Non-linear Learning for Statistical Machine Translation

Shujian Huang, Huadong Chen, Xinyu Dai and Jiajun Chen
State Key Laboratory for Novel Software Technology
Nanjing University
Nanjing 210023, China
{huangsj, chenhd, daixy, chenjj}@nlp.nju.edu.cn

Abstract

Modern statistical machine translation (SMT) systems usually use a linear combination of features to model the quality of each translation hypothesis. The linear combination assumes that all the features are in a linear relationship and constrains that each feature interacts with the rest features in an linear manner, which might limit the expressive power of the model and lead to a under-fit model on the current data. In this paper, we propose a non-linear modeling for the quality of translation hypotheses based on neural networks, which allows more complex interaction between features. A learning framework is presented for training the non-linear models. We also discuss possible heuristics in designing the network structure which may improve the non-linear learning performance. Experimental results show that with the basic features of a hierarchical phrase-based machine translation system, our method produce translations that are better than a linear model.

1 Introduction

One of the core problems in the research of statistical machine translation is the modeling of translation hypotheses. Each modeling method defines a score of a target sentence $\mathbf{e} = e_1 e_2 ... e_i ... e_I$, given a source sentence $\mathbf{f} = f_1 f_2 ... f_j ... f_J$, where each e_i is the ith target word and f_j is the jth source word. The well-known modeling method starts from the Source-Channel model (Brown et al., 1993)(Equation 1). The scoring of $\mathbf{e}$ decomposes to the calculation of a translation model and a language model.

$$Pr(\mathbf{e}|\mathbf{f}) = Pr(\mathbf{e})Pr(\mathbf{f}|\mathbf{e})/Pr(\mathbf{f}) \quad (1)$$

The modeling method is extended to log-linear models by Och and Ney (2002), as shown in Equation 2, where $h_m(\mathbf{e}|\mathbf{f})$ is the mth feature function and λ_m is the corresponding weight.

$$\begin{aligned} Pr(\mathbf{e}|\mathbf{f}) &= p_{\lambda_1^M}(\mathbf{e}|\mathbf{f}) \\ &= \frac{exp[\sum_{m=1}^{M} \lambda_m h_m(\mathbf{e}|\mathbf{f})]}{\sum_{\mathbf{e}'} exp[\sum_{m=1}^{M} \lambda_m h_m(\mathbf{e}'|\mathbf{f})]} \end{aligned} \quad (2)$$

Because the normalization term in Equation 2 is the same for all translation hypotheses of the same source sentence, the score of each hypothesis, denoted by s_L, is actually a linear combination of all features, as shown in Equation 3.

$$s_L(\mathbf{e}) = \sum_{m=1}^{M} \lambda_m h_m(\mathbf{e}|\mathbf{f}) \quad (3)$$

The log-linear models are flexible to incorporate new features and show significant advantage over the traditional source-channel models, thus become the state-of-the-art modeling method and are applied in various translation settings (Yamada and Knight, 2001; Koehn et al., 2003; Chiang, 2005; Liu et al., 2006).

It is worth noticing that log-linear models try to separate good and bad translation hypotheses using a linear hyper-plane. However, complex interactions between features make it difficult to linearly separate good translation hypotheses from bad ones (Clark et al., 2014).

Taking common features in a typical phrase-based (Koehn et al., 2003) or hierarchical phrase-based (Chiang, 2005) machine translation system as an example, the language model feature favors shorter hypotheses; the word penalty feature encourages longer hypotheses. The phrase translation probability feature selects phrases that occurs more frequently in the training corpus, which sometimes is long with a lower translation probability, as in translating named entities or idioms;

Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing, pages 825–835,
Beijing, China, July 26-31, 2015.

sometimes is short but with a high translation probability, as in translating verbs or pronouns. These three features jointly decide the choice of translations. Simply use the weighted sum of their values may not be the best choice for modeling translations. As a result, log-linear models may under-fit the data. This under-fitting may prevents the further improvement of translation quality.

