Learning Continuous Phrase Representations for Translation Modeling

Jianfeng Gao Xiaodong He Wen-tau Yih Li Deng
Microsoft Research
One Microsoft Way
Redmond, WA 98052, USA
{jfgao,xiaohe,scottyih,deng}@microsoft.com

Abstract

This paper tackles the sparsity problem in estimating phrase translation probabilities by learning continuous phrase representations, whose distributed nature enables the sharing of related phrases in their representations. A pair of source and target phrases are projected into continuous-valued vector representations in a low-dimensional latent space, where their translation score is computed by the distance between the pair in this new space. The projection is performed by a neural network whose weights are learned on parallel training data. Experimental evaluation has been performed on two WMT translation tasks. Our best result improves the performance of a state-of-the-art phrase-based statistical machine translation system trained on WMT 2012 French-English data by up to 1.3 BLEU points.

1 Introduction

The phrase translation model, also known as the *phrase table*, is one of the core components of phrase-based statistical machine translation (SMT) systems. The most common method of constructing the phrase table takes a two-phase approach (Koehn et al. 2003). First, the bilingual phrase pairs are extracted heuristically from an automatically word-aligned training data. The second phase, which is the focus of this paper, is parameter estimation where each phrase pair is assigned with some scores that are estimated based on counting these phrases or their words using the same word-aligned training data.

Phrase-based SMT systems have achieved state-of-the-art performance largely due to the fact that long phrases, rather than single words, are used as translation units so that useful context information can be captured in selecting translations. However, longer phrases occur less often in training data, leading to a severe data sparseness problem in parameter estimation. There has been a plethora of research reported in the literature on improving parameter estimation for the phrase translation model (e.g., DeNero et al. 2006; Wuebker et al. 2010; He and Deng 2012; Gao and He 2013).

This paper revisits the problem of scoring a phrase translation pair by developing a Continuous-space Phrase Translation Model (CPTM). The translation score of a phrase pair in this model is computed as follows. First, we represent each phrase as a bag-of-words vector, called *word vector* henceforth. We then project the word vector, in either the source language or the target language, into a respective continuous feature vector in a common low-dimensional space that is language independent. The projection is performed by a multi-layer neural network. The projected feature vector forms the *continuous representation* of a phrase. Finally, the translation score of a source-target phrase pair is computed by the distance between their feature vectors.

The main motivation behind the CPTM is to alleviate the data sparseness problem associated with the traditional counting-based methods by grouping phrases with a similar meaning across different languages. This style of grouping is made possible because of the distributed nature of the continuous-space representations for phrases. No such sharing was possible in the original symbolic space for representing words or phrases. In this model, semantically or grammatically related phrases, in both the source and the target languages, would tend to have similar (close) feature vectors in the continuous space, guided by the training objective. Since the translation score is a smooth function of these feature vectors, a small

change in the features should only lead to a small change in the translation score.

The primary research task in developing the CPTM is learning the continuous representation of a phrase that is effective for SMT. Motivated by recent studies on continuous-space language models (e.g., Bengio et al. 2003; Mikolov et al. 2011; Schwenk et al., 2012), we use a neural network to project a word vector to a feature vector. Ideally, the projection would discover those latent features that are useful to differentiate *good* translations from *bad* ones, for a given source phrase. However, there is no training data with explicit annotation on the quality of phrase translations. The phrase translation pairs are *hidden* in the parallel source-target sentence pairs, which are used to train the traditional translation models. The quality of a phrase translation can only be judged implicitly through the translation quality of the sentences, as measured by BLEU, which contain the phrase pair. In order to overcome this challenge and let the BLEU metric guide the projection learning, we propose a new method to learn the parameters of a neural network. This new method, via the choice of an appropriate objective function in training, automatically forces the feature vector of a source phrase to be closer to the feature vectors of its candidate translations. As a result, the BLEU score is improved when these translations are selected by an SMT decoder to produce final, sentence-level translations. The new learning method makes use of the L-BFGS algorithm and the expected BLEU as the objective function defined on N-best lists.

To the best of our knowledge, the CPTM proposed in this paper is the first continuous-space phrase translation model that makes use of joint representations of a phrase in the source language and its translation in the target language (to be detailed in Section 4) and that is shown to lead to significant improvement over a standard phrase-based SMT system (to be detailed in Section 6).

Like the traditional phrase translation model, the translation score of each bilingual phrase pair is modeled explicitly in our model. However, instead of estimating the phrase translation score on aligned parallel data, our model intends to capture the grammatical and semantic similarity between a source phrase and its paired target phrase by projecting them into a common, continuous space that is language independent.

The rest of the paper is organized as follows. Section 2 reviews previous work. Section 3 reviews the log-linear model for phrase-based SMT and Sections 4 presents the CPTM. Section 5 describes the way the model parameters are estimated, followed by the experimental results in Section 6. Finally, Section 7 concludes the paper.

2 Related Work

Representations of words or documents as continuous vectors have a long history. Most of the earlier latent semantic models for learning such vectors are designed for information retrieval (Deerwester et al. 1990; Hofmann 1999; Blei et al. 2003). In contrast, recent work on continuous space language models, which estimate the probability of a word sequence in a continuous space (Bengio et al. 2003; Mikolov et al. 2010), have advanced the state of the art in language modeling, outperforming the traditional n-gram model on speech recognition (Mikolov et al. 2012; Sundermeyer et al. 2013) and machine translation (Mikolov 2012; Auli et al. 2013).

Because these models are developed for monolingual settings, word embedding from these models is not directly applicable to translation. As a result, variants of such models for cross-lingual scenarios have been proposed so that words in different languages are projected into the shared latent vector space (Dumais et al. 1997; Platt et al. 2010; Vinokourov et al. 2002; Yih et al. 2011; Gao et al. 2011; Huang et al. 2013; Zou et al. 2013). In principle, a phrase table can be derived using any of these cross-lingual models, although decoupling the derivation from the SMT training often results in suboptimal performance (e.g., measured in BLEU), as we will show in Section 6.

