# A Recursive Recurrent Neural Network for Statistical Machine Translation

**Shujie Liu**[1], **Nan Yang**[2], **Mu Li**[1] **and Ming Zhou**[1]
[1]Microsoft Research Asia, Beijing, China
[2]University of Science and Technology of China, Hefei, China
shujliu, v-nayang, muli, mingzhou@microsoft.com

## Abstract

In this paper, we propose a novel recursive recurrent neural network ($R^2NN$) to model the end-to-end decoding process for statistical machine translation. $R^2NN$ is a combination of recursive neural network and recurrent neural network, and in turn integrates their respective capabilities: (1) new information can be used to generate the next hidden state, like recurrent neural networks, so that language model and translation model can be integrated naturally; (2) a tree structure can be built, as recursive neural networks, so as to generate the translation candidates in a bottom up manner. A semi-supervised training approach is proposed to train the parameters, and the phrase pair embedding is explored to model translation confidence directly. Experiments on a Chinese to English translation task show that our proposed $R^2NN$ can outperform the state-of-the-art baseline by about 1.5 points in BLEU.

## 1 Introduction

Deep Neural Network (DNN), which essentially is a multi-layer neural network, has re-gained more and more attentions these years. With the efficient training methods, such as (Hinton et al., 2006), DNN is widely applied to speech and image processing, and has achieved breakthrough results (Kavukcuoglu et al., 2010; Krizhevsky et al., 2012; Dahl et al., 2012).

Applying DNN to natural language processing (NLP), representation or embedding of words is usually learnt first. Word embedding is a dense, low dimensional, real-valued vector. Each dimension of the vector represents a latent aspect of the word, and captures its syntactic and semantic properties (Bengio et al., 2006). Word embedding is usually learnt from large amount of monolingual corpus at first, and then fine tuned for special distinct tasks. Collobert et al. (2011) propose a multi-task learning framework with DNN for various NLP tasks, including part-of-speech tagging, chunking, named entity recognition, and semantic role labelling. Recurrent neural networks are leveraged to learn language model, and they keep the history information circularly inside the network for arbitrarily long time (Mikolov et al., 2010). Recursive neural networks, which have the ability to generate a tree structured output, are applied to natural language parsing (Socher et al., 2011), and they are extended to recursive neural tensor networks to explore the compositional aspect of semantics (Socher et al., 2013).

DNN is also introduced to Statistical Machine Translation (SMT) to learn several components or features of conventional framework, including word alignment, language modelling, translation modelling and distortion modelling. Yang et al. (2013) adapt and extend the CD-DNN-HMM (Dahl et al., 2012) method to HMM-based word alignment model. In their work, bilingual word embedding is trained to capture lexical translation information, and surrounding words are utilized to model context information. Auli et al. (2013) propose a joint language and translation model, based on a recurrent neural network. Their model predicts a target word, with an unbounded history of both source and target words. Liu et al. (2013) propose an additive neural network for SMT decoding. Word embedding is used as the input to learn translation confidence score, which is combined with commonly used features in the conventional log-linear model. For distortion modeling, Li et al. (2013) use recursive auto encoders to make full use of the entire merging phrase pairs, going beyond the boundary words with a maximum entropy classifier (Xiong et al., 2006).



*Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics*, pages 1491–1500,
Baltimore, Maryland, USA, June 23-25 2014. 

Different from the work mentioned above, which applies DNN to components of conventional SMT framework, in this paper, we propose a novel $R^2NN$ to model the end-to-end decoding process. $R^2NN$ is a combination of recursive neural network and recurrent neural network. In $R^2NN$, new information can be used to generate the next hidden state, like recurrent neural networks, and a tree structure can be built, as recursive neural networks. To generate the translation candidates in a commonly used bottom-up manner, recursive neural networks are naturally adopted to build the tree structure. In recursive neural networks, all the representations of nodes are generated based on their child nodes, and it is difficult to integrate additional global information, such as language model and distortion model. In order to integrate these crucial information for better translation prediction, we combine recurrent neural networks into the recursive neural networks, so that we can use global information to generate the next hidden state, and select the better translation candidate.