In this paper, we propose a non-linear modeling of translation hypotheses based on neural networks. The traditional features of a machine translation system are used as the input to the network. By feeding input features to nodes in a hidden layer, complex interactions among features are modeled, resulting in much stronger expressive power than traditional log-linear models. (Section 3)

Employing a neural network for SMT modeling has two issues to be tackled. The first issue is the parameter learning. Log-linear models rely on minimum error rate training (MERT) (Och, 2003) to achieve best performance. When the scoring function become non-linear, the intersection points of these non-linear functions could not be effectively calculated and enumerated. Thus MERT is no longer suitable for learning the parameters. To solve the problem, we present a framework for effective training including several criteria to transform the training problem into a binary classification task, a unified objective function and an iterative training algorithm. (Section 4)

The second issue is the structure of neural network. Single layer neural networks are equivalent to linear models; two-layer networks with sufficient nodes are capable of learning any continuous function (Bishop, 1995). Adding more layers into the network could model complex functions with less nodes, but also brings the problem of vanishing gradient (Erhan et al., 2009). We adapt a two-layer feed-forward neural network to keep the training process efficient. We notice that one major problem that prevents a neural network training reaching a good solution is that there are too many local minimums in the parameter space. Thus we discuss how to constrain the learning of neural networks with our intuitions and observations of the features. (Section 5)

Experiments are conducted to compare various settings and verify the effectiveness of our proposed learning framework. Experimental results show that our framework could achieve better translation quality even with the same traditional features as previous linear models. (Section 6)

2 Related work

Many research has been attempting to bring non-linearity into the training of SMT. These efforts could be roughly divided into the following three categories.

The first line of research attempted to re-interpret original features via feature transformation or additional learning. For example, Maskey and Zhou (2012) use a deep belief network to learn representations of the phrase translation and lexical translation probability features. Clark et al. (2014) used discretization to transform real-valued dense features into a set of binary indicator features. Lu et al. (2014) learned new features using a semi-supervised deep auto encoder. These work focus on the explicit representation of the features and usually employ extra learning procedure. Our proposed method only takes the original features, with no transformation, as the input. Feature transformation or combination are performed implicitly during the training of the network and integrated with the optimization of translation quality.

The second line of research attempted to use non-linear models instead of log-linear models, which is most similar in spirit with our work. Duh and Kirchhoff (2008) used the boosting method to combine several results of MERT and achieved improvement in a re-ranking setting. Liu et al. (2013) proposed an additive neural network which employed a two-layer neural network for embedding-based features. To avoid local minimum, they still rely on a pre-training and post-training from MERT or PRO. Comparing to these efforts, our proposed method takes a further step that it is integrated with iterative training, instead of re-ranking, and works without the help of any pre-trained linear models.

The third line of research attempted to add non-linear features/components into the log-linear learning framework. Neural network based models are trained as language models (Vaswani et al., 2013; Auli and Gao, 2014), translation models (Gao et al., 2014) or joint language and translation models (Auli et al., 2013; Devlin et al., 2014). Liu et al. (2013) also introduced word embedding for source and target sides of the translation

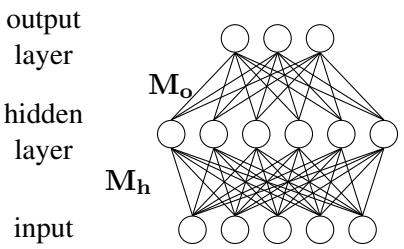

Figure 1: A two-layer feed-forward neural network.

rules as local features. In this paper, we focus on enhancing the expressive power of the modeling, which is independent of the research of enhancing translation systems with new designed features. We believe additional improvement could be achieved by incorporating more features into our framework.

3 Non-linear Translation

The non-linear modeling of translation hypotheses could be used in both phrase-based system and syntax-based systems. In this paper, we take the hierarchical phrase based machine translation system (Chiang, 2005) as an example and introduce how we fit the non-linearity into the system.

3.1 Two-layer Neural Networks

We employ a two-layer neural network as the non-linear model for scoring translation hypotheses. The structure of a typical two-layer feed-forward neural network includes an input layer, a hidden layer, and a output layer (as shown in Figure 1).

We use the input layer to accept input features, the hidden layer to combine different input features, the output layer with only one node to output the model score for each translation hypothesis based on the value of hidden nodes. More specifically, the score of hypothesis $\mathbf{e}$, denoted as s_N, is defined as:

$$s_N(\mathbf{e}) = \sigma_o(M_o \cdot \sigma_h(M_h \cdot h_1^m(\mathbf{e}|\mathbf{f}) + b_h) + b_o) \quad (4)$$

where M, b is the weight matrix, bias vector of the neural nodes, respectively; σ is the activation function, which is often set to non-linear functions such as the tanh function or sigmoid function; subscript h and o indicates the parameters of hidden layer and output layer, respectively.

