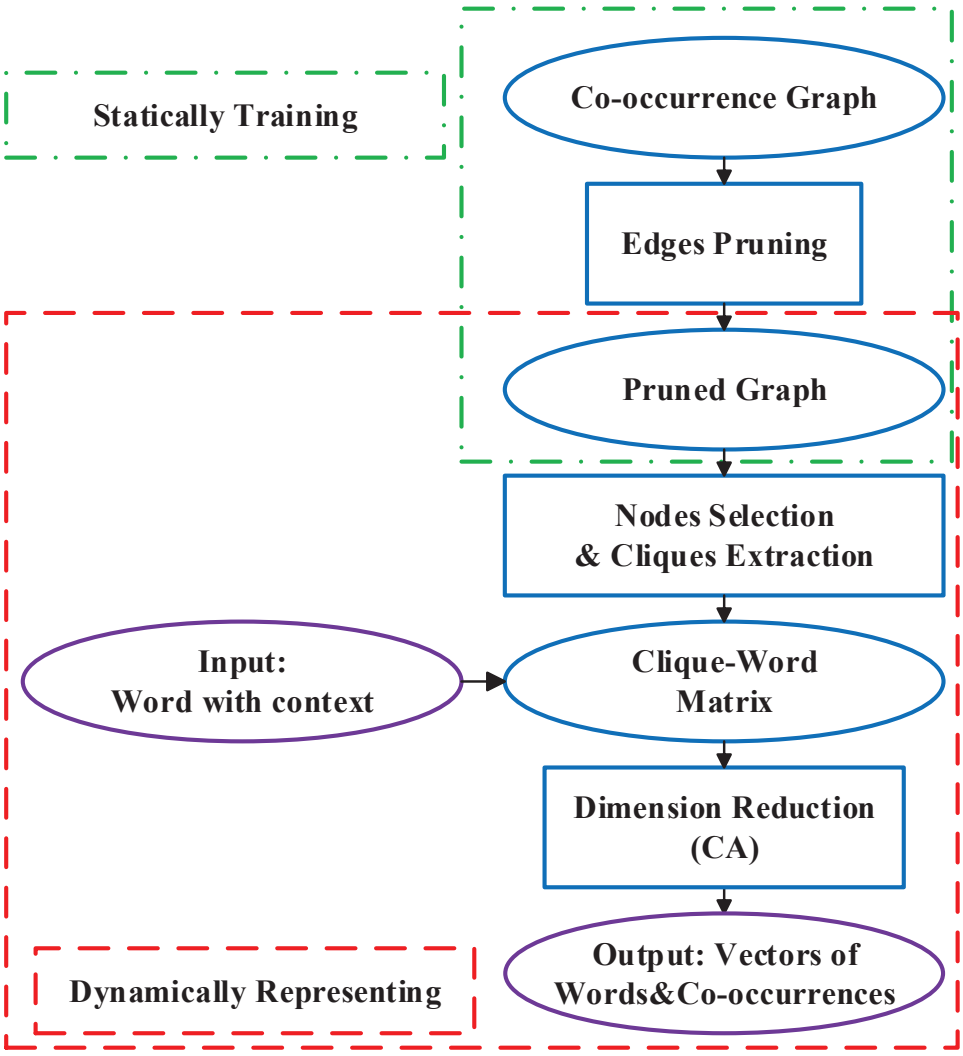

Figure 4–1 Pipeline of training and using BGSM.

To visualize the results, top two dimensions are chosen and illustrated into spatial map. We prefer to illustrate the spatial relationship between BCCs and words in the same map, instead of the spatial map of words only. BCCs are represented by points and words by regions. Label of word is approximately (to avoid overlapping) placed at the barycentre of region (gray line) delineated by a set of BCCs that contain the word. BCCs are clustered [138] into three groups (green line). We only present several typical words, and the words too far from most of other words are discarded due to limited space in the figures.

Figures 5–2 and 5–3 illustrate the spatial representation of all the co-occurrence words when

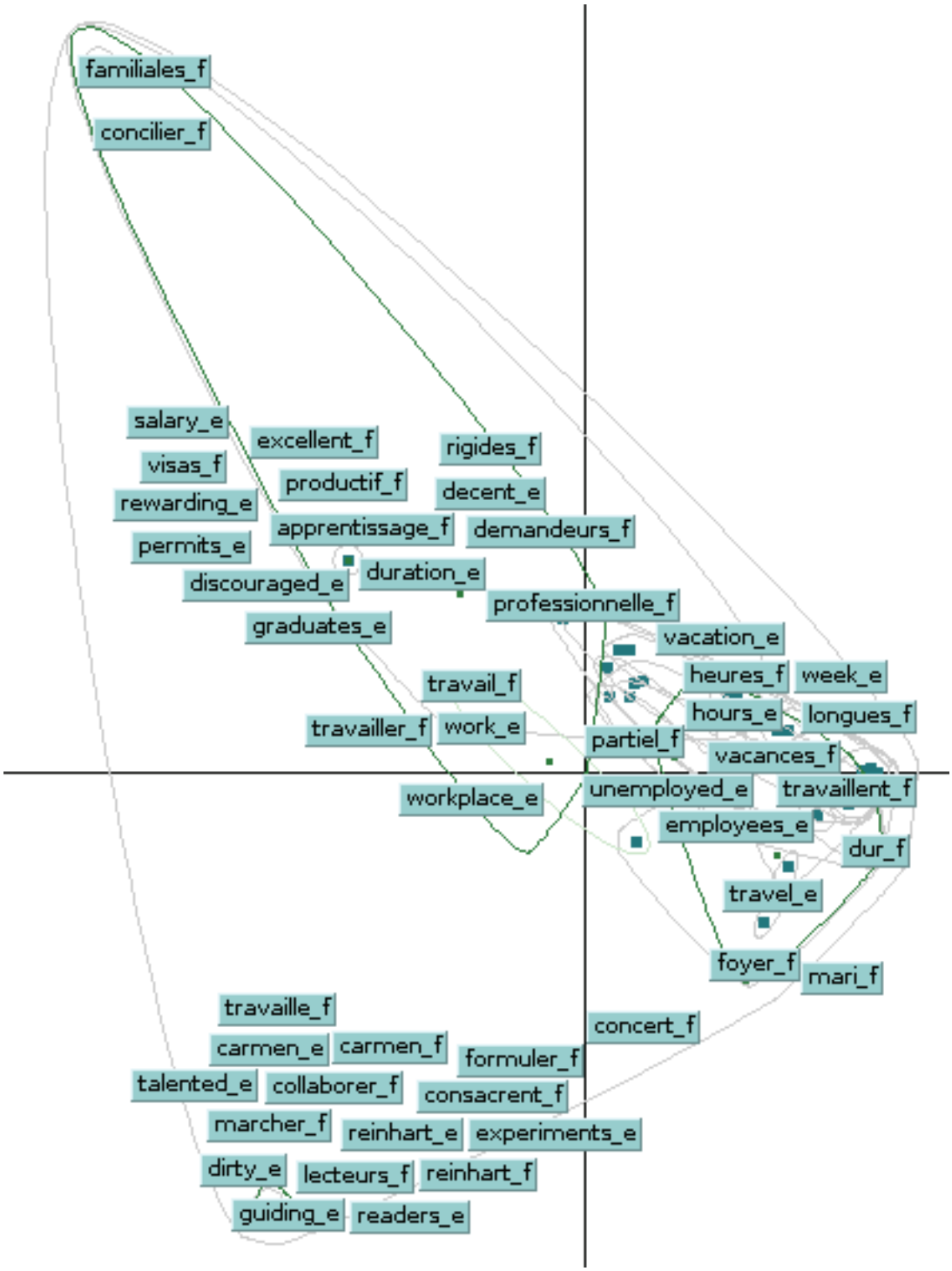

Figure 4–2 Spatial map of *work_e* (without context) as input (The green lines indicate the borderline of clusters).

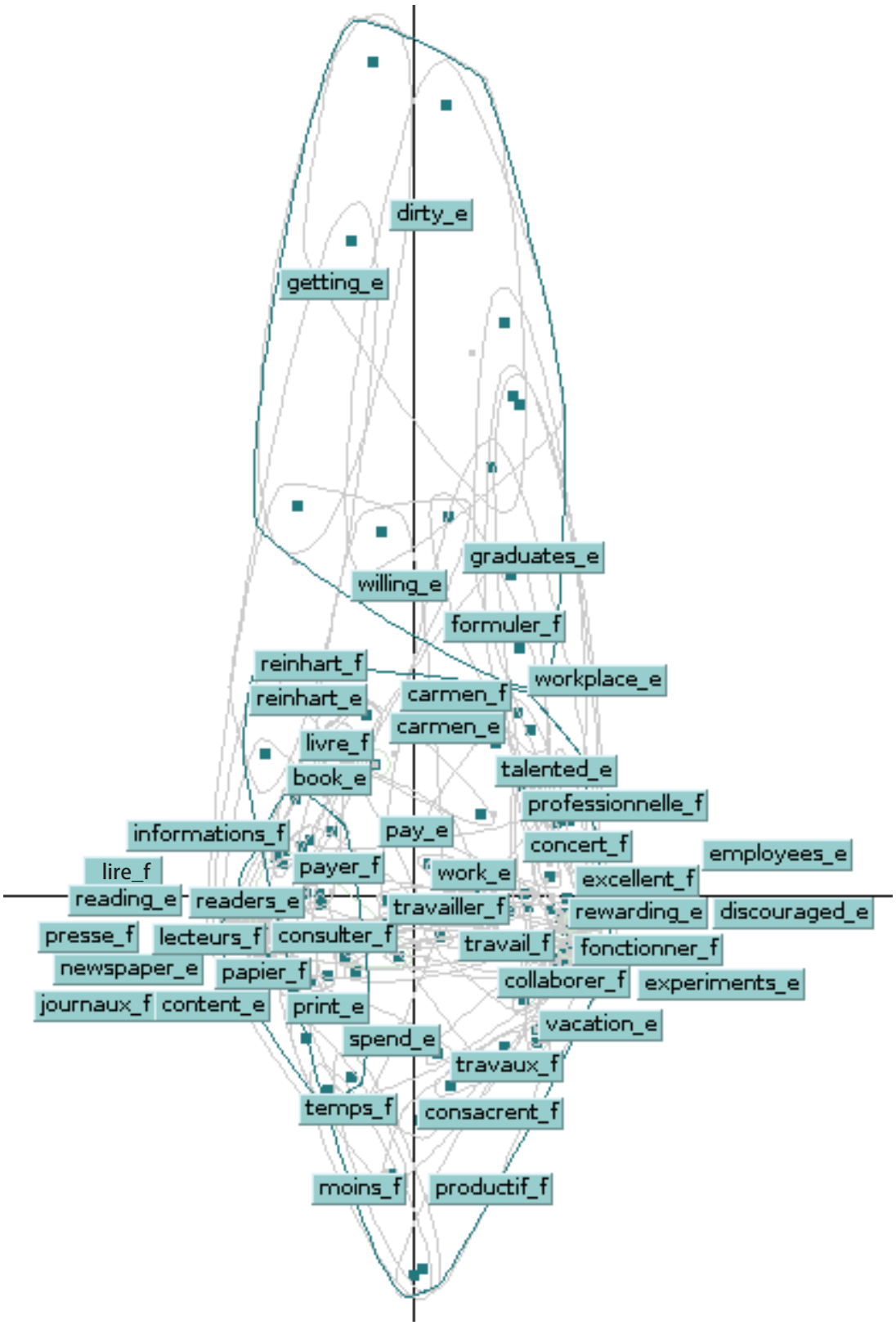

Figure 4–3 Spatial map of *work_e* with context *readers_e* as input.

we input *work_e* without context and with *readers_e* as context using BGSM, respectively. We can obtain the following observations from them:

(a) There are several good translation pairs, such as (*work_e,travailler_f*), (*readers_e*, *lecteurs_f*), (*reading_e,lire_f*) and (*book_e,livre_f*).

(b) For the *work_e* as input, it is placed at the center and the other words are mainly clustered into three semantic groups: *employment* (the right), *evaluation of job* (the upper left), and *literary works*[1] (the bottom). We can not determine which sense of *work_e* belongs to.

(c) For *work_e+readers_e* as input, we can determine sense of *work_e* by the words close to both of *work_e* and *readers_e*, such as *book_e*, *print_e* and *paper_f*.

4.3 Phrase Translation Probability Estimation

BGSM represents words as vectors dynamically on various geometric coordinates according to contextual words. For each source word in phrase-table, its contextual words are fixed, so all the translation candidate target words can be represented as vectors in the same geometric coordinates. This makes BGSM possible to be applied to phrase-based SMT for selecting translated phrase candidates.

4.3.1 Bilingual Phrase Semantic Representation

The phrase-table of phrase-based SMT model can be simply formalized as[2],

$$(F,\ E,\ scores,\ word\text{-}alignment), \tag{4–7}$$

where $F(w_{f_1}, w_{f_2}, ...w_{f_i}, ..., w_{f_k})$ and $E(w_{e_1}, w_{e_2}, ..., w_{e_j}, ..., w_{e_l})$ are source and its aligned target phrase, respectively, and $scores$ indicate various feature scores including direct translation probability, lexical weighting and phrase penalty. The phrase length is limited to seven, which is the default setting for phrase-based SMT.

BGSM is applied to represent words in phrase-table as six-dimension vectors. It should be noted that the clique extraction depends on contextual words, and CA then projects clique-word matrix onto corresponding semantic geometric coordinates accordingly. So the same contextual words should be used for all the words in F and its aligned P_{E_β}, in order to represent them on

[1] Part of borderline of this cluster is overlapped by some words.

[2] The alignment is from standard IBM alignment model [6].

the same geometric coordinates. For each word w_{fi} (or w_{ei}) in phrase pair (F, E), we consider two strategies for selecting the context words:

Strategy-A: only the source words in F are used as the contextual words $\{w_{f_1}, w_{f_2}, ..., w_{f_k}\}$.

Strategy-B: both of the source words in F and target words in all the aligned P_{E_β} are used as its contextual words $\{w_{f_1}, w_{f_2}, ..., w_{f_k}, w_{e_1}, w_{e_2}, ..., w_{e_l}\}$.

Word w_{fi} (or w_{ei}) is represented as vector V_{wf_i} (or V_{we_j}). It should be noted that all the source and target words for the same source phrase F are represented as vectors in the same geometric coordinates. Some words may not belong to any BCC (partially because the graph is pruned). These *unknown* words would be represented as default vectors[1].

Consequently, all the words in the (F, E) are represented as vectors.

4.3.2 Semantic Similarity Measurement

Because the lengths of phrases are different, Normalized Euclidean Distance (NED) is adopted to measure the distance between source and target phrases incorporated with word-alignment model:

$$NED(F,E) = \sqrt{\frac{\sum_{align(i,j)} ED^2(V_{wf_i}, V_{we_j})}{N_{align(i,j)}}}, \tag{4–8}$$

where $ED(V_{wf_i}, V_{we_j})$ stands for Euclidean Distance of word vector V_{wf_i} and V_{we_j}, $align(i,j)$ is from the word-alignment model in Eq. (4–7), and $N_{align(i,j)}$ is number of alignments.

The distance will be normalized, so that the sum of the Similarity $Sim(P_{E_\beta}|F)$ of the E_β from the same P_F equals to one (if there is only one translation candidate for F, the $Sim(E|F)$ will be simply set to one),

$$Sim(E_\beta|F) = \frac{\sum_E NED^2(F,E_\beta) - NED^2(F,E_\beta)}{(N_{E_\beta} - 1) \times \sum_E NED^2(F,E_\beta)}, \tag{4–9}$$

where N_E indicates how many E_β are aligned to F. Using the same pipeline, $Sim(F|E)$ can also be calculated. Both $Sim(E|F)$ and $Sim(F|E)$ can be added as additional features into phrase table for SMT decoding.

[1] Source and target default *unknown* vectors are empirically set at the space locations that are far from all of the other words (vectors).

4.4 Bilingual Phrase Generation

Some phrases are not in corpus, however, they may have similar meanings with the phrases in the corpus. These phrases may also be useful for translation. Take the source French phrase *la bonne réponse* as example, the corresponding aligned target English phrase is *the right answer* is in the corpus and phrase-table. The other phrases, such as *the correct answer* or *the right response*, may not be in the corpus or phrase-table, however, they are also good translation candidates for translation.