We propose a three-step semi-supervised training approach to optimizing the parameters of $R^2NN$, which includes recursive auto-encoding for unsupervised pre-training, supervised local training based on the derivation trees of forced decoding, and supervised global training using early update strategy. So as to model the translation confidence for a translation phrase pair, we initialize the phrase pair embedding by leveraging the sparse features and recurrent neural network. The sparse features are phrase pairs in translation table, and recurrent neural network is utilized to learn a smoothed translation score with the source and target side information. We conduct experiments on a Chinese-to-English translation task to test our proposed methods, and we get about 1.5 BLEU points improvement, compared with a state-of-the-art baseline system.

The rest of this paper is organized as follows: Section 2 introduces related work on applying DNN to SMT. Our $R^2NN$ framework is introduced in detail in Section 3, followed by our three-step semi-supervised training approach in Section 4. Phrase pair embedding method using translation confidence is elaborated in Section 5. We introduce our conducted experiments in Section 6, and conclude our work in Section 7.

## 2 Related Work

Yang et al. (2013) adapt and extend CD-DNN-HMM (Dahl et al., 2012) to word alignment. In their work, initial word embedding is firstly trained with a huge mono-lingual corpus, then the word embedding is adapted and fine tuned bilingually in a context-depended DNN HMM framework. Word embeddings capturing lexical translation information and surrounding words modeling context information are leveraged to improve the word alignment performance. Unfortunately, the better word alignment result generated by this model, cannot bring significant performance improvement on a end-to-end SMT evaluation task.

To improve the SMT performance directly, Auli et al. (2013) extend the recurrent neural network language model, in order to use both the source and target side information to scoring translation candidates. In their work, not only the target word embedding is used as the input of the network, but also the embedding of the source word, which is aligned to the current target word. To tackle the large search space due to the weak independence assumption, a lattice algorithm is proposed to rerank the n-best translation candidates, generated by a given SMT decoder.

Liu et al. (2013) propose an additive neural network for SMT decoding. RNNLM (Mikolov et al., 2010) is firstly used to generate the source and target word embeddings, which are fed into a one-hidden-layer neural network to get a translation confidence score. Together with other commonly used features, the translation confidence score is integrated into a conventional log-linear model. The parameters are optimized with development data set using mini-batch conjugate sub-gradient method and a regularized ranking loss.

DNN is also brought into the distortion modeling. Going beyond the previous work using boundary words for distortion modeling in BTG-based SMT decoder, Li et al. (2013) propose to apply recursive auto-encoder to make full use of the entire merged blocks. The recursive auto-encoder is trained with reordering examples extracted from word-aligned bilingual sentences. Given the representations of the smaller phrase pairs, recursive auto-encoder can generate the representation of the parent phrase pair with a re-ordering confidence score. The combination of reconstruction error and re-ordering error is used to be the objective function for the model training.

## 3 Our Model

In this section, we leverage DNN to model the end-to-end SMT decoding process, using a novel recursive recurrent neural network (R$^2$NN), which is different from the above mentioned work applying DNN to components of conventional SMT framework. R$^2$NN is a combination of recursive neural network and recurrent neural network, which not only integrates the conventional global features as input information for each combination, but also generates the representation of the parent node for the future candidate generation.

In this section, we briefly recall the recurrent neural network and recursive neural network in Section 3.1 and 3.2, and then we elaborate our R$^2$NN in detail in Section 3.3.

### 3.1 Recurrent Neural Network

Recurrent neural network is usually used for sequence processing, such as language model (Mikolov et al., 2010). Commonly used sequence processing methods, such as Hidden Markov Model (HMM) and n-gram language model, only use a limited history for the prediction. In HMM, the previous state is used as the history, and for n-gram language model (for example $n$ equals to 3), the history is the previous two words. Recurrent neural network is proposed to use unbounded history information, and it has recurrent connections on hidden states, so that history information can be used circularly inside the network for arbitrarily long time.

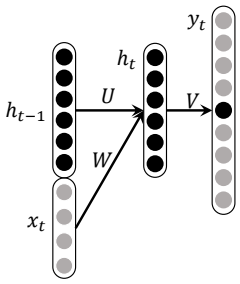


Figure 1: Recurrent neural network

As shown in Figure 1, the network contains three layers, an input layer, a hidden layer, and an output layer. The input layer is a concatenation of $h_{t-1}$ and $x_t$, where $h_{t-1}$ is a real-valued vector, which is the history information from time 0 to $t-1$. $x_t$ is the embedding of the input word at time $t$. Word embedding $x_t$ is integrated with previous history $h_{t-1}$ to generate the current hidden layer, which is a new history vector $h_t$. Based on $h_t$, we can predict the probability of the next word, which forms the output layer $y_t$. The new history $h_t$ is used for the future prediction, and updated with new information from word embedding $x_t$ recurrently.

