Recursive Autoencoders for ITG-based Translation

Peng Li, Yang Liu and **Maosong Sun**
State Key Laboratory of Intelligent Technology and Systems
Tsinghua National Laboratory for Information Science and Technology
Department of Computer Science and Technology, Tsinghua University, Beijing 100084, China
pengli09@gmail.com, {liuyang2011,sms}@tsinghua.edu.cn

Abstract

While inversion transduction grammar (ITG) is well suited for modeling ordering shifts between languages, how to make applying the two reordering rules (i.e., straight and inverted) dependent on actual blocks being merged remains a challenge. Unlike previous work that only uses boundary words, we propose to use recursive autoencoders to make full use of the entire merging blocks alternatively. The recursive autoencoders are capable of generating vector space representations for variable-sized phrases, which enable predicting orders to exploit syntactic and semantic information from a neural language modeling's perspective. Experiments on the NIST 2008 dataset show that our system significantly improves over the MaxEnt classifier by 1.07 BLEU points.

1 Introduction

Phrase-based models (Koehn et al., 2003; Och and Ney, 2004) have been widely used in practical machine translation (MT) systems due to their effectiveness, simplicity, and applicability. First, as sequences of consecutive words, phrases are capable of memorizing local word selection and reordering, making them an effective mechanism for translating idioms or translations with word insertions or omissions. Moreover, n-gram language models can be seamlessly integrated into phrase-based decoders since partial translations grow left to right in decoding. Finally, phrase-based systems can be applicable to most domains and languages, especially for resource-scarce languages without high-accuracy parsers.

However, as phrase-based decoding casts translation as a string concatenation problem and permits arbitrary permutations, it proves to be NP-complete (Knight, 1999). Therefore, phrase reordering modeling has attracted intensive attention in the past decade (e.g., Och et al., 2004; Tillman, 2004; Zens et al., 2004; Al-Onaizan and Papineni, 2006; Xiong et al., 2006; Koehn et al., 2007; Galley and Manning, 2008; Feng et al., 2010; Green et al., 2010; Bisazza and Federico, 2012; Cherry, 2013).

Among them, reordering models based on **inversion transduction grammar (ITG)** (Wu, 1997) are one of the important ongoing research directions. As a formalism for bilingual modeling of sentence pairs, ITG is particularly well suited to predicting ordering shifts between languages. As a result, a number of authors have incorporated ITG into left-to-right decoding to constrain the reordering space and reported significant improvements (e.g., Zens et al., 2004; Feng et al., 2010). Along another line, Xiong et al. (2006) propose a maximum entropy (MaxEnt) reordering model based on ITG. They use the CKY algorithm to recursively merge two blocks (i.e., a pair of source and target strings) into larger blocks, either in a straight or an inverted order. Unlike lexicalized reordering models (Tillman, 2004; Koehn et al., 2007; Galley and Manning, 2008) that are defined on individual bilingual phrases, the MaxEnt ITG reordering model is a two-category classifier (i.e., straight or inverted) for two arbitrary bilingual phrases of which the source phrases are adjacent. This potentially alleviates the data sparseness

Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing, pages 567–577,
Seattle, Washington, USA, 18-21 October 2013.

problem since there are usually a large number of reordering training examples available (Xiong et al., 2006). As a result, the MaxEnt ITG model and its extensions (Xiong et al., 2008; Xiong et al., 2010) have achieved competing performance as compared with state-of-the-art phrase-based systems.

Despite these successful efforts, the ITG reordering classifiers still face a major challenge: how to extract features from training examples (i.e., a pair of bilingual strings). It is hard to decide which words are representative for predicting reordering, either manually or automatically, especially for long sentences. As a result, Xiong et al. (2006) only use boundary words (i.e., the first and the last words in a string) to predict the ordering. What if we look inside? Is it possible to avoid manual feature engineering and learn semantic representations from the data?

Fortunately, the rapid development of intersecting deep learning with natural language processing (Bengio et al., 2003; Collobert and Weston, 2008; Collobert et al., 2011; Glorot et al., 2011; Bordes et al., 2011; Socher et al., 2011a; Socher et al., 2011b; Socher et al., 2011c; Socher et al., 2012; Bordes et al., 2012; Huang et al., 2012; Socher et al., 2013; Hermann and Blunsom, 2013) brings hope for alleviating this problem. In these efforts, natural language words are represented as real-valued vectors, which can be naturally fed to neural networks as input. More importantly, it is possible to learn vector space representations for multi-word phrases using **recursive autoencoders** (Socher et al., 2011c), which opens the door to leveraging semantic representations of phrases in reordering models from a neural language modeling point of view.

In this work, we propose an ITG reordering classifier based on recursive autoencoders. The neural network consists of four autoencoders (i.e., the first source phrase, the first target phrase, the second source phrase, and the second target phrase) and a softmax layer. The recursive autoencoders, which are trained on reordering examples extracted from word-aligned bilingual corpus, are capable of producing vector space representations for arbitrary multi-word strings in decoding. Therefore, our model takes the whole phrases rather than only boundary words into consideration when predicting phrase permutations. Experiments on the NIST 2008 dataset show that our system significantly improves over the MaxEnt classifier by 1.07 in terms of case-insensitive BLEU score.

