Word Alignment Modeling with Context Dependent Deep Neural Network

Nan Yang[1], **Shujie Liu**[2], **Mu Li**[2], **Ming Zhou**[2], **Nenghai Yu**[1]
[1]University of Science and Technology of China, Hefei, China
[2]Microsoft Research Asia, Beijing, China
`{v-nayang,shujliu,muli,mingzhou}@microsoft.com`
`ynh@ustc.edu.cn`

Abstract

In this paper, we explore a novel bilingual word alignment approach based on DNN (Deep Neural Network), which has been proven to be very effective in various machine learning tasks (Collobert et al., 2011). We describe in detail how we adapt and extend the CD-DNN-HMM (Dahl et al., 2012) method introduced in speech recognition to the HMM-based word alignment model, in which bilingual word embedding is discriminatively learnt to capture lexical translation information, and surrounding words are leveraged to model context information in bilingual sentences. While being capable to model the rich bilingual correspondence, our method generates a very compact model with much fewer parameters. Experiments on a large scale English-Chinese word alignment task show that the proposed method outperforms the HMM and IBM model 4 baselines by 2 points in F-score.

1 Introduction

Recent years research communities have seen a strong resurgent interest in modeling with deep (multi-layer) neural networks. This trending topic, usually referred under the name *Deep Learning*, is started by ground-breaking papers such as (Hinton et al., 2006), in which innovative training procedures of deep structures are proposed. Unlike shallow learning methods, such as Support Vector Machine, Conditional Random Fields, and Maximum Entropy, which need hand-craft features as input, DNN can learn suitable features (representations) automatically with raw input data, given a training objective.

DNN did not achieve expected success until 2006, when researchers discovered a proper way to intialize and train the deep architectures, which contains two phases: layer-wise unsupervised pre-training and supervised fine tuning. For pre-training, Restricted Boltzmann Machine (RBM) (Hinton et al., 2006), auto-encoding (Bengio et al., 2007) and sparse coding (Lee et al., 2007) are proposed and popularly used. The unsupervised pre-training trains the network one layer at a time, and helps to guide the parameters of the layer towards better regions in parameter space (Bengio, 2009). Followed by fine tuning in this region, DNN is shown to be able to achieve state-of-the-art performance in various area, or even better (Dahl et al., 2012) (Kavukcuoglu et al., 2010). DNN also achieved breakthrough results on the ImageNet dataset for objective recognition (Krizhevsky et al., 2012). For speech recognition, (Dahl et al., 2012) proposed context-dependent neural network with large vocabulary, which achieved 16.0% relative error reduction.

DNN has also been applied in Natural Language Processing (NLP) field. Most works convert atomic lexical entries into a dense, low dimensional, real-valued representation, called *word embedding*; Each dimension represents a latent aspect of a word, capturing its semantic and syntactic properties (Bengio et al., 2006). Word embedding is usually first learned from huge amount of monolingual texts, and then fine-tuned with task-specific objectives. (Collobert et al., 2011) and (Socher et al., 2011) further apply Recursive Neural Networks to address the structural prediction tasks such as tagging and parsing, and (Socher et al., 2012) explores the compositional aspect of word representations.

Inspired by successful previous works, we propose a new DNN-based word alignment method, which exploits contextual and semantic similarities between words. As shown in example (a) of Figure 1, in word pair {"juda" $\Rightarrow$"mammoth"}, the Chinese word "juda" is a common word, but

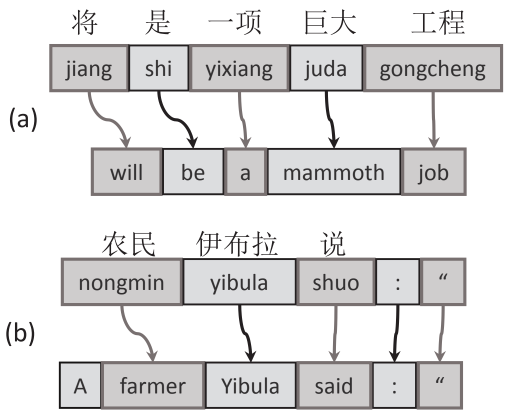

Figure 1: Two examples of word alignment

the English word “mammoth” is not, so it is very hard to align them correctly. If we know that “mammoth” has the similar meaning with “big”, or “huge”, it would be easier to find the corresponding word in the Chinese sentence. As we mentioned in the last paragraph, *word embedding* (trained with huge monolingual texts) has the ability to map a word into a vector space, in which, similar words are near each other.