### 3.2 Recursive Neural Network

In addition to the sequential structure above, tree structure is also usually constructed in various NLP tasks, such as parsing and SMT decoding. To generate a tree structure, recursive neural networks are introduced for natural language parsing (Socher et al., 2011). Similar with recurrent neural networks, recursive neural networks can also use unbounded history information from the sub-tree rooted at the current node. The commonly used binary recursive neural networks generate the representation of the parent node, with the representations of two child nodes as the input.

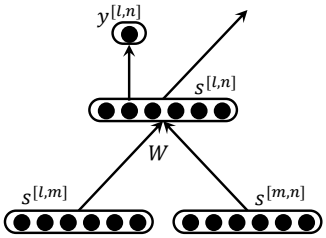


Figure 2: Recursive neural network

As shown in Figure 2, $s^{[l,m]}$ and $s^{[m,n]}$ are the representations of the child nodes, and they are concatenated into one vector to be the input of the network. $s^{[l,n]}$ is the generated representation of the parent node. $y_{[l,n]}$ is the confidence score of how plausible the parent node should be created. $l, m, n$ are the indexes of the string. For example, for nature language parsing, $s^{[l,n]}$ is the representation of the parent node, which could be a $NP$ or $VP$ node, and it is also the representation of the whole sub-tree covering from $l$ to $n$.

### 3.3 Recursive Recurrent Neural Network

Word embedding $x_t$ is integrated as new input information in recurrent neural networks for each prediction, but in recursive neural networks, no additional input information is used except the two representation vectors of the child nodes. However, some global information , which cannot be generated by the child representations, is crucial

for SMT performance, such as language model score and distortion model score. So as to integrate such global information, and also keep the ability to generate tree structure, we combine the recurrent neural network and the recursive neural network to be a recursive recurrent neural network (R$^2$NN).

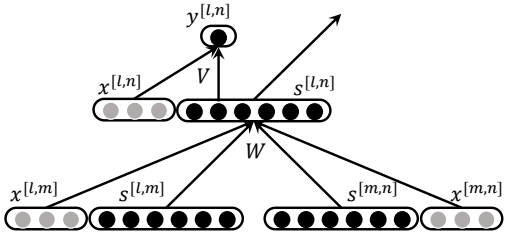


Figure 3: Recursive recurrent neural network

As shown in Figure 3, based on the recursive network, we add three input vectors $x^{[l,m]}$ for child node $[l,m]$, $x^{[m,n]}$ for child node $[m,n]$, and $x^{[l,n]}$ for parent node $[l,n]$. We call them recurrent input vectors, since they are borrowed from recurrent neural networks. The two recurrent input vectors $x^{[l,m]}$ and $x^{[m,n]}$ are concatenated as the input of the network, with the original child node representations $s^{[l,m]}$ and $s^{[m,n]}$. The recurrent input vector $x^{[l,n]}$ is concatenated with parent node representation $s^{[l,n]}$ to compute the confidence score $y^{[l,n]}$.

The input, hidden and output layers are calculated as follows:

$$\hat{x}^{[l,n]} = x^{[l,m]} \bowtie s^{[l,m]} \bowtie x^{[m,n]} \bowtie s^{[m,n]} \quad (1)$$

$$s_j^{[l,n]} = f(\sum_i \hat{x}_i^{[l,n]} w_{ji}) \quad (2)$$

$$y^{[l,n]} = \sum_j (s^{[l,n]} \bowtie x^{[l,n]})_j v_j \quad (3)$$

where $\bowtie$ is a concatenation operator in Equation 1 and Equation 3, and $f$ is a non-linear function, here we use $HTanh$ function, which is defined as:

$$HTanh(x) = \begin{cases} -1, & x < -1 \\ x, & -1 \leq x \geq 1 \\ 1, & x > 1 \end{cases} \quad (4)$$