2 Recursive Autoencoders for ITG-based Translation

2.1 Inversion Transduction Grammar

Inversion transduction grammar (ITG) (Wu, 1997) is a formalism for synchronous parsing of bilingual sentence pairs. Xiong et al. (2006) apply bracketing transduction grammar (BTG), which is a simplified version of ITG, to phrase-based translation using the following production rules:

$$X \rightarrow [X^1, X^2] \quad (1)$$

$$X \rightarrow \langle X^1, X^2 \rangle \quad (2)$$

$$X \rightarrow f/e \quad (3)$$

where X is a **block** that consists of a pair of source and target strings, f is a source phrase, and e is a target phrase. X^1 and X^2 are two neighboring blocks of which the two source phrases are adjacent. While rule (1) merges two target phrases in a **straight** order, rule (2) merges in an **inverted** order. Besides these two **reordering rules**, rule (3) is a **lexical rule** that translates a source phrase f into a target phrase e. This is exactly a bilingual phrase used in conventional phrase-based systems.

An ITG derivation, which consists of a sequence of production rules, explains how a sentence pair is generated simultaneously. Figure 1 shows an ITG derivation for a Chinese sentence and its English translation. We distinguish between two types of blocks:

1. **atomic blocks**: blocks generated by applying lexical rules,
2. **composed blocks**: blocks generated by applying reordering rules.

In Figure 1, the sentence pair is segmented into five atomic blocks:

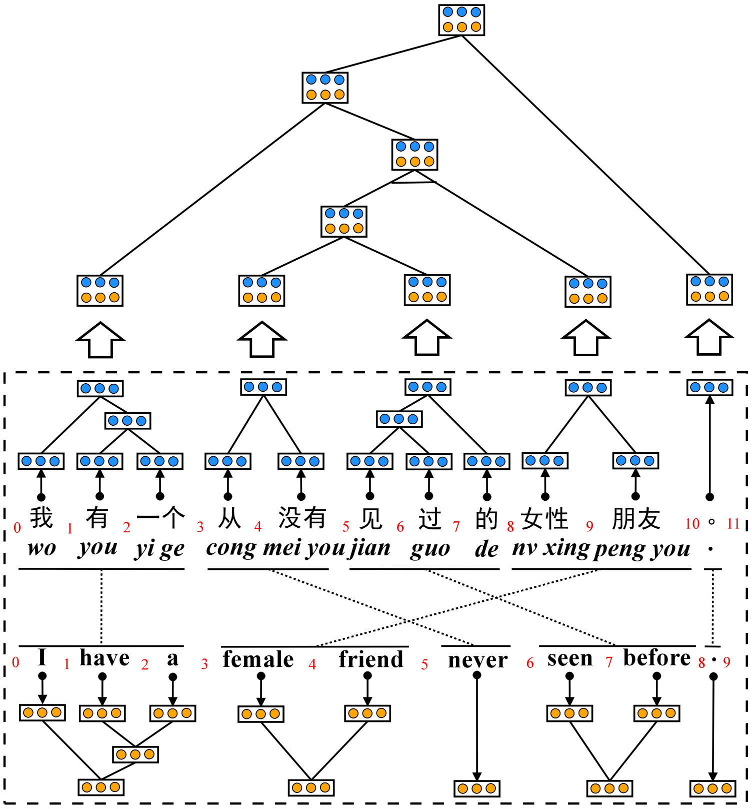
$X_{0,3,0,3}$: *wo you yi ge* ↔ I have a

$X_{3,5,5,6}$: *cong mei you* ↔ never

$X_{5,8,6,8}$: *jian guo de* ↔ seen before

$X_{8,10,3,5}$: *nv xing peng you* ↔ female friend

$X_{10,11,8,9}$: . ↔ .


0 我 1 有 2 一个 3 从 4 没有 5 见 6 过 7 的 8 女性 9 朋友 10 。 11
wo you yi ge cong mei you jian guo de nv xing peng you ·
0 I 1 have 2 a 3 female 4 friend 5 never 6 seen 7 before 8 · 9

(1)	$X_{0,11,0,9} \rightarrow [X_{0,10,0,8}, X_{10,11,8,9}]$
(2)	$X_{0,10,0,8} \rightarrow [X_{0,3,0,3}, X_{3,10,3,8}]$
(3)	$X_{0,3,0,3} \rightarrow$ *wo you yi ge* / I have a
(4)	$X_{3,10,3,8} \rightarrow \langle X_{3,8,5,8}, X_{8,10,3,5} \rangle$
(5)	$X_{3,8,5,8} \rightarrow [X_{3,5,5,6}, X_{5,8,6,8}]$
(6)	$X_{3,5,5,6} \rightarrow$ *cong mei you* / never
(7)	$X_{5,8,6,8} \rightarrow$ *juan guo de* / seen before
(8)	$X_{8,10,3,5} \rightarrow$ *nv xing peng you*/ female friend
(9)	$X_{10,11,8,9} \rightarrow$. / .

Figure 1: An ITG derivation for a Chinese sentence and its translation. We use $X_{i,j,k,l} = \langle f_i^j, e_k^l \rangle$ to represent a block. Our neural ITG reordering model first assigns vector space representations to single words and then produces vectors for phrases using recursive autoencoders, which form atomic blocks. The atomic blocks are recursively merged into composed blocks, the vector space representations of which are produced by recursive autoencoders simultaneously. The neural classifier makes decisions at each node using the vectors of all its descendants.

where $X_{3,5,5,6}$ indicates that the block consists of a source phrase spanning from position 3 to position 5 (i.e., *"cong mei you"*) and a target phrase spanning from position 5 to position 6 (i.e., "never"). More formally, a block $X_{i,j,k,l} = \langle f_i^j, e_k^l \rangle$ is a pair of a source phrase $f_i^j = f_{i+1} \dots f_j$ and a target phrase $e_k^l = e_{k+1} \dots e_l$. Obviously, these atomic blocks are generated by lexical rules.