For example (b) in Figure 1, for the word pair {“yibula” ⇒ “Yibula”}, both the Chinese word “yibula” and English word “Yibula” are rare name entities, but the words around them are very common, which are {“nongmin”, “shuo”} for Chinese side and {“farmer”, “said”} for the English side. The pattern of the context {“nongmin X shuo” ⇒ “farmer X said”} may help to align the word pair which fill the variable X, and also, the pattern {“yixiang X gongcheng” ⇒ “a X job”} is helpful to align the word pair {“juda” ⇒“mammoth”} for example (a).

Based on the above analysis, in this paper, both the words in the source and target sides are firstly mapped to a vector via a discriminatively trained word embeddings, and word pairs are scored by a multi-layer neural network which takes rich contexts (surrounding words on both source and target sides) into consideration; and a HMM-like distortion model is applied on top of the neural network to characterize structural aspect of bilingual sentences.

In the rest of this paper, related work about DNN and word alignment are first reviewed in Section 2, followed by a brief introduction of DNN in Section 3. We then introduce the details of leveraging DNN for word alignment, including the details of our network structure in Section 4 and the training method in Section 5. The merits of our approach are illustrated with the experiments described in Section 6, and we conclude our paper in Section 7.

2 Related Work

DNN with unsupervised pre-training was firstly introduced by (Hinton et al., 2006) for MNIST digit image classification problem, in which, RBM was introduced as the layer-wise pre-trainer. The layer-wise pre-training phase found a better local maximum for the multi-layer network, thus led to improved performance. (Krizhevsky et al., 2012) proposed to apply DNN to do object recognition task (ImageNet dataset), which brought down the state-of-the-art error rate from 26.1% to 15.3%. (Seide et al., 2011) and (Dahl et al., 2012) apply Context-Dependent Deep Neural Network with HMM (CD-DNN-HMM) to speech recognition task, which significantly outperforms traditional models.

Most methods using DNN in NLP start with a word embedding phase, which maps words into a fixed length, real valued vectors. (Bengio et al., 2006) proposed to use multi-layer neural network for language modeling task. (Collobert et al., 2011) applied DNN on several NLP tasks, such as part-of-speech tagging, chunking, name entity recognition, semantic labeling and syntactic parsing, where they got similar or even better results than the state-of-the-art on these tasks. (Niehues and Waibel, 2012) shows that machine translation results can be improved by combining neural language model with n-gram traditional language. (Son et al., 2012) improves translation quality of n-gram translation model by using a bilingual neural language model. (Titov et al., 2012) learns a context-free cross-lingual word embeddings to facilitate cross-lingual information retrieval.

For the related works of word alignment, the most popular methods are based on generative models such as IBM Models (Brown et al., 1993) and HMM (Vogel et al., 1996). Discriminative approaches are also proposed to use hand crafted features to improve word alignment. Among them, (Liu et al., 2010) proposed to use phrase and rule pairs to model the context information in a log-linear framework. Unlike previous discriminative methods, in this work, we do not resort to any hand crafted features, but use DNN to induce “features” from raw words.

3 DNN structures for NLP

The most important and prevalent features available in NLP are the words themselves. To apply DNN to NLP task, the first step is to transform a discrete word into its *word embedding*, a low dimensional, dense, real-valued vector (Bengio et al., 2006). Word embeddings often implicitly encode syntactic or semantic knowledge of the words. Assuming a finite sized vocabulary V, word embeddings form a $(L \times |V|)$-dimension embedding matrix W_V, where L is a pre-determined embedding length; mapping words to embeddings is done by simply looking up their respective columns in the embedding matrix W_V. The lookup process is called a *lookup layer LT* , which is usually the first layer after the input layer in neural network.

After words have been transformed to their embeddings, they can be fed into subsequent classical network layers to model highly non-linear relations:

$$z^l = f_l(M^l z^{l-1} + b^l) \quad (1)$$

where z^l is the output of lth layer, M^l is a $|z^l| \times |z^{l-1}|$ matrix, b^l is a $|z^l|$-length vector, and f_l is an $activation$ function. Except for the last layer, f_l must be non-linear. Common choices for f_l include sigmoid function, hyperbolic function, "hard" hyperbolic function etc. Following (Collobert et al., 2011), we choose "hard" hyperbolic function as our activation function in this work:

$$htanh(x) = \begin{cases} 1 & \text{if } x \text{ is greater than 1} \\ -1 & \text{if } x \text{ is less than -1} \\ x & \text{otherwise} \end{cases} \quad (2)$$

If probabilistic interpretation is desired, a *softmax* layer (Bridle, 1990) can be used to do normalization:

$$z_i^l = \frac{e^{z_i^{l-1}}}{\sum_{j=1}^{|z^l|} e^{z_j^{l-1}}} \quad (3)$$

The above layers can only handle fixed sized input and output. If input must be of variable length, *convolution* layer and *max* layer can be used, (Collobert et al., 2011) which transform variable length input to fixed length vector for further processing.