Figure 4 illustrates the R$^2$NN architecture for SMT decoding. For a source sentence "laizi faguo he eluosi de", we first split it into phrases "laizi", "faguo he eluosi" and "de". We then check whether translation candidates can be found in the translation table for each span, together with the phrase pair embedding and recurrent input vector (global features). We call it the rule matching phase. For a translation candidate of the span node $[l,m]$, the black dots stand for the node representation $s^{[l,m]}$, while the grey dots for recurrent input vector $x^{[l,m]}$. Given $s^{[l,m]}$ and $x^{[l,m]}$ for matched translation candidates, conventional CKY decoding process is performed using R$^2$NN. R$^2$NN can combine the translation pairs of child nodes, and generate the translation candidates for parent nodes with their representations and plausible scores. Only the n-best translation candidates are kept for upper combination, according to their plausible scores.

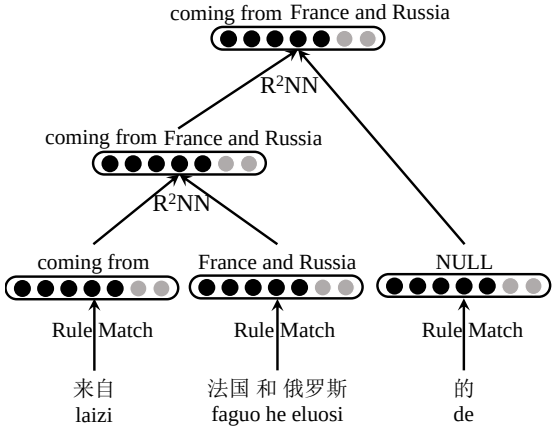


Figure 4: R$^2$NN for SMT decoding

We extract phrase pairs using the conventional method (Och and Ney, 2004). The commonly used features, such as translation score, language model score and distortion score, are used as the recurrent input vector $x$. During decoding, recurrent input vectors $x$ for internal nodes are calculated accordingly. The difference between our model and the conventional log-linear model includes:

- R$^2$NN is not linear, while the conventional model is a linear combination.
- Representations of phrase pairs are automatically learnt to optimize the translation performance, while features used in conventional model are hand-crafted.
- History information of the derivation can be recorded in the representation of internal nodes, while conventional model cannot.

Liu et al. (2013) apply DNN to SMT decoding, but not in a recursive manner. A feature is learnt via a one-hidden-layer neural network, and the embedding of words in the phrase pairs are used as the input vector. Our model generates the representation of a translation pair based on its child nodes. Li et al. (2013) also generate the representation of phrase pairs in a recursive way. In their work, the representation is optimized to learn a distortion model using recursive neural network, only based on the representation of the child nodes. Our R$^2$NN is used to model the end-to-end translation process, with recurrent global information added. We also explore phrase pair embedding method to model translation confidence directly, which is introduced in Section 5.

In the next two sections, we will answer the following questions: (a) how to train the model, and (b) how to generate the initial representations of translation pairs.

## 4 Model Training

In this section, we propose a three-step training method to train the parameters of our proposed R$^2$NN, which includes unsupervised pre-training using recursive auto-encoding, supervised local training on the derivation tree of forced decoding, and supervised global training using early update training strategy.

### 4.1 Unsupervised Pre-training

We adopt the Recursive Auto Encoding (RAE) (Socher et al., 2011) for our unsupervised pre-training. The main idea of auto encoding is to initialize the parameters of the neural network, by minimizing the information lost, which means, capturing as much information as possible in the hidden states from the input vector.

As shown in Figure 5, RAE contains two parts, an encoder with parameter $W$, and a decoder with parameter $W'$. Given the representations of child nodes $s_1$ and $s_2$, the encoder generates the representation of parent node $s$. With the parent node representation $s$ as the input vector, the decoder reconstructs the representation of two child nodes $s_1'$ and $s_2'$. The loss function is defined as following so as to minimize the information lost:

$$L_{RAE}(s_1, s_2) = \frac{1}{2}(\|s_1 - s_1'\|^2 + \|s_2 - s_2'\|^2) \quad (5)$$

where $\|\cdot\|$ is the Euclidean norm.