3.2 Features

We use the standard features of a typical hierarchical phrase based translation system(Chiang, 2005). Adding new features into the framework is left as a future direction. The features as listed as following:

- $p(\alpha|\gamma)$ and $p(\gamma|\alpha)$: conditional probability of translating α as γ and translating α as γ, where α and γ is the left and right hand side of a initial phrase or hierarchical translation rule, respectively;
- $p_w(\alpha|\gamma)$ and $p_w(\gamma|\alpha)$: lexical probability of translating words in α as words in γ and translating words in γ as words in α;
- p_{lm}: language model probability;
- wc: accumulated count of individual words generated during translation;
- pc: accumulated count of initial phrases used;
- rc: accumulated count of hierarchical rule phrases used;
- gc: accumulated count of glue rule used in this hypothesis;
- uc: accumulated count of unknown source word. which has no entry in the translation table;
- nc: accumulated count of source phrases that translate into null;

3.3 Decoding

The basic decoding algorithm could be kept almost the same as traditional phrase-based or syntax-based translation systems (Yamada and Knight, 2001; Koehn et al., 2003; Chiang, 2005; Liu et al., 2006). For example, in the experiments of this paper, we use a CKY style decoding algorithm following Chiang (2005).

Our non-linear translation system is different from traditional systems in the way to calculate the score for each hypothesis. Instead of calculating the score as a linear combination, we use neural networks (Section 3.1) to perform a non-linear combination of feature values.

We also use the cube-pruning algorithm (Chiang, 2005) to keep the decoding efficient. Although the non-linearity in model scores may cause more search errors (Huang and Chiang,

2007) finding the highest scoring hypothesis, in practice it still achieves reasonable results.

4 Non-linear Learning Framework

Traditional machine translation systems rely on MERT to tune the weights of different features. MERT performs efficient search by enumerating the score function of all the hypotheses and using intersections of these linear functions to form the "upper-envelope" of the model score function (Och, 2003). When the scoring function is non-linear, it is not feasible to find the intersections of these functions. In this section, we discuss alternatives to train the parameters for non-linear models.

4.1 Training Criteria

The task of machine translation is a complex problem with structural output space. Decoding algorithms search for the translation hypothesis with the highest score, according to a given scoring function, from an exponentially large set of candidate hypotheses. The purpose of training is to select the scoring function, so that the function score the hypotheses "correctly". The correctness is often introduced by some extrinsic metrics, such as BLEU (Papineni et al., 2002).

We denote the scoring function as $s(\mathbf{f}, \mathbf{e}; \vec{\theta})$, or simply s, which is parameterized by $\vec{\theta}$; denote the set of all translation hypotheses as C; denote the extrinsic metric as $eval(\cdot)$ [1]. Note that, in linear cases, s is a linear function as in Equation 3, while in the non-linear case described in this paper, s is the scoring function in Equation 4.

Ideally, the training objective is to select a scoring function $\hat{s}$, from all functions $\mathcal{S}$, that scores the correct translation (or references) $\hat{\mathbf{e}}$, higher than any other hypotheses (Equation 5).

$$\hat{s} = \{s \in \mathcal{S} | s(\hat{\mathbf{e}}) > s(\mathbf{e}) \, \forall \mathbf{e} \in C\} \quad (5)$$

In practice, the candidate set C is exponentially large and hard to enumerate; the correct translation $\hat{\mathbf{e}}$ may not even exist in the current search space for various reasons, e.g. unknown source word. As a result, we use the n-best set C_{nbest} to approximate C, use the extrinsic metric $eval(\cdot)$ to evaluate the quality of hypotheses in C_{nbest} and use the following three alternatives as approximations to the ideal objective.

[1] In our experiments, we use sentence level BLEU with +1 smoothing as the evaluation metric.

Best v.s. Rest (BR) To score the best hypothesis in the n-best set $\tilde{\mathbf{e}}$ higher than the rest hypotheses. This objective is very similar to MERT in that it tries to optimize the score of $\tilde{\mathbf{e}}$ and doesn't concern about the ranking of rest hypotheses. In this case, $\tilde{\mathbf{e}}$ is an approximation of $\hat{\mathbf{e}}$.