Since the BGSM can be used to represent words as vectors and measure their similarities by measuring their distance. Why do not we try to find the similar words to replace the original words in the phrase-table, to generate a new phrase with similar meaning as the original phrase?

4.4.1 Phrase Pair Generation

In Section 4.3, we focus on measuring the distance between source and target phrases in phrase-table. In this section, we focus on measuring the distance between source and target phrases outside phrase-table and select new phrase pair to enhance SMT.

As mentioned in Section 4.3, for each word w (source or target), both of the source words in F and target words are used as its contextual words $\{w_{f_1}, w_{f_2}, ..., w_{f_k}, w_{e_1}, w_{e_2}, ..., w_{e_l}\}$ (Strategy-B). Word w and its co-occurrence words are represented as vectors. For a aligned word pair (w_{fi}, w_{ej}), they are represented as vectors (V_{fi}, V_{ej}) and their co-occurrence words $\{w_{co}\}$ are represented as vectors $\{V_{co}\}$. We need to find new translation candidate w'_{ej} in $\{w_{co}\}$ to form new phrase pair (w_{fi}, w'_{ej}). The Euclidean Distance,

$$ED(V_{wf_i}, V'_{we_j}), \tag{4–10}$$

is used to measure the distances between w_{fi} and all of the target words w'_{ej} in w_{co}. The w'_{ej} close with w_{fi} in distance is selected as the new translated candidate word. The pipeline is similar with the one in Section 3.3.2. The w_{ej} in original phrase $E(w_{e_1}, w_{e_2}, ..., w_{e_j}, ..., w_{e_l})$ is replaced with w'_{ej} and the generated target candidate phrase $E'(w_{e_1}, w_{e_2}, ..., w_{e'_j}, ..., w_{e_l})$ for source phrase F is formed consequently. For a target phrase $E'(w_{e_1}, w_{e_2}, ..., w_{e_j}, ..., w_{e_l})$, only one word w_{e_j} is replaced with w'_{e_j} and other words maintain as they are[1]. Using the same pipeline, the generated source candidate phrase $F'(w_{f_1}, w_{f_2}, ..., w_{f'_i}, ..., w_{f_k})$ for target phrase E can also be formed[2].

[1] We are aware that two or more words can be replaced, however it may lead to serious sense bias, and the experiments also show replacing more than two words does not perform well (not shown for limited space).

[2] We generate both of source and target phrases, because some phrases in test data may also not be in phrase-table but have

$Sim(E'|F)$ and $Sim(F'|E)$ can be calculated using Eqs. (4–8) and (4–9). Because there are no original phrase translation probabilities $\psi(E'|F)$ or $\psi(F'|E)$ for the generated (F, E') and (E, F') in the original phrase-table. so $Sim(E'|F)$ and $Sim(F'|E)$ are used for the generated (F, E') and (E, F') as $\psi(E'|F)$ or $\psi(F'|E)$. The updated lexical weighting $lex(E'|F)$ and inverse lexical weighting $lex(F'|E)$ are computed by IBM model [6].

The generated phrases are filled-up [102] into original phrase table. That is, a penalty score is added as additional feature, for original phrase pairs, the penalty is set as one; for the generated ones the penalty is set as natural logarithm base e (= 2.71828...). All of scores weights in phrase table will be further tuned using MERT [139].

4.4.2 Prase-table Size Tuning

Using the phrase generation approach, a lot of new phrase pairs can be generated. We need to select the most reasonable ones inside them. The *Distance Ratio* (DR) of normalized distance in Eq. (4–9) between generated phrase pair (F, E') and original phrase pair (F, E),

$$DR(E', E) = \frac{NED(F, E')}{NED(F, E)}, \tag{4–11}$$

is used to measure the usefulness of generated word pairs.

A threshold ε is set up to keep the closest generated phrase pair only. Namely, for a source phrase F, only the E' whose $DR(E', E)$ in Eq. (4–11) smaller than ε are selected as the generated word pair (F, E'). Using the same pipeline, the size of generated source candidate phrases is also tuned. For SMT task, the threshold is tuned by using the SMT performance on development data.

The proposed BGSM word embedding can be applied to phrase pair translation probability estimation and generation. The experimental results show that the proposed model can effectively enhance phrase-based SMT decoding and achieve a significant improvement with high computational efficiency. It also outperform the existing related word embedding methods for SMT.

similar meaning with them.

Chapter 5

Experiments and Results

Although some of the experiments share the same corpus, the detail setting are quite different. So we introduce the data setting up independently at each section.

Since the CSLM conversion method is partially designed for large LM conversion, which is related with LM pruning or adaptation, we combine the LM adaptation parts with the CSLM conversion part in this section.

5.1 CSLM Conversion

5.1.1 Experiment Setting up

The main data set in this section is from Chinese to English NTCIR-9 patent translation task [140]. It consists of one Million (M), two thousand and two thousand sentences for parallel training, development, and test data , respectively.

The same settings of the NTCIR-9 phrase-based translation (Chinese to English) baseline system [140] are followed besides the part of LMs for the purpose of comparison. Moses phrase-based SMT system [7], GIZA++ [141] for alignment and MERT [139] for tuning are adopted. We use the case-insensitive BLEU scores to measure translation performance. `mteval-v13a.pl` is used for calculating BLEU scores[1].

5.1.1.1 Corpus

Three corpora with different sizes are used:

1) Corpus42M: 1M English sentences with 42M words in NCTIR-9 training data;

2) Corpus746M: The English texts from the NTCIR-8 patent translation task [142], which consists of 746 M words;

3) Corpus5B: English texts extracted from US patent data (from 1993 to 2005) with 5B words.

It should be noted that the Corpus42M is added to Corpus746M and Corpus5B.

[1] It is available at `http://www.itl.nist.gov/iad/mig/tests/mt/2009/`

5.1.1.2 BNLM

All the BNLMs are trained using SRILM [59, 91] and the same vocabulary of Corpus42M is used. A 5-gram BNLM with interpolated KN smoothing using Corpus42M is trained as the baseline BNLM, , without any discarding or count cutoffs. This BNLM is called *KN42M*.

A larger 5-gram BNLM with interpolated Kneser-Ney smoothing is also trained by using Corpus746M, by discarding 3,4,5-grams which occur only once. This BNLM is called *KN746M(cutoff)*.

The 5-gram LMs are trained with interpolated KN and Good-Turing (GT) smoothing methods[1], on the Corpus5B without cutoff. These LMs are called *KN5B* and *GT5B*, respectively. The KN/GT5B will be pruned into smaller ones in different sizes with different pruning methods.

5.1.1.3 CSLM

We train a 5-gram CSLM on the same Corpus42M using the CSLM toolkit [65]. The detail settings are as follows: 256 dimensions for each word in projection layer, 384 dimensions for hidden layer and 8192 dimensions for output layer (short-list). These settings are recommended by CSLM toolkit. This CSLM is called CSLM42M, in which KN42M is used as the background BNLM.

Arsoy *et. al.* [88, 89] use around 55 M words as the corpus, including 84K words as vocabulary, and 20K words as short-list. They only construct a 4-gram LM. In this thesis, we use around 42 M words as corpus, including 456K words as vocabulary, and 8K words, which covers 92.89% of words in the training corpus, as short-list for both NNGC and the Arsoy method. The size of short-list is selected according to the corpus by experiments before we construct the CSLM. That is, we choose 1K (default setting by the CSLM toolkit), 4K, 8K, 16K and 24K as the short-list, and then we test the coverage probability that the words in training corpus are in the short-list. The results are around 93% (this is because the frequency of words is different, and the rare words seldom appear although they are in the vocabulary) of the words will be in the short-list for 8K as short-list, and 94% or 95% if we use 16K or 24K. Please note this 1% or 2% improvement is not PPL or BLEU, it is just the probability of whether use CSLM or BNLM to calculate the probabilities of n-grams. That is, 93% of the n-grams will be calculated using the CSLM and 7% using BNLM. The time computational complexity for the CSLM is nearly linear with the size of vocabulary. If we use a 20K short-list, it will take 2.5 times time for training and converting CSLM. Therefore we choose 8K as the short-list for both methods,

[1] We also apply BLMP for Witten-Bell smoothing, and the results are similar to Good-Turing.

because more than 8K V_0 doest not affect the coverage of words in short-list so much, but it will take much more time.

We adopt the similar size setting for corpus and short-list as [88, 89]. Our vocabulary is much larger than theirs, because the whole vocabulary must be used for decoding in SMT, compared with only a small vocabulary used in speech recognition. The ratio of $|V_0|/|V|$ affects how much time cost will reduced for the output layer. In Arsoy's experiments, $|V|$ is small (84K), and nearly 75% time is saved by using 20K as short-list. For NNGC, $|V|$ is 456K and short-list was 8K. Therefore, nearly 98% time cost is reduced.

It should be noted that every setting of the CSLM for all of the methods is the same for fair comparison in this thesis.

5.1.1.4 SMT Models

Fourteen standard SMT feature scores are used, i.e, one LM score, seven distortion scores, four translation model scores, one word penalty score, and one phrase pair number penalty score.

All the above models are trained on Corpus42M, except LMs. The weights of the 14 standard SMT features are tuned using MERT independently.

All of the experiments are conducted using the same computer with 2.70GHz CPUs.

5.1.2 CSLM Conversion

5.1.2.1 Setup for proposed NNGC method

The CSLM can only predict the probability of the same order i-grams as they are. That is, the bigram CSLM can only calculate the probabilities of bigrams. So different order CSLMs must be constructed if we want to convert a *5*-gram LM.

The 2,3,4,5-CSLMs are trained with the same setting for the CSLM described in Section 4.1.3 on Corpus42M using the CSLM toolkit [65], with KN42M as the background BNLM. These CSLMs are called 2,3,4,5-CSLM42Ms, respectively.

By using the method described in Section 5.1.2, we re-write KN42M with CSLM42Ms. This re-written BNLM is then interpolated with KN42M, whose interpolation weight is determined by grid search. Namely, the interpolation weight is empirically set as 0.9, 0.7, 0.5, 0.3, 0.1 to create an interpolated LM. Then the feature weight parameters of this interpolated LM are tuned the on the first half development data (1000 sentences). Subsequently, the interpolation weight, which obtains the highest BLEU score on the second half development data, is chosen. Once the

interpolation weight is determined, MERT is applied to the whole 2,000 sentence development data again to tune the weight parameters[1]. This converted BNLM is called CONV-KN42M.

5.1.2.2 Setup for Compared Methods

The same CSLM42Ms are used for Arsoy method. All the words in short-list are added to the tail word of i-grams to build the (i+1)-grams. Note that the probabilities of all the target/tail words in short-list can be calculated by neurons at the same time. Then the probabilities of (i+1)-grams are calculated using the (i+1)-CSLM. So a large converted (i+1)-gram LM will be grown, and then it needs to be pruned into the same size as the original (i+1)-grams using entropy-based LM pruning. Using the similar pipeline, (i+2)-grams are grown by using (i+1)-grams recursively. At last, the CONV is interpolated with the original BNLM[2] using the same methods in Section 5.1.2.1.

5.1.2.3 Comparison between Converting Artificial N-grams and Natural N-grams

The KN42M indicates the BNLM trained on Corpus42M using the interpolated Kneser-Ney smoothing. The Arsoy42M indicates the converted CSLM42M using Arsoy method, which uses artificial n-grams. The converted CSLM and the original BNLM are interpolated. Since the n-grams in the converted CSLM and BNLM are quite different in [88, 89], the interpolated LM is larger than both of them. Consequently, the interpolated LM is pruned into the same size as the original BNLM. The CONV-KN42M is the converted CSLM42M using NNGC, which uses natural n-grams from KN42M, and CONV-KN42M (interpolated) is the CONV-KN42M interpolated with KN42M. All the LMs above are trained on Corpus42M.

The comparison of Arsoy method and ours are shown in Table 5–1. Paired bootstrap re-sampling test [143][3] is performed, with 2000 samples for each significance test. The results of statistical significance test is shown in Table 5–2.

Tables 5–1 and 5–2 indicate:

1) Both Arsoy method and NNGC improve the PPL compared with the original BNLM. This indicates that using the CSLM for converting can improve the probabilities estimation.

[1]The interpolation weight can also be determined by minimizing the PPL on development data. In practice, this method does not work well according to our experiments. So we optimize the weights by directly maximize the BLEU score.

[2]Since this interpolated LM is larger than the original one, it is pruned into the same size as the original BNLM according to [89].

[3]We used the code available at `http://www.ark.cs.cmu.edu/MT`

LMs	Size	n-grams	PPL	BLEU	Ratio in Phrase Table
KN42M	3.0G	73.9M	108.8	32.35	100%
Arsoy42M (pruned)	3.1G	71.8M	104.7	32.38	45.9%
Arsoy42M (interpolated)	5.4G	140.1M	103.5	32.91	52.7%
CONV-KN42M	3.0G	73.9M	106.3	32.88	100%
CONV-KN42M (interpolated)	3.0G	73.9M	106.1	**33.07**	100%

Table 5–1 Comparison experiments between converting artificial n-grams and natural n-grams. The 'Ratio in Phrase Table' indicates how much percentage of the n-grams are in phrase table.

	Arsoy42M (interpolated)	CONV-KN42M	Arsoy42M (pruned)	KN42M
CONV-KN42M (interpolated)	$>$	$>$	$\gg$	$\gg$
Arsoy42M (interpolated)		$-$	$\gg$	$\gg$
CONV-KN42M			$\gg$	$\gg$
Arsoy42M (pruned)				$-$

Table 5–2 Significance tests for comparison between converting artificial n-grams and natural N-grams. The marks ($\gg$, $>$ and $-$) indicate whether or not the LM to the left of a mark is better than the one above the mark at certain levels. "$\gg$" indicates significance level at $\alpha = 0.01$, "$>$" at $\alpha = 0.05$, and "$-$" indicates that it is not significantly better at $\alpha = 0.05$.

2) The BLEU score of CONV-KN42M is better than Arsoy42M (pruned) by around 0.5, and CONV-KN42M (interpolated) is slightly better than Arsoy42M (interpolated) by 0.16. For the ratio of n-grams in phrase table, Arsoy method can generate more artificial n-grams, however some n-grams, which are important for SMT (such as the n-grams in the phrase table), are pruned during their converting. For NNGC, all the n-grams in the CONV are inside the phrase table.

5.1.2.4 Comparison for Converting Efficiency

In this subsection, we show the converting efficiency of Arsoy method and NNGC. 1M trigrams are used as the input for the CSLM42M for both methods. In converting, Arsoy method will generate much more n-grams and then a large part of these n-grams should be pruned[1]. The converting time is shown in Table 5–3.

LMs	Converting Time
Arsoy' method	24 hours 24 minutes
NNGC CONV	41 minutes

Table 5–3 Converting efficiency.

Table 5–3 shows that Arosy's converting method took much more time (nearly 40 times) than ours. Arsoy method generates all the possible n-grams ended with the words in the short-list (usually thousands times of the original one) and then prune them into smaller ones. Although the probabilities of all the tail words in short-list can be calculated at the same time using neurons in the output layer in the CSLM, Arsoy method still takes much more time than ours. For our NNGC method, the same n-grams with the n-grams in the BNLM are converted.

5.1.2.5 Comparison for Decoding Efficiency

Vaswani *et. al.* [79] applies several techniques to speeding up NPLM in SMT, meanwhile Arsoy method and NNGC method store the probabilities of the CSLM into the BNLM format. In this subsection, we compare the decoding efficiency of BNLM, Arsoy, CONV, CSLM and NPLM. The numbers of hidden layers are set the same as the CSLM and other settings followed the

[1]The pruning actually takes a lot of time, but we do not take it into account for fair comparison of converting time. In contrast, NNGC will only produce the same sized n-grams.

default setting of NPLM toolkit [79], with all the same setting in SMT decoding. The 2,000 sentences of the NTCIR-9 test data are used as the evaluation data. The decoding time and SMT performance are shown in Table 5–4.

LMs	Decoding Time (sec.)	BLEU
KN42M	15.3	32.25
Arsoy42M (pruned)	15.3	32.38
Arsoy42M (interpolated)	15.9	32.91
CONV-KN42M (our)	**15.2**	33.07
CSLM42M	186.5	**33.16**
NPLM42M	50.4	32.78
NPLM42M (re-normalized)	456.8	32.85

Table 5–4 Decoding efficiency and performance.