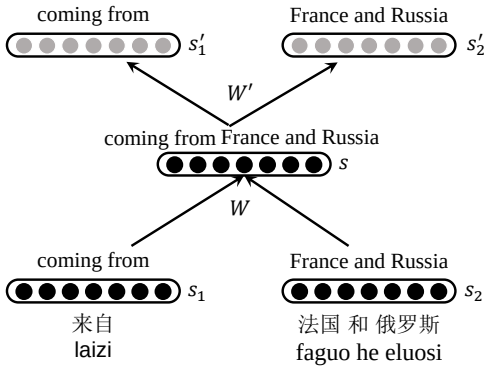


Figure 5: Recursive auto encoding for unsupervised pre-training

The training samples for RAE are phrase pairs $\{s_1, s_2\}$ in translation table, where $s_1$ and $s_2$ can form a continuous partial sentence pair in the training data. When RAE training is done, only the encoding model $W$ will be fine tuned in the future training phases.

### 4.2 Supervised Local Training

We use contrastive divergence method to fine tune the parameters $W$ and $V$. The loss function is the commonly used ranking loss with a margin, and it is defined as follows:

$$L_{SLT}(W, V, s^{[l,n]}) = max(0, 1 - y_{oracle}^{[l,n]} + y_t^{[l,n]}) \quad (6)$$

where $s^{[l,n]}$ is the source span. $y_{oracle}^{[l,n]}$ is the plausible score of a oracle translation result. $y_t^{[l,n]}$ is the plausible score for the best translation candidate given the model parameters $W$ and $V$. The loss function aims to learn a model which assigns the good translation candidate (the oracle candidate) higher score than the bad ones, with a margin 1.

Translation candidates generated by forced decoding (Wuebker et al., 2010) are used as oracle translations, which are the positive samples. Forced decoding performs sentence pair segmentation using the same translation system as decoding. For each sentence pair in the training data, SMT decoder is applied to the source side, and any candidate which is not the partial sub-string of the target sentence is removed from the n-best list during decoding. From the forced decoding result, we can get the ideal derivation tree in the decoder's search space, and extract positive/oracle translation candidates.

### 4.3 Supervised Global Training

The supervised local training uses the n-odes/samples in the derivation tree of forced decoding to update the model, and the trained model tends to over-fit to local decisions. In this subsection, a supervised global training is proposed to tune the model according to the final translation performance of the whole source sentence.

Actually, we can update the model from the root of the decoding tree and perform back propagation along the tree structure. Due to the inexact search nature of SMT decoding, search errors may inevitably break theoretical properties, and the final translation results may be not suitable for model training. To handle this problem, we use early update strategy for the supervised global training. Early update is testified to be useful for SMT training with large scale features (Yu et al., 2013). Instead of updating the model using the final translation results, early update approach optimizes the model, when the oracle translation candidate is pruned from the n-best list, meaning that, the model is updated once it performs a unrecoverable mistake. Back propagation is performed along the tree structure, and the phrase pair embeddings of the leaf nodess are updated.

The loss function for supervised global training is defined as follows:

$$L_{SGT}(W, V, s^{[l,n]}) = -\log(\frac{\sum_{y^{[l,n]}_{oracle}} \exp(y^{[l,n]}_{oracle})}{\sum_{t \in nbest} \exp(y^{[l,n]}_{t})}) \quad (7)$$

where $y^{[l,n]}_{oracle}$ is the model score of a oracle translation candidate for the span $[l, n]$. Oracle translation candidates are candidates get from forced decoding. If the span $[l, n]$ is not the whole source sentence, there may be several oracle translation candidates, otherwise, there is only one, which is exactly the target sentence. There are much fewer training samples than those for supervised local training, and it is not suitable to use ranking loss for global training any longer. We use negative log-likelihood to penalize all the other translation candidates except the oracle ones, so as to leverage all the translation candidates as training samples.

## 5 Phrase Pair Embedding

The next question is how to initialize the phrase pair embedding in the translation table, so as to generate the leaf nodes of the derivation tree. There are more phrase pairs than mono-lingual words, but bilingual corpus is much more difficult to acquire, compared with monolingual corpus.

| Embedding | #Data | #Entry | #Parameter |
|---|---|---|---|
| Word | 1G | 500K | $20 \times 500K$ |
| Word Pair | 7M | $(500K)^2$ | $20 \times (500K)^2$ |
| Phrase Pair | 7M | $(500K)^4$ | $20 \times (500K)^4$ |

Table 1: The relationship between the size of training data and the number of model parameters. The numbers for word embedding is calculated on English Giga-Word corpus version 3. For word pair and phrase pair embedding, the numbers are calculated on IWSLT 2009 dialog training set. The word count of each side of phrase pairs is limited to be 2.