Recently, there is growing interest in applying continuous-space models for translation. The most related to this study is the work of continuous space n-gram translation models (Schwenk et al. 2007; Schwenk 2012; Son et al. 2012), where the feed-forward neural network language model is extended to represent translation probabilities. However, these earlier studies focused on the *n-gram translation models*, where the translation probability of a phrase or a sentence is decomposed as a product of n-gram probabilities as in a standard n-gram language model. Therefore, it is not clear how their approaches can be applied to the phrase translation model[1], which is much more

[1] Niehues et al. (2011) use different translation units in order to integrate the n-gram translation model into the phrase-based approach. However, it is not clear how a continuous version of such a model can be trained efficiently because the factor models used by Son et al. cannot be applied directly.

widely used in modern SMT systems. In contrast, our model learns jointly the representations of a phrase in the source language as well as its translation in the target language. The recurrent continuous translation models proposed by Kalchbrenner and Blunsom (2013) also adopt the recurrent language model (Mikolov et al. 2010). But unlike the n-gram translation models above, they make no Markov assumptions about the dependency of the words in the target sentence. Continuous space models have also been used for generating translations for new words (Mikolov et al. 2013a) and ITG reordering (Li et al. 2013).

There has been a lot of research on improving the phrase table in phrase-based SMT (Marcu and Wong 2002; Lamber and Banchs 2005; Denero et al. 2006; Wuebker et al. 2010; Zhang et al., 2011; He and Deng 2012; Gao and He 2013). Among them, (Gao and He 2013) is most relevant to the work described in this paper. They estimate phrase translation probabilities using a discriminative training method under the N-best reranking framework of SMT. In this study we use the same objective function to learn the continuous representations of phrases, integrating the strengths associated with these earlier studies.

3 The Log-Linear Model for SMT

Phrase-based SMT is based on a log-linear model which requires learning a mapping between input $F \in \mathcal{F}$ to output $E \in \mathcal{E}$. We are given

- Training samples (F_i, E_i) for $i = 1 \ldots N$, where each source sentence F_i is paired with a reference translation in target language E_i;
- A procedure GEN to generate a list of N-best candidates $\mathrm{GEN}(F_i)$ for an input F_i, where GEN in this study is the baseline phrase-based SMT system, i.e., an in-house implementation of the Moses system (Koehn et al. 2007) that does not use the CPTM, and each $E \in \mathrm{GEN}(F_i)$ is labeled by the sentence-level BLEU score (He and Deng 2012), denoted by $\mathrm{sBleu}(E_i, E)$, which measures the quality of E with respect to its reference translation E_i;
- A vector of features $\mathbf{h} \in \mathbb{R}^M$ that maps each (F_i, E) to a vector of feature values[2]; and
- A parameter vector $\boldsymbol{\lambda} \in \mathbb{R}^M$, which assigns a real-valued weight to each feature.

The components $\mathrm{GEN}(.)$, $\mathbf{h}$ and $\boldsymbol{\lambda}$ define a log-linear model that maps F_i to an output sentence as follows:

$$E^* = \underset{(E,A)\in \mathrm{GEN}(F_i)}{\operatorname{argmax}} \boldsymbol{\lambda}^{\mathrm{T}} \mathbf{h}(F_i, E, A) \qquad (1)$$

which states that given $\boldsymbol{\lambda}$ and $\mathbf{h}$, argmax returns the highest scoring translation E^*, maximizing over correspondences A. In phrase-based SMT, A consists of a segmentation of the source and target sentences into phrases and an alignment between source and target phrases. Since computing the argmax exactly is intractable, it is commonly performed approximatedly by beam search (Och and Ney 2004). Following Liang et al. (2006), we assume that every translation candidate is always coupled with a corresponding A, called the *Viterbi derivation*, generated by (1).

4 A Continuous-Space Phrase Translation Model (CPTM)

The architecture of the CPTM is shown in Figures 1 and 2, where for each pair of source and target phrases (f_i, e_j) in a source-target sentence pair, we first project them into feature vectors $\mathbf{y}_{f_i}$ and $\mathbf{y}_{e_j}$ in a latent, continuous space via a neural network with one hidden layer (as shown in Figure 2), and then compute the translation score, $\mathrm{score}(f_i, e_j)$, by the distance of their feature vectors in that space.

We start with a bag-of-words representation of a phrase $\mathbf{x} \in \mathbb{R}^d$, where $\mathbf{x}$ is a word vector and d is the size of the vocabulary consisting of words in both source and target languages, which is set to 200K in our experiments. We then learn to project $\mathbf{x}$ to a low-dimensional continuous space $\mathbb{R}^k$:

$$\phi(\mathbf{x}): \mathbb{R}^d \rightarrow \mathbb{R}^k$$

The projection is performed using a fully connected neural network with one hidden layer and tanh activation functions. Let $\mathbf{W}_1$ be the projection matrix from the input layer to the hidden layer and $\mathbf{W}_2$ the projection matrix from the hidden layer to the output layer, we have

$$\mathbf{y} \equiv \phi(\mathbf{x}) = \tanh\left(\mathbf{W}_2{}^{\mathrm{T}}\left(\tanh\left(\mathbf{W}_1{}^{\mathrm{T}}\mathbf{x}\right)\right)\right) \qquad (2)$$

[2] Our baseline system uses a set of standard features suggested in Koehn et al. (2007), which is also detailed in Section 6.

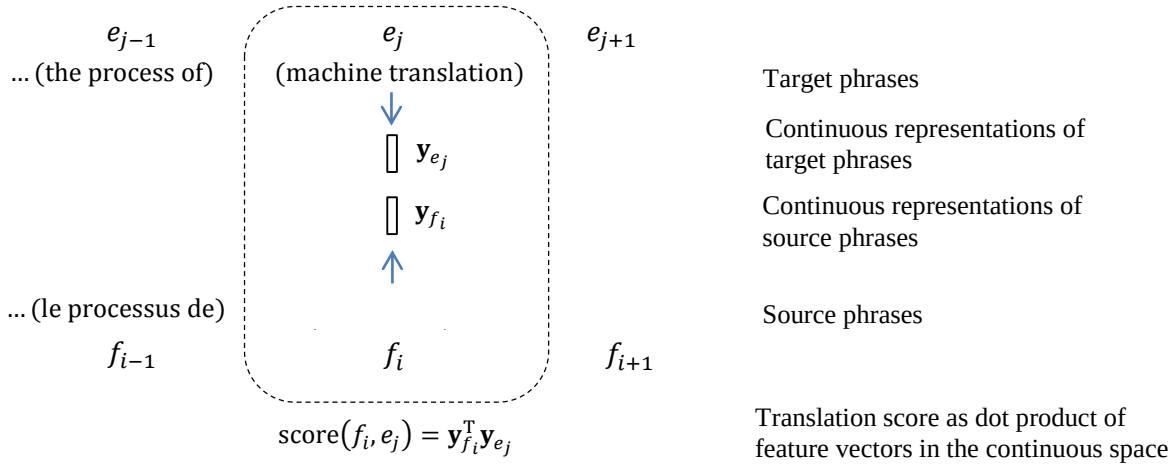

Figure 1. The architecture of the CPTM, where the mapping from a phrase to its continuous representation is shown in Figure 2.