Multi-layer neural networks are trained with the standard *back propagation* algorithm (LeCun, 1985). As the networks are non-linear and the task specific objectives usually contain many local maximums, special care must be taken in the optimization process to obtain good parameters. Techniques such as *layerwise pre-training*(Bengio et al., 2007) and many tricks(LeCun et al., 1998) have been developed to train better neural networks. Besides that, neural network training also involves some hyperparameters such as learning rate, the number of hidden layers. We will address these issues in section 4.

4 DNN for word alignment

Our DNN word alignment model extends classic HMM word alignment model (Vogel et al., 1996). Given a sentence pair (**e**, **f**), HMM word alignment takes the following form:

$$P(\mathbf{a}, \mathbf{e}|\mathbf{f}) = \prod_{i=1}^{|\mathbf{e}|} P_{lex}(e_i|f_{a_i}) P_d(a_i - a_{i-1}) \quad (4)$$

where P_{lex} is the lexical translation probability and P_d is the jump distance distortion probability.

One straightforward way to integrate DNN into HMM is to use neural network to compute the emission (lexical translation) probability P_{lex}. Such approach requires a softmax layer in the neural network to normalize over all words in source vocabulary. As vocabulary for natural languages is usually very large, it is prohibitively expensive to do the normalization. Hence we give up the probabilistic interpretation and resort to a non-probabilistic, discriminative view:

$$s_{NN}(\mathbf{a}|\mathbf{e}, \mathbf{f}) = \prod_{i=1}^{|\mathbf{e}|} t_{lex}(e_i, f_{a_i}|\mathbf{e}, \mathbf{f}) t_d(a_i, a_{i-1}|\mathbf{e}, \mathbf{f}) \quad (5)$$

where t_{lex} is a lexical translation score computed by neural network, and t_d is a distortion score.

In the classic HMM word alignment model, context is not considered in the lexical translation probability. Although we can rewrite $P_{lex}(e_i|f_{a_i})$ to $P_{lex}(e_i|\text{context of } f_{a_i})$ to model context, it introduces too many additional parameters and leads to serious over-fitting problem due to data sparseness. As a matter of fact, even without any contexts, the lexical translation table in HMM already contains $O(|V_e| * |V_f|)$ parameters, where $|V_e|$ and V_f denote source and target vocabulary sizes. In contrast, our model does not maintain a separate translation score parameters for every source-target word pair, but computes t_{lex} through a multi-layer network, which naturally handles contexts on both sides without explosive growth of number of parameters.

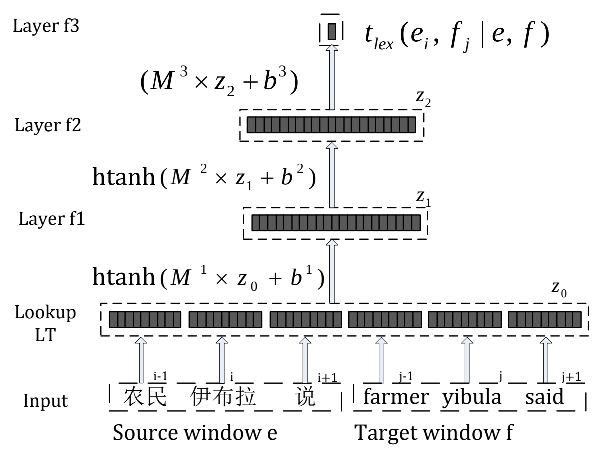

Figure 2: Network structure for computing context dependent lexical translation scores. The example computes translation score for word pair (*yibula*, *yibulayin*) given its surrounding context.

Figure 2 shows the neural network we used to compute context dependent lexical translation score t_{lex}. For word pair (e_i, f_j), we take fixed length windows surrounding both e_i and f_j as input: $(e_{i-\frac{sw}{2}}, \dots, e_{i+\frac{sw}{2}}, f_{j-\frac{tw}{2}}, \dots, f_{j+\frac{tw}{2}})$, where sw, tw stand window sizes on source and target side respectively. Words are converted to embeddings using the lookup table LT, and the catenation of embeddings are fed to a classic neural network with two hidden-layers, and the output of the network is the our lexical translation score:

$$\begin{aligned} & t_{lex}(e_i, f_j|\mathbf{e}, \mathbf{f}) \\ & = f_3 \circ f_2 \circ f_1 \circ LT(window(e_i), window(f_j)) \end{aligned} \quad (6)$$

f_1 and f_2 layers use $htanh$ as activation functions, while f_3 is only a linear transformation with no activation function.