From the results of Table 5–4, we reach the following conclusions:

1) The decoding time using CONV-KN42M and BNLM (KN42M) are nearly all the same. This indicates that NNGC method can run as fast as the BNLM does.

2) The decoding time using CSLM is much slower than BNLM, and decoding time using NPLM (without normalization) is much faster than CSLM and much slower than BNLM. This shows the advantage of NNGC in decoding efficiency.

3) In comparison with the LMs with similar decoding time, the BLEU of CONV is slightly better than Arsoy method and much better than KN42M.

4) In comparison with NNLMs directly uses in decoding, which are much slower than BNLM, the BLEU of CONV is slightly better than NPLM and slightly worse than CSLM.

5.1.3 LM Pruning

In this subsection, we compare the performance of pruned LMs using different methods. The experiments on LM pruning are divided into four groups: the original, cutoff pruning, entropy-based, and BLMP methods. The experiment results are listed in Tables 5–5 and 5–6.

For the cutoff LM pruning, the default setting 1-1-2-2-2 in SRILM is applied, which means that we does not cutoff unigrams and bigrams, but cutoff n-gram whose frequency is less than two for the higher order n-grams. They are represented by KN/GT5B-Cutoff in Tables 5–5 and 5–6. KN/GT5B-Connect-* (where * stands for any ID) are LMs tuned into different sizes

LMs	Size	n-grams	PPL	BLEU
KN5B	163.1G	3945.3M	73.4	34.32
KN5B-Cutoff	36.1G	956.7M	74.8	33.89
KN5B-Entropy-1	80.0G	2029.8M	74.0	34.04
KN5B-Entropy-2	41.3G	1099.5M	77.9	33.67
KN5B-Entropy-3	21.8G	607.9M	82.8	33.41
KN5B-Entropy-4	8.2G	201.9M	93.8	33.16
KN5B-Connect-1	84.0G	2195.9M	79.8	34.58^{++}
KN5B-Connect-2	43.8G	1192.9M	82.0	33.92^{+}
KN5B-Connect-3	19.3G	555.7M	86.9	33.79^{++}
KN5B-Connect-4	8.4G	208.0M	94.2	33.38^{++}
KN5B-Inter-1	119.9G	2995.5M	**71.8**	$\mathbf{34.66}^{++}$
KN5B-Inter-2	71.0G	1871.4M	**72.4**	$\mathbf{34.44}^{++}$
KN5B-Inter-3	35.5G	985.8M	**75.2**	$\mathbf{34.28}^{++}$
KN5B-Inter-4	15.9G	459.0M	**81.0**	$\mathbf{33.93}^{++}$

Table 5–5 Comparison of different pruning methods on the 5B corpus for KN smoothing. The marks ($++$ and $+$) indicate whether or not the LM to the left of a mark is better than the one above the mark at certain levels. "$++$" indicates significance level at $\alpha = 0.01$, "$+$" at $\alpha = 0.05$.

using BLMP method. The sizes of the entropy-based pruned LMs, KN/GT5B-Entropy-*, are tuned to be similar to those of BLMP KN/GT5B-Connect-* pruned LMs by setting thresholds appropriately.

The corresponding (in similar size) entropy-based pruned LMs and BLMP pruned LMs, such as KN5B-Connect-1 and KN5B-Entropy-1, are interpolated with the parameter 0.5 into KN5B-Inter-1 using SRILM. The IDs after the pruned LMs (1/2/3/4) indicates the different sizes (in decreasing order) of LMs for the same pruning method.

From Tables 5–5 and 5–6, we got the following observations:

1) KN5B-Connect-1 and GT5B-Connect-1 are the pruned LMs with all the connecting phrases. That is, all the connecting phrases and phrases in the phrase table are kept and all the other phrases were pruned. The BLEUs of these two pruned LM are similar as the original ones (KN/GT5B). This indicates that BLMP kept all the useful information and discarded all the useless information from the original huge LM.

LMs	Size	n-grams	PPL	BLEU
GT5B	162.8G	3945.3M	83.5	34.58
GT5B-Cutoff	37.8G	956.7M	82.3	34.07
GT5B-Entropy-1	77.7G	2055.5M	83.3	34.41
GT5B-Entropy-2	43.8G	1206.3M	82.7	34.12
GT5B-Entropy-3	20.3G	576.3M	81.8	34.41
GT5B-Entropy-4	8.5G	229.7M	82.8	34.07
GT5B-Connect-1	85.2G	2195.9M	84.2	**34.42**
GT5B-Connect-2	43.6G	1192.9M	84.5	34.14
GT5B-Connect-3	19.2G	555.7M	87.5	33.58^{--}
GT5B-Connect-4	8.3G	208.0M	92.1	33.97
GT5B-Inter-1	115.1G	2935.1M	**80.5**	34.33
GT5B-Inter-2	70.3G	1894.3M	**78.4**	$\mathbf{34.50}^{++}$
GT5B-Inter-3	33.5G	947.4M	**77.3**	$\mathbf{34.67}^{+}$
GT5B-Inter-4	14.3G	409.9M	**78.3**	$\mathbf{34.42}^{++}$

Table 5–6 Comparison of different pruning methods on 5B corpus for GT smoothing.

2) For BLMP method, as LMs became smaller, the PPL increases and BLEU decreases accordingly for nearly all the cases in both KN and GT smoothing. Since PPL and BLEU are positively correlated, PPL can be used to predict the performance of pruned LM in SMT. This also indicates that we could use PPL to setup a threshold for tuning the size of pruned LMs.

3) For the entropy-based pruning method, as LMs became smaller, PPL does not always increase and BLEU does not always decrease accordingly, and the PPL and BLEU of some pruned LMs even decreases/increases. It is interesting that some small pruned LMs performed better than larger LMs in GT smoothing.

4) For KN smoothing, the entropy-based pruned LMs obtains better PPL on the LMs with similar sizes, but the BLMP pruned LMs always obtains significantly better BLEU[1]. These show that BLMP performed significantly better in SMT for KN smoothing, although with worse PPL.

5) For GT smoothing, the entropy-based method also obtains better PPL on the LMs with

[1]Some researchers find that KN smoothing performs bad in the entropy-based LM pruning [114, 144, 145]. The PPL of entropy-based pruned LMs for KN smoothing increases aggressively as their sizes decrease, compared with other smoothing methods.

similar sizes. BLMP method obtains similar BLEU on large LMs and the entropy-based method obtains better BLEU on small LMs.

6) Some interpolated LMs obtained comparable BLEU, with comparable 1/4 of the size, in comparison with the unpruned LMs. The interpolated LMs obtain better PPL and BLEU in nearly all cases, compared to either the entropy-based or BLMP pruned LMs of the same IDs.

7) By comparing n-grams in the entropy-based pruned LMs and BLMP pruned LMs, such as KN5B-Entropy-* and KN5B-Connect-*, the percentage of the overlapping n-grams between the entropy-based pruned LMs and BLMP pruned LMs in similar sizes are 60.6% / 38.2% / 29.2% / 10.6%, corresponding to KN5B-Entropy-/KN5B-Connect-1/2/3/4. The overlapping situation is similar for GT smoothing. This indicates that these two pruning methods are quite different, and the pruned LMs have increasingly different n-grams as the pruning keeps going on.

5.1.3.1 Discussions

The above results show that BLMP performed well in SMT. In this subsection, which kind of n-grams are useful during decoding was investigated. We calculate the log-probabilities of every reference sentence of test data to simulate the performance of 5-gram LMs in SMT decoding, and count the ratio of different order n-grams which are used for each pruned LM. Then the Average Length of the N-grams Hit (ALNH) in SMT decoding for different LMs is calculated as follows,

$$ALNH = \sum_{i=1}^{5} P_{\text{i-gram}} \times i, \tag{5–1}$$

where $P_{\text{i-gram}}$ means the ratio of the i-grams hit in SMT decoding. The results are illustrated in Figure 5–1.

As shown in Figure 5–1, ALNH in SMT decoding of BLMP pruned LMs is longer than that of the entropy-based pruned LMs for KN smoothing, and ALNH of BLMP pruned LMs and the entropy-based pruned LMs are similar for GT smoothing. ALNH of interpolated LMs is longer than both of the two methods for KN and GT smoothing.

As we know, LM refers to the probabilities of (i-1)-grams together with the adjusted weights using back-off, if no corresponding i-grams are hit. The statistics above indicated that more high-order n-grams are hit for BLMP pruned LMs in SMT compared with the entropy-based pruned LMs. In other words, less back-off were applied with the BLMP pruned LMs in SMT

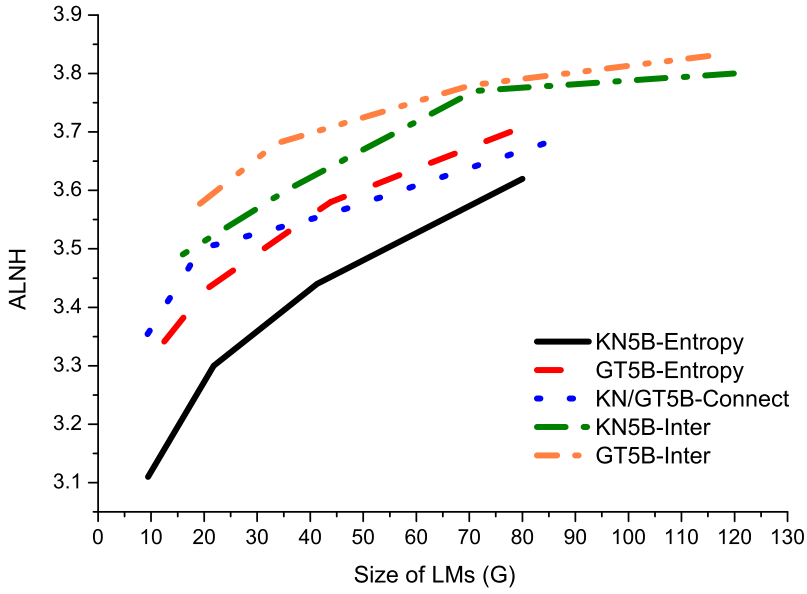

Figure 5–1 ALNH of different pruning methods in SMT decoding.

Because the n-grams in BLMP pruned LMs for KN and GT (KN/GT5B-Connect) are the same and only possess different probabilities and back-off weights, so their ALNH are also the same.

decoding[1].

There is also a positive correlation between ALNH in Figure 5–1 and BLEU in Tables 5–5 and 5–6. Most of the LMs with larger ALNH also obtain higher BLEU in SMT. This indicates that useful n-grams in SMT were not only the n-grams used in the decoding, but also those useful n-grams that are *long*. Although our current method only focus on probability, the results suggest that we should also focus on lengths of the n-grams for pruning in the future work.

5.1.4 Converting Large LM

As mentioned in Section 4.1, large additional corpus may let the LMs better. In this subsection, experiments on converting large LM are conducted on both NTCIR and NIST data sets.

5.1.4.1 Effects of Corpus Size

In Section 5.1.2, the results show that the proposed NNGC CONV outperforms the original BNLM and Arsoy method on small corpus containing 42M words. In this subsection, we show

[1]For the 5-gram LM, if 1/2/3/4-grams are hit in decoding, it indicated that back-off are applied.

the results of NNGC method on a large corpus containing around 1B words.

Both the Corpus42M and the Corpus746M are used for this experiments. The KN42M and CONV-KN42M are the same as in Table 5–1. We obtain the CONV-KN746M by re-writing the KN746M (cutoff) with CSLM42M in the same way as CONV-KN42M.

Table 5–7 shows BLEU scores on the test data. The '1st pass' column shows the BLEU scores in the first pass decoding. The 'reranking' column shows BLEU scores when CSLM42M is used to rerank the 100-best lists[1]. CSLM42M is added as the additional 15th feature, as we apply it to reranking. Z-MERT [146] is used to tune the weight parameters.

LMs	Size	n-grams	PPL	1st pass	rerank	ALNH	weights
KN42M	3.0G	73.9M	108.8	32.35	32.96	3.03	0.055
CONV-KN42M	3.0G	73.9M	106.1	33.07	33.47	3.03	0.060
KN746M (cutoff)	7.5G	183.7M	95.1	33.27	33.78	3.43	0.073
CONV-KN746M	7.5G	183.7M	**92.9**	**33.59**	**33.91**	3.43	0.081

Table 5–7 Comparison of BLEU scores for LMs with different sizes

Statistical significance test is listed in Table 5–8.

From the results of Tables 5–7 and 5–8, we can observe:

1) CSLM Reranking increases the BLEU scores for all the first pass LMs. This observation is consistent with those of previous work [65, 77].

2) For both first-pass and reranking, CONV42M is better than the KN42M and CONV746M is also better KN746M. This indicates that NNGC method improves BNLMs, even if BNLM is trained on a much larger corpus (containing around 1B words here and 5B words in next subsection) than CSLM.

3) The BLEU in the first-pass of CONV42M (33.07) and CONV746M (33.59) are comparable with the reranking results of KN42M (32.96) and KN746M (33.78), respectively. No significance differences are shown between these results.

4) The last column in Tables 5–7 indicates that the tuned SMT feature weights for different LMs. There is also a positive correlation between LMs and weights. The LMs with better PPL and BLEU had larger weights.

The results above indicated that NNGC can scale up well on large corpora.

[1]In practice, we also conducted experiments using 1000-best lists for reranking(not shown), and the results are similar with 100-best lists reranking.

	KN746M (rerank)	CONV-KN746M (1st)	CONV-KN42M (rerank)	KN746M (cutoff) (1st)	CONV-KN42M (1st)	KN42M (rerank)	KN42 (1st)
CONV-KN746M (rerank)	$-$	$\gg$	$\gg$	$\gg$	$\gg$	$\gg$	$\gg$
KN746M (rerank)		$-$	$>$	$\gg$	$\gg$	$\gg$	$\gg$
CONV-KN746M (1st)			$-$	$>$	$\gg$	$\gg$	$\gg$
CONV-KN42M (rerank)				$\gg$	$\gg$	$\gg$	$\gg$
KN746M (cutoff) (1st)					$-$	$\gg$	$\gg$
CONV-KN42M (1st)						$-$	$\gg$
KN42M(rerank)							$\gg$

Table 5–8 Significance tests for LMs with different sizes.

The '1st' is the short for the '1st pass'.

5.1.5 Combination Performance

5.1.5.1 Conversion of Pruned LMs

Although NNGC using natural n-grams works well on large corpora, it is time consuming to convert a BNLM trained from an even larger corpus (such as containing around 5B words). Since BLMP method can prune the useless n-grams from large LM, without sacrificing their SMT performance significantly, instead of converting the huge BNLM, we can first make the pruning before converting it.

The smallest pruned LMs (KN/GT5B-Entropy-4 and KN/GT5B-Connect-4) that are nearly 1/20 size of the original LM (KN/GT5B), and KN/GT5B-Connect-3 that are nearly 1/10 size of the original LM (KN/GT5B) in Tables 5–5 and 5–6, are selected for conducting converting experiments. We take them as the input BNLM instead of the original one. The same setting for natural n-grams converting experiments are set here with the pruned LMs as the input. We also conduct significance tests to show whether the BLEU of CONV is significantly better than the corresponding BNLM, and the results are shown in Table 5–9.

From the results of Table 5–9, we can observe:

LMs	Size	n-grams	PPL	BLEU
KN5B	163.1G	3945.3M	73.4	34.32
GT5B	162.8G	3945.3M	83.5	**34.58**
KN5B-Connect-3	19.3G	555.7M	86.9	33.79
CONV-KN5B-Connect-3	19.3G	555.7M	83.6	34.24^{++}
GT5B-Connect-3	19.2G	555.7M	87.5	33.58
CONV-GT5B-Connect-3	19.2G	555.7M	82.8	$\mathbf{34.30}^{++}$
KN5B-Entropy-4	8.2G	201.9M	93.8	33.16
CONV-KN5B-Entropy-4	8.2G	201.9M	89.5	33.57^{++}
GT5B-Entropy-4	8.5G	229.7M	82.8	34.07
CONV-GT5B-Entropy-4	8.5G	229.7M	79.8	34.19^{++}
KN5B-Connect-4	8.4G	208.0M	94.2	33.38
CONV-KN5B-Connect-4	8.4G	208.0M	90.1	33.83^{++}
GT5B-Connect-4	8.3G	208.0M	92.1	33.97
CONV-GT5B-Connect-4	8.3G	208.0M	88.5	$\mathbf{34.32}^{++}$
CONV-KN746M	7.5G	183.7M	85.9	33.59
CONV-KN42M	3.0G	73.9M	106.1	33.07

Table 5–9 Converting pruned BNLM using CSLM.