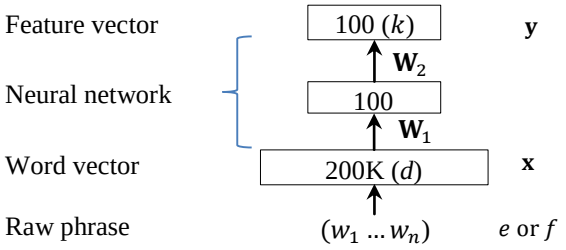

Figure 2. A neural network model for phrases giving rise to their continuous representations. The model with the same form is used for both source and target languages.

The translation score of a source phrase f and a target phrase e can be measured as the similarity (or distance) between their feature vectors. We choose the dot product as the similarity function[3]:

$$\text{score}(f, e) \equiv \text{sim}_{\boldsymbol{\theta}}(\mathbf{x}_f, \mathbf{x}_e) = \mathbf{y}_f^{\mathrm{T}} \mathbf{y}_e \qquad (3)$$

According to (2), we see that the value of the scoring function is determined by the projection matrices $\boldsymbol{\theta} = \{\mathbf{W}_1, \mathbf{W}_2\}$.

The CPTM of (2) and (3) can be incorporated into the log-linear model for SMT (1) by introducing a new feature h_{M+1} and a new feature weight λ_{M+1}. The new feature is defined as

$$h_{M+1}(F_i, E, A) = \sum_{(f,e)\in A} \text{sim}_{\boldsymbol{\theta}}(\mathbf{x}_f, \mathbf{x}_e) \qquad (4)$$

Thus, the phrase-based SMT system, into which the CPTM is incorporated, is parameterized by $(\boldsymbol{\lambda}, \boldsymbol{\theta})$, where $\boldsymbol{\lambda}$ is a vector of a handful of parameters used in the log-linear model of (1), with one weight for each feature; and $\boldsymbol{\theta}$ is the projection matrices used in the CPTM defined by (2) and (3). In our experiments we take three steps to learn $(\boldsymbol{\lambda}, \boldsymbol{\theta})$:

1. We use a baseline phrase-based SMT system to generate for each source sentence in training data an N-best list of translation hypotheses[4].
2. We set $\boldsymbol{\lambda}$ to that of the baseline system and let $\lambda_{M+1} = 1$, and optimize $\boldsymbol{\theta}$ w.r.t. a loss function on training data[5].
3. We fix $\boldsymbol{\theta}$, and optimize $\boldsymbol{\lambda}$ using MERT (Och 2003) to maximize BLEU on dev data.

In the next section, we will describe Step 2 in detail as it is directly related to the CPTM training.

[3] In our experiments, we compare dot product and the cosine similarity functions and find that the former works better for nonlinear multi-layer neural networks, and the latter works better for linear neural networks. For the sake of clarity, we choose dot product when we describe the CPTM and its training in Sections 4 and 5, respectively.

[4] The baseline SMT needs to be reasonably good in the sense that the oracle BLEU score on the generated n-best lists needs to be significantly higher than that of the top-1 translations so that the CPTM can be effectively trained.

[5] The initial value of λ_{M+1} can also be tuned using the dev set. However, we find in a pilot study that it is good enough to set it to 1 when the values of all the baseline feature weights, used in the log-linear model of (1), are properly normalized, such as by setting $\lambda_m = \lambda_m / C$ for $m = 1 \dots M$, where C is the unnormalized weight value of the target language model.

5 Training CPTM

This section describes the loss function we employ with the CPTM and the algorithm to train the neural network weights.

We define the loss function $\mathcal{L}(\boldsymbol{\theta})$ as the negative of the N-best list based expected BLEU, denoted by $\text{xBleu}(\boldsymbol{\theta})$. In the reranking framework of SMT outlined in Section 3, $\text{xBleu}(\boldsymbol{\theta})$ over one training sample (F_i, E_i) is defined as

$$\text{xBleu}(\boldsymbol{\theta}) = \sum_{E \in \text{GEN}(F_i)} P(E|F_i)\text{sBleu}(E_i, E) \quad (5)$$

where $\text{sBleu}(E_i, E)$ is the sentence-level BLEU score, and $P(E|F_i)$ is the translation probability from F_i to E computed using *softmax* as

$$P(E|F_i) = \frac{\exp\left(\gamma \boldsymbol{\lambda}^{\mathrm{T}} \mathbf{h}(F_i, E, A)\right)}{\sum_{E' \in \text{GEN}(F_i)} \exp\left(\gamma \boldsymbol{\lambda}^{\mathrm{T}} \mathbf{h}(F_i, E', A)\right)} \quad (6)$$

where $\boldsymbol{\lambda}^{\mathrm{T}}\mathbf{h}$ is the log-linear model of (1), which also includes the feature derived from the CPTM as defined by (4), and γ is a tuned smoothing factor.

Let $\mathcal{L}(\boldsymbol{\theta})$ be a loss function which is differentiable w.r.t. the parameters of the CPTM, $\boldsymbol{\theta}$. We can compute the gradient of the loss and learn $\boldsymbol{\theta}$ using gradient-based numerical optimization algorithms, such as L-BFGS or stochastic gradient descent (SGD).