Two blocks of which the source phrases are adjacent can be merged into a larger one in two ways: concatenating the target phrases in a straight order using rule (1) or in an inverted order using rule (2). For example, atomic blocks $X_{3,5,5,6}$ and $X_{5,8,6,8}$ are merged into a composed block $X_{3,8,5,8}$ in a straight order, which is further merged with an atomic block $X_{8,10,3,5}$ into another composed block $X_{3,10,3,8}$ in an inverted order. This process recursively proceeds until the entire sentence pair is generated.

The major challenge of applying ITG to machine translation is to decide when to merge two blocks in a straight order and when in an inverted order. Therefore, the ITG reordering model can be seen as a two-category classifier $P(o|X^1, X^2)$, where $o \in \{straight, inverted\}$.

A naive way is to assign fixed probabilities to two reordering rules, which is referred to as *flat model* by Xiong et al. (2006):

$$P(o|X^1, X^2) = \begin{cases} p & o = straight \\ 1 - p & o = inverted \end{cases} \quad (4)$$

The drawback of the flat model is ignoring the actual blocks being merged. Intuitively, different blocks should have different preferences between the two orders.

To alleviate this problem, Xiong et al. (2006) propose a maximum entropy (MaxEnt) classifier:

$$P(o|X^1, X^2) = \frac{\exp(\theta \cdot h(o, X^1, X^2))}{\sum_{o'} \exp(\theta \cdot h(o', X^1, X^2))} \quad (5)$$

where $h(\cdot)$ is a vector of features defined on the blocks and the order, θ is a vector of feature weights.

While MaxEnt is a flexible and powerful framework for including arbitrary features, feature engineering becomes a major challenge for the MaxEnt classifier. Xiong et al. (2006) find that boundary words (i.e., the first and the last words in a string) are informative for predicting reordering. Actually, it is hard to decide which internal words in a long composed blocks are representative and informative. Therefore, they only use boundary words as the main features.

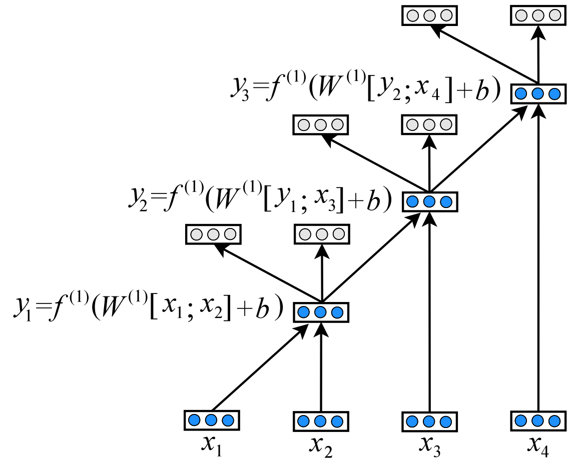

$y_3 = f^{(1)}(W^{(1)}[y_2; x_4] + b)$
$y_2 = f^{(1)}(W^{(1)}[y_1; x_3] + b)$
$y_1 = f^{(1)}(W^{(1)}[x_1; x_2] + b)$
x_1
x_2
x_3
x_4

Figure 2: A recursive autoencoder for multi-word strings. The example is adapted from (Socher et al., 2011c). Blue and grey nodes are original and reconstructed ones, respectively.

However, it seems not enough to just consider boundary words and ignore all internal words when making order predictions, especially for long sentences.[1] Indeed, Xiong et al. (2008) find that the MaxEnt classifier with boundary words as features is prone to make wrong predictions for long composed blocks. As a result, they have to impose a hard constraint to always prefer merging long composed blocks in a monotonic way.

Therefore, it is important to consider more than boundary words to make more accurate reordering predictions. We need a new mechanism to achieve this goal.

2.2 Recursive Autoencoders

2.2.1 Vector Space Representations for Words

In neural networks, a natural language word is represented as a real-valued vector (Bengio et al., 2003; Collobert and Weston, 2008). For example, we can use $[0.1\ 0.8\ 0.4]^T$ to represent "female" and

[1] Strictly speaking, the ITG reordering model is not a phrase reordering model since phrase pairs are only the atomic blocks. Instead, it is defined to work on arbitrarily long strings because composed blocks become larger and larger until the entire sentence pair is generated.

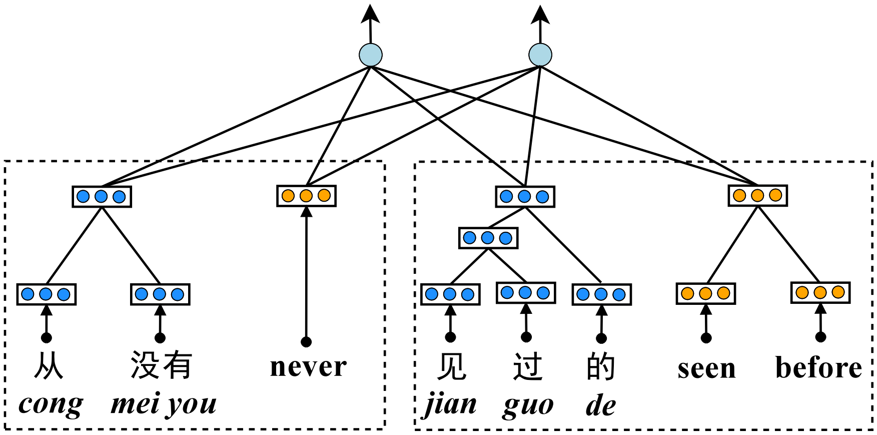


Figure 3: A neural ITG reordering model. The binary classifier makes decisions based on the vector space representations of the source and target sides of merging blocks.