The LMs in this table were the same as the LMs with the same names in Table 5–5 and 5–6. The BLEU in bold indicate it is the highest one among the LMs with similar size.

1) The converted pruned CONVs outperform the pruned BNLM significantly. This indicates that NNGC works well together with BLMP to convert a large LM efficiently.

2) As the size of CONV increased, both PPL and BLEU are further improved. These results strengthen that the performance of converted LM becomes better, as the sizes of input BNLM increased.

3) The BLEU of CONV-GT* are significantly better than the corresponding GT*. This indicates that NNGC also worked well together with GT smoothing.

In addition, the proposed BLMP can also be applied to Arsoy method. That is, all the words in short-list are added to the tails of i-grams in the corpus in order to produce (i+1)-grams called *artificial n-grams*. BLMP is applied to these artificial *n*-grams to select connecting phrases. These connecting phrases (*n*-grams) are used as input of CSLM and their probabilities are obtained as output. These generated *n*-grams are interpolated with the original BNLM to form the larger converted LM consequently.

Table 5–10 shows that we use BLMP to prune both Arsoy's converted LMs nd NNGC converted LMs.

1) Arsoy's converting and the proposed BLMP can work well together (Arsoy-Connect-3/4) to enhance LM and SMT performance in comparison with KN42M.

2) The NNGC with BLMP (CONV-KN5B-Connect-3/4) outperforms Arsoy's converting with BLMP (Arsoy-Connect-3/4) significantly. This result indicates natural n-grams are more helpful in comparison with artificial ones.

5.1.5.2 Experiments on NIST Data Set

We also evaluate the performance of our proposed methods on NIST Chinese to English data set. OpenMT06 in the NIST open machine translation 2006 Evaluation[1] is used. The parallel corpus follows the setting of [147], and it mainly consists of news and blog texts. The training set consists of 430K sentences and 12M words. The English corpus from United Nation data set[2] is used as the large monolingual corpus[3], which includes around out-of-domain 7M sentences and 206M words as the parallel corpus. The data sets of NIST Eval 2002 to 2005 are used as development data for MERT tuning. The NIST Eval 2006 are used as evaluation data. The same settings of Section 5.1.1 for SMT are followed.

[1] http://www.itl.nist.gov/iad/mig/tests/mt/2006/

[2] LDC2013T06 Info Local CD/DVDL310 "1993-2007 United Nations Parallel Text" (3 DVDs)

[3] The English side of small bilingual corpus is added to the large monolingual corpus.

LMs	Size	n-grams	PPL	BLEU
KN5B-Connect-3	19.3G	555.7M	86.9	33.79
Arsoy-Connect-3	19.6G	578.5M	100.3	32.98
CONV-KN5B-Connect-3	19.3G	555.7M	**83.6**	**34.24**
KN5B-Connect-4	8.4G	208.0M	94.2	33.38
Arsoy-Connect-4	8.0G	198.6M	101.2	33.05
CONV-KN5B-Connect-4	8.4G	208.0M	**90.1**	**33.83**
KN42M	3.0G	73.9M	108.8	32.35
Arsoy42M (pruned)	3.1G	71.8M	104.7	32.38
Arsoy42M (interpolated)	5.4G	140.1M	**103.5**	32.91
CONV-KN42M	3.0G	73.9M	106.1	**33.07**

Table 5–10 BLMP with Arsoy method.

We compare the performances of small/large BNLM (NIST-KN12/206M), Arsoy method (NIST-Arsoy12M), the proposed NNGC method (CONV-NIST-KN12M), the BLMP method (NIST-KN206M-connect-1/2), and their converted LMs (CONV-NIST-KN206M-connect-1/2).

LMs	Size	n-grams	PPL	BLEU
NIST-KN12M	1.1G	23.9M	145.8	31.88
NIST-Arsoy12M	1.1G	23.9M	142.2	32.23
CONV-NIST-KN12M	1.1G	23.9M	**141.0**	32.45
NIST-KN206M-connect-1	2.7G	67.4M	176.8	31.96
CONV-NIST-KN206M-connect-1	2.7G	67.4M	174.2	**32.65**
NIST-KN206M-connect-2	4.6G	119.4M	179.3	31.77
CONV-NIST-KN206M-connect-2	4.6G	119.4M	177.8	32.37
NIST-KN206M	12.8G	260.0M	177.9	31.65

Table 5–11 Experiments on NIST data set.

From the results of Table 5–11, we have the following observations:

1) Because the large extra monolingual corpus and parallel corpus are in different domains, the BLEU of BNLM constructed from large corpus (NIST-KN206M) is slightly worse than NIST-KN12M. The pruned LM (NIST-KN206M-connect-1) is slightly better than NIST-

KN206M and NIST-KN12M. This indicates that the bilingual LM pruning method can also be applied to LM adaptation.

2) The proposed NNGC method works well and performed slightly better than Arsoy method on the NIST data set. NNGC and BLMP also work well together (CONV-NIST-KN206M-connect-1/2) on the NIST data set.

5.1.6 Summary

We have proposed a NNGC CSLM converting method for better probability estimation, and a BLMP pruning approach for enhancing LMs for SMT decoding and improving the efficiency of CSLM converting. Our methods have the following two attractive features:

1) NNGC and BLMP can select the n-grams with linguistic sense which are potentially to be used in decoding.

2) The proposed pruning and converting methods have lower computational complexity in comparison with the existing methods.

A series of experiments on various data sets have been conducted to evaluate the performance and efficiency of our proposed methods. The experimental results show that both the converting and pruning methods outperform the existing approaches for SMT. We have also shown that the NNGC converting method and BLMP pruning method can work well together to convert a huge LM trained from 5B words efficiently.

5.2 SMT Adaptation

5.2.1 Data Sets

The proposed methods are evaluated on two data sets. 1) IWSLT 2014 French (FR) to English (EN) corpus[1] [64] is used as in-domain data, which mainly contains TED talks. Dev2010 and test2010, are selected as development (dev) and test data, respectively. Out-of-domain corpora contains Common Crawl, Europarl v7, News Commentary v10 and United Nation (UN) FR-EN parallel corpora[2]. 2) NIST 2006 Chinese (CN) to English corpus[3] is used as in-domain corpus, which follows the setting of [147] and mainly consists of news and blog texts. Chinese to English UN data set (LDC2013T06) and NTCIR-9 [140] patent data are used as out-of-domain bilingual

[1] It is available at `https://wit3.fbk.eu`

[2] It is available at `http://statmt.org/wmt15/translation-task.html`

[3] http://www.itl.nist.gov/iad/mig/tests/mt/2006/

(Bil) parallel corpora. The English patent data in NTCIR-8 [142] is also used as additional out-of-domain monolingual (Mono) corpus. NIST Eval 2002-2005 and NIST Eval 2006 are used as dev and test data, respectively.

IWSLT FR-EN	Sentences	Tokens
in-domain	178.1K	3.5M
out-of-domain	17.8M	450.0M
dev	0.9K	20.1K
test	1.6K	31.9K
NIST CN-EN	**Sentences**	**Tokens**
in-domain	430.8K	12.6M
out-of-domain (Bil)	8.8M	249.4M
out-of-domain (Mono)	33.7M	1.0B
dev (average of four)	4.4K	145.8K
test (average of four)	1.6K	46.7K

Table 5–12 Statistics on data sets.

Statistics on data sets. 'B' indicates billions.

5.2.2 Common Setting

The basic settings of IWSLT-2014 FR to EN and NIST-06 CN to EN translation baseline systems are followed. Moses phrase-based SMT system is applied [148], together with GIZA++ [141], SRILM for interpolated KN 5-gram LM and MERT [139]. Translation performances are measured by case-insensitive BLEU [72], METEOR [74, 75] with significance test [143]. MERT (BLEU based) is run three times for each system and the average BLEU/METEOR scores are recorded. CSTM [69] are applied to NN translation models: phrase length limit is set as seven, shared 320-dimension projection layer for each word (that is 2240 for seven words), 768-dimension projection layer, 512-dimension hidden layer. The dimensions of input/output layers for both in/out-of-domain CSTMs follows the size of vocabularies of source/target words from in-domain corpora. That is, 72K/57K for IWSLT 149/112K for NIST[1].

[1]Since out-of-domain corpora are large, part of them are resampled (resample coefficient 0.01 for IWSLT and NIST corpora) to make sure whole training is finished in one week.

Three mostly related existing methods are selected as baselines[1], [96] for NN based adaptation baseline, [103] for translation model adaptation baseline and [102] for phrase adaptation baseline. 'in-domain', 'out-of-domain' and 'mix-domain' indicate training all models using corresponding corpora, 'in+out-PT/LM' indicates replacing in-domain PT/LM with out-of-domain PT/LM and 'in+NN/connect-PT/LM' indicates replacing in-domain PT/LM with the proposed methods.

5.2.3 Results and Conclusion

Only the best performed results (for both the baselines and proposed methods) on development data are chosen to be evaluated on test data and shown in Table 5–13.

1) Directly using 'out-of-domain' or 'mix-domain' data will cause SMT performances decrease in comparison with 'in-domain' data.

2) Both of NN ('in+NN-PT') and connecting phrase ('in+connect-PT/LM') based methods can significantly enhance performance.

3) These two methods can work jointly ('in+NN+connect-PT+LM') and significantly improve +1.6/1.1 BLEU and +0.8/1.4 METEOR for IWSLT/NIST.

4) 31.7/42.1% (IWSLT/NIST) of phrase pairs in 'in+NN-PT' and 'in+connect-PT/LM' are overlapped. This suggests that the proposed two methods capture quite different phrase selection features.

5) The proposed 'in+NN+connect-PT+LM' outperforms other adaptation methods.

5.2.4 Discussions

Some adapted phrase examples of IWSLT FR-EN task are in Table 5.2.4. For NN based method, some phrases with similar meaning are adapted, such as *third world countries* and *developing countries*. For connecting phrase method, some phrases which are combination of phrases are adapted, such as *the reason* and *why I like* form *the reason why I like*.

[1]We try our best to make necessary and sufficient comparison with all existing related works as we are aware that there are various SMT adaptation works such as [97, 100, 111]. However, there does not exist a commonly used evaluation corpus for this task, and either detailed implementations or experimental settings are absent for most published works, which makes us limit our comparison to the most related and typical methods that can be exactly re-implemented or have implementations released.

IWSLT FR-EN	PT Size	BLEU	METEOR
in-domain	9.8M	31.94	34.07
out-of-domain	759.0M	27.34	32.22
mix-domain	765.4M	30.07	33.19
Duh's method	412.3M	32.65	34.31
Sennrich's method	765.4M	32.41	34.32
Bisazza's method	767.8M	32.24	34.28
in+out-PT	757.0M	30.27	32.75
in+out-LM	9.8M	32.31	34.26
in+connect-PT	184.5M	32.55	34.36
in+connect-LM	9.8M	32.52	34.39
in+connect-PT+LM	184.5M	33.26	34.60
in+NN-PT	296.8M	32.54	34.25
in+NN+connect-PT	405.0M	33.10	34.50
in+NN+connect-PT+LM	405.0M	**33.5**	**34.87**
NIST CN-EN	PT Size	BLEU	METEOR
in-domain	27.2M	32.10	29.29
out-of-domain	365.8M	27.85	22.48
mix-domain	370.9M	31.37	28.80
Duh's method	160.5M	32.51	29.36
Sennrich's method	370.9M	32.36	29.88
Bisazza's method	386.2M	32.15	29.72
in+out-PT	365.8M	27.05	26.49
in+out-LM	27.2M	32.43	29.80
in+connect-PT	142.6M	32.38	30.13
in+connect-LM	27.2M	32.70	30.03
in+connect-PT+LM	142.6M	32.63	29.97
in+NN-PT	187.6M	32.82	30.23
in+NN+connect-PT	260.6M	32.81	**30.65**
in+NN+connect-PT+LM	260.6M	**33.15**	30.50

Table 5–13 Adaptation performances.

All the performances in bold are significantly better than corresponding baselines at level 0.01.

Methods	Source Phrase	Original Target Phrase	Adapted Phrase
NN	*les pays en voie de développement*	*1. developing countries* *2. the developing countries* *3. all developing countries*	*1. developing countries* *2. third world countries* *3. countries in the developing world*
Connect	*la raison pour laquelle je tiens*	*1. the reason I want* *2. why I like* *3. I therefore wish*	*1. the reason why I like* *2. the reason why I want* *3. the reason I would like*

Table 5–14 Some examples of adapted phrases, which are ranked by translation probabilities. The probabilities of overlapped phrases between original target phrase and adapted phrase are interpolated.

5.2.5 Summary

We propose a combinational phrase adaptation method based on NN and linguistic hypotheses, which can select useful additional out-of-domain phrase (pairs) into in-domain PT/LM. Experimental results show that it can improve SMT performance significantly and outperform existing methods.

5.3 SMT Generation

In this section, we focus on LM generation. For translation model generation, it is described in Section 5.4.4.2 as a baseline system.

5.3.1 Experiment Setting

We follow the same settings of NTCIR-9 Chinese to English translation baseline system [140], and the only difference is to use various LMs for comparison. Moses phrase-based SMT is applied [148], together with GIZA++ [141] for alignment and MERT [139] for tuning. 14 standard features of SMT are used: seven distortion scores, one LM score, four translation model scores, one word penalty score, and one phrase pair number penalty score. Various LMs are used to calculate the LM scores. Case-insensitive BLEU is used to evaluate SMT performance. The tool, `mteval-v13a.pl`, is used for calculating BLEU scores[1].

Chinese to English NTCIR-9 patent translation subtask [140][2] is used. The parallel train-

[1]It is available at `http://www.itl.nist.gov/iad/mig/tests/mt/2009/`

[2]We are aware that there is extra large monolingual English corpus for NTCIR-9, unlike TED, whose large extra monolingual corpus is hard to find. NTCIR-9 corpus is selected for a fair comparison among Wang *et. al*. [50]'s (use NTCIR-9 corpus),

ing, development, and test data consists of one million(M), two thousand, and two thousand sentences, respectively. This SMT system is called SMT1M. A subset is randomly selected from the whole train data containing 100K sentences and all other settings the same are as the SMT1M. This SMT system is called SMT100K.

As an example, considering a phrase table built from SMT100K sentences, it consists of nearly 9.5M phrases. So it is too time and space consuming to construct the connecting phrases using all the phrases in the phrase table (9.5M $\times$ 9.5M $\approx$ 90T). For SMT1M, it will take more time and space. In practice, only a small part of top useful phrases (1%-5% by experience for SMT100K and SMT1M) in the phrase table are considered by a ranking method according to Eq. (3–14)[1].

Following SRILM [59, 91], a *5*-gram interpolated KN smoothed BNLM is trained, by using the 1M/100K sentences or 42M/4M words without cutoff. These LMs are called BNLM42M and BNLM4M.

Using the CSLM toolkit [65], 2,3,4,5-CSLMs are trained on the same 1M/100K sentences, respectively. The settings for the CSLMs are the same as CSLM conversion experiments, which are also recommended in the CSLM toolkit and our previous work [50]. These CSLMs are called CSLM42M and CSLM4M.

In this thesis, around 42M words are used as the corpus, including 456K words as vocabulary, and 8K words, which covers 92.89% of words in the training corpus, as short-list for both our method and baseline methods. Arsoy *et. al.* use around 55M words as the corpus, 84K words as vocabulary, and 20K words as short-list. The sizes of our corpus and short-list are similar with theirs in [88, 89]. However, our vocabulary is much larger than theirs, because the whole vocabulary must be used for SMT decoding, compared with only a small vocabulary used for speech recognition. The ratio of $|V_0|/|V|$ affect how much the time complexity will be reduced for the output layer. With a small $|V|$ (84K) in Arsoy *et. al.*'s [88, 89], 76% time cost could be saved as 20K is chosen for the short-list size. For our method, $|V|$ being 456K and 8K being short-list, 98% time cost is saved.