5.1 Computing the Gradient

Since the loss does not explicitly depend on $\boldsymbol{\theta}$, we use the chain rule for differentiation:

$$\frac{\partial \mathcal{L}(\boldsymbol{\theta})}{\partial \boldsymbol{\theta}} = \sum_{(f,e)} \frac{\partial \mathcal{L}(\boldsymbol{\theta})}{\partial \text{sim}_{\boldsymbol{\theta}}(\mathbf{x}_f, \mathbf{x}_e)} \frac{\partial \text{sim}_{\boldsymbol{\theta}}(\mathbf{x}_f, \mathbf{x}_e)}{\partial \boldsymbol{\theta}}$$

$$= \sum_{(f,e)} -\delta_{(f,e)} \frac{\partial \text{sim}_{\boldsymbol{\theta}}(\mathbf{x}_f, \mathbf{x}_e)}{\partial \boldsymbol{\theta}} \quad (7)$$

which takes the form of summation over all phrase pairs occurring either in a training sample (stochastic mode) or in the entire training data (batch mode). $\delta_{(f,e)}$ in (7) is known as the *error* term of the phrase pair (f, e), and is defined as

$$\delta_{(f,e)} = -\frac{\partial \mathcal{L}(\boldsymbol{\theta})}{\partial \text{sim}_{\boldsymbol{\theta}}(\mathbf{x}_f, \mathbf{x}_e)} \quad (8)$$

It describes how the overall loss changes with the translation score of the phrase pair (f, e). We will leave the derivation of $\delta_{(f,e)}$ to Section 5.1.2, and will first describe how the gradient of $\text{sim}_{\boldsymbol{\theta}}(\mathbf{x}_f, \mathbf{x}_e)$ w.r.t. $\boldsymbol{\theta}$ is computed.

5.1.1 Computing $\partial \text{sim}_{\boldsymbol{\theta}}(\mathbf{x}_f, \mathbf{x}_e)/\partial \boldsymbol{\theta}$

Without loss of generality, we use the following notations to describe a neural network:

- $\mathbf{W}_l$ is the projection matrix for the *l*-th layer of the neural network;
- $\mathbf{x}$ is the input word vector of a phrase;
- $\mathbf{z}^l$ is the sum vector of the *l*-th layer; and
- $\mathbf{y}^l = \sigma(\mathbf{z}^l)$ is the output vector of the *l*-th layer, where σ is an activation function;

Thus, the CPTM defined by (2) and (3) can be represented as

$$\mathbf{z}^1 = {\mathbf{W}_1}^{\mathrm{T}}\mathbf{x}$$
$$\mathbf{y}^1 = \sigma(\mathbf{z}^1)$$
$$\mathbf{z}^2 = {\mathbf{W}_2}^{\mathrm{T}}\mathbf{y}^1$$
$$\mathbf{y}^2 = \sigma(\mathbf{z}^2)$$
$$\text{sim}_{\boldsymbol{\theta}}(\mathbf{x}_f, \mathbf{x}_e) = \left(\mathbf{y}_f^2\right)^{\mathrm{T}}\mathbf{y}_e^2$$

The gradient of the matrix $\mathbf{W}_2$ which projects the hidden vector to the output vector is computed as:

$$\frac{\partial \text{sim}_{\boldsymbol{\theta}}(\mathbf{x}_f, \mathbf{x}_e)}{\partial \mathbf{W}_2} = \frac{\partial \left(\mathbf{y}_f^2\right)^{\mathrm{T}}}{\partial \mathbf{W}_2}\mathbf{y}_e^2 + \left(\mathbf{y}_f^2\right)^{\mathrm{T}} \frac{\partial \mathbf{y}_e^2}{\partial \mathbf{W}_2}$$

$$= \mathbf{y}_f^1 \left(\mathbf{y}_e^2 \circ \sigma'\left(\mathbf{z}_f^2\right)\right)^{\mathrm{T}} + \mathbf{y}_e^1 \left(\mathbf{y}_f^2 \circ \sigma'\left(\mathbf{z}_e^2\right)\right)^{\mathrm{T}} \quad (9)$$

where $\circ$ is the element-wise multiplication (Hadamard product). Applying the back propagation principle, the gradient of the projection matrix mapping the input vector to the hidden vector $\mathbf{W}_1$ is computed as

$$\frac{\partial \text{sim}_{\boldsymbol{\theta}}(\mathbf{x}_f, \mathbf{x}_e)}{\partial \mathbf{W}_1}$$
$$= \mathbf{x}_f \left(\mathbf{W}_2 \left(\mathbf{y}_e^2 \circ \sigma'\left(\mathbf{z}_f^2\right)\right) \circ \sigma'\left(\mathbf{z}_f^1\right)\right)^{\mathrm{T}}$$
$$+ \mathbf{x}_e \left(\mathbf{W}_2 \left(\mathbf{y}_f^2 \circ \sigma'\left(\mathbf{z}_e^2\right)\right) \circ \sigma'\left(\mathbf{z}_e^1\right)\right)^{\mathrm{T}} \quad (10)$$

The derivation can be easily extended to a neural network with multiple hidden layers.

5.1.2 Computing $\delta_{(f,e)}$

To simplify the notation, we rewrite our loss function of (5) and (6) over one training sample as

$$\mathcal{L}(\boldsymbol{\theta}) = -\mathrm{xBleu}(\boldsymbol{\theta}) = -\frac{\mathrm{G}(\boldsymbol{\theta})}{\mathrm{Z}(\boldsymbol{\theta})} \quad (11)$$

where

$$\mathrm{G}(\boldsymbol{\theta}) = \sum_E \mathrm{sBleu}(E, E_i) \exp\left(\boldsymbol{\lambda}^{\mathrm{T}} \mathbf{h}(F_i, E, A)\right)$$

$$\mathrm{Z}(\boldsymbol{\theta}) = \sum_E \exp\left(\boldsymbol{\lambda}^{\mathrm{T}} \mathbf{h}(F_i, E, A)\right)$$

Combining (8) and (11), we have

$$\delta_{(f,e)} = \frac{\partial \mathrm{xBleu}(\boldsymbol{\theta})}{\partial \mathrm{sim}_{\boldsymbol{\theta}}(\mathbf{x}_f, \mathbf{x}_e)} \quad (12)$$

$$= \frac{1}{\mathrm{Z}(\boldsymbol{\theta})} \left(\frac{\partial \mathrm{G}(\boldsymbol{\theta})}{\partial \mathrm{sim}_{\boldsymbol{\theta}}(\mathbf{x}_f, \mathbf{x}_e)} - \frac{\partial \mathrm{Z}(\boldsymbol{\theta})}{\partial \mathrm{sim}_{\boldsymbol{\theta}}(\mathbf{x}_f, \mathbf{x}_e)} \mathrm{xBleu}(\boldsymbol{\theta}) \right)$$