Table 1 shows the relationship between the size of training data and the number of model parameters. For word embedding, the training size is 1G bits, and we may have 500K terms. For each term, we have a vector with length 20 as parameters, so there are $20 \times 500K$ parameters totally. But for source-target word pair, we may only have 7M bilingual corpus for training (taking IWSLT data set as an example), and there are $20 \times (500K)^2$ parameters to be tuned. For phrase pairs, the situation becomes even worse, especially when the limitation of word count in phrase pairs is relaxed. It is very difficult to learn the phrase pair embedding brute-forcedly as word embedding is learnt (Mikolov et al., 2010; Collobert et al., 2011), since we may not have enough training data.

A simple approach to construct phrase pair embedding is to use the average of the embeddings of the words in the phrase pair. One problem is that, word embedding may not be able to model the translation relationship between source and target phrases at phrase level, since some phrases cannot be decomposed. For example, the meaning of "hot dog" is not the composition of the meanings of the words "hot" and "dog". In this section, we split the phrase pair embedding into two parts to model the translation confidence directly: translation confidence with sparse features and translation confidence with recurrent neural network. We first get two translation confidence vectors separately using sparse features and recurrent neural network, and then concatenate them to be the phrase pair embedding. We call it translation confidence based phrase pair embedding (TCBPPE).

### 5.1 Translation Confidence with Sparse Features

Large scale feature training has drawn more attentions these years (Liang et al., 2006; Yu et al., 2013). Instead of integrating the sparse features directly into the log-linear model, we use them as the input to learn a phrase pair embedding. For the top 200,000 frequent translation pairs, each of them is a feature in itself, and a special feature is added for all the infrequent ones.

The one-hot representation vector is used as the input, and a one-hidden-layer network generates a confidence score. To train the neural network, we add the confidence scores to the conventional log-linear model as features. Forced decoding is utilized to get positive samples, and contrastive divergence is used for model training. The neural network is used to reduce the space dimension of sparse features, and the hidden layer of the network is used as the phrase pair embedding. The length of the hidden layer is empirically set to 20.

### 5.2 Translation Confidence with Recurrent Neural Network

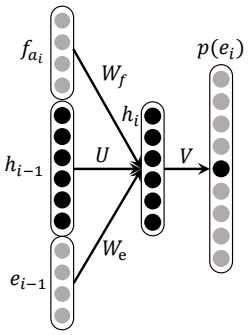


Figure 6: Recurrent neural network for translation confidence

We use recurrent neural network to generate two smoothed translation confidence scores based on source and target word embeddings. One is source to target translation confidence score and the other is target to source. These two confidence scores are defined as:

$$T_{S2T}(s,t) = \sum_i \log p(e_i|e_{i-1}, f_{a_i}, h_i) \quad (8)$$

$$T_{T2S}(s,t) = \sum_j \log p(f_j|f_{j-1}, e_{\hat{a}_j}, h_j) \quad (9)$$

where, $f_{a_i}$ is the corresponding target word aligned to $e_i$, and it is similar for $e_{\hat{a}_j}$. $p(e_i|e_{i-1}, f_{a_i}, h_i)$ is produced by a recurrent network as shown in Figure 6. The recurrent neural network is trained with word aligned bilingual corpus, similar as (Auli et al., 2013).

## 6 Experiments and Results

In this section, we conduct experiments to test our method on a Chinese-to-English translation task. The evaluation method is the case insensitive IBM BLEU-4 (Papineni et al., 2002). Significant testing is carried out using bootstrap re-sampling method proposed by (Koehn, 2004) with a 95% confidence level.

### 6.1 Data Setting and Baseline

The data is from the IWSLT 2009 dialog task. The training data includes the BTEC and SLDB training data. The training data contains 81k sentence pairs, 655K Chinese words and 806K English words. The language model is a 5-gram language model trained with the target sentences in the training data. The test set is development set 9, and the development set comprises both development set 8 and the Chinese DIALOG set.