The coverage rate, C_r, of short-list is defined as:

$$C_r = \frac{N_{short-list}}{N_{corpus}} \times 100\%, \tag{5–2}$$

Arsoy *et. al.* [88, 89]'s and our methods. The experiments on TED corpus will be shown in Section 5.3.5, and the experiments using additional monolingual corpus will be shown in Section 5.3.6.

[1]We also empirically compare the proposed ranking method with the phrase table pruning method in Johanson *et. al.* [149], and the ranking method shows better performance. Thus we choose the ranking method at last.

where $N_{short-list}$ indicates the number of words hit in short-list and N_{corpus} indicates the number of words in corpus. The size of short-list is selected according to the corpus by experiments before CSLM is constructed. The results in Table 5–15 show that the coverage rates become saturated after short-list size is larger than 8K. It should be noted that these 1% and 2% of coverage rate differences determine which LM, CSLM or BNLM, is more likely used to calculate the probabilities of *n*-grams. However, it will lead to a big gap of computational cost. For a 8K sized short-list, nearly 93% of the *n*-grams in the training data will be calculated using CSLM and 7% using BNLM. The time complexity for CSLM is approximately linear with the size of output layer (short-list). If the short-list is 20K as that in Arsoy's method, it will take nearly 2.5 times of computing time for training and converting CSLM. Therefore, the 8K short-list is adopted for both methods. In comparison with 20K short-list, 8K makes little loss on the coverage rate, but brings about much higher efficiency.

Sizes of Short-list	Coverage Rates (%)
1K (training)	75.20
4K (training)	86.28
8K (training)	92.89
16K (training)	93.82
24K (training)	94.69
8K (test)	91.08

Table 5–15 Coverage rates of words in short-list.

The proportions of n-grams in the test data covered by CSLM (8K as short-list) and/or BNLM[1] are also counted. For each n-gram, there are four situations shown in Table 5–16.

It should be noted that the common settings of CSLM in our previous work [50], Arsoy *et. al.*'s approach and the proposed bilingual growing method are all the same for fair comparison with the same 2.70GHz CPUs in this thesis. Code for the bilingual CSLM growing is available online[2].

[1] If the n-grams are not covered by BNLM, BNLM will refer to lower-order probabilities with the adjusted weights using back-off.

[2] it is available at `http://bcmi.sjtu.edu.cn/wangrui/program/blmg.zip`

	CSLM	BNLM	short-list	back-off	Proportions (%)
A	In	Out	In	Yes	65.49
B	Out	In	Out	No	2.50
C	In	In	In	No	25.59
D	Out	Out	Out	Yes	6.42

Table 5–16 Proportions of n-grams covered by CSLM and/or BNLM.

5.3.2 SMT Results

Experiments are firstly conducted on SMT100K. That is, CSLM4M and BNLM4M are used for LMs growing, and then the grown LMs are applied to SMT100K. The results are shown in Table 5–17. Then, the experiments are conducted on SMT1M. That is, CSLM42M and BNLM42M are used for LMs growing, and then the grown LMs are applied to SMT1M. The results are shown in Table 5–18. At last, the grown LMs from CSLM4M and BNLM4M are applied to SMT1M to study whether the grown LMs built from a small corpus can outperform the original BNLMs built from larger corpus. In other words, CSLM4M and BNLM4M are used for LMs growing, and then the grown LMs are applied to SMT1M. The results are shown in Table 5–19.

The results of the LM experiments in SMT are divided into five groups: the original BNLMs (BNLM4M/42M), CSLMs in decoding (CSLM4/42M, mainly following the setting of [69, 150]), our previous converting method [50], Arsoy *et. al.* [88, 89]'s growing, and neural network based bilingual growing methods. For our previous method [50] which is referred to Wang4M/42M, the probabilities are only re-written from CSLM into the BNLM. Therefore, only an LM with the same size can be conducted. For the proposed BLMG method, five **Bil**ingual grown LMs for every SMT system (**Bil**4M/42M-1 to 5[1]) are conducted in increasing sizes, and the largest grown LM is around 10 times larger than the original one by the number of *n*-grams. For the method in [88, 89], 5 grown LMs (Arsoy4M/42M-1 to 5) for each SMT system are also conducted in increasing sizes. Entropy based method is adopted to prune them into similar sizes as the grown LMs using our method (Bil4M/42M-1 to 5).

Arsoy grown LMs, our previous converted LM, and the proposed bilingual grown LMs are

[1]We firstly select the top connecting phrases, which are around 10 times larger than the *n*-gram (phrases) in the original BNLM, and then choose the top 20%, 40%, 60%, 80% and 100% of them to construct the Bil42M-1 to Bil42M-5. The IDs after the Bil42M indicate the sizes of grown LMs (in ascending order). The distribution of *n*-grams in different orders ($n = 2, 3, 4, 5$) is the same as the original BNLM.

interpolated with the original BNLMs[1]. Two methods are used for tuning SMT feature weights, in order to reduce the randomness of MERT. **BLEU-s** indicates that the **same** weights of the baseline (BNLM4M/42M) features are used for all the SMT systems. **BLEU-i** indicates that the MERT is run **independently** by three times and the average BLEU scores are recorded. The trends of PPL and BLEU-s are illustrated in Figures 5–2 and 5–3, respectively.

The paired bootstrap re-sampling test [143][2] is performed. 2000 samples are used for each significance test.

Comparison are also conducted on re-ranking for CSLM and our bilingual grown LMs. CSLM42M and Bil42M are used to re-rank the 1000-best lists of SMT1M. That is, the BNLM scores in the 1000-best lists are replaced with the CSLM42M/Bil42M scores, and then the global scores are re-ranked. For CSLM re-ranking in Table 5–20, the PPL column indicates that the PPLs are calculated using CSLM, and the BLEU(Re-ranking) column indicates that the feature weight of CSLM42M/Bil42M is tuned using Z-MERT [146]. The results are presented in Table 5–20.

LMs	*n*-grams	PPL	BLEU-s	BLEU-i	ALH
BNLM4M	10.3M	144.8	27.48	27.48	2.57
CSLM4M	N/A	**126.6**	N/A	28.03	N/A
Wang4M	10.3M	140.5	27.69	27.72	2.57
Arsoy4M-1	33.3M	135.1	27.70	27.90	2.68
Arsoy4M-2	53.3M	134.7	27.72	27.81	2.69
Arsoy4M-3	68.4M	134.3	27.82	27.71	2.71
Arsoy4M-4	89.5M	134.2	27.83	27.92	2.72
Arsoy4M-5	107.9M	134.1	27.74	27.83	2.73
Bil4M-1	33.5M	134.8	27.94+	27.97	2.74
Bil4M-2	49.4M	134.1	28.05++	28.06+	2.78
Bil4M-3	65.6M	133.2	28.00+	27.99++	2.80
Bil4M-4	90.1M	132.5	27.99+	**28.07**+	2.82
Bil4M-5	110.3M	132.2	**28.10**++	28.02+	**2.83**

Table 5–17 Performance of the grown LMs in SMT100K.

[1] In this thesis, the default 0.5 is used as the default interpolation weights.

[2] The implementation of our system follows `http://www.ark.cs.cmu.edu/MT`

LMs	*n*-grams	PPL	BLEU-s	BLEU-i	ALNH
BNLM42M	73.9M	108.8	32.19	32.19	3.03
CSLM42M	N/A	**97.5**	N/A	33.18	N/A
Wang42M	73.9M	104.4	32.60	32.62	3.03
Arsoy42M-1	217.6M	103.3	32.55	32.75	3.14
Arsoy42M-2	323.8M	103.1	32.61	32.64	3.18
Arsoy42M-3	458.5M	103.0	32.39	32.71	3.20
Arsoy42M-4	565.6M	102.8	32.67	32.51	3.21
Arsoy42M-5	712.2M	102.5	32.49	32.60	3.22
Bil42M-1	223.5M	101.9	32.81+	33.02+	3.20
Bil42M-2	343.6M	101.0	32.92+	33.11++	3.24
Bil42M-3	464.5M	100.6	33.08++	**33.25**++	3.26
Bil42M-4	571.0M	100.3	**33.15**++	33.12++	3.28
Bil42M-5	705.5M	100.1	33.11++	33.24++	**3.31**

Table 5–18 Performance of the grown LMs in SMT1M.

LMs	*n*-grams	PPL	BLEU-s	BLEU-i	ALH
BNLM42M	73.9M	108.8	32.19	32.19	**3.03**
Wang42M	73.9M	**104.4**	**32.60**	**32.62**	**3.03**
Bil4M-1	33.5M	135.2	29.45	29.98	2.74
Bil4M-2	49.4M	134.1	29.47	30.62	2.78
Bil4M-3	65.6M	133.2	29.46	30.49	2.80
Bil4M-4	90.1M	132.5	29.57	30.31	2.82
Bil4M-5	110.3M	132.2	29.60	30.56	2.83

Table 5–19 Performance of the grown LMs (Bil4M) in SMT1M.

LMs	*n*-grams	PPL	BLEU (Re-ranking)
BNLM42M	73.9M	108.8	32.19
CSLM42M	N/A	**97.5**	32.46
Bil42M-1	223.5M	101.9	32.31
Bil42M-2	343.6M	101.0	32.44
Bil42M-3	464.5M	100.6	32.37
Bil42M-4	571.0M	100.3	**32.51**
Bil42M-5	705.5M	100.1	32.28

Table 5–20 Performance of the grown LMs and CSLM in re-ranking.

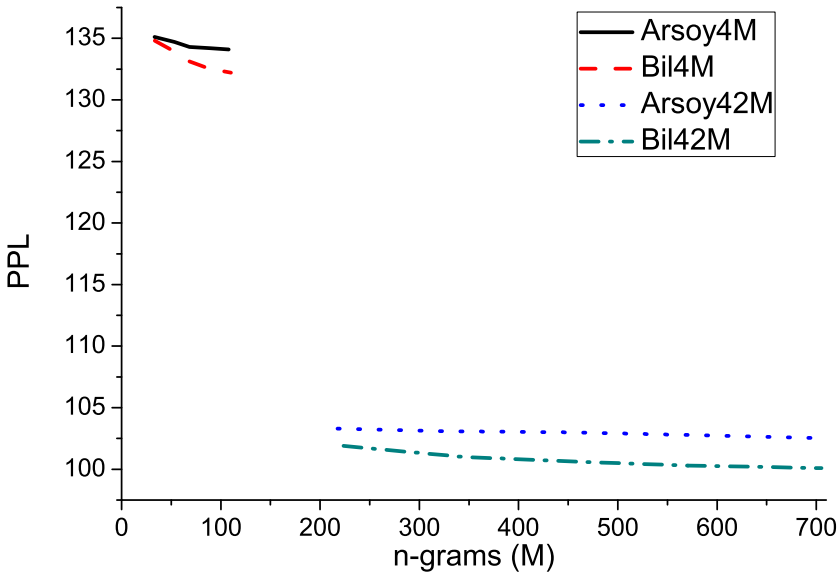

Figure 5–2 Trend of PPL as the LMs grow (lower is better).

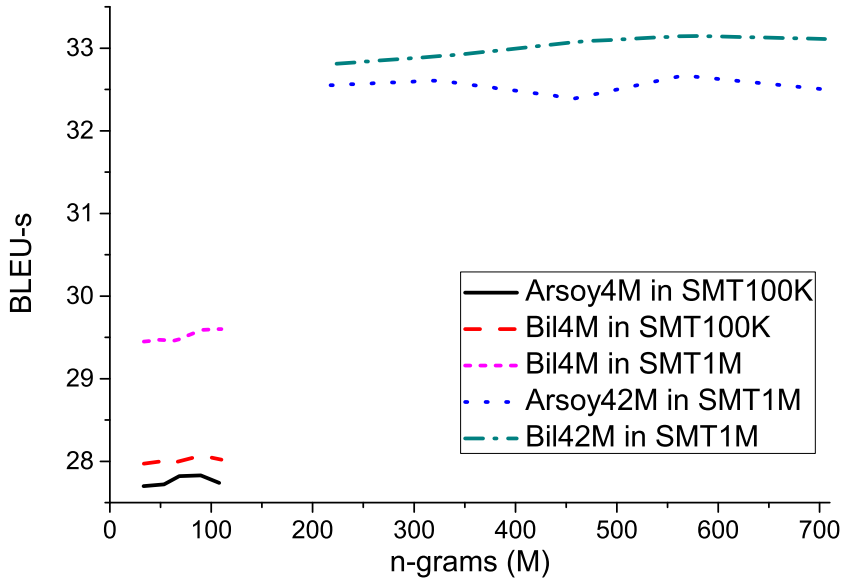

Figure 5–3 Trend of BLEU as the LMs grow (higher is better).

From the results shown in Tables 5–17, 5–18, 5–19, and 5–20 and Figures 5–2 and 5–3, the following observations are obtained:

1) Nearly all the BLMG LM outperform the original BNLM on the PPL and BLEU. These indicate that our BLMG method can estimate probability better. Compared with the CSLM in re-ranking and decoding methods, the bilingual grown LMs obtain higher PPLs, but similar BLEUs.

2) BLEU scores trend to increase and PPLs always decrease. Bil42M-1 keeps the top 20% ranked connecting phrases, which are the most useful connecting phrases, and therefore it performs similar as Bil42M-5, which contains all the selected connecting phrases.

3) Compared with the grown LMs in [88, 89] with the similar size, the proposed BLMG LMs obtain better PPL and significantly better BLEU. As the sizes of LM increase, the PPL and BLEU improvements of baselines become saturated much more quickly than the proposed method.

4) The grown LMs (Bil4M) in SMT1M perform much better than in SMT100K, but much worse than the BNLM42M with the similar size in SMT1M. This indicates that the grown LMs built from small corpus can indeed improve the performance on PPL and BLEU. But they do not perform well compared to the LMs originally built from large corpus[1].

[1]Given a large extra monolingual corpora (in-domain and out-of-domian), our connecting phrase method will be suitable for LM domain adaptation. Please refer to Section 5.3.6 for details.

5.3.3 Efficiency Comparison

In this subsection, the efficiency of our new bilingual LM growing method is investigated.

5.3.3.1 Efficiency for Constructing Connecting Phrases

The average time for constructing 1M trigrams connecting phrases for SMT100K and SMT1M is shown in Table 5–21. The results indicate that the constructing time is much less than the growing time in Table 5–22 for the same 1M trigrams.

SMT Systems	Constructing Time (sec.)
SMT100K	11.3
SMT1M	17.9

Table 5–21 Constructing connecting phrases time.

5.3.3.2 Growing Time

The growing time of Arsoy's method and the proposed bilingual growing method is evaluated. The 1M trigrams are used as the input n-grams for CSLMs (SMT100K for CSLM4M and SMT1M for CSLM42M) for both methods. The time of LMs growing is recorded. For the growing step, Arsoy's method will generate much more n-grams and then prunes it into smaller ones[1]. The proposed method only produces the same size n-grams. The growing time used by these two methods is presented in Table 5–22.

CSLMs	Arosy	Our
CSLM4M	24 hours 10 minutes	37 minutes
CSLM42M	24 hours 24 minutes	41 minutes

Table 5–22 Growing time.

From Table 5–22, we can see that Arosy's growing method takes much more time (nearly 40 times) than ours. The Arsoy's method grows all the possible n-grams ended with the words in the short-list (usually thousands times of the original one) and then prunes them into smaller

[1]The pruning actually takes a lot of time, but we do not take it into account for fair comparison of growing time.

ones. The probabilities of all the tail words in short-list can be calculated at once using CSLM neurons, Arsoy's method still takes much more time than ours. For the proposed method, which n-grams to be grown are decided according to the appearing probabilities in Eq. (3–15) before they are put into CSLM. The growing time for the proposed method is approximately linear with the size of input n-grams. In practice, a grown LM which is around 8 times larger than the original one has the best performance in SMT as shown in Tables 5–17, 5–18 and 5–19.

5.3.3.3 Decoding Time

Vaswani *et. al.* [79] propose several techniques (such as NCE training algorithm) to improve the efficiency of using NPLM in SMT. Meanwhile our proposed growing method stores the probabilities of the CSLM into BNLM format to improve the efficiency of using CSLM in SMT. In this subsection, we evaluate the decoding efficiency of the BNLM, CSLM, NPLM[1] and our bilingual grown LM with all the same settings in SMT decoding. The 2,000 English reference sentences of the NTCIR-9 test data are used for evaluation. The log-probabilities of every sentence of the test data are calculated using different LMs for SMT decoding stimulation. The decoding time is shown in Table 5–23.