Because $\boldsymbol{\theta}$ is only relevant to h_{M+1} which is defined in (4), we have

$$\frac{\partial \boldsymbol{\lambda}^{\mathrm{T}} \mathbf{h}(F_i, E, A)}{\partial \mathrm{sim}_{\boldsymbol{\theta}}(\mathbf{x}_f, \mathbf{x}_e)} = \lambda_{M+1} \frac{\partial h_{M+1}(F_i, E, A)}{\partial \mathrm{sim}_{\boldsymbol{\theta}}(\mathbf{x}_f, \mathbf{x}_e)}$$

$$= \lambda_{M+1} N(f, e; A) \quad (13)$$

where $N(f, e; A)$ is the number of times the phrase pair (f, e) occurs in A. Combining (12) and (13), we end up with the following equation

$$\delta_{(f,e)} = \sum_{(E,A) \in GEN(F_i)} \mathrm{U}(\boldsymbol{\theta}, E) P(E|F_i) \lambda_{M+1} N(f, e; A) \quad (14)$$

where

$$\mathrm{U}(\boldsymbol{\theta}, E) = \mathrm{sBleu}(E_i, E) - \mathrm{xBleu}(\boldsymbol{\theta}).$$

5.2 The Training Algorithm

In our experiments we train the parameters of the CPTM, $\boldsymbol{\theta}$, using the L-BFGS optimizer described in Andrew and Gao (2007), together with the loss function described in (5). The gradient is computed as described in Sections 5.1. Although SGD has been advocated for neural network training due to its simplicity and its robustness to local minima (Bengio 2009), we find that in our task that the L-BFGS minimizes the loss in a desirable fashion empirically when iterating over the complete training data (batch mode). For example, the convergence of the algorithm was found to be smooth, despite the non-convexity in our loss. Another merit of batch training is that the gradient over all training data can be computed efficiently. As shown in Section 5.1, computing $\partial \mathrm{sim}_{\theta}(\mathrm{x}_f, \mathrm{x}_e)/\partial\theta$ requires large-scale matrix multiplications, and is expensive for multi-layer neural networks. Eq. (7) suggests that $\partial \mathrm{sim}_{\theta}(\mathrm{x}_f, \mathrm{x}_e)/\partial\theta$ and $\delta_{(f,e)}$ can be computed separately, thus making the computation cost of the former term only depends on the number of phrase pairs in the phrase table, but not the size of training data. Therefore, the training method described here can be used on larger amounts of training data with little difficulty.

As described in Section 4, we take three steps to learn the parameters for both the log-linear model of SMT and the CPTM. While steps 1 and 3 can be easily parallelized on a computer cluster, the CPTM training is performed on a single machine. For example, given a phrase table containing 16M pairs and a 1M-sentence training set, it takes a couple of hours to generate the N-best lists on a cluster, and about 10 hours to train the CPTM on a Xeon E5-2670 2.60GHz machine.

For a non-convex problem, model initialization is important. In our experiments we always initialize $\mathbf{W}_1$ using a bilingual topic model trained on parallel data (see detail in Section 6.2), and $\mathbf{W}_2$ as an identity matrix. In principle, the loss function of (5) can be further regularized (e.g. by adding a term of L_2 norm) to deal with overfitting. However, we did not find clear empirical advantage over the simpler early stop approach in a pilot study, which is adopted in the experiments in this paper.

6 Experiments

This section evaluates the CPTM presented on two translation tasks using WMT data sets. We first describe the data sets and baseline setup. Then we present experiments where we compare different versions of the CPTM and previous models.

6.1 Experimental Setup

Baseline. We experiment with an in-house phrase-based system similar to Moses (Koehn et al. 2007), where the translation candidates are scored by a set of common features including maximum likelihood estimates of source given target phrase mappings $P_{MLE}(e|f)$ and vice versa $P_{MLE}(f|e)$, as well as lexical weighting estimates $P_{LW}(e|f)$ and $P_{LW}(f|e)$, word and phrase penalties, a linear distortion feature, and a lexicalized reordering feature. The baseline includes a standard 5-gram modified Kneser-Ney language model trained on the target side of the parallel corpora described below. Log-linear weights are estimated with the MERT algorithm (Och 2003).

Evaluation. We test our models on two different data sets. First, we train an English to French system based on the data of WMT 2006 shared task (Koehn and Monz 2006). The parallel corpus includes 688K sentence pairs of parliamentary proceedings for training. The development set contains 2000 sentences, and the test set contains other 2000 sentences, all from the official WMT 2006 shared task.

Second, we experiment with a French to English system developed using 2.1M sentence pairs of training data, which amounts to 102M words, from the WMT 2012 campaign. The majority of the training data set is parliamentary proceedings except for 5M words which are newswire. We use the 2009 newswire data set, comprising 2525 sentences, as the development set. We evaluate on four newswire domain test sets from 2008, 2010 and 2011 as well as the 2010 system combination test set, containing 2034 to 3003 sentences.

In this study we perform a detailed empirical comparison using the WMT 2006 data set, and verify our best models and results using the larger WMT 2012 data set.

The metric used for evaluation is case insensitive BLEU score (Papineni et al. 2002). We also perform a significance test using the Wilcoxon signed rank test. Differences are considered statistically significant when the *p*-value is less than 0.05.

6.2 Results of the CPTM

Table 1 shows the results measured in BLEU evaluated on the WMT 2006 data set, where Row 1 is the baseline system. Rows 2 to 4 are the systems enhanced by integrating different versions of the CPTM. Rows 5 to 7 present the results of previous models. Row 8 is our best system. Table 2 shows the main results on the WMT 2012 data set.

CPTM is the model described in Sections 4. As illustrated in Figure 2, the number of the nodes in the input layer is the vocabulary size d. Both the hidden layer and the output layer have 100 nodes[6]. That is, $\mathbf{W}^1$ is a $d \times 100$ matrix and $\mathbf{W}^2$ a 100×100 matrix. The result shows that **CPTM** leads to a substantial improvement over the baseline system with a statistically significant margin of 1.0 BLEU points as in Table 1.

We have developed a set of variants of **CPTM** to investigate two design choices we made in developing the CPTM: (1) whether to use a linear projection or a multi-layer nonlinear projection; and (2) whether to compute the phrase similarity using word-word similarities as suggested by e.g., the lexical weighting model (Koehn et al. 2003). We compare these variants on the WMT 2006 data set, as shown in Table 1.