The training data for monolingual word embedding is Giga-Word corpus version 3 for both Chinese and English. Chinese training corpus contains 32M sentences and 1.1G words. English training data contains 8M sentences and 247M terms. We only train the embedding for the top 100,000 frequent words following (Collobert et al., 2011). With the trained monolingual word embedding, we follow (Yang et al., 2013) to get the bilingual word embedding using the IWSLT bilingual training data.

Our baseline decoder is an in-house implementation of Bracketing Transduction Grammar (BTG) (Wu, 1997) in CKY-style decoding with a lexical reordering model trained with maximum entropy (Xiong et al., 2006). The features of the baseline are commonly used features as standard BTG decoder, such as translation probabilities, lexical weights, language model, word penalty and distortion probabilities. All these commonly used features are used as recurrent input vector $x$ in our R$^2$NN.

### 6.2 Translation Results

As we mentioned in Section 5, constructing phrase pair embeddings from word embeddings may be not suitable. Here we conduct experiments to ver-

ify it. We first train the source and target word embeddings separately using large monolingual data, following (Collobert et al., 2011). Using monolingual word embedding as the initialization, we fine tune them to get bilingual word embedding (Yang et al., 2013).

The word embedding based phrase pair embedding (WEPPE) is defined as:

$$Epp_{web}(s,t) = \sum_i E_{wms}(s_i) \bowtie \sum_j E_{wbs}(s_j) \bowtie \sum_k E_{wmt}(t_k) \bowtie \sum_l E_{wbt}(t_l) \quad (10)$$

where $\bowtie$ is a concatenation operator. $s$ and $t$ are the source and target phrases. $E_{wms}(s_i)$ and $E_{wmt}(t_k)$ are the monolingual word embeddings, and $E_{wbs}(s_i)$ and $E_{wbt}(t_k)$ are the bilingual word embeddings. Here the length of the word embedding is also set to 20. Therefore, the length of the phrase pair embedding is $20 \times 4 = 80$ .

We compare our phrase pair embedding methods and our proposed R$^2$NN with baseline system, in Table 2. We can see that, our R$^2$NN models with WEPPE and TCBPPE are both better than the baseline system. WEPPE cannot get significant improvement, while TCBPPE does, compared with the baseline result. TCBPPE is much better than WEPPE.

| Setting | Development | Test |
|---|---|---|
| Baseline | 46.81 | 39.29 |
| WEPPE+R$^2$NN | 47.23 | 39.92 |
| TCBPPE+R$^2$NN | 48.70 $\uparrow$ | 40.81 $\uparrow$ |

Table 2: Translation results of our proposed R$^2$NN Model with two phrase embedding methods, compared with the baseline. Setting "WEPPE+R$^2$NN" is the result with word embedding based phrase pair embedding and our R$^2$NN Model, and "TCBPPE+R$^2$NN" is the result of translation confidence based phrase pair embedding and our R$^2$NN Model. The results with $\uparrow$ are significantly better than the baseline.

Word embedding can model translation relationship at word level, but it may not be powerful to model the phrase pair respondents at phrasal level, since the meaning of some phrases cannot be decomposed into the meaning of words. And also, translation task is difference from other NLP tasks, that, it is more important to model the translation confidence directly (the confidence of one target phrase as a translation of the source phrase), and our TCBPPE is designed for such purpose.

### 6.3 Effects of Global Recurrent Input Vector

In order to compare R$^2$NN with recursive network for SMT decoding, we remove the recurrent input vector in R$^2$NN to test its effect, and the results are shown in Table 3. Without the recurrent input vectors, R$^2$NN degenerates into recursive neural network (RNN).

| Setting | Development | Test |
|---|---|---|
| WEPPE+R$^2$NN | 47.23 | 40.81 |
| WEPPE+RNN | 37.62 | 33.29 |
| TCBPPE+R$^2$NN | 48.70 | 40.81 |
| TCBPPE+RNN | 45.11 | 37.33 |

Table 3: Experimental results to test the effects of recurrent input vector. WEPPE /TCBPPE+RNN are the results removing recurrent input vectors with WEPPE /TCBPPE.

From Table 3 we can find that, the recurrent input vector is essential to SMT performance. When we remove it from R$^2$NN, WEPPE based method drops about 10 BLEU points on development data and more than 6 BLEU points on test data. TCBPPE based method drops about 3 BLEU points on both development and test data sets. When we remove the recurrent input vectors, the representations of recursive network are generated with the child nodes, and it does not integrate global information, such as language model and distortion model, which are crucial to the performance of SMT.