LMs	Decoding Time (sec.)
BNLM42M	**15.3**
CSLM42M	186.5
NPLM42M	50.4
NPLM42M (normalized)	456.8
Bil42M-3	16.5

Table 5–23 Decoding time.

From Table 5–23, the following conclusions can be reached:

1) The decoding time using Bil42M-3 and BNLM are quite close. This indicates that the proposed converting method can run as fast as the BNLM.

2) The decoding time using CSLM is much slower than the BNLM and Bil42M-3. This demonstrates the advantage of decoding speed of the proposed method.

[1]The numbers of hidden layers are set the same as the CSLM and other settings follow the default setting of NPLM toolkit [79].

3) The decoding time using the NPLM (without normalization) is much faster than the CSLM and much slower than the BNLM/Bil42M-3. This also shows the the efficiency advantage of our proposed method.

5.3.4 Function of Connecting Phrase

The above results show that the proposed LM growing method performs better in terms of both PPL and BLEU, because more useful connecting phrases have been added to the LM. This subsection will show how these connecting phrases perform in SMT in detail.

Given each sentence from the test data, the probability of every target word is calculated using the grown LMs, to simulate the performance of the LMs in SMT decoding. Then the ratio of the different *n*-grams used for each grown LMs on all the test data is counted.

The results of the *i*-grams hit in SMT decoding are shown in Tables 5–17, 5–18 and 5–19. Average Length of the *N*-grams Hit (ALNH) as mentioned in Section 5.1.3.1 is shown in the last column and is illustrated in Figure 5–4.

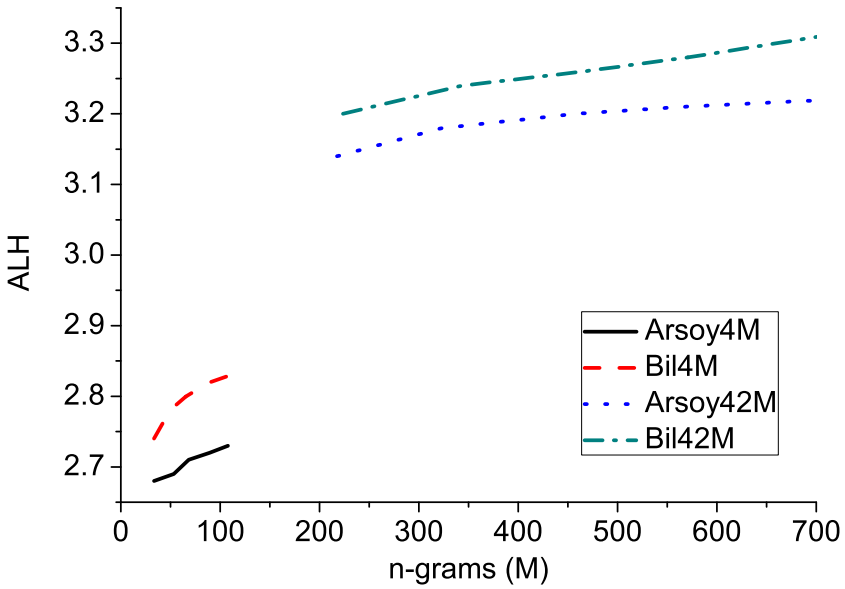

Figure 5–4 Trend of ALH in SMT decoding (Longer is better).

The ALNHs of Bil42/4M are longer than Arsoy42/4M. There are also certain positive correlations among ALNH, PPL and BLEUs.

As we know, the LM refers to the probabilities of (*i*-1)-grams together with the adjusted weights using back-off, if no any corresponding *i*-gram is hit. The statistics above indicate

that more high-order *n*-grams are hit using the proposed grown LMs in SMT in comparison to the Arsoy's grown LMs. In another word, less back-off is applied to the proposed connecting phrase-based grown LMs in SMT decoding.

5.3.5 Experiments on TED Corpus

As TED corpora is in special domain, so it is difficult to find a large extra monolingual corpus in the same domain. We conduct the SMT experiments on TED corpora by using the proposed LM growing method in this subsection.

We follow the IWSLT 2014 evaluation campaign baselines[1], with only a few modifications. That is, the *n*-gram orders and LM toolkits. The French (FR) to English (EN) and Chinese (CN) to English language pairs are chosen. The data sets, dev2010 and test2010, are selected as development data and evaluation data, respectively. The statistics on TED parallel data used for setting up the baselines are described in Table 5–24.

FR-EN	Sentences
training	186.8K
dev2010	0.9K
test2010	1.6K
CN-EN	**Sentences**
training	186.8K
dev2010	0.9K
test2010	1.6K

Table 5–24 Statistics on TED parallel data.

The same LM growing method is applied to TED corpora as on NTCIR corpora. That is, the bilingual data is used to grow the LMs, and then the monolingual grown LMs are integrated into SMT system. The results of TED experiments are divided into three groups: baseline method using BNLM (BNLN)[2], our previous method using converted CSLM (Wang) [50], and

[1] It is available at `https://wit3.fbk.eu`

[2] Our CN-EN baseline is a little better than the baseline (BLEU = 11.21) in IWSLT 2014. For the FR-EN language pair, only the EN-FR baseline is shown (BLEU = 29.44) in IWSLT 2014. Please refer to `https://wit3.fbk.eu/score.php?release=2014-01` for details.

our proposed bilingual grown LMs (Bil1-1,2,3) on both language pairs. The results are shown in Tables 5–25 and 5–26.

LMs	*n*-grams	PPL	BLEU-s
BNLM	8.2M	90.0	32.30
Wang	8.2M	85.4	32.67
Bil-1	27.1M	83.2	32.87
Bil-2	52.4M	82.6	33.05
Bil-3	79.8M	**81.8**	**33.21**

Table 5–25 IWSLT FR-EN (TED) experiments.

LMs	*n*-grams	PPL	BLEU-s
BNLM	7.8M	87.1	12.41
Wang	7.8M	85.3	12.73
Bil-1	23.1M	79.2	12.92
Bil-2	49.7M	78.3	13.16
Bil-3	73.4M	**77.6**	**13.24**

Table 5–26 IWSLT CN-EN (TED) experiments.

Tables 5–25 and 5–26 demonstrate that the proposed growing method can improve the performances of LMs in both PPL and BLEU for different corpora and language pairs.

5.3.6 Experiments on Additional Monolingual Corpora

So far, the concerned LMs are limited to the target side of bilingual corpora. In this subsection, the CSLMs using additional monolingual corpora are constructed and compared with our bilingual grown LMs without using additional monolingual corpora.

We conduct experiments on both in-domain and out-of-domain corpora. For the in-domain NTCIR-9 patent experiments, the 2005 US patent English data set distributed in the NTCIR-8 patent translation task [142] is used as the additional monolingual corpus, which consists of around 5M sentences[1]. For the out-of-domain TED experiments, the NIST English OpenMT06

[1]The original corpus consists of 25M sentences, and we choose a part of these sentences randomly.

data set[1], which consists of around 400K sentences, is used following the same setting of [147]. TED data set belongs to a specific domain, so it is not easy to obtain in-domain additional monolingual corpus in fact. The additional US patent data and NIST data are added to the original NTCIR-9 and TED data, respectively, to construct the large monolingual corpora. These corpora are called Corpus-US-patent and Corpus-NIST, respectively. The statistics on these corpora are shown in Table 5–27.

Corpora	Sentences	Words
Corpus-NTCIR	1.0M	42.3M
Corpus-US-patent	5.9M	202.9M
Corpus-TED	186.8K	2.7M
Corpus-NIST	591.7K	15.2M

Table 5–27 Statistics of additional monolingual corpora.

The LM experimental settings are conducted in two ways: (a) both CSLMs and BNLMs are built from the large corpora (Corpus-US-patent/NIST), and the large BNLMs are converted using large CSLMs. These converted CSLMS are called Converted CSLM-US-patent/NIST-(a); (b) only the BNLMs are built from the large corpora, and the large BNLMs are converted using small CSLMs built from NTCIR-9/TED corpora. These converted CSLMS are called Converted CSLM-US-patent/NIST-(b).

The SMT experimental settings follow Section 5.3.1, except for using different LMs. That is, the same CN-EN phrase table and other models used in Section 5.3.1 are applied, and LMs are built from different size corpora. The bilingual grown LMs (Bil42M-2 (NTCIR) and Bil-1 (TED)), with the similar size as the corresponding Converted CSLMs built from large additional corpora, are selected for fair comparison.

The results are shown in Tables 5–28 and 5–29. For the NTCIR patent experiments, the Converted CSLM-US-patent using additional monolingual in-domain patent corpus perform better than our bilingual grown LM. For the TED SMT experiments, our bilingual grown LMs perform better than the Converted CSLM-NIST using the out-of-domain additional monolingual corpus. These results suggest that our bilingual LM growing method is useful for corpus adaptation.

[1]It is available at `http://www.itl.nist.gov/iad/mig/tests/mt/2006/`. The data mainly consists of news and blog texts.

LMs	*n*-grams	PPL	BLEU-s
BNLM-NTCIR	73.9M	108.8	32.19
BNLM-US-patent	312.4M	95.7	32.76
Converted CSLM-US-patent-(a)	312.4M	**91.3**	**33.24**
Converted CSLM-US-patent-(b)	312.4M	97.5	32.85
Bil42M-2 (NTCIR)	343.6M	101.0	32.92

Table 5–28 NTCIR experiments with additional monolingual corpus.

LMs	*n*-grams	PPL	BLEU-s
BNLM-TED	7.8M	87.1	12.41
BNLM-NIST	26.5M	111.2	8.03
Converted CSLM-NIST-(a)	26.5M	102.1	10.87
Converted CSLM-NIST-(b)	26.5M	96.3	11.32
Bil-1 (TED)	23.1M	**79.2**	**12.92**

Table 5–29 TED (CN-EN) experiments with additional monolingual corpus.

5.3.7 Experiments on Hierarchical Phrase-based and Syntax-based SMT

The experiments of applying our bilingual grown CSLMs to hierarchical phrase-based and syntax-based SMT are also conducted. Namely, the bilingual grown CSLMs are firstly constructed using phrase table, and then applied to hierarchical phrase-based or syntax-based SMT. The same settings for the NTCIR-9 hierarchical translation baseline system [140] are followed for hierarchical phrase-based SMT, and WAT-2014 String-to-Tree translation baseline system[1] [151] for syntax-based SMT.

Experiments are conducted on NTCIR-9 Chinese-to-English and IWSLT-2014 TED French-to-English corpora, and results are shown in Tables 5–30, 5–31, 5–32 and 5–33.

The results in Tables 5–30, 5–31, 5–32 and 5–33 demonstrate that the proposed LM growing method can improve the BLEU for different translation models.

[1]To accelerate the speed of parsing and training, the maximum length limit of sentences is set a little shorter than the baseline.

LMs	*n*-grams	PPL	BLEU-s
BNLM	73.9M	108.8	33.14
Bil42M-1	223.5M	101.9	33.38
Bil42M-3	464.5M	100.6	33.64
Bil42M-5	705.5M	**100.1**	**33.71**

Table 5–30 Hierarchical phrase-based SMT on CN-EN NTCIR.

LMs	*n*-grams	PPL	BLEU-s
BNLM	73.9M	108.8	29.03
Bil42M-1	223.5M	101.9	29.39
Bil42M-3	464.5M	100.6	**29.51**
Bil42M-5	705.5M	**100.1**	29.42

Table 5–31 Syntax-based SMT on CN-EN NTCIR.

LMs	*n*-grams	PPL	BLEU-s
BNLM	8.2M	90.0	31.79
Bil-1	27.1M	83.2	32.03
Bil-2	52.4M	82.6	**32.35**
Bil-3	79.8M	**81.8**	32.12

Table 5–32 Hierarchical phrase-based SMT on FR-EN TED.

LMs	*n*-grams	PPL	BLEU-s
BNLM	8.2M	90.0	28.64
Bil-1	27.1M	83.2	28.87
Bil-2	52.4M	82.6	29.10
Bil-3	79.8M	**81.8**	**29.21**

Table 5–33 Syntax-based SMT on FR-EN TED.

5.3.8 Summary

A novel proposed LM growing method has two attractive features. First, it constructs a large efficient LM using neural network. The pre-computed CSLM probabilities inside/outside the corpus are stored in BNLM format, and therefore the grown LMs can perform as precisely as CSLM and run as fast as the BNLM. Second, it takes the phrase-table into consideration, and makes the grown LM obtain bilingual information for SMT. The key points of our method are to construct and select the connecting phrases, which are likely to appear in SMT decoding but outside phrase-table. The ranking functions are carefully designed for constructing the connecting phrases precisely and efficiently.

Various metrics are applied to evaluate our new method, and the experimental results show that the proposed method significantly outperforms the existing LM converting/growing methods in SMT performance. Both the growing time and decoding time are also significantly reduced. From the experimental results, we can see that the proposed method is promising and can be applied to LM adaptation for additional monolingual out-of-domain corpus.

5.4 Bilingual Graph based Semantic Model

5.4.1 Lexical Translation

We follow the bilingual embeddings on the word translation task proposed by[23]. In their task, the authors extracted the 6K most frequent words from the WMT11 English-Spanish data[1], and then used the online Google Translate service to derive dictionaries by translating these source words into the target language (individually for English and Spanish). Since their method requires translation-pairs for training, they used the first 5K most frequent words to learn the "*translation matrix*", and then evaluated their method on the remaining 1K words used as a test set. To translate a source word, one finds its k nearest neighbours in the target language embedding space, and then evaluate the translation precision $P@k$ as the fraction of target translations that are within the top-k words returned using the specific method. GBWE does not require translation-pairs for training, so we simply train our model using WMT11 data, tune the parameters (output dimension and threshold γ for) on 5k pairs and test on the 1K test-pairs.

We use as baselines the same three methods reported in [23]: *Edit Distance*, *Word Co-occurrence* and *Translation Matrix* methods. We also compare with state-of-the-art[2] bilingual

[1] `http://www.statmt.org/wmt11/`

[2] We are aware there are some other related works focus on this task[152, 153]. However they did not release their codes

word embedding method BilBOWA [49].

Methods	En-Sp P@1	Sp-En P@1	En-Sp P@5	Sp-En P@5	Average Accuracy
Edit Distance	13	18	24	27	20.5
Word Co-occurrence	30	19	20	30	24.7
Translation Matrix	33	35	51	52	42.7
BilBOWA	**39**	**44**	51	55	47.2
BGSM	36	42	**55**	**60**	**48.2**

Table 5–34 Bilingual word embeddings on word translation task.

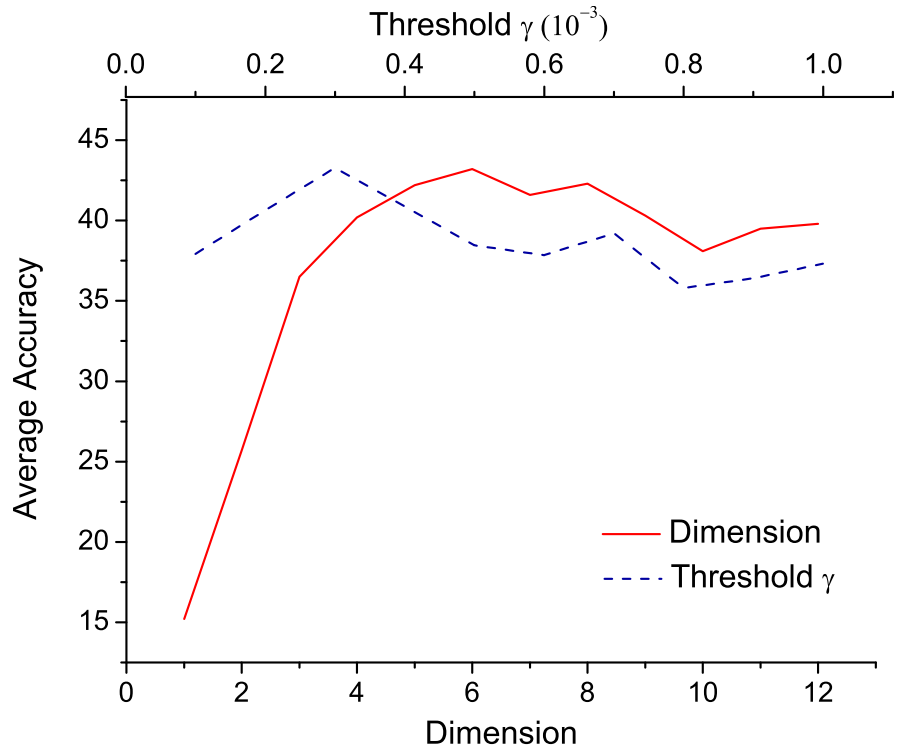

Figure 5–5 Output dimension and node pruning threshold γ tuning for lexical translation.