For the distortion t_d, we could use a lexicalized distortion model:

$$t_d(a_i, a_{i-1}|\mathbf{e}, \mathbf{f}) = t_d(a_i - a_{i-1}|window(f_{a_{i-1}})) \quad (7)$$

which can be computed by a neural network similar to the one used to compute lexical translation scores. If we map jump distance $(a_i - a_{i-1})$ to B buckets, we can change the length of the output layer to B, where each dimension in the output stands for a different bucket of jump distances. But we found in our initial experiments on small scale data, lexicalized distortion does not produce better alignment over the simple jump-distance based model. So we drop the lexicalized distortion and reverse to the simple version:

$$t_d(a_i, a_{i-1}|\mathbf{e}, \mathbf{f}) = t_d(a_i - a_{i-1}) \quad (8)$$

Vocabulary V of our alignment model consists of a source vocabulary V_e and a target vocabulary V_f. As in (Collobert et al., 2011), in addition to real words, each vocabulary contains a special unknown word symbol $\langle unk \rangle$ to handle unseen words; two sentence boundary symbols $\langle s \rangle$ and $\langle /s \rangle$, which are filled into surrounding window when necessary; furthermore, to handle null alignment, we must also include a special null symbol $\langle null \rangle$. When f_j is null word, we simply fill the surrounding window with the identical null symbols.

To decode our model, the lexical translation scores are computed for each source-target word pair in the sentence pair, which requires going through the neural network $(|\mathbf{e}| \times |\mathbf{f}|)$ times; after that, the *forward-backward* algorithm can be used to find the viterbi path as in the classic HMM model.

The majority of tunable parameters in our model resides in the lookup table LT, which is a $(L \times (|V_e| + |V_f|))$-dimension matrix. For a reasonably large vocabulary, the number is much smaller than the number of parameters in classic HMM model, which is in the order of $(|V_e| \times |V_f|)$. [1]

The ability to model context is not unique to our model. In fact, discriminative word alignment can model contexts by deploying arbitrary features (Moore, 2005). Different from previous discriminative word alignment, our model does not use manually engineered features, but learn "features" automatically from raw words by the neural network. (Berger et al., 1996) use a maximum entropy model to model the bag-of-words context for word alignment, but their model treats each word as a distinct feature, which can not leverage the similarity between words as our model.

5 Training

Although unsupervised training technique such as *Contrastive Estimation* as in (Smith and Eisner, 2005), (Dyer et al., 2011) can be adapted to train

[1]In practice, the number of non-zero parameters in classic HMM model would be much smaller, as many words do not co-occur in bilingual sentence pairs. In our experiments, the number of non-zero parameters in classic HMM model is about 328 millions, while the NN model only has about 4 millions.

our model from raw sentence pairs, they are too computational demanding as the lexical translation probabilities must be computed from neural networks. Hence, we opt for a simpler supervised approach, which learns the model from sentence pairs with word alignment. As we do not have a large manually word aligned corpus, we use traditional word alignment models such as HMM and IBM model 4 to generate word alignment on a large parallel corpus. We obtain bidirectional alignment by running the usual *grow-diag-final* heuristics (Koehn et al., 2003) on unidirectional results from both directions, and use the results as our training data. Similar approach has been taken in speech recognition task (Dahl et al., 2012), where training data for neural network model is generated by forced decoding with traditional Gaussian mixture models.

Tunable parameters in neural network alignment model include: word embeddings in lookup table LT, parameters W^l, b^l for linear transformations in the hidden layers of the neural network, and distortion parameters s_d of jump distance. We take the following ranking loss with margin as our training criteria:

$$loss(\theta) = \sum_{\text{every } (\mathbf{e},\mathbf{f})} max\{0, 1 - s_\theta(\mathbf{a}^+|\mathbf{e},\mathbf{f}) + s_\theta(\mathbf{a}^-|\mathbf{e},\mathbf{f})\} \tag{9}$$

where θ denotes all tunable parameters, $\mathbf{a}^+$ is the gold alignment path, $\mathbf{a}^-$ is the highest scoring incorrect alignment path under θ, and s_θ is model score for alignment path defined in Eq. 5 . One nuance here is that the gold alignment after *grow-diag-final* contains many-to-many links, which cannot be generated by any path. Our solution is that for each source word alignment multiple target, we randomly choose one link among all candidates as the golden link.