Best v.s. Worst (BW) To score the best hypothesis higher than the worst hypothesis in the n-best set. This objective is motivated by the practice of separating the "hope" and "fear" translation hypotheses (Chiang, 2012). We take a simpler strategy which uses the best and worst hypothesis in C_{nbest} as the "hope" and "fear" hypothesis, respectively, in order to avoid multi-pass decoding.

Pairwise (PW) To score the better hypothesis in sampled hypothesis pairs higher than the worse one in the same pair. This objective is adapted from the Pairwise Ranking Optimization (PRO) (Hopkins and May, 2011), which tries to ranking all the hypotheses instead of selecting the best one. We use the same sampling strategy as their original paper.

Note that each of the above criteria transforms the original problem of selecting best hypotheses from an exponential space to a certain pairwise comparison problem, which could be easily trained using binary classifiers.

4.2 Training Objective

For the binary classification task, we use a hinge loss following Watanabe (2012). Because the network has a lot of parameters compared with the linear model, we use a L_1 norm instead of L_2 norm as the regularization term, to favor sparse solutions. We define our training objective function in Equation 6.

$$arg\min_{\theta} \frac{1}{N} \sum_{\mathbf{f} \in D} \sum_{(\mathbf{e}_1, \mathbf{e}_2) \in T(f)} \delta(\mathbf{f}, \mathbf{e}_1, \mathbf{e}_2; \theta) + \lambda \cdot ||\theta||_1$$

with

$$\delta(\cdot) = max\{s(\mathbf{f}, \mathbf{e}_1; \theta) - s(\mathbf{f}, \mathbf{e}_2; \theta) + 1, 0\} \quad (6)$$

where D is the given training data; $(\mathbf{e}_1, \mathbf{e}_2)$ is a training hypothesis-pair, with $\mathbf{e}_1$ to be the one with

higher $eval(\cdot)$ score; N is the total number of hypothesis-pairs in D; $T(f)$, or simply T, is the set of hypothesis-pairs for each source sentence f.

The set T is decided by the criterion used for training. For the BR setting, the best hypothesis is paired with every other hypothesis in the n-best list (Equation 7); while for the BW setting, it is only paired with the worst hypothesis (Equation 8). The generation of T in PW setting is the same with PRO sampling, we refer the readers to the original paper of Hopkins and May (2011).

$$T_{BR} = \{(\mathbf{e}_1, \mathbf{e}_2) | \mathbf{e}_1 = arg \max_{\mathbf{e} \in C_{nbest}} eval(\mathbf{e}), \mathbf{e}_2 \in C_{nbest} \text{ and } \mathbf{e}_1 \neq \mathbf{e}_2\} \quad (7)$$

$$T_{BW} = \{(\mathbf{e}_1, \mathbf{e}_2) | \mathbf{e}_1 = arg \max_{\mathbf{e} \in C_{nbest}} eval(\mathbf{e}), \mathbf{e}_2 = arg \min_{\mathbf{e} \in C_{nbest}} eval(\mathbf{e})\} \quad (8)$$

4.3 Training Procedure

In standard training algorithm for classification, the training instances stays the same in each iteration. In machine translation, decoding algorithms usually return a very different n-best set with different parameters. This is due to the exponentially large size of search space. MERT and PRO extend the current n-best set by merging the n-best set of all previous iterations into a pool (Och, 2003; Hopkins and May, 2011). In this way, the enlarged n-best set may give a better approximation of the true hypothesis set C and may lead to better and more stable training results.

We argue that the training should still focus on hypotheses obtained in current round, because in each iteration the searching for the n-best set is independent of previous iterations. To compromise the above two goals, in our practice, training hypothesis pairs are first generated from the current n-best set, then merged with the pairs generated from all previous iterations. In order to make the model focus more on pairs from current iteration, we assign pairs in previous iterations a small constant weight and assign pairs in current iteration a relatively large constant weight [2]. This is inspired by the AdaBoost algorithm (Schapire, 1999) in weighting instances.

Following the spirit of MERT, we propose a iterative training procedure (Algorithm 1). The training starts by randomly initialized model parameters θ^0 (line 1). In ith iteration, the decoding algorithm decodes each sentence $\mathbf{f}$ to get the n-best set C_{nbest} (line 5). Training hypothesis pairs T are extracted from C_{nbest} according to the training criterion described in Section 4.2 (line 6). Newly collected pairs T_i are combined with pairs from previous iterations before used for training (line 9). θ^{i+1} is obtained by solving Equation 6 using the Conjugate Sub-Gradient method (Le et al., 2011) (line 10).