$[0.7\ 0.1\ 0.5]^T$ to represent "friend". Such vector space representations enable natural language words to be fed to neural networks as input.

Formally, we denote each word as a vector $x \in \mathbb{R}^n$. These word vectors are then stacked into a word embedding matrix $L \in \mathbb{R}^{n \times |V|}$, where $|V|$ is the vocabulary size. Given a sentence that is an ordered list of m words, each word has an associated vocabulary index k into the word embedding matrix L that we use to retrieve the word's vector space representation. This look-up operation can be seen as a simple projection layer:

$$x_i = Lb_k \in \mathbb{R}^n \tag{6}$$

where b_k is a binary vector which is zero in all positions except for the kth index.

In Figure 1, we assume $n = 3$ for simplicity and can retrieve vectors for Chinese and English words from two embedding matrices, respectively.

2.2.2 Vector Space Representations for Multi-Word Strings

To apply neural networks to ITG-based translation, it is important to generate vector space representations for atomic and composed blocks.

For example, since the vector of "female" is $[0.1\ \ 0.8\ \ 0.4]^T$ and the vector of "friend" is $[0.7\ 0.1\ 0.5]^T$, what is the vector of the phrase "female friend"? If we denote "female friend" as p (i.e., parent), "female" as c_1 (i.e., the first child), and "friend" as c_2 (i.e., the second child), this can be done by applying a function $f^{(1)}$:

$$p = f^{(1)}(W^{(1)}[c_1; c_2] + b^{(1)}) \tag{7}$$

where $[c_1; c_2] \in \mathbb{R}^{2n \times 1}$ is the concatenation of c_1 and c_2, $W^{(1)} \in \mathbb{R}^{n \times 2n}$ is a parameter matrix, $b^{(1)} \in \mathbb{R}^{n \times 1}$ is a bias term, and $f^{(1)}$ is an element-wise activation function such as $\tanh(\cdot)$, which is used in our experiments.

Note that the resulting vector for the parent is also an n-dimensional vector, e.g, $[0.6\ 0.9\ 0.2]^T$. The same neural network can be recursively applied to two strings until the vector of the entire sentence is generated. As ITG derivation builds a binary parse tree, the neural network can be naturally integrated into CKY parsing.

To assess how well the learned vector p represents its children, we can **reconstruct** the children in a reconstruction layer:

$$[c'_1; c'_2] = f^{(2)}(W^{(2)}p + b^{(2)}) \tag{8}$$

where c'_1 and c'_2 are the reconstructed children, $W^{(2)}$ is a parameter matrix for reconstruction, $b^{(2)}$ is a bias term for reconstruction, and $f^{(2)}$ is an element-wise activation function, which is also set as $\tanh(\cdot)$ in our experiments. Similarly, the same reconstruction neural network can be applied to each node in an ITG parse.

These neural networks are called **recursive autoencoders** (Socher et al., 2011c). Figure 2 illustrates an application of a recursive autoencoder to a binary tree. The blue and grey nodes are the original and reconstructed nodes, respectively. The autoencoder is re-used at each node of the tree. The binary tree is composed of a set of triplets in the form of $(p \rightarrow c_1\ c_2)$, where p is a parent vector and c_1 and c_2 are children vectors of p. Each child can be either an input word vector or a multi-word vector. Therefore, the tree in Figure 2 can be represented as three triplets: $(y_1 \rightarrow x_1\ x_2)$, $(y_2 \rightarrow y_1\ x_3)$, and $(y_3 \rightarrow y_2\ x_4)$.

In Figure 1, we use recursive autoencoders to generate vector space representations for Chinese and English phrases, which form the atomic blocks for further block merging.

2.2.3 A Neural ITG Reordering Model

Once the vectors for blocks are generated, it is straightforward to introduce a neural ITG reordering model. As shown in Figure 3, the neural network consists of an input layer and a softmax layer. The input layer is composed of the vectors of the first source phrase, the first target phrase, the second source phrase, and the second target phrase. Note that all phrases in the same language use the the same recursive autoencoder. The softmax layer outputs the probabilities of the two merging orders:

$$P(o|X^1, X^2) = \frac{\exp(g(o, X^1, X^2))}{\sum_{o'} \exp(g(o', X^1, X^2))} \quad (9)$$

$$g(o, X^1, X^2) = f(W^o c(X^1, X^2) + b^o) \quad (10)$$

where $o \in \{straight, inverted\}$, $W^o \in \mathbb{R}^{1 \times 4n}$ is a parameter matrix, $b^o \in \mathbb{R}$ is a bias term, and $c(X^1, X^2) \in \mathbb{R}^{4n \times 1}$ is the concatenation of the vectors of the four phrases.

3 Training

There are three sets of parameters in our recursive autoencoders:

1. θ_L: word embedding matrix L for both source and target languages (Section 2.2.1);

2. θ_{rec}: recursive autoencoder parameter matrices $W^{(1)}$, $W^{(2)}$ and bias terms $b^{(1)}$, $b^{(2)}$ for both source and target languages (Section 2.2.2);

3. θ_{reo}: neural ITG reordering model parameter matrix W^o and bias term b^o (Section 2.2.3).

All these parameters are learned automatically from the training data. For clarity, we will use θ to denote all these parameters in the rest of the paper.