As shown in Table 5–34 and Figure 5–5, GWEB outperforms other methods in $P@5$ tasks and average accuracy, but does not is $P@1$ tasks. The reason may lie in that there is no contextual information for lexical translation task. Although GWEB can predict five good translation

or details implements as BilBOWA did.

candidates, it does not know which sense the word should belong to. So we will investigate how GWEB will preform in phrase translation, if given contextual information.

5.4.2 Setting Up

To evaluate performance of BGSM in various language and domain SMT systems, IWSLT-2014 French (FR) to English (EN)[1], NTCIR-9 Chinese (CN) to English patent translation task [140] and NISTOpenMT08[2] are chosen. The statistics on parallel data are described in Table 5–35.

Corpus	IWSLT	NCTIR	NIST
training	186.8K	1.0M	2.4M
dev	0.9K	2.0K	1.6K
test	1.6K	2.0K	1.3K

Table 5–35 Sentences statistics on parallel corpora.

5.4.3 Baseline Systems

The same basic settings for the IWSLT-2014 [119], NTCIR-9 translation baseline systems [140] and NIST08 are followed. The standard Moses phrase-based SMT system is applied [148] together with GIZA++ for alignment, SRILM for language modeling and MERT for tuning. The tool, `mteval-v13a.pl`[3], is used for calculating BLEU scores (we run MERT three times and record the average BLEU score). The paired bootstrap re-sampling test[4] is performed. Significant tests are done for each round of test. Marks at the right of BLEU scores indicate whether our proposed methods are significantly better/worse than the corresponding baseline ('++/−−': significantly better/worse at *significance level* $\alpha = 0.01$; '+/−': $\alpha = 0.05$). All the experiments in this thesis are conducted on the same machine with 2.70GHz CPU.

We are aware that there are several state-of-the-art end-to-end neural SMT methods [44, 45]. However, the proposed BGSM is a bilingual word embedding method and applied to SMT as

[1]It is available at `https://wit3.fbk.eu`

[2]Zou et. al. zou-EtAl:2013:EMNLP only released their vectors rather than their code (`http://ai.stanford.edu/~wzou/mt/`), so we have to conduct experiments on NIST08 Chinese-English translation task as they did. The training data consists of part of NIST OpenMT06, United Nations Parallel Text (1993-2007) and corpora of [154, 155] that were used by [25]. NIST Eval 2006 is used as development data and NIST Eval 2008 as test data.

[3]It is available at `http://www.itl.nist.gov/iad/mig/tests/mt/2009/`

[4]The implementation of our system follows `http://www.ark.cs.cmu.edu/MT`

additional features, so we only compare with most related bilingual word embedding or generation methods for SMT. For phrase pair translation probability estimation task, two typical NN based bilingual embedding methods, Continuous Space Translation Model (CSTM[1]) [69] and [25], are select as baselines. However, their method is quite similar with ours, because we both use co-occurrence matrix (graph). So we apply bilingual co-occurrence matrix to GloVe.. The embedding of each method is added as additional features to the phrase-based SMT baseline, with all of the other setting the same. For bilingual phrase generation methods, CSTM is used to generate phrase pairs[2]. We also compare with [127], which uses graph method for translation candidates generation (They use word graph directly, and we use word graph to extract sense unit BCCs in comparison).

5.4.4 Results and Analysis

Only the best performed results (for both the baselines and proposed methods) on development data are chosen to be evaluated on test data and shown.

5.4.4.1 Phrase Pair Translation Probability Estimation Results

	IWSLT	NTCIR	NIST
Baseline	31.80	32.19	30.12
+Zou	N / A	N / A	30.36
+CSTM	32.19	32.37	30.25
+BGSM-A	**32.32+**	**32.56**	**30.38**
+BGSM-B	**32.61++**	**33.04++**	**30.44+**

Table 5–36 Phrase Pair Translation Probability Estimation Results (BLEU).

Zou et. al. only released their vectors instead of codes. So their method is only applied to NIST08 as they did. Please refer to Footnote 2 for detail.

Table 5–36 indicates that our proposed method can improve the SMT performance up to +0.85 BLEU, and slightly better than the existing NN methods up to +0.67 BLEU.

[1]The recommended settings of CSTM are followed.

[2]They have discussed phrase generation method as experimental evidence in their thesis, and we follow their basic idea.

Besides, the Strategy-B performs better than Strategy-A. The reason may be that more contextual (both target and source) information are used for Strategy-B in comparison with that only source contextual information are used for Strategy-A.

5.4.4.2 Bilingual Phrase Pair Generation Results

'Baseline + BPG' indicates adding the generated phrase pairs to original phrase table. 'BGSM + BPG' indicates adding the generated phrase pairs to original phrase table, as well as replacing the translation probabilities $\psi(P_E|P_F)$ and $\psi(P_F|P_E)$ in original phrase table with the $Sim(P_E|P_F)$ and $Sim(P_F|P_E)$ calculated by BGSM (please refer to Section 4.4.1 for detail).

Corpora	Methods	Phrase-table Size	BLEU
IWSLT	Baseline	9.8M	31.80
	+CSTM	23.1M	32.19
	+Saluja	31.5M	32.35
	+BPG	25.6M	32.37
	+BPG+BGSM	25.6M	**33.13++**
NTCIR	Baseline	71.8M	32.19
	+CSTM	143.8M	32.42
	+Saluja	341.3M	32.68
	+BPG	312.6M	32.54+
	+BPG+BGSM	312.6M	**33.47++**

Table 5–37 bilingual phrase generation (BPG) results.
Phrase-table size indicates number of phrase pairs in phrase-table.

The results in Table 5–37 indicate that the proposed BPG method and BGSM method can work well together and enhance SMT performance significantly up to +1.33 BLEU. They also outperform the best performances of existing methods up to +0.79 BLEU.

5.4.4.3 Efficiency Comparison

Both of our proposed method and CSTM pre-compute the probability scores of the phrase pairs and then use these scores in SMT decoding. The difference is that we use graph-based method and they use NN-based method. In this subsection, we compare the efficiencies for model training and computing the probability scores of phrases pairs using these two models. Two thousand

phrase pairs are randomly selected from the whole IWSLT-2014 FR-EN corpus. The CPU time of training models and calculating their probability scores using CSTM and BGSM is shown in Table 5–38.

Methods	Training Time	Calculating Time
CSTM	59.5 Hours	17.1 Minutes
BGSM-A	1.1 Hours	8.9 Minutes
BGSM-B	1.1 Hours	15.6 Minutes

Table 5–38 CPU time on IWSLT-2014.

The results in Table 5–38 demonstrate that the proposed method is much more efficient than CSTM, especially for training, it can be nearly 60 times as fast as CSTM.

5.4.5 Summary

Although there are several bilingual word embedding methods, nearly all of them only consider word information itself. In comparison, we split the word embedding procedure into two steps:

1) Using a graph based method, a novel bilingual sense unit bilingual contexonym clique is proposed. which can describe senses of a word better instead of word itself.

2) A context-based dynamic bilingual clique-word matrix is constructed and CA applied to summarize this matrix into low dimensions. The BGSM is consequently constructed for dynamically bilingual word embedding.

Conclusion

5.5 Conclusion

Continuous-Space based, especially NN based SMT has sparked widespread concern recently. It manages to learn distributed word, phrase or sentence representation, which makes the probability estimation more accurate in comparison with existing linear models.

Aiming at overcoming some drawbacks for NN based SMT, we propose several methods to improve SMT performance in Language modeling, adaptation, generation and bilingual word embedding.

For LM aspect, we convert CSLMs into BNLMs. The empirical results show that the converted CSLMs can predict the probabilities as accurate as CSLMs, and run as fast as BNLMs. It also outperforms the existing CSLM conversion methods.

For adaptation aspect, we propose a combinational phrase adaptation method based on NN and linguistic hypothesis. In comparison with traditional sentence level adaptation, phrase adaptation provide more accurate granularity. Empirical experiments show that the proposed adaptation methods can improve SMT performance significantly, and outperform existing adaptation methods.

For generation aspect, the proposed LM growing method can be regarded as an optimization method of the probabilities of the items in an n-gram LM which requires a small computational cost, using an NNLM which has a high generalization ability and requires a large computational cost. The empirical experiments also show it can improve LM performance and SMT performance significantly.

For bilingual word embedding aspect, we present a novel Bilingual Graph-based Semantic Model (BGSM). By means of maximum complete sub-graph (clique) for context selection, BGSM is capable of effectively modeling word sense representation instead of word itself. The proposed model is applied to phrase pair translation probability estimation and generation for SMT. The results show that BGSM can enhance SMT both in performance and efficiency in comparison with existing methods.

5.6 Future Work

NN based deep learning methods cannot be implemented by computer in a reasonable time until very recently. Therefore, most of the proposed methods in this thesis focus on how to save the training and using time of NN based methods. As the calculating ability of computer keeps being improved, the computational cost problem is not so important as before, so we will focus on SMT performance in our future work.

Recently, a series of advanced deep learning technologies, such as LSTM and attention based NN have been applied to SMT. In our future work, we will also try to apply some linguistic motivated feature to these deep learning methods.

References

[1] M. D. Hauser, N. Chomsky and W. T. Fitch. "*The Faculty of Language: What Is It, Who Has It, and How Did It Evolve?*" *Science*, **2002**.

[2] J. Hirschberg and C. D. Manning. "*Advances in natural language processing*". *Science*, **2015**.

[3] W. Weaver. "*Letter to Norbert Wiener, March 4, 1947*". *Rockefeller Foundation Archives*, **1947**.

[4] P. F. Brown *et al.* "*A Statistical Approach to Machine Translation*". *Computational Linguistics*, **1990**.

[5] P. F. Brown *et al.* "*The Mathematics of Statistical Machine Translation: Parameter Estimation*". *Computational Linguistics*, **1993**.

[6] A. L. Berger *et al.* "*The Candide System for Machine Translation*". In: *HLT*, **1994**.

[7] P. Koehn, F. J. Och and D. Marcu. "*Statistical phrase-based translation*". In: *NAACL*, **2003**.

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

[9] G. E. Hinton. "*Learning distributed representations of concepts*". In: *CogSci*, **1986**.

[10] D. E. Rumelhart, G. E. Hinton and R. J. Williams. "*Learning representations by back-propagating errors*". *Nature*, **1986**.

[11] G. A. Miller *et al.* "*Introduction to wordnet: An on-line lexical database**". *International journal of lexicography*, **1990**.

[12] Ed. by P. Vossen. *EuroWordNet: A Multilingual Database with Lexical Semantic Networks*. Kluwer Academic Publishers, **1998**.

[13] S. Edelman. "*Representation is representation of similarities*". *Behavioral and Brain Sciences*, **1998**.

[14] T. K. Landauer and S. T. Dutnais. "*A solution to Plato's problem: The latent semantic analysis theory of acquisition, induction, and representation of knowledge*". *Psychological review*, **1997**.

[15] D. M. Blei, A. Y. Ng and M. I. Jordan. "*Latent dirichlet allocation*". *Journal of machine Learning research*, **2003**.

[16] E. Sumita. "*Lexical Transfer Using a Vector-space Model*". In: *ACL*, **2000**.

[17] M. Carpuat and D. Wu. "*Improving Statistical Machine Translation Using Word Sense Disambiguation*". In: *EMNLP*, **2007**.

[18] Y. Peirsman and S. Padó. "*Cross-lingual Induction of Selectional Preferences with Bilingual Vector Spaces*". In: *NAACL*, **2010**.

[19] Y. Bengio *et al.* "*A neural probabilistic language model*". *Journal of Machine Learning Research*, **2003**.

[20] E. Huang *et al.* "*Improving Word Representations via Global Context and Multiple Word Prototypes*". In: *ACL*, **2012**.

[21] T. Mikolov, W.-t. Yih and G. Zweig. "*Linguistic Regularities in Continuous Space Word Representations*". In: *NAACL*, **2013**.

[22] T. Mikolov *et al.* "*Distributed representations of words and phrases and their compositionality*". In: *NIPS*, **2013**.

[23] T. Mikolov, Q. V. Le and I. Sutskever. "*Exploiting similarities among languages for machine translation*". *arXiv*, **2013**.

[24] T. Mikolov *et al.* "*Efficient estimation of word representations in vector space*". *ICLR 2013*, **2013**.

[25] W. Y. Zou *et al.* "*Bilingual Word Embeddings for Phrase-Based Machine Translation*". In: *EMNLP*, **2013**.

[26] J. Gao *et al.* "*Learning Continuous Phrase Representations for Translation Modeling*". In: *ACL*, **2014**.

[27] S. Lauly *et al.* "*An Autoencoder Approach to Learning Bilingual Word Representations*". In: *NIPS*, **2014**.

[28] K. Cho *et al.* "*Learning Phrase Representations using RNN Encoder–Decoder for Statistical Machine Translation*". In: *EMNLP*, **2014**.

[29] O. Levy and Y. Goldberg. "*Neural Word Embedding as Implicit Matrix Factorization*". In: *NIPS*, **2014**.

[30] H. Schwenk. "*Continuous space language models*". *Computer Speech and Language*, **2007**.

[31] T. Mikolov *et al.* "*Recurrent Neural Network based Language Model*". In: *INTERSPEECH*, **2010**.

[32] H.-S. Le *et al.* "*Structured Output Layer neural network language model*". In: *ICASSP*, **2011**.

[33] M. Auli *et al.* "*Joint Language and Translation Modeling with Recurrent Neural Networks*". In: *EMNLP*, **2013**.

[34] L. Liu *et al.* "*Additive Neural Networks for Statistical Machine Translation*". In: *ACL*, **2013**.

[35] J. Devlin *et al.* "*Fast and Robust Neural Network Joint Models for Statistical Machine Translation*". In: *ACL*, **2014**.

[36] M. Sundermeyer *et al.* "*Translation Modeling with Bidirectional Recurrent Neural Networks*". In: *EMNLP*, **2014**.

[37] F. Meng *et al.* "*Encoding Source Language with Convolutional Neural Network for Machine Translation*". In: *ACL*, **2015**.

[38] S. Huang *et al.* "*Non-linear Learning for Statistical Machine Translation*". In: *ACL*, **2015**.

[39] A. V. Miceli Barone and G. Attardi. "*Non-projective Dependency-based Pre-Reordering with Recurrent Neural Network for Machine Translation*". In: *ACL*, **2015**.

[40] H. Yu and X. Zhu. "*Recurrent Neural Network based Rule Sequence Model for Statistical Machine Translation*". In: *ACL*, **2015**.

[41] J. Zhang *et al.* "*Learning Word Reorderings for Hierarchical Phrase-based Statistical Machine Translation*". In: *ACL*, **2015**.

[42] B. Hu *et al.* "*Context-Dependent Translation Selection Using Convolutional Neural Network*". In: *ACL*, **2015**.

[43] D. Bahdanau, K. Cho and Y. Bengio. "*Neural Machine Translation by Jointly Learning to Align and Translate*". In: *ICLR*, **2015**.

[44] I. Sutskever, O. Vinyals and Q. V. Le. "*Sequence to sequence learning with neural networks*". In: *NIPS*, **2014**.

[45] S. Jean *et al.* "*On Using Very Large Target Vocabulary for Neural Machine Translation*". In: *ACL*, **2015**.

[46] T. Luong *et al.* "*Addressing the Rare Word Problem in Neural Machine Translation*". In: *ACL*, **2015**.

[47] J. Turian, L.-A. Ratinov and Y. Bengio. "*Word Representations: A Simple and General Method for Semi-Supervised Learning*". In: *ACL*, **2010**.