Algorithm 1 Iterative Training Algorithm

Input: the set of training sentences D, max number of iteration I

1: $\theta^0 \leftarrow$ RandomInit(),
2: **for** $i = 0$ to I **do**
3: $\quad T_i \leftarrow \emptyset$;
4: $\quad$ **for** each $\mathbf{f} \in D$ **do**
5: $\quad\quad C_{nbest} \leftarrow$ NbestDecode($\mathbf{f}$; θ^i)
6: $\quad\quad T \leftarrow$ GeneratePair(C_{nbest})
7: $\quad\quad T_i \leftarrow T_i \cup T$
8: $\quad$ **end for**
9: $\quad T_{all} \leftarrow$ WeightedCombine($\cup_{k=0}^{i-1} T_k, T_i$)
10: $\quad \theta^{i+1} \leftarrow$ Optimize(T_{all}, θ^i)
11: **end for**

5 Structure of the Network

Although neural networks bring strong expressive power to the modeling of translation hypothesis, training a neural network is prone to resulting in local minimum which may affect the training results. We speculate that one reason for these local minimums is that the structure of a well-connected network has too many parameters. Take a neural network with k nodes in the input layer and m nodes in the hidden layer as an example. Every node in the hidden layer is connected to each of the k input nodes. This simple structure resulting in at least $k \times m$ parameters.

In Section 4.2, we use L_1 norm in the objective function in order to get sparser solutions. In this section, we propose some constrained network structures according to our prior knowledge of the features. These structures have much less parameters or simpler structures comparing to original neural networks, thus reduce the possibility of getting stuck in local minimums.

[2] In our experiments, we empirically set the constants to be 0.1 and 0.9, respectively.

5.1 Network with two-degree Hidden Layer

We find the first pitfall of the standard two-layer neural network is that each node in the hidden layer receives input from every input layer node. Features used in SMT are usually manually designed, which has their concrete meanings. For a network of several hidden nodes, combining every features into every hidden node may be redundant and not necessary to represent the quality of a hypothesis.

As a result, we take a harsh step and constrain the nodes in hidden layer to have a in-degree of two, which means each hidden node only accepts inputs from two input nodes. We do not use any other prior knowledge about features in this setting. So for a network with k nodes in the input layer, the hidden layer should contain $C_k^2 = k(k-1)/2$ nodes to accept all combinations from the input layer. We name this network structure as Two-Degree Hidden Layer Network (TDN).

It is easy to see that a TDN has $C_k^2 \times 2 = k(k-1)$ parameters for the hidden layer because of the constrained degree. This is one order of magnitude less than a standard two-layer network with the same number of hidden nodes, which has $C_k^2 \times k = k^2(k-1)/2$ parameters.

Note that we perform a 2-degree combination that looks similar in spirit with those combination of atomic features in large scale discriminative learning for other NLP tasks, such as POS tagging and parsing. However, unlike the practice in these tasks that directly combines values of different features to generate a new feature type, we first linearly combine the value of these features and perform non-linear transformation on these values via an activation function.

5.2 Network with Grouped Features

It might be a too strong constraint to require the hidden node have in-degree of 2. In order to relax this constraint, we need more prior knowledge from the features.

Our first observation is that there are different types of features. These types are different from each other in terms of value ranges, sources, importance, etc. For example, language model features usually take a very small value of probability, and word count feature takes a integer value and usually has a much higher weight in linear case than other count features.

The second observation is that features of the same type may not have complex interaction with each other. For example, it is reasonable to combine language model features with word count features in a hidden node. But it may not be necessary to combine the count of initial phrases and the count of unknown words into a hidden node.

Based on the above two intuitions, we design a new structure of network that has the following constraints: given a disjoint partition of features: G_1, G_2,..., G_k, every hidden node takes input from a set of input nodes, where any two nodes in this set come from two different feature groups. Under this constraint, the in-degree of a hidden node is at most k. We name this network structure as Grouped Network (GN).

In practice, we divide the basic features in Section 3.2 into five groups: language model features, translation probability features, lexical probability features, the word count feature, and the rest of count features. This division considers not only the value ranges, but also types of features and the possibility of them interact with each other.