### 6.4 Sparse Features and Recurrent Network Features

To test the contributions of sparse features and recurrent network features, we first remove all the recurrent network features to train and test our R$^2$NN model, and then remove all the sparse features to test the contribution of recurrent network features.

| Setting | Development | Test |
|---|---|---|
| TCBPPE+R$^2$NN | 48.70 | 40.81 |
| SF+R$^2$NN | 48.23 | 40.19 |
| RNN+R$^2$NN | 47.89 | 40.01 |

Table 4: Experimental results to test the effects of sparse features and recurrent network features.

The results are shown in Table 6.4. From the results, we can find that, sparse features are more effective than the recurrent network features a little bit. The sparse features can directly model the translation correspondence, and they may be more effective to rank the translation candidates, while recurrent neural network features are smoothed lexical translation confidence.

## 7 Conclusion and Future Work

In this paper, we propose a Recursive Recurrent Neural Network($R^2$NN) to combine the recurrent neural network and recursive neural network. Our proposed $R^2$NN cannot only integrate global input information during each combination, but also can generate the tree structure in a recursive way. We apply our model to SMT decoding, and propose a three-step semi-supervised training method. In addition, we explore phrase pair embedding method, which models translation confidence directly. We conduct experiments on a Chinese-to-English translation task, and our method outperforms a state-of-the-art baseline about 1.5 points BLEU.

From the experiments, we find that, phrase pair embedding is crucial to the performance of SMT. In the future, we will explore better methods for phrase pair embedding to model the translation equivalent between source and target phrases. We will apply our proposed $R^2$NN to other tree structure learning tasks, such as natural language parsing.

## References


Michael Auli, Michel Galley, Chris Quirk, and Geoffrey Zweig. 2013. Joint language and translation modeling with recurrent neural networks. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1044–1054, Seattle, Washington, USA, October. Association for Computational Linguistics.

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.

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.

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. 2004. Statistical significance tests for machine translation evaluation. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 388–395.

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.

Peng Li, Yang Liu, and Maosong Sun. 2013. Recursive autoencoders for ITG-based translation. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 567–577, Seattle, Washington, USA, October. Association for Computational Linguistics.

Percy Liang, Alexandre Bouchard-Côté, Dan Klein, and Ben Taskar. 2006. An end-to-end discriminative approach to machine translation. In *Proceedings of the 21st International Conference on Computational Linguistics and the 44th annual meeting of the Association for Computational Linguistics*, pages 761–768. Association for Computational Linguistics.

Lemao Liu, Taro Watanabe, Eiichiro Sumita, and Tiejun Zhao. 2013. Additive neural networks for statistical machine translation. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics*, pages 791–801, Sofia, Bulgaria, August. Association for Computational Linguistics.

Tomas Mikolov, Martin Karafiát, Lukas Burget, Jan Cernockỳ, and Sanjeev Khudanpur. 2010. Recurrent neural network based language model. In *Proceedings of the Annual Conference of International Speech Communication Association*, pages 1045–1048.

Franz Josef Och and Hermann Ney. 2004. The alignment template approach to statistical machine translation. *Computational linguistics*, 30(4):417–449.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th annual meeting on association for computational linguistics*, pages 311–318. Association for Computational Linguistics.

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, John Bauer, and Christopher D Manning. 2013. Parsing with compositional vector grammars. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics*, volume 1, pages 455–465.

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

Joern Wuebker, Arne Mauser, and Hermann Ney. 2010. Training phrase translation models with leaving-one-out. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, pages 475–484. Association for Computational Linguistics.

Deyi Xiong, Qun Liu, and Shouxun Lin. 2006. Maximum entropy based phrase reordering model for statistical machine translation. In *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics*, volume 44, page 521.

Nan Yang, Shujie Liu, Mu Li, Ming Zhou, and Nenghai Yu. 2013. Word alignment modeling with context dependent deep neural network. In *51st Annual Meeting of the Association for Computational Linguistics*.

Heng Yu, Liang Huang, Haitao Mi, and Kai Zhao. 2013. Max-violation perceptron and forced decoding for scalable MT training. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1112–1123, Seattle, Washington, USA, October. Association for Computational Linguistics.