For training word embedding matrix, there are two settings commonly used. In the first setting, the word embedding matrix is initialized randomly. This works well in a supervised scenario, in which a neural network updates the matrix in order to optimize some task-specific objectives (Collobert et al., 2011; Socher et al., 2011c). In the second setting, the word embedding matrix is pre-trained using an unsupervised neural language model (Bengio et al., 2003; Collobert and Weston, 2008) with huge amount of unlabeled data. In this work, we prefer to the first setting because the word embedding matrices can be trained to minimize errors with respect to reordering modeling.

There are two kinds of errors involved

1. **reconstruction error**: how well the learned vector space representations represent the corresponding strings?

2. **reordering error**: how well the classifier predicts the merging order?

As described in Section 2.2.2, the input vector c_1 and c_2 of a recursive autoencoder can be reconstructed using Eq. 8 as c'_1 and c'_2. We use Euclidean distance between the input and the reconstructed vectors to measure the *reconstruction error*:

$$E_{rec}([c_1; c_2]; \theta) = \frac{1}{2} \left\| [c_1; c_2] - [c'_1; c'_2] \right\|^2 . \quad (11)$$

Given a sentence, there are exponentially many ways to obtain its vector space representation. Note that each way corresponds to a binary tree like Figure 2. To find a binary tree with minimal reconstruction error, we follow Socher et al. (2011c) to use a greedy algorithm. Taking Figure 2 as an example, the greedy algorithm begins with computing the reconstruction error $E_{rec}(\cdot)$ for each pair of consecutive vectors, i.e., $E_{rec}([x_1; x_2]; \theta)$, $E_{rec}([x_2; x_3]; \theta)$ and $E_{rec}([x_3; x_4]; \theta)$. Suppose $E_{rec}([x_1; x_2]; \theta)$ is the smallest, the algorithm will replace x_1 and x_2 with their vector representation y_1 produced by the recursive autoencoder. Then, the algorithm evaluates $E_{rec}([y_1; x_3]; \theta)$ and $E_{rec}([x_3; x_4]; \theta)$ and repeats the above replacing steps until only one vector remains. Socher et al. (2011c) find that the greedy algorithm runs fast without significant loss in performance as compared with CKY-style algorithms.

Given a training example set $S = \{t_i = (o_i, X_i^1, X_i^2)\}$, the average reconstruction error on the source side on the training set is defined as

$$E_{rec,s}(S;\theta) = \frac{1}{N_s}\sum_i \sum_{p\in T_R^\theta(t_i,s)} E_{rec}([p.c_1, p.c_2];\theta) \tag{12}$$

where $T_R^\theta(t_i, s)$ denotes all the intermediate nodes on the source side in binary trees, N_s is the number of these intermediate nodes, and $p.c_k$ is the kth child vector of p. The average reconstruction error on the target side, denoted by $E_{rec,t}(S;\theta)$, can be computed in a similar way.

Therefore, the reconstruction error is defined as

$$E_{rec}(S;\theta) = E_{rec,s}(S;\theta) + E_{rec,t}(S;\theta). \tag{13}$$

Given a training example $t_i = (o_i, X_i^1, X_i^2)$, we assume the probability distribution d_{t_i} for its label is $[1,0]$ when $o_i = straight$, and $[0,1]$ when $o_i = inverted$. Then the cross-entropy error is

$$E_c(t_i;\theta) = -\sum_o d_{t_i}(o)\log\left(P_\theta(o|X^1, X^2)\right) \tag{14}$$

where $o \in \{straight, inverted\}$. As a result, the reordering error is defined as

$$E_{reo}(S;\theta) = \frac{1}{|S|}\sum_i E_c(t_i;\theta). \tag{15}$$

Therefore, the joint training objective function is

$$J = \alpha E_{rec}(S;\theta) + (1-\alpha)E_{reo}(S;\theta) + R(\theta) \tag{16}$$

where α is a parameter used to balance the preference between reconstruction error and reordering error, $R(\theta)$ is the regularizer and defined as [2]

$$R(\theta) = \frac{\lambda_L}{2}\|\theta_L\|^2 + \frac{\lambda_{rec}}{2}\|\theta_{rec}\|^2 + \frac{\lambda_{reo}}{2}\|\theta_{reo}\|^2. \tag{17}$$

As Socher et al. (2011c) stated, a naive way for lowering the reconstruction error is to make the magnitude of the hidden layer very small, which is not desirable. In order to prevent such behavior, we normalize all the output vectors of the hidden layers to have length 1 in the same way as (Socher et al., 2011c). Namely we set $p = \frac{p}{||p||}$ after computing p as in Eq. 7, and $c_1' = \frac{c_1'}{||c_1'||}$, $c_2' = \frac{c_2'}{||c_2'||}$ in Eq. 8.

Following Socher et al. (2011c), we use L-BFGS to estimate the parameters with respect to the joint training objective. Given a set of parameters, we construct binary trees for all the phrases using the greedy algorithm. The derivatives for these fixed binary trees can be computed via backpropagation through structures (Goller and Kuchler, 1996).

4 Experiments

4.1 Data Preparation

We evaluated our system on Chinese-English translation. The training corpus contains 1.23M sentence pairs with 32.1M Chinese words and 35.4M English words. We used SRILM (Stolcke, 2002) to train a 4-gram language model on the Xinhua portion of the GIGAWORD corpus, which contains 398.6M words. We used the NIST 2006 MT Chinese-English dataset as the development set and NIST 2008 dataset as the test set. The evaluation metric is case-insensitive BLEU. Because of the expensive computational cost for training our neural ITG reordering model, only the reordering examples extracted from about 1/5 of the entire parallel training corpus were used to train our neural ITG reordering model.