Because our multi-layer neural network is inherently non-linear and is non-convex, directly training against the above criteria is unlikely to yield good results. Instead, we take the following steps to train our model.

5.1 Pre-training initial word embedding with monolingual data

Most parameters reside in the word embeddings. To get a good initial value, the usual approach is to pre-train the embeddings on a large monolingual corpus. We replicate the work in (Collobert et al., 2011) and train word embeddings for source and target languages from their monolingual corpus respectively. Our vocabularies V_s and V_t contain the most frequent 100,000 words from each side of the parallel corpus, and all other words are treated as unknown words. We set word embedding length to 20, window size to 5, and the length of the only hidden layer to 40. Follow (Turian et al., 2010), we randomly initialize all parameters to [-0.1, 0.1], and use stochastic gradient descent to minimize the ranking loss with a fixed learning rate 0.01. Note that embedding for null word in either V_e and V_f cannot be trained from monolingual corpus, and we simply leave them at the initial value untouched.

Word embeddings from monolingual corpus learn strong syntactic knowledge of each word, which is not always desirable for word alignment between some language pairs like English and Chinese. For example, many Chinese words can act as a verb, noun and adjective without any change, while their English counter parts are distinct words with quite different word embeddings due to their different syntactic roles. Thus we have to modify the word embeddings in subsequent steps according to bilingual data.

5.2 Training neural network based on local criteria

Training the network against the sentence level criteria Eq. 5 directly is not efficient. Instead, we first ignore the distortion parameters and train neural networks for lexical translation scores against the following local pairwise loss:

$$max\{0, 1 - t_\theta((e,f)^+|\mathbf{e},\mathbf{f}) + t_\theta((e,f)^-|\mathbf{e},\mathbf{f})\} \tag{10}$$

where $(e,f)^+$ is a correct word pair, $(e,f)^-$ is a wrong word pair in the same sentence, and t_θ is as defined in Eq. 6 . This training criteria essentially means our model suffers loss unless it gives correct word pairs a higher score than random pairs from the same sentence pair with some margin.

We initialize the lookup table with embeddings obtained from monolingual training, and randomly initialize all W^l and b^l in linear layers to [-0.1, 0.1]. We minimize the loss using stochastic gradient descent as follows. We randomly cycle through all sentence pairs in training data; for each correct word pair (including null alignment), we generate a positive example, and generate two negative examples by randomly corrupting either side of the pair with another word in the sentence pair. We set learning rate to 0.01. As there is no clear stopping criteria, we simply run the stochastic optimizer through parallel corpus for N iterations. In this work, N is set to 50.

To make our model concrete, there are still hyper-parameters to be determined: the window size sw and tw, the length of each hidden layer L_l. We empirically set sw and tw to 11, L_1 to 120, and L_2 to 10, which achieved a minimal loss on a small held-out data among several settings we tested.

5.3 Training distortion parameters

We fix neural network parameters obtained from the last step, and tune the distortion parameters s_d with respect to the sentence level loss using standard stochastic gradient descent. We use a separate parameter for jump distance from -7 and 7, and another two parameters for longer forward/backward jumps. We initialize all parameters in s_d to 0, set the learning rate for the stochastic optimizer to 0.001. As there are only 17 parameters in s_d, we only need to run the optimizer over a small portion of the parallel corpus.

5.4 Tuning neural network based on sentence level criteria

Up-to-now, parameters in the lexical translation neural network have not been trained against the sentence level criteria Eq. 5. We could achieve this by re-using the same online training method used to train distortion parameters, except that we now fix the distortion parameters and let the loss back-propagate through the neural networks. Sentence level training does not take larger context in modeling word translations, but only to optimize the parameters regarding to the sentence level loss. This tuning is quite slow, and it did not improve alignment on an initial small scale experiment; so, we skip this step in all subsequent experiment in this work.

6 Experiments and Results

We conduct our experiment on Chinese-to-English word alignment task. We use the manually aligned Chinese-English alignment corpus (Haghighi et al., 2009) which contains 491 sentence pairs as test set. We adapt the segmentation on the Chinese side to fit our word segmentation standard.

6.1 Data

Our parallel corpus contains about 26 million unique sentence pairs in total which are mined from web.

The monolingual corpus to pre-train word embeddings are also crawled from web, which amounts to about 1.1 billion unique sentences for English and about 300 million unique sentences for Chinese. As pre-processing, we lowercase all English words, and map all numbers to one special token; and we also map all email addresses and URLs to another special token.

6.2 Settings

We use classic HMM and IBM model 4 as our baseline, which are generated by Giza++ (Och and Ney, 2000). We train our proposed model from results of classic HMM and IBM model 4 separately. Since classic HMM, IBM model 4 and our model are all uni-directional, we use the standard *grow-diag-final* to generate bi-directional results for all models.