6 Experiments and Results

6.1 General Settings

We conduct experiments on a large scale machine translation tasks. The parallel data comes from LDC, including LDC2002E18, LDC2003E14, LDC2004E12, LDC2004T08, LDC2005T10, LDC2007T09, which consists of 8.2 million of sentence pairs. Monolingual data includes Xinhua portion of Gigaword corpus. We use multi-references data MT03 as training data, MT02 as development data, and MT04, MT05 as test data. These data are mainly in the same genre, avoiding the extra consideration of domain adaptation.

Data	Usage	Sents.
LDC	TM train	8,260,093
Gigaword	LM train	14,684,074
MT03	train	919
MT02	dev	878
MT04	test	1,789
MT05	test	1,083

Table 1: Experimental data and statistics.

The Chinese side of the corpora is word segmented using ICTCLAS[3]. Our translation sys-

[3]http://ictclas.nlpir.org/

Criteria	MT03(train)	MT02(dev)	MT04	MT05
BR_c	35.02	36.63	34.96	34.15
BR	38.66	40.04	38.73	37.50
BW	39.55	39.36	38.72	37.81
PW	38.61	38.85	38.73	37.98

Table 2: BLEU4 in percentage on different training criteria ("BR", "BW" and "PW" refer to experiments with "Best v.s. Rest", "Best v.s. Worst" and "Pairwise" training criteria, respectively. "BR_c" indicates generate hypothesis pairs from n-best set of current iteration only presented in Section 4.3.

tem is an in-house implementation of the hierarchical phrase-based translation system(Chiang, 2005). We set the beam size to 20. We train a 5-gram language model on the monolingual data with MKN smoothing(Chen and Goodman, 1998). For each parameter tuning experiments, we ran the same training procedure 3 times and present the average results. The translation quality is evaluated use 4-gram case-insensitive BLEU (Papineni et al., 2002). Significant test is performed using bootstrap re-sampling implemented by Clark et al. (2011). We employ a two-layer neural network with 11 input layer nodes, corresponding to features listed in Section 3.2 and 1 output layer node. The number of nodes in the hidden layer varies in different settings. The sigmoid function is used as the activation function for each node in the hidden layer. For the output layer we use a linear activation function. We try different λ for the L_1 norm from 0.01 to 0.00001 and use the one with best performance on the development set. We solve the optimization problem with ALGLIB package[4].

6.2 Experiments of Training Criteria

This set experiments evaluates different training criteria discussed in Section 4.1. We generate hypothesis-pair according to BW, BR and PW criteria, respectively, and perform training with these pairs. In the PW criterion, we use the sampling method of PRO (Hopkins and May, 2011) and get the 50 hypothesis pairs for each sentence. We use 20 hidden nodes for all three settings to make a fair comparison.

The results are presented in Table 2. The first two rows compare training with and without the weighted combination of hypothesis pairs we discussed in Section 4.3. As the result suggested, with the weighted combination of hypothesis pairs from previous iterations, the performance improves significantly on both test sets.

[4]http://www.alglib.net/

Although the system performance on the dev set varies, the performance on test sets are almost comparable. This suggest that although the three training criteria are based on different assumptions, their are basically equivalent for training translation systems.

Criteria	Pairs/iteration	Accuracy(%)
BR	19	70.7
BW	1	79.5
PW	100	67.3

Table 3: Comparison of different training criteria in number of new instances per iteration and training accuracy.

We also compares the three training criteria in their number of new instances per iteration and final training accuracy (Table 3). Compared to BR which tries to separate the best hypothesis from the rest hypotheses in the n-best set, and PW which tries to obtain a correct ranking of all hypotheses, BW only aims at separating the best and worst hypothesis of each iteration, which is a easier task for learning a classifiers. It requires the least training instances and achieves the best performance in training. Note that, the accuracy for each system in Table 3 are the accuracy each system achieves after training stops. They are not calculated on the same set of instances, thus not directly comparable. We use the differences in accuracy as an indicator for the difficulties of the corresponding learning task.

For the rest of this paper, we use the BW criterion because it is much simpler compared to sampling method of PRO (Hopkins and May, 2011).

6.3 Experiments of Network Structures

We make several comparisons of the network structures and compare them with a baseline hierarchical phrase-based translation system (HPB).