For the neural ITG reordering model, we set the dimension of the word embedding vectors to 25 empirically, which is a trade-off between computational cost and expressive power. We use the early stopping principle to determine when to stop L-BFGS. The hyper-parameters α, λ_L, λ_{rec} and λ_{reo} are optimized by random search (Bergstra and Bengio, 2012). As preliminary experiments show that classification accuracy has a high correlation with BLEU score, we optimize these hyper-parameters with respect to classification accuracy instead of BLEU to reduce computational cost. We randomly select 400,000 reordering examples as training set, 500 as development set, and another 500 as test set. The numbers of straight and inverted reordering examples in the development/test set are set to be equal to avoid biases. We draw α uniformly from 0.05

[2] The bias terms $b^{(1)}$, $b^{(2)}$ and b^o are not regularized. We do not exclude them from the equation explicitly just for clarity.

System	NIST 2006 (tune)	NIST 2008
maxent	30.40	23.75
neural	**31.61***	**24.82***

Table 1: BLEU scores on the NIST 2006 and 2008 datasets. *: significantly better ($p < 0.01$). "maxent" denotes the baseline maximum entropy system and "neural" denotes our recursive autoencoder system.

length	$>$	$=$	$<$
$[1, 10]$	43	121	57
$[11, 20]$	181	67	164
$[21, 30]$	170	11	152
$[31, 40]$	105	3	90
$[41, 50]$	69	1	53
$[51, 119]$	40	0	30

Table 2: Number of sentences that our system has a higher ($>$), equal ($=$) or lower ($<$) sentence-level BLEU-4 score on the NIST 2008 dataset.

to 0.3, and λ_L, λ_{rec}, λ_{reo} exponentially from 10^{-8} to 10^{-2}. We use the following hyper-parameters in our experiments: $\alpha = 0.11764$, $\lambda_L = 7.59 \times 10^{-5}$, $\lambda_{rec} = 1.30 \times 10^{-5}$ and $\lambda_{reo} = 3.80 \times 10^{-4}$. [3]

The baseline system is a re-implementation of (Xiong et al., 2006). Our system is different from the baseline by replacing the MaxEnt reordering model with a neural model. Both the systems have the same pruning settings: the threshold pruning parameter is set to 0.5 and the histogram pruning parameter to 40. For minimum-error-rate training, both systems generate 200-best lists.

4.2 MT Evaluation

Table 1 shows the case-insensitive BLEU-4 scores of the baseline system and our system on the development and test sets. Our system outperforms the baseline system by 1.21 BLEU points on the development set and 1.07 on the test set. Both the differences are statistically significant at $p = 0.01$ level (Riezler and Maxwell, 2005).

Table 2 shows the number of sentences that our system has a higher ($>$), equal ($=$) or lower ($<$) BLEU score on the NIST 2008 dataset. We find that our system is superior to the baseline system for long sentences.

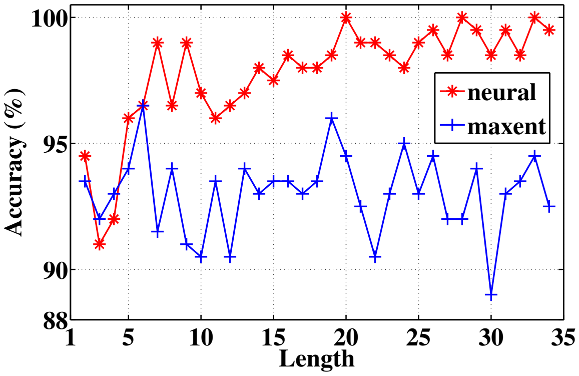


Figure 4: Comparison of reordering classification accuracies between the MaxEnt and neural classifiers over varying phrase lengths. "Length" denotes the sum of the lengths of two source phrases in a reordering example. Our classifier (neural) outperforms the MaxEnt classifier (maxent) consistently, especially for predicting long-distance reordering.

# of examples	NIST 2006 (tune)	NIST 2008
100,000	30.88	23.78
200,000	30.75	23.89
400,000	30.80	24.35
800,000	31.01	24.45
6,004,441	31.61	24.82

Table 3: The effect of reordering training data size on BLEU scores. The BLEU scores rise with the increase of training data size. Due to the computational cost, we only used 1/5 of the entire bilingual corpus to train our neural reordering model.

Figure 4 compares classification accuracies of the neural and MaxEnt classifiers. "Length" denotes the sum of the lengths of two source phrases in a reordering example. For each length, we randomly select 200 unseen reordering examples to calculate the classification accuracy. Our classifier outperforms the baseline consistently, especially for long composed blocks.

Xiong et al. (2008) find that the performance of the baseline system can be improved by forbidding inverted reordering if the phrase length exceeds a pre-defined distortion limit. This heuristic increases the BLEU score of the baseline system significantly to 24.46 but is still significantly worse ($p < 0.05$) than our system without the heuristic. We find that imposing this heuristic fails to improve our system

[3] The choice of α is very important for achieving high BLEU scores. We tried a number of intervals and found that the classification accuracy is most stable in the interval [0.100,0.125].

cluster 1	cluster 2	cluster 3	cluster 4	cluster 5
1.18	works for	alternative duties	these people who	of the three
accessibility	verify on	one-day conference	the reasons why	on the fundamental
wheelchair	tunnels from	armed groups	the story of how	over the entire
candies	transparency in	chinese language works	the system which	through its own
cough	opinion at	eating habits	the trend towards	with the best

Table 4: Words and phrases that are close in the Euclidean space. The words and phrases in the same cluster have similar behaviors from a reordering point of view rather than relatedness, suggesting that the vector representations produced by the recursive autoencoders are helpful for capturing reordering regularities.

significantly. One possible reason is that there is limited room for improvement as our system makes fewer wrong predictions for long composed blocks.