Models are evaluated on the manually aligned test set using standard metric: precision, recall and F1-score.

6.3 Alignment Result

It can be seen from Table 1, the proposed model consistently outperforms its corresponding baseline whether it is trained from alignment of classic HMM or IBM model 4. It is also clear that the

setting	prec.	recall	F-1
HMM	0.768	0.786	0.777
HMM+NN	0.810	0.790	0.798
IBM4	0.839	0.805	0.822
IBM4+NN	**0.885**	**0.812**	**0.847**

Table 1: Word alignment result. The first row and third row show baseline results obtained by classic HMM and IBM4 model. The second row and fourth row show results of the proposed model trained from HMM and IBM4 respectively.

results of our model also depends on the quality of baseline results, which is used as training data of our model. In future we would like to explore whether our method can improve other word alignment models.

We also conduct experiment to see the effect on end-to-end SMT performance. We train hierarchical phrase model (Chiang, 2007) from different word alignments. Despite different alignment scores, we do not obtain significant difference in translation performance. In our C-E experiment, we tuned on NIST-03, and tested on NIST-08. Case-insensitive BLEU-4 scores on NIST-08 test are 0.305 and 0.307 for models trained from IBM-4 and NN alignment results. The result is not surprising considering our parallel corpus is quite large, and similar observations have been made in previous work as (DeNero and Macherey, 2011) that better alignment quality does not necessarily lead to better end-to-end result.

6.4 Result Analysis

6.4.1 Error Analysis

From Table 1 we can see higher F-1 score of our model mainly comes from higher precision, with recall similar to baseline. By analyzing the results, we found out that for both baseline and our model, a large part of missing alignment links involves stop words like English words "the", "a", "it" and Chinese words "de". Stop words are inherently hard to align, which often requires grammatical judgment unavailable to our models; as they are also extremely frequent, our model fully learns their alignment patterns of the baseline models, including errors. On the other hand, our model performs better on low-frequency words, especially proper nouns. Take person names for example. Most names are low-frequency words, on which baseline HMM and IBM4 models show the "garbage collector" phenomenon. In our model, different person names have very similar word embeddings on both English side and Chinese side, due to monolingual pre-training; what is more, different person names often appear in similar contexts. As our model considers both word embeddings and contexts, it learns that English person names should be aligned to Chinese person names, which corrects errors of baseline models and leads to better precision.

6.4.2 Effect of context

To examine how context contribute to alignment quality, we re-train our model with different window size, all from result of IBM model 4. From Figure 3, we can see introducing context increase the quality of the learned alignment, but the benefit is diminished for window size over 5. On the other hand, the results are quite stable even with large window size 13, without noticeable over-

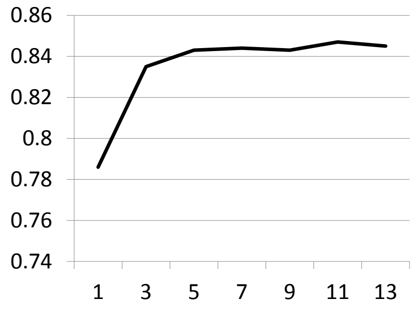

Figure 3: Effect of different window sizes on word alignment F-score.

fitting problem. This is not surprising considering that larger window size only requires slightly more parameters in the linear layers. Lastly, it is worth noticing that our model with no context (window size 1) performs much worse than settings with larger window size and baseline IBM4. Our explanation is as follows. Our model uses the simple jump distance based distortion, which is weaker than the more sophisticated distortions in IBM model 4; thus without context, it does not perform well compared to IBM model 4. With larger window size, our model is able to produce more accurate translation scores based on more contexts, which leads to better alignment despite the simpler distortions.

IBM4+NN	F-1
1-hidden-layer	0.834
2-hidden-layer	0.847
3-hidden-layer	0.843

Table 3: Effect of different number of hidden layers. Two hidden layers outperform one hidden layer, while three hidden layers do not bring further improvement.