#	Systems	WMT test2006
1	**Baseline**	33.06
2	**CPTM**	**34.10**$^{\alpha}$
3	**CPTM$_{\mathbf{L}}$**	33.60$^{\alpha\beta}$
4	**CPTM$_{\mathbf{W}}$**	33.25$^{\beta}$
5	**BLTM$_{\mathbf{PR}}$**	33.15$^{\beta}$
6	**DPM**	33.29$^{\beta}$
7	**MRF$_{\mathbf{P}}$**	33.91$^{\alpha}$
8	**Comb (2 + 7)**	**34.39**$^{\alpha\beta}$

Table 1: BLEU results for the English to French task using translation models and systems built on the WMT 2006 data set. The superscripts α and β indicate statistically significant difference ($p < 0.05$) from **Baseline** and **CPTM**, respectively.

CPTM$_{\mathbf{L}}$ (Row 3 in Table 1) uses a linear neural network to project a word vector of a phrase $\mathbf{x}$ to a feature vector $\mathbf{y}$: $\mathbf{y} \equiv \phi(\mathbf{x}) = \mathbf{W}^{\mathrm{T}}\mathbf{x}$, where $\mathbf{W}$ is a $d \times 100$ projection matrix. The translation score of a source phrase f and a target phrase e is measured as the similarity of their feature vectors. We choose cosine similarity because it works better than dot product for linear projection.

CPTM$_{\mathbf{W}}$ (Row 4 in Table 1) computes the phrase similarity using word-word similarity scores. This follows the common smoothing strategy of addressing the data sparseness problem in modeling phrase translations, such as the lexical weighting model (Koehn et al. 2003) and the word factored n-gram translation model (Son et al. 2012). Let w denote a word, and f and e the source and target phrases, respectively. We define

$$\mathrm{sim}(f,e) = \frac{1}{|f|}\sum_{w\in f}\mathrm{sim}_\tau(w,e) + \frac{1}{|e|}\sum_{w\in e}\mathrm{sim}_\tau(w,f)$$

where $\mathrm{sim}_\tau(w,e)$ (or $\mathrm{sim}_\tau(w,f)$) is the word-phrase similarity, and is defined as a smooth approximation of the maximum function

$$\mathrm{sim}_\tau(w,e) = \frac{\sum_{w'\in e}\mathrm{sim}(w,w')\exp\big(\tau\mathrm{sim}(w,w')\big)}{\sum_{w'\in e}\exp\big(\tau\mathrm{sim}(w,w')\big)}$$

[6] We can achieve slightly better results using more nodes in the hidden and output layers, say 500 nodes. But the model training is too slow to perform a detailed study within a reasonable time. Therefore, all the models reported in this paper use 100 nodes.

#	Systems	dev	news2011	news2010	news2008	newssyscomb2010
1	**Baseline**	23.58	25.24	24.35	20.36	24.14
2	**MRF$_P$**	24.07$^{\alpha}$	26.00$^{\alpha}$	24.90	20.84$^{\alpha}$	25.05$^{\alpha}$
3	**CPTM**	24.12$^{\alpha}$	26.25$^{\alpha}$	25.05$^{\alpha}$	21.15$^{\alpha\beta}$	24.91$^{\alpha}$
4	**Comb (2 + 3)**	**24.46**$^{\alpha\beta}$	**26.56**$^{\alpha\beta}$	**25.52**$^{\alpha\beta}$	**21.64**$^{\alpha\beta}$	**25.22**$^{\alpha}$

Table 2: BLEU results for the French to English task using translation models and systems built on the WMT 2012 data set. The superscripts α and β indicate statistically significant difference ($p < 0.05$) from **Baseline** and **MRF$_p$**, respectively.

where $\mathrm{sim}_\tau(w, e)$ (or $\mathrm{sim}_\tau(w, f)$) is the word-phrase similarity, and is defined as a smooth approximation of the maximum function
where τ is the tuned smoothing parameter.

Similar to **CPTM**, **CPTM$_W$** also uses a nonlinear projection to map each word (not a phrase vector as in **CPTM**) to a feature vector.

Two observations can be made by comparing **CPTM** in Row 2 to its variants in Table 1. First of all, it is more effective to model the phrase translation directly than decomposing it into word-word translations in the CPTMs. Second, we see that the nonlinear projection is able to generate more effective features, leading to better results than the linear projection.

We also compare the best version of the CPTM i.e., **CPTM**, with three related models proposed previously. We start the discussion with the results on the WMT 2006 data set in Table 1.

Rows 5 and 6 in Table 1 are two state-of-the-art latent semantic models that are originally trained on clicked query-document pairs (i.e., clickthrough data extracted from search logs) for query-document matching (Gao et al. 2011). To adopt these models for SMT, we view source-target sentence pairs as clicked query-document pairs, and trained both models using the same methods as in Gao et al. (2011) on the parallel bilingual training data described earlier. Specifically, **BTLM$_{PR}$** is an extension to PLSA, and is the best performer among different versions of the Bi-Lingual Topic Model (BLTM) described in Gao et al. (2011). BLTM with Posterior Regularization (**BLTM$_{PR}$**) is trained on parallel training data using the EM algorithm with a constraint enforcing a source sentence and its paralleled target sentence to not only share the same prior topic distribution, but to also have similar fractions of words assigned to each topic. We incorporated the model into the log-linear model for SMT (1) as follows. First of all, the topic distribution of a source sentence F_i, denoted by $P(z|F_i)$, is induced from the learned topic-word distributions using EM. Then, each translation candidate E in the N-best list $\mathrm{GEN}(F_i)$ is scored as

$$P(E|F_i) = \prod_{w \in E} \sum_z P(w|z) P(z|F_i)$$

$P(F_i|E)$ can be similarly computed. Finally, the logarithms of the two probabilities are incorporated into the log-linear model of (1) as two additional features. **DPM** is the Discriminative Projection Model described in Gao et al. (2011), which is an extension of LSA. **DPM** uses a matrix to project a word vector of a sentence to a feature vector. The projection matrix is learned on parallel training data using the S2Net algorithm (Yih et al. 2011). **DPM** can be incorporated into the log-linear model for SMT (1) by introducing a new feature h_{M+1} for each phrase pair, which is defined as the cosine similarity of the phrases in the project space.

As we see from Table 1, both latent semantic models, although leading to some slight improvement over **Baseline**, are much less effective than **CPTM**.