The above results suggest that our system does go beyond using boundary words and make a better use of the merging blocks by using vector space representations.

Table 3 shows the effect of training dataset size on BLEU scores. We find that BLEU scores on both the development and test sets rise with the increase of the training dataset size. As the training process is very time-consuming, only the reordering examples extracted from 1/5 of the entire parallel training corpus are used in our experiments to train our model. Obviously, with more efficient training algorithms, making full use of all the reordering examples extracted from the entire corpus will result in better results. We leave this for future work.

4.3 Qualitative Analysis on Vector Representations

Table 4 shows a number of words and phrases that are close (measured by Euclidean distance) in the n-dimensional space. We randomly select about 370K target side phrases used in our experiments and cluster them into 983 clusters using k-means algorithm (MacQueen, 1967). The distance between two phrases are measured by the Euclidean distance between their vector representations. As shown in Table 4, cluster 1 mainly consists of nouns, cluster 2 mainly contains verb/noun+preposition structures, cluster 3 contains compound phrases, cluster 4 consists of phrases which should be followed by a clause, and cluster 5 mainly contains the beginning parts of prepositional phrases that tend to be followed by a noun phrase or word. We find that the words and phrases in the same cluster have similar behaviors from a reordering point of view rather than relatedness. This indicates that the vector representations produced by the recursive autoencoders are helpful for capturing reordering regularities.

5 Conclusion

We have presented an ITG reordering classifier based on recursive autoencoders. As recursive autoencoders are capable of producing vector space representations for arbitrary multi-word strings in decoding, our neural ITG system achieves an absolute improvement of 1.07 BLEU points over the baseline on the NIST 2008 Chinese-English dataset.

There are a number of interesting directions we would like to pursue in the near future. First, replacing the MaxEnt classifier with a neural one redefines the conditions for risk-free hypothesis recombination. We find that the number of hypotheses that can be recombined reduces in our system. Therefore, we plan to use forest reranking (Huang, 2008) to alleviate this problem. Second, it is interesting to follow Socher et al. (2013) to combine linguistically-motivated labels with recursive neural networks. Another problem with our system is that the decoding speed is much slower than the baseline system because of the computational overhead introduced by RAEs. It is necessary to investigate more efficient decoding algorithms. Finally, it is possible to apply our method to other phrase-based and even syntax-based systems.

Acknowledgments

This research is supported by the 863 Program under the grant No. 2012AA011102, by the Boeing Tsinghua Joint Research Project on Language Processing (Agreement TBRC-008-SDB-2011 Phase 3

(2013)), by the Singapore National Research Foundation under its International Research Centre @ Singapore Funding Initiative and administered by the IDM Programme Office, and by a Research Fund No. 20123000007 from Tsinghua MOE-Microsoft Joint Laboratory. Many thanks go to Chunyang Liu and Chong Kuang for their great help for setting up the computing platform. We also thank Min-Yen Kan, Meng Zhang and Yu Zhao for their insightful discussions.

References

Yaser Al-Onaizan and Kishore Papineni. 2006. Distortion models for statistical machine translation. In *Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the Association for Computational Linguistics*, pages 529–536, Sydney, Australia, July.

Yoshua Bengio, Réjean Ducharme, Pascal Vincent, and Christian Janvin. 2003. A neural probabilistic language model. *Journal of Machine Learning Research*, 3:1137–1155, March.

James Bergstra and Yoshua Bengio. 2012. Random search for hyper-parameter optimization. *The Journal of Machine Learning Research*, 13(1):281–305, February.

Arianna Bisazza and Marcello Federico. 2012. Modified distortion matrices for phrase-based statistical machine translation. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 478–487, Jeju Island, Korea, July.

Antoine Bordes, Jason Weston, Ronan Collobert, and Yoshua Bengio. 2011. Learning structured embeddings of knowledge bases. In *Proceedings of the Twenty-Fifth AAAI Conference on Artificial Intelligence*, pages 301–306.

Antoine Bordes, Xavier Glorot, Jason Weston, and Yoshua Bengio. 2012. Joint learning of words and meaning representations for open-text semantic parsing. In *International Conference on Artificial Intelligence and Statistics*, pages 127–135.

Colin Cherry. 2013. Improved reordering for phrase-based translation using sparse features. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 22–31, Atlanta, Georgia, June.

Ronan Collobert and Jason Weston. 2008. A unified architecture for natural language processing: Deep neural networks with multitask learning. In *Proceedings of the 25th International Conference on Machine Learning*, pages 160–167.

R. Collobert, J. Weston, L. Bottou, M. Karlen, K. Kavukcuoglu, and P. Kuksa. 2011. Natural language processing (almost) from scratch. *Journal of Machine Learning Research*, 12:2493–2537.

Yang Feng, Haitao Mi, Yang Liu, and Qun Liu. 2010. An efficient shift-reduce decoding algorithm for phrased-based machine translation. In *Proceedings of the 23rd International Conference on Computational Linguistics: Posters*, pages 285–293, Beijing, China, August.