6.4.3 Effect of number of hidden layers

Our neural network contains two hidden layers besides the lookup layer. It is natural to ask whether adding more layers would be beneficial. To answer this question, we train models with 1, 2 and 3 layers respectively, all from result of IBM model 4. For 1-hidden-layer setting, we set the hidden layer length to 120; and for 3-hidden-layer setting, we set hidden layer lengths to 120, 100, 10 respectively. As can be seen from Table 3, 2-hidden-layer outperforms the 1-hidden-layer setting, while another hidden layer does not bring

word	good	history	british	served	labs	zetian	laggards
LM	bad	tradition	russian	worked	networks	hongzhang	underperformers
	great	culture	japanese	lived	technologies	yaobang	transferees
	strong	practice	dutch	offered	innovations	keming	megabanks
	true	style	german	delivered	systems	xingzhi	mutuals
	easy	literature	canadian	produced	industries	ruihua	non-starters
WA	nice	historical	uk	offering	lab	hongzhang	underperformers
	great	historic	britain	serving	laboratories	qichao	illiterates
	best	developed	english	serve	laboratory	xueqin	transferees
	pretty	record	classic	delivering	exam	fuhuan	matriculants
	excellent	recording	england	worked	experiments	bingkun	megabanks

Table 2: Nearest neighbors of several words according to their embedding distance. *LM* shows neighbors of word embeddings trained by monolingual language model method; *WA* shows neighbors of word embeddings trained by our word alignment model.

improvement. Due to time constraint, we have not tuned the hyper-parameters such as length of hidden layers in 1 and 3-hidden-layer settings, nor have we tested settings with more hidden-layers. It would be wise to test more settings to verify whether more layers would help.

6.4.4 Word Embedding

Following (Collobert et al., 2011), we show some words together with its nearest neighbors using the Euclidean distance between their embeddings. As we can see from Table 2, after bilingual training, "bad" is no longer in the nearest neighborhood of "good" as they hold opposite semantic meanings; the nearest neighbor of "history" is now changed to its related adjective "historical". Neighbors of proper nouns such as person names are relatively unchanged. For example, neighbors of word "zetian" are all Chinese names in both settings. As Chinese language lacks morphology, the single form and plural form of a noun in English often correspond to the same Chinese word, thus it is desirable that the two English words should have similar word embeddings. While this is true for relatively frequent nouns such as "lab" and "labs", rarer nouns still remain near their monolingual embeddings as they are only modified a few times during the bilingual training. As shown in last column, neighborhood of "laggards" still consists of other plural forms even after bilingual training.

7 Conclusion

In this paper, we explores applying deep neural network for word alignment task. Our model integrates a multi-layer neural network into an HMM-like framework, where context dependent lexical translation score is computed by neural network, and distortion is modeled by a simple jump-distance scheme. Our model is discriminatively trained on bilingual corpus, while huge monolingual data is used to pre-train word-embeddings. Experiments on large-scale Chinese-to-English task show that the proposed method produces better word alignment results, compared with both classic HMM model and IBM model 4.

For future work, we will investigate more settings of different hyper-parameters in our model. Secondly, we want to explore the possibility of unsupervised training of our neural word alignment model, without reliance of alignment result of other models. Furthermore, our current model use rather simple distortions; it might be helpful to use more sophisticated model such as ITG (Wu, 1997), which can be modeled by Recursive Neural Networks (Socher et al., 2011).

Acknowledgments

We thank anonymous reviewers for insightful comments. We also thank Dongdong Zhang, Lei Cui, Chunyang Wu and Zhenyan He for fruitful discussions.

References

Yoshua Bengio, Holger Schwenk, Jean-Sébastien Senécal, Fréderic Morin, and Jean-Luc Gauvain. 2006. Neural probabilistic language models. *Innovations in Machine Learning*, pages 137–186.

Yoshua Bengio, Pascal Lamblin, Dan Popovici, and

Hugo Larochelle. 2007. Greedy layer-wise training of deep networks. *Advances in neural information processing systems*, 19:153.

Yoshua Bengio. 2009. Learning deep architectures for ai. *Foundations and Trends® in Machine Learning*, 2(1):1–127.

Adam L. Berger, Vincent J. Della Pietra, and Stephen A. Della Pietra. 1996. A maximum entropy approach to natural language processing. *Comput. Linguist.*, 22(1):39–71, March.

JS Bridle. 1990. Neurocomputing: Algorithms, architectures and applications, chapter probabilistic interpretation of feedforward classification network outputs, with relationships to statistical pattern recognition.

Peter F Brown, Vincent J Della Pietra, Stephen A Della Pietra, and Robert L Mercer. 1993. The mathematics of statistical machine translation: Parameter estimation. *Computational linguistics*, 19(2):263–311.

David Chiang. 2007. Hierarchical phrase-based translation. *computational linguistics*, 33(2):201–228.

Ronan Collobert, Jason Weston, Léon Bottou, Michael Karlen, Koray Kavukcuoglu, and Pavel Kuksa. 2011. Natural language processing (almost) from scratch. *The Journal of Machine Learning Research*, 12:2493–2537.