Finally, we compare the CPTM with the Markov Random Field model using phrase features (**MRF$_P$** in Tables 1 and 2), proposed by Gao and He (2013)[7], on both the WMT 2006 and WMT 2012 datasets. **MRF$_p$** is a state-of-the-art large scale discriminative training model that uses the same expected BLEU training criterion, which has proven to give superior performance across a range of MT tasks recently (He and Deng 2012, Setiawan and Zhou 2013, Gao and He 2013).

Unlike **CPTM**, **MRF$_p$** is a linear model that simply treats each phrase pair as a single feature. Therefore, although both are trained using the

[7] Gao and He (2013) reported results of MRF models with different feature sets. We picked the MRF using phrase features only (**MRF$_P$**) for comparison since we are mainly interested in phrase representation.

same expected BLEU based objective function, **CPTM** and **MRF$_p$** model the translation relationship between two phrases from different angles. **MRF$_p$** estimates one translation score for each phrase pair explicitly without parameter sharing, while in **CPTM**, all phrases share the same neural network that projects raw phrases to the continuous space, providing a more smoothed estimation of the translation score for each phrase pair.

The results in Tables 1 and 2 show that **CPTM** outperforms **MRF$_P$** on most of the test sets across the two WMT data sets, but the difference between them is often not significant. Our interpretation is that although **CPTM** provides a better smoothed estimation for low-frequent phrase pairs, which otherwise suffer the data sparsity issue, **MRF$_p$** provides a more precise estimation for those high-frequent phrase pairs. That is, **CPTM** and **MRF$_p$** capture complementary information for translation. We thus combine **CPTM** and **MRF$_P$** (**Comb** in Tables 1 and 2) by incorporating two features, each for one model, into the log-linear model of SMT (1). We observe that for both translation tasks, accuracy improves by up to 0.8 BLEU over **MRF$_P$** alone (e.g., on the news2008 test set in Table 2). The results confirm that **CPTM** captures complementary translation information to **MRF$_p$**. Overall, we improve accuracy by up to 1.3 BLEU over the baseline on both WMT data sets.

7 Conclusions

The work presented in this paper makes two major contributions. First, we develop a novel phrase translation model for SMT, where joint representations are exploited of a phrase in the source language and of its translation in the target language, and where the translation score of the pair of source-target phrases are represented as the distance between their feature vectors in a low-dimensional, continuous space. The space is derived from the representations generated using a multilayer neural network. Second, we present a new learning method to train the weights in the multilayer neural network for the end-to-end BLEU metric directly. The training method is based on L-BFGS. We describe in detail how the gradient in closed form, as required for efficient optimization, is derived. The objective function, which takes the form of the expected BLEU computed from N-best lists, is very different from the usual objective functions used in most existing architectures of neural networks, e.g., cross entropy (Hinton et al. 2012) or mean square error (Deng et al. 2012). We hence have provided details in the derivation of the gradient, which can serve as an example to guide the derivation of neural network learning with other non-standard objective functions in the future.

Our evaluation on two WMT data sets show that incorporating the continuous-space phrase translation model into the log-linear framework significantly improves the accuracy of a state-of-the-art phrase-based SMT system, leading to a gain up to 1.3 BLEU. Careful implementation of the L-BFGS optimization based on the BLEU-centric objective function, together with the associated closed-form gradient, is a key to the success.

A natural extension of this work is to expand the model and learning algorithm from shallow to deep neural networks. The deep models are expected to produce more powerful and flexible semantic representations (e.g., Tur et al., 2012), and thus greater performance gain than what is presented in this paper.

8 Acknowledgements

We thank Michael Auli for providing a dataset and for helpful discussions. We also thank the four anonymous reviewers for their comments.

References

Andrew, G. and Gao, J. 2007. Scalable training of L1-regularized log-linear models. In *ICML*.

Auli, M., Galley, M., Quirk, C. and Zweig, G. 2013 Joint language and translation modeling with recurrent neural networks. In *EMNLP*.

Bengio, Y. 2009. Learning deep architectures for AI. *Fundamental Trends Machine Learning*, vol. 2, no. 1, pp. 1–127.

Bengio, Y., Duharme, R., Vincent, P., and Janvin, C. 2003. A neural probabilistic language model. *JMLR*, 3:1137-1155.

Blei, D. M., Ng, A. Y., and Jordan, M. J. 2003. Latent Dirichlet allocation. *Journal of Machine Learning Research*, 3: 993-1022.

Collobert, R., Weston, J., Bottou, L., Karlen, M., Kavukcuoglu, K., and Kuksa, P. 2011. Natural language processing (almost) from scratch. *Journal of Machine Learning Research*, vol. 12.

Deerwester, S., Dumais, S. T., Furnas, G. W., Landauer, T., and Harshman, R. 1990. Indexing by latent semantic analysis. *Journal of the American Society for Information Science*, 41(6): 391-407

DeNero, J., Gillick, D., Zhang, J., and Klein, D. 2006. Why generative phrase models underperform surface heuristics. In *Workshop on Statistical Machine Translation*, pp. 31-38.

Deng, L., Yu, D., and Platt, J. 2012. Scalable stacking and learning for building deep architectures. In *ICASSP*.

Diamantaras, K. I., and Kung, S. Y. 1996. *Principle Component Neural Networks: Theory and Applications.* Wiley-Interscience.

Dumais S., Letsche T., Littman M. and Landauer T. 1997. Automatic cross-language retrieval using latent semantic indexing. In AAAI-97 Spring Symposium Series: Cross-Language Text and Speech Retrieval.

Ganchev, K., Graca, J., Gillenwater, J., and Taskar, B. 2010. Posterior regularization for structured latent variable models. *Journal of Machine Learning Research*, 11 (2010): 2001-2049.

Gao, J., and He, X. 2013. Training MRF-based translation models using gradient ascent. In *NAACL-HLT*, pp. 450-459.

Gao, J., Toutanova, K., Yih., W-T. 2011. Clickthrough-based latent semantic models for web search. In *SIGIR*, pp. 675-684.

He, X., and Deng, L. 2012. Maximum expected bleu training of phrase and lexicon translation models. In *ACL*, pp. 292-301.

Hinton, G., and Salakhutdinov, R., 2010. Discovering Binary Codes for Documents by Learning Deep Generative Models. *Topics in Cognitive Science*, pp. 1-18.