Michel Galley and Christopher D. Manning. 2008. A simple and effective hierarchical phrase reordering model. In *Proceedings of the 2008 Conference on Empirical Methods in Natural Language Processing*, pages 848–856, Honolulu, Hawaii, October.

Xavier Glorot, Antoine Bordes, and Yoshua Bengio. 2011. Domain adaptation for large-scale sentiment classification: A deep learning approach. In *Proceedings of the 28th International Conference on Machine Learning (ICML-11)*, pages 513–520.

Christoph Goller and Andreas Kuchler. 1996. Learning task-dependent distributed representations by backpropagation through structure. In *Proceedings of 1996 IEEE International Conference on Neural Networks (Volume:1)*, volume 1, pages 347–352.

Spence Green, Michel Galley, and Christopher D. Manning. 2010. Improved models of distortion cost for statistical machine translation. In *Proceedings of Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 867–875, Los Angeles, California, June.

Karl Moritz Hermann and Phil Blunsom. 2013. The role of syntax in vector space models of compositional semantics. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 894–904, Sofia, Bulgaria, August.

Eric Huang, Richard Socher, Christopher Manning, and Andrew Ng. 2012. Improving word representations via global context and multiple word prototypes. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 873–882, Jeju Island, Korea, July.

Liang Huang. 2008. Forest reranking: Discriminative parsing with non-local features. In *Proceedings of the 46th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 586–594, Columbus, Ohio, June.

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

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.

Philipp Koehn, Hieu Hoang, Alexandra Birch, Chris Callison-Burch, Marcello Federico, Nicola Bertoldi, Brooke Cowan, Wade Shen, Christine Moran, Richard Zens, Chris Dyer, Ondrej Bojar, Alexandra Constantin, and Evan Herbst. 2007. Moses: Open source toolkit for statistical machine translation. In *Proceedings of the 45th Annual Meeting of the Association for Computational Linguistics Companion Volume Proceedings of the Demo and Poster Sessions*, pages 177–180, Prague, Czech Republic, June.

James MacQueen. 1967. Some methods for classification and analysis of multivariate observations. In *Proceedings of the Fifth Berkeley Symposium on Mathematical Statistics and Probability*, volume 1.

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

Franz Josef Och, Daniel Gildea, Sanjeev Khudanpur, Anoop Sarkar, Kenji Yamada, Alex Fraser, Shankar Kumar, Libin Shen, David Smith, Katherine Eng, Viren Jain, Zhen Jin, and Dragomir Radev. 2004. A smorgasbord of features for statistical machine translation. In *Proceedings of the Human Language Technology Conference of the North American Chapter of the Association for Computational Linguistics: HLT-NAACL 2004: Main Proceedings*, pages 161–168, Boston, Massachusetts, USA, May.

Stefan Riezler and John T. Maxwell. 2005. On some pitfalls in automatic evaluation and significance testing for MT. In *Proceedings of the ACL Workshop on Intrinsic and Extrinsic Evaluation Measures for Machine Translation and/or Summarization*, pages 57–64, Ann Arbor, Michigan, June.

Richard Socher, Eric H. Huang, Jeffrey Pennin, Andrew Y. Ng, and Christopher D. Manning. 2011a. Dynamic pooling and unfolding recursive autoencoders for paraphrase detection. In *Proceedings of Advances in Neural Information Processing Systems 24*, pages 801–809.

Richard Socher, Cliff C. Lin, Andrew Y. Ng, and Christopher D. Manning. 2011b. Parsing natural scenes and natural language with recursive neural networks. In *Proceedings of the 26th International Conference on Machine Learning (ICML)*, pages 129–136.

Richard Socher, Jeffrey Pennington, Eric H. Huang, Andrew Y. Ng, and Christopher D. Manning. 2011c. Semi-supervised recursive autoencoders for predicting sentiment distributions. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing*, pages 151–161, Edinburgh, Scotland, UK., July.

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, Jeju Island, Korea, July.

Richard Socher, John Bauer, Christopher D. Manning, and Ng Andrew Y. 2013. Parsing with compositional vector grammars. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 455–465, Sofia, Bulgaria, August.

Andreas Stolcke. 2002. SRILM - an extensible language modeling toolkit. In *Proceedings of International Conference on Spoken Language Processing, vol. 2*, pages 901–904, September.

Christoph Tillman. 2004. A unigram orientation model for statistical machine translation. In *Proceedings of the Human Language Technology Conference of the North American Chapter of the Association for Computational Linguistics: HLT-NAACL 2004: Short Papers*, pages 101–104, Boston, Massachusetts, USA, May.

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

Deyi Xiong, Qun Liu, and Shouxun Lin. 2006. Maximum entropy based phrase reordering model for statistical machine translation. In *Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the Association for Computational Linguistics*, pages 521–528, Sydney, Australia, July.

Deyi Xiong, Min Zhang, Aiti Aw, Haitao Mi, Qun Liu, and Shouxun Lin. 2008. Refinements in BTG-based statistical machine translation. In *Proceedings of the Third International Joint Conference on Natural Language Processing: Volume-I*, pages 505–512.

Deyi Xiong, Min Zhang, Aiti Aw, and Haizhou Li. 2010. Linguistically annotated reordering: Evaluation and analysis. *Computational Linguistics*, 36(3):535–568, September.

Richard Zens, Hermann Ney, Taro Watanabe, and Eiichiro Sumita. 2004. Reordering constraints for phrase-based statistical machine translation. In *Proceedings of the 20th International Conference on Computational Linguistics*, pages 205–211, Geneva, Switzerland, Aug 23–Aug 27.