Table 4 shows the translation performance of

Systems	MT03(train)	MT02(dev)	MT04	MT05	Test Average
HPB	39.25	39.07	38.81	38.01	38.41
TLayer_{20}	39.55*	39.36*	38.72	37.81	38.27(-0.14)
TLayer_{30}	39.70$^+$	39.71*	38.89	37.90	38.40(-0.01)
TLayer_{50}	39.26	38.97	38.72	38.79$^+$	38.76(+0.35)
TLayer_{100}	39.42	38.77	38.65	38.65$^+$	38.69(+0.28)
TLayer_{200}	39.69	38.68	38.72	38.80$^+$	38.74(+0.32)
TDN	39.60$^+$	38.94	38.99*	38.13	38.56(+0.15)
GN	**39.73**$^+$	**39.41**$^+$	**39.45**$^+$	**38.51**$^+$	**38.98(+0.57)**

Table 4: BLEU4 in percentage for comparing of systems using different network structures (HPB refers to the baseline hierarchical phrase-based system. TLayer, TDN, GN refer to the standard 2-layer network, Two-Degree Hidden Layer Network, Grouped Network, respectively. Subscript of TLayer indicates the number of nodes in the hidden layer.) $^+$, * marks results that are significant better than the baseline system with $p < 0.01$ and $p < 0.05$.

Systems	# Hidden Nodes	# Parameters	Training Time per iter.(s)
HPB	-	11	1041
TLayer_{20}	20	261	671
TLayer_{30}	30	391	729
TLayer_{50}	50	651	952
TLayer_{100}	100	1,301	1,256
TLayer_{200}	200	2,601	2,065
TDN	55	221	808
GN	214	1,111	1,440

Table 5: Comparison of network scales and training time of different systems, including the number of nodes in the hidden layer, the number of parameters, the average training time per iteration (15 iterations). The notations of systems are the same as in Table4.

different systems[5]. All 5 two-layer feed forward neural networks models could achieve comparable or better performance comparing to the baseline system. We can see that training a larger network may lead to better translation quality (from TLayer_{20} and TLayer_{30} to TLayer_{50}). However, increasing the number of hidden node to 100 and 200 does not bring further improvement. One possible reason is that training a larger network with arbitrary connections brings in too many parameters which may be difficult to train with limited training data.

TDN and GN are the two network structures proposed in Section 5. With the constraint that all input to the hidden node should be of degree 2, TDN performs comparable to the baseline system. With the grouped feature, we could design networks such as GN, which shows significant improvement over the baseline systems (+0.57) and achieves the best performance among all neural systems.

Table 4 shows statistics related to the efficiency issue of different systems. The baseline system (HPB) uses MERT for training. HPB has a very small number of parameters and searches for the best parameters exhaustively in each iteration. The non-linear systems with few nodes (TLayer_{20} and TLayer_{30}) train faster than HPB in each iteration because they perform back-propagation instead of exhaustive search. We iterate 15 iterations for each non-linear system, while MERT takes about 10 rounds to reach its best performance.

When the number of nodes in the hidden layer increases (from 20 to 200), the number of parameters in the system also increases, which requires longer time to compute the score for each hypothesis and to update the parameters through back-propagation. The network with 200 hidden nodes takes about twice the time to train for each iteration, compared to the linear system[6].

TDN and GN have larger numbers of hidden

[5] TLayer_{20} is the same system as BW in Table 2

[6] Matrix operation is CPU intensive. The cost will increase when multiple tasks are running.

nodes. However, because of our intuitions in designing the structure of the networks, the degree of the hidden node is constrained. So these two networks are sparser in parameters and take significant less training time than standard neural networks. For example, GN has a comparable number of hidden nodes with TLayer_{200}, but only has half of its parameters and takes about 70% time to train in each iteration. In other words, our proposed network structure provides more efficient training in these cases and achieve better results.

7 Conclusion

In this paper, we discuss a non-linear framework for modeling translation hypothesis for statistical machine translation system. We also present a learning framework including training criterion and algorithms to integrate our modeling into a state of the art hierarchical phrase based machine translation system. Compared to previous effort in bringing in non-linearity into machine translation, our method uses a single two-layer neural networks and performs training independent with any previous linear training methods (e.g. MERT). Our method also trains its parameters without any pre-training or post-training procedure. Experiment shows that our method could improve the baseline system even with the same feature as input, in a large scale Chinese-English machine translation task.