George E Dahl, Dong Yu, Li Deng, and Alex Acero. 2012. Context-dependent pre-trained deep neural networks for large-vocabulary speech recognition. *Audio, Speech, and Language Processing, IEEE Transactions on*, 20(1):30–42.

John DeNero and Klaus Macherey. 2011. Model-based aligner combination using dual decomposition. In *Proc. ACL*.

Chris Dyer, Jonathan Clark, Alon Lavie, and Noah A Smith. 2011. Unsupervised word alignment with arbitrary features. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies-Volume 1*, pages 409–419. Association for Computational Linguistics.

Aria Haghighi, John Blitzer, John DeNero, and Dan Klein. 2009. Better word alignments with supervised itg models. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP: Volume 2-Volume 2*, pages 923–931. Association for Computational Linguistics.

Geoffrey E Hinton, Simon Osindero, and Yee-Whye Teh. 2006. A fast learning algorithm for deep belief nets. *Neural computation*, 18(7):1527–1554.

Koray Kavukcuoglu, Pierre Sermanet, Y-Lan Boureau, Karol Gregor, Michaël Mathieu, and Yann LeCun. 2010. Learning convolutional feature hierarchies for visual recognition. *Advances in Neural Information Processing Systems*, pages 1090–1098.

Philipp Koehn, Franz Josef Och, and Daniel Marcu. 2003. Statistical phrase-based translation. In *Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology-Volume 1*, pages 48–54. Association for Computational Linguistics.

Alex Krizhevsky, Ilya Sutskever, and Geoff Hinton. 2012. Imagenet classification with deep convolutional neural networks. In *Advances in Neural Information Processing Systems 25*, pages 1106–1114.

Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. 1998. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324.

Yann LeCun. 1985. A learning scheme for asymmetric threshold networks. *Proceedings of Cognitiva*, 85:599–604.

Honglak Lee, Alexis Battle, Rajat Raina, and Andrew Y Ng. 2007. Efficient sparse coding algorithms. *Advances in neural information processing systems*, 19:801.

Shujie Liu, Chi-Ho Li, and Ming Zhou. 2010. Discriminative pruning for discriminative itg alignment. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics, ACL*, volume 10, pages 316–324.

Y MarcAurelio Ranzato, Lan Boureau, and Yann LeCun. 2007. Sparse feature learning for deep belief networks. *Advances in neural information processing systems*, 20:1185–1192.

Robert C Moore. 2005. A discriminative framework for bilingual word alignment. In *Proceedings of the conference on Human Language Technology and Empirical Methods in Natural Language Processing*, pages 81–88. Association for Computational Linguistics.

Jan Niehues and Alex Waibel. 2012. Continuous space language models using restricted boltzmann machines. In *Proceedings of the nineth International Workshop on Spoken Language Translation (IWSLT)*.

Franz Josef Och and Hermann Ney. 2000. Giza++: Training of statistical translation models.

Frank Seide, Gang Li, and Dong Yu. 2011. Conversational speech transcription using context-dependent deep neural networks. In *Proc. Interspeech*, pages 437–440.

Noah A Smith and Jason Eisner. 2005. Contrastive estimation: Training log-linear models on unlabeled data. In *Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics*, pages 354–362. Association for Computational Linguistics.

Richard Socher, Cliff C Lin, Andrew Y Ng, and Christopher D Manning. 2011. Parsing natural scenes and natural language with recursive neural networks. In *Proceedings of the 26th International Conference on Machine Learning (ICML)*, volume 2, page 7.

Richard Socher, Brody Huval, Christopher D Manning, and Andrew Y Ng. 2012. Semantic compositionality through recursive matrix-vector spaces. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 1201–1211. Association for Computational Linguistics.

Le Hai Son, Alexandre Allauzen, and François Yvon. 2012. Continuous space translation models with neural networks. In *Proceedings of the 2012 conference of the north american chapter of the association for computational linguistics: Human language technologies*, pages 39–48. Association for Computational Linguistics.

Ivan Titov, Alexandre Klementiev, and Binod Bhattarai. 2012. Inducing crosslingual distributed representations of words.

Joseph Turian, Lev Ratinov, and Yoshua Bengio. 2010. Word representations: a simple and general method for semi-supervised learning. *Urbana*, 51:61801.

Stephan Vogel, Hermann Ney, and Christoph Tillmann. 1996. Hmm-based word alignment in statistical translation. In *Proceedings of the 16th conference on Computational linguistics-Volume 2*, pages 836–841. Association for Computational Linguistics.

Dekai Wu. 1997. Stochastic inversion transduction grammars and bilingual parsing of parallel corpora. *Computational linguistics*, 23(3):377–403.