Hinton, G., Deng, L., Yu, D., Dahl, G., Mohamed, A., Jaitly, N., Senior, A., Vanhoucke, V., Nguyen, P., Sainath, T., and Kingsbury, B., 2012. Deep neural networks for acoustic modeling in speech recognition. *IEEE Signal Processing Magazine*, vol. 29, no. 6, pp. 82-97.

Hofmann, T. 1999. Probabilistic latent semantic indexing. In *SIGIR*, pp. 50-57.

Huang, P-S., He, X., Gao, J., Deng, L., Acero, A. and Heck, L. 2013. Learning deep structured semantic models for web search using clickthrough data. In *CIKM*.

Kalchbrenner, N. and Blunsom, P. 2013. Recurrent continuous translation models. In *EMNLP*.

Koehn, P., Hoang, H., Birch, A., Callison-Burch, C., Federico, M., Bertoldi, N., Cowan, B., Shen, W., Moran, C., Zens, R., Dyer, C., Bojar, O., Constantin, A., and Herbst, E. 2007. Moses: open source toolkit for statistical machine translation. In *ACL 2007*, demonstration session.

Koehn, P. and Monz, C. 2006. Manual and automatic evaluation of machine translation between European languages. In *Workshop on Statistical Machine Translation*, pp. 102-121.

Koehn, P., Och, F., and Marcu, D. 2003. Statistical phrase-based translation. In *HLT-NAACL*, pp. 127-133.

Lambert, P. and Banchs, R. E. 2005. Data inferred multi-word expressions for statistical machine translation. In *MT Summit X*, Phuket, Thailand.

Li, P., Liu, Y., and Sun, M. 2013. Recursive autoencoders for ITG-based translation. In *EMNLP*.

Liang,P., Bouchard-Cote,A., Klein, D. and Taskar, B. 2006. An end-to-end discriminative approach to machine translation. In *COLING-ACL*.

Marcu, D., and Wong, W. 2002. A phrase-based, joint probability model for statistical machine translation. In *EMNLP*.

Mikolov, T., Karafiat, M., Burget, L., Cernocky, J., and Khudanpur, S. 2010. Recurrent neural network based language model. In *INTERSPEECH*, pp. 1045-1048.

Mikolov, T., Kombrink, S., Burget, L., Cernocky, J., and Khudanpur, S. 2011. Extensions of recurrent neural network language model. In *ICASSP*, pp. 5528-5531.

Mikolov, T. 2012. Statistical Language Model based on Neural Networks. *Ph.D. thesis*, Brno University of Technology.

Mikolov, T., Le, Q. V., and Sutskever, H. 2013a. Exploiting similarities among languages for machine translation. CoRR. 2013; abs/1309.4148.

Mikolov, T., Yih, W. and Zweig, G. 2013b. Linguistic Regularities in Continuous Space Word Representations. In *NAACL-HLT*.

Mimno, D., Wallach, H., Naradowsky, J., Smith, D. and McCallum, A. 2009. Polylingual topic models. In *EMNLP*.

Niehues J., Herrmann, T., Vogel, S., and Waibel, A. 2011. Wider context by using bilingual language models in machine translation.

Och, F. 2003. Minimum error rate training in statistical machine translation. In *ACL*, pp. 160-167.

Och, F., and Ney, H. 2004. The alignment template approach to statistical machine translation. *Computational Linguistics*, 29(1): 19-51.

Papineni, K., Roukos, S., Ward, T., and Zhu W-J. 2002. BLEU: a method for automatic evaluation of machine translation. In *ACL*.

Platt, J., Toutanova, K., and Yih, W. 2010. Translingual Document Representations from Discriminative Projections. In *EMNLP*.

Rosti, A-V., Hang, B., Matsoukas, S., and Schwartz, R. S. 2011. Expected BLEU training for graphs: bbn system description for WMT system combination task. In *Workshop on Statistical Machine Translation*.

Schwenk, H., Costa-Jussa, M. R. and Fonollosa, J. A. R. 2007. Smooth bilingual n-gram translation. In *EMNLP-CoNLL*, pp. 430-438.

Schwenk, H. 2012. Continuous space translation models for phrase-based statistical machine translation. In *COLING*.

Schwenk, H., Rousseau, A., and Mohammed A. 2012. Large, pruned or continuous space language models on a GPU for statistical machine translation. In *NAACL-HLT Workshop on the future of language modeling for HLT*, pp. 11-19.

Setiawan, H. and Zhou, B., 2013. Discriminative training of 150 million translation parameters and its application to pruning. In *NAACL*.

Socher, R., Huval, B., Manning, C., Ng, A., 2012. Semantic Compositionality through Recursive Matrix-Vector Spaces. In *EMNLP*.

Socher, R., Lin, C., Ng, A. Y., and Manning, C. D. 2011. Parsing natural scenes and natural language with recursive neural networks. In *ICML*.

Son, L. H., Allauzen, A., and Yvon, F. 2012. Continuous space translation models with neural networks. In *NAACL-HLT*, pp. 29-48.

Sundermeyer, M., Oparin, I., Gauvain, J-L. Freiberg, B., Schluter, R. and Ney, H. 2013. Comparison of feed forward and recurrent neural network language models. In *ICASSP*, pp. 8430–8434.

Tur, G, Deng, L., Hakkani-Tur, D., and He, X., 2012. Towards deeper understanding: deep convex networks for semantic utterance classification. In *ICASSP*.

Vinokourov,A., Shawe-Taylor,J. and Cristianini,N. 2002. Inferring a semantic representation of text via cross-language correlation analysis. In *NIPS*.

Weston, J., Bengio, S., and Usunier, N. 2011. Large scale image annotation: learning to rank with joint word-image embeddings. In *IJCAI*.

Wuebker, J., Mauser, A., and Ney, H. 2010. Training phrase translation models with leaving-one-out. In *ACL*, pp. 475-484.

Yih, W., Toutanova, K., Platt, J., and Meek, C. 2011. Learning discriminative projections for text similarity measures. In *CoNLL*.

Zhang, Y., Deng, L., He, X., and Acero, A. 2011. A novel decision function and the associated decision-feedback learning for speech translation. In *ICASSP*.

Zhila, A., Yih, W., Meek, C., Zweig, G. and Mikolov, T. 2013. Combining heterogeneous models for measuring relational similarity. In *NAACL-HLT*.

Zou, W. Y., Socher, R., Cer, D., and Manning, C. D. 2013. Bilingual word embeddings for phrase-based machine translation. In *EMNLP*.