[48] A. Klementiev, I. Titov and B. Bhattarai. "*Inducing crosslingual distributed representations of words*". **2012**.

[49] S. Gouws, Y. Bengio and G. Corrado. "*Bilbowa: Fast bilingual distributed representations without word alignments*". In: *ICML*, **2015**.

[50] R. Wang *et al.* "*Converting Continuous-Space Language Models into N-Gram Language Models for Statistical Machine Translation*". In: *EMNLP*, **2013**.

[51] R. Wang *et al.* "*Bilingual Continuous-Space Language Model Growing for Statistical Machine Translation*". *IEEE/ACM Trans. on ASLP*, **2015**.

[52] R. Wang *et al.* "*Converting Continuous-Space Language Models into N-gram Language Models with Efficient Bilingual Pruning for Statistical Machine Translation*". *ACM Trans. on ALLIP*, **2016**.

[53] R. Wang *et al.* "*Neural Network Based Bilingual Language Model Growing for Statistical Machine Translation*". In: *EMNLP*, **2014**.

[54] R. Wang *et al.* "*A Bilingual Graph-based Semantic Model for Statistical Machine Translation*". In: *IJCAI*, **2016**.

[55] D. Chiang. "*A Hierarchical Phrase-based Model for Statistical Machine Translation*". In: *ACL*, **2005**.

[56] K. Yamada and K. Knight. "*A syntax-based statistical translation model*". In: *ACL*, **2001**.

[57] S. F. Chen and J. Goodman. "*An empirical study of smoothing techniques for language modeling*". In: *ACL*, **1996**.

[58] S. F. Chen and J. Goodman. "*An Empirical Study of Smoothing Techniques for Language Modeling*". *Computer Speech & Language*, **1999**.

[59] A. Stolcke. "*SRILM-an extensible language modeling toolkit*". In: *ICSLP*, **2002**.

[60] H. Schwenk, D. Dchelotte and J.-L. Gauvain. "*Continuous space language models for statistical machine translation*". In: *COLING/ACL*, **2006**.

[61] L. H. Son *et al.* "*Training continuous space language models: some practical issues*". In: *EMNLP*, **2010**.

[62] H. Schwenk, A. Rousseau and M. Attik. "*Large, pruned or continuous space language models on a GPU for statistical machine translation*". In: *NAACL-HLT*, **2012**.

[63] L. H. Son, A. Allauzen and F. Yvon. "*Continuous space translation models with neural networks*". In: *NAACL*, **2012**.

[64] J. Niehues and A. Waibel. "*Continuous Space Language Models using Restricted Boltzmann Machines*". In: *IWSLT*, **2012**.

[65] H. Schwenk. "*Continuous-Space Language Models for Statistical Machine Translation*". *The Prague Bulletin of Mathematical Linguistics*, **2010**.

[66] A. Mnih and G. E. Hinton. "*A Scalable Hierarchical Distributed Language Model*". In: *NIPS*, **2008**.

[67] S. Hochreiter and J. Schmidhuber. "*Long short-term memory*". *Neural computation*, **1997**.

[68] T. K. Moon. "*The expectation-maximization algorithm*". *Signal processing magazine, IEEE*, **1996**.

[69] H. Schwenk. "*Continuous Space Translation Models for Phrase-Based Statistical Machine Translation*". In: *COLING*, **2012**.

[70] K. Knight. "*Decoding Complexity in Word-replacement Translation Models*". *Computational Linguistics*, **1999**.

[71] P. Koehn. "*Pharaoh: a beam search decoder for phrase-based statistical machine translation models*". In: *Machine translation: From real users to research*, **2004**.

[72] K. Papineni *et al.* "*BLEU: a method for automatic evaluation of machine translation*". In: *ACL*, **2002**.

[73] M. Snover *et al.* "*A study of translation edit rate with targeted human annotation*". In: *AMTA*, **2006**.

[74] A. Lavie and A. Agarwal. "*METEOR: An Automatic Metric for MT Evaluation with Improved Correlation with Human Judgments*". In: *ACL Workshop*, **2005**.

[75] M. Denkowski and A. Lavie. "*Meteor Universal: Language Specific Translation Evaluation for Any Target Language*". In: *EACL Workshop*, **2014**.

[76] T. Brants *et al.* "*Large Language Models in Machine Translation*". In: *EMNLP*, **2007**.

[77] Z. Huang, J. Devlin and S. Matsoukas. "*BBN's Systems for the Chinese-English Sub-task of the NTCIR-10 PatentMT evaluation*". In: *NTCIR*, **2013**.

[78] T. Mikolov *et al.* "*Strategies for training large scale neural network language models*". In: *ICASSP*, **2011**.

[79] A. Vaswani *et al.* "*Decoding with Large-Scale Neural Language Models Improves Translation*". In: *EMNLP*, **2013**.

[80] V. Nair and G. E. Hinton. "*Rectified Linear Units Improve Restricted Boltzmann Machines*". In: *ICML*, **2010**.

[81] M. Gutmann and A. Hyvärinen. "*Noise-Contrastive Estimation: A New Estimation Principle for Unnormalized Statistical Models*". In: *AISTATS*, **2010**.

[82] N. Kalchbrenner and P. Blunsom. "*Recurrent Continuous Translation Models*". In: *EMNLP*, **2013**.

[83] S. Liu *et al.* "*A Recursive Recurrent Neural Network for Statistical Machine Translation*". In: *ACL*, **2014**.

[84] X. Peng and D. Gildea. "*Type-based MCMC for Sampling Tree Fragments from Forests*". In: *EMNLP*, **2014**.

[85] P. Li *et al.* "*A Neural Reordering Model for Phrase-based Translation*". In: *COLING*, **2014**.

[86] S. Chandar *et al.* "*An Autoencoder Approach to Learning Bilingual Word Representations*". *CoRR*, **2014**.

[87] A. Deoras *et al.* "*Variational approximation of long-span language models for lvcsr*". In: *ICASSP*, **2011**.

[88] E. Arısoy *et al.* "*Converting Neural Network Language Models Into Back-off Language Models For Efficient Decoding In Automatic Speech Recognition*". In: *ICASSP*, **2013**.

[89] E. Arsoy *et al.* "*Converting Neural Network Language Models into Back-off Language Models for Efficient Decoding in Automatic Speech Recognition*". *IEEE/ACM Trans. on ASLP*, **2014**.

[90] A. Stolcke. "*Entropy-based Pruning of Backoff Language Models*". In: *Broadcast News Workshop*, **1998**.

[91] A. Stolcke *et al.* "*SRILM at sixteen: Update and outlook*". In: *ASRU*, **2011**.

[92] K. Heafield *et al.* "*Scalable Modified Kneser-Ney Language Model Estimation*". In: *ACL*, **2013**.

[93] P. Koehn and J. Schroeder. "*Experiments in Domain Adaptation for Statistical Machine Translation*". In: *WMT*, **2007**.

[94] A. Axelrod, X. He and J. Gao. "*Domain Adaptation via Pseudo In-Domain Data Selection*". In: *EMNLP*, **2011**.

[95] P. Banerjee *et al.* "*Translation Quality-Based Supplementary Data Selection by Incremental Update of Translation Models*". In: *COLING*, **2012**.

[96] K. Duh *et al.* "*Adaptation Data Selection using Neural Language Models: Experiments in Machine Translation*". In: *ACL*, **2013**.

[97] C. Hoang and K. Sima'an. "*Latent Domain Phrase-based Models for Adaptation*". In: *EMNLP*, **2014**.

[98] C. Hoang and K. Sima'an. "*Latent Domain Translation Models in Mix-of-Domains Haystack*". In: *COLING*, **2014**.

[99] J. R. Bellegarda. "*Statistical language model adaptation: review and perspectives*". *Speech Communication*, **2004**.

[100] Y. Deng, J. Xu and Y. Gao. "*Phrase Table Training for Precision and Recall: What Makes a Good Phrase and a Good Phrase Pair?*" In: *ACL*, **2008**.

[101] G. Foster, C. Goutte and R. Kuhn. "*Discriminative Instance Weighting for Domain Adaptation in Statistical Machine Translation*". In: *EMNLP*, **2010**.

[102] A. Bisazza *et al.* "*Fill-up versus interpolation methods for phrase-based SMT adaptation.*" In: *IWSLT*, **2011**.

[103] R. Sennrich. "*Perplexity Minimization for Translation Model Domain Adaptation in Statistical Machine Translation*". In: *EACL*, **2012**.

[104] S. Mansour and H. Ney. "*Phrase Training Based Adaptation for Statistical Machine Translation*". In: *NAACL*, **2013**.

[105] M. Carpuat *et al.* "*SenseSpotting: Never let your parallel data tie you to an old domain*". In: *ACL*, **2013**.

[106] B. Chen, R. Kuhn and G. Foster. "*Vector Space Model for Adaptation in Statistical Machine Translation*". In: *ACL*, **2013**.

[107] B. Chen, G. Foster and R. Kuhn. "*Adaptation of Reordering Models for Statistical Machine Translation*". In: *NAACL*, **2013**.

[108] R. Sennrich, H. Schwenk and W. Aransa. "*A Multi-Domain Translation Model Framework for Statistical Machine Translation*". In: *ACL*, **2013**.

[109] P. Mathur, S. Venkatapathy and N. Cancedda. "*Fast Domain Adaptation of SMT models without in-Domain Parallel Data*". In: *COLING*, **2014**.

[110] Y. Shi, M. Larson and C. M. Jonker. "*Recurrent neural network language model adaptation with curriculum learning*". *Computer Speech and Language*, **2015**.

[111] S. Joty *et al.* "*How to Avoid Unwanted Pregnancies: Domain Adaptation using Neural Network Models*". In: *EMNLP*, **2015**.

[112] N. Durrani *et al.* "*Using Joint Models for Domain Adaptation in Statistical Machine Translation*". *MT Summit*, **2015**.

[113] J. Gao and M. Zhang. "*Improving language model size reduction using better pruning criteria*". In: *ACL*, **2002**.

[114] V. Siivola, T. Hirsimäki and S. Virpioja. "*On Growing and Pruning Kneser-Ney Smoothed N-Gram Models*". *IEEE Trans. on ASLP*, **2007**.

[115] F. Jelinek. "*Self-organized Language Modeling for Speech Recognition*". In: *Readings in Speech Recognition*, **1990**.

[116] R. Jonson. "*Generating Statistical Language Models from Interpretation Grammars in Dialogue Systems*". In: *Proceedings of 11th Conference of the European Association of Computational Linguistics*, **2006**.

[117] P. Koehn and J. Schroeder. "*Experiments in Domain Adaptation for Statistical Machine Translation*". In: *WMT*, **2007**.

[118] J. Zhang and C. Zong. "*Learning a Phrase-based Translation Model from Monolingual Data with Application to Domain Adaptation*". In: *ACL*, **2013**.

[119] M. Cettolo, C. Girardi and M. Federico. "*WIT^3: Web Inventory of Transcribed and Translated Talks*". In: *EMTA*, **2012**.

[120] E. S. Ristad and R. G. Thomas. "*New Techniques for Context Modeling*". In: *ACL*, **1995**.

[121] T. Niesler and P. Woodland. "*A variable-length category-based n-gram language model*". In: *ICASSP*, **1996**.

[122] M. Siu and M. Ostendorf. "*Variable n-grams and extensions for conversational speech language modeling*". *IEEE Trans. on ASLP*, **2000**.

[123] O. Levy and Y. Goldberg. "*Linguistic Regularities in Sparse and Explicit Word Representations*". In: *CoNLL*, **2014**.

[124] I. Vulić and M.-F. Moens. "*Bilingual Word Embeddings from Non-Parallel Document-Aligned Data Applied to Bilingual Lexicon Induction*". In: *ACL*, **2015**.

[125] S. Ploux and H. Ji. "*A Model for Matching Semantic Maps Between Languages (French/English, English/French)*". *Computational Linguistics*, **2003**.

[126] H. Ji and S. Ploux. "*Lexical Knowledge Representation with Contexonyms*". In: *WMT*, **2003**.

[127] A. Saluja *et al.* "*Graph-based Semi-Supervised Learning of Translation Models from Monolingual Data*". In: *ACL*, **2014**.

[128] J. Zhang *et al.* "*Bilingually-constrained Phrase Embeddings for Machine Translation*". In: *ACL*, **2014**.

[129] J. Pennington, R. Socher and C. Manning. "*Glove: Global Vectors for Word Representation*". In: *EMNLP*, **2014**.

[130] T. Shi *et al.* "*Learning Cross-lingual Word Embeddings via Matrix Co-factorization*". In: *ACL*, **2015**.

[131] M. Faruqui and C. Dyer. "*Improving Vector Space Word Representations Using Multilingual Correlation*". In: *EACL*, **2014**.

[132] A. Lu *et al.* "*Deep Multilingual Correlation for Improved Word Embeddings*". In: *NAACL*, **2015**.

[133] J. A. Bondy and U. S. R. Murty. *Graph theory with applications*. Macmillan London, **1976**.

[134] R. Luce and A. Perry. "*A method of matrix analysis of group structure*". *Psychometrika*, **1949**.

[135] R. M. Karp. *Reducibility Among Combinatorial Problems - Complexity of Computer Computations*. Plenum Press, **1972**.

[136] H. O. Hirschfeld. "*A connection between correlation and contingency*". In: *Mathematical Proceedings of the Cambridge Philosophical Society*, **1935**.

[137] J.-P. Benzécri. "*L'Analyse des Correspondances*". In: *L'Analyse des Données*. Dunod Paris, **1973**.

[138] J. H. Ward Jr. "*Hierarchical grouping to optimize an objective function*". *Journal of the American statistical association*, **1963**.

[139] F. J. Och. "*Minimum Error Rate Training in Statistical Machine Translation*". In: *ACL*, **2003**.

[140] I. Goto *et al.* "*Overview of the Patent Machine Translation Task at the NTCIR-9 Workshop*". In: *NTCIR-9 Workshop*, **2011**.

[141] F. J. Och and H. Ney. "*A systematic comparison of various statistical alignment models*". *Computational Linguistics*, **2003**.

[142] A. Fujii *et al.* "*Overview of the patent translation task at the NTCIR-8 workshop*". In: *NTCIR Workshop*, **2010**.

[143] P. Koehn. "*Statistical Significance Tests for Machine Translation Evaluation*". In: *EMNLP*, **2004**.

[144] C. Chelba *et al.* "*Study on Interaction between Entropy Pruning and Kneser-Ney Smoothing*". In: *INTERSPEECH*, **2010**.

[145] B. Roark, C. Allauzen and M. Riley. "*Smoothed Marginal Distribution Constraints for Language Modeling*". In: *ACL*, **2013**.

[146] O. F. Zaidan. "*Z-MERT: A Fully Configurable Open Source Tool for Minimum Error Rate Training of Machine Translation Systems*". *The Prague Bulletin of Mathematical Linguistics*, **2009**.

[147] X. Wang *et al.* "*Empirical Study of Unsupervised Chinese Word Segmentation Methods for SMT on Large-scale Corpora*". In: *ACL*, **2014**.

[148] P. Koehn *et al.* "*Moses: Open Source Toolkit for Statistical Machine Translation*". In: *ACL*, **2007**.

[149] H. Johnson *et al.* "*Improving Translation Quality by Discarding Most of the Phrasetable*". In: *EMNLP*, **2007**.

[150] H. Schwenk, A. Rousseau and M. Attik. "*Large, Pruned or Continuous Space Language Models on a GPU for Statistical Machine Translation*". In: *NAACL-HLT Workshop*, **2012**.

[151] T. Nakazawa *et al.* "*Overview of the 1st Workshop on Asian Translation*". In: *WAT*, **2014**.

[152] J. Coulmance *et al.* "*Trans-gram, Fast Cross-lingual Word-embeddings*". In: *EMNLP*, **2015**.

[153] C. Xing *et al.* "*Normalized Word Embedding and Orthogonal Transform for Bilingual Word Translation*". In: *NAACL*, **2015**.

[154] M. Galley *et al.* "*NIST Open Machine Translation 2008 Evaluation: Stanford University's System Description*". In: *NIST OpenMT*, **2008**.

[155] D. Cer *et al.* "*Phrasal: A Statistical Machine Translation Toolkit for Exploring New Model Features*". In: *NAACL*, **2010**.