In training neural networks with hidden nodes, we use heuristics to reduce the complexity of network structures and obtain extra advantages over standard networks. It shows that heuristics and intuitions of the data and features are still important to a machine translation system.

Neural networks are able to perform feature learning by using hidden nodes to model the interaction among a large vector of raw features, as in image and speech processing (Krizhevsky et al., 2012; Hinton et al., 2012). We are trying to model the interaction between hand-crafted features, which is indeed similar in spirit with learning features from raw features. Although our features already have concrete meaning, e.g. the probability of translation, the fluency of target sentence, etc. Combining these features may have extra advantage in modeling the translation process.

As future work, it is necessary to integrate more features into our learning framework. It is also interesting to see how the non-linear modeling fits in to more complex learning tasks which involves domain specific learning techniques.

Acknowledgments

The authors would like to thank Yue Zhang and the anonymous reviewers for their valuable comments. This work is supported by the National Natural Science Foundation of China (No. 61300158, 61223003), the Jiangsu Provincial Research Foundation for Basic Research (No. BK20130580).

References

Michael Auli and Jianfeng Gao. 2014. Decoder integration and expected BLEU training for recurrent neural network language models. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics, ACL 2014, June 22-27, 2014, Baltimore, MD, USA, Volume 2: Short Papers*, pages 136–142.

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, EMNLP 2013, 18-21 October 2013, Grand Hyatt Seattle, Seattle, Washington, USA, A meeting of SIGDAT, a Special Interest Group of the ACL*, pages 1044–1054.

Christopher M. Bishop. 1995. *Neural Networks for Pattern Recognition*. Oxford University Press, Inc., New York, NY, USA.

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

S. F. Chen and J. T. Goodman. 1998. An empirical study of smoothing techniques for language modeling. Technical report, Computer Science Group, Harvard University, Technical Report TR-10-98.

David Chiang. 2005. A hierarchical phrase-based model for statistical machine translation. In *annual meeting of the Association for Computational Linguistics*.

David Chiang. 2012. Hope and fear for discriminative training of statistical translation models. *J. Mach. Learn. Res.*, 13(1):1159–1187, April.

Jonathan H. Clark, Chris Dyer, Alon Lavie, and Noah A. Smith. 2011. Better hypothesis testing for statistical machine translation: Controlling for optimizer instability. In *Proceedings of the 49th Annual*

Meeting of the Association for Computational Linguistics: Human Language Technologies: Short Papers - Volume 2, HLT '11, pages 176–181, Stroudsburg, PA, USA. Association for Computational Linguistics.

Jonathan Clark, Chris Dyer, and Alon Lavie. 2014. Locally non-linear learning for statistical machine translation via discretization and structured regularization. *Transactions of the Association for Computational Linguistics*, 2:393–404.

Jacob Devlin, Rabih Zbib, Zhongqiang Huang, Thomas Lamar, Richard M. Schwartz, and John Makhoul. 2014. Fast and robust neural network joint models for statistical machine translation. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics, ACL 2014, June 22-27, 2014, Baltimore, MD, USA, Volume 1: Long Papers*, pages 1370–1380.

Kevin Duh and Katrin Kirchhoff. 2008. Beyond log-linear models: Boosted minimum error rate training for n-best re-ranking. In *Proceedings of the 46th Annual Meeting of the Association for Computational Linguistics on Human Language Technologies: Short Papers*, HLT-Short '08, pages 37–40, Stroudsburg, PA, USA. Association for Computational Linguistics.

Dumitru Erhan, Pierre antoine Manzagol, Yoshua Bengio, Samy Bengio, and Pascal Vincent. 2009. The difficulty of training deep architectures and the effect of unsupervised pre-training. In David V. Dyk and Max Welling, editors, *Proceedings of the Twelfth International Conference on Artificial Intelligence and Statistics (AISTATS-09)*, volume 5, pages 153–160. Journal of Machine Learning Research - Proceedings Track.

Jianfeng Gao, Xiaodong He, Wen-tau Yih, and Li Deng. 2014. Learning continuous phrase representations for translation modeling. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics, ACL 2014, June 22-27, 2014, Baltimore, MD, USA, Volume 1: Long Papers*, pages 699–709.

Geoffrey Hinton, Li Deng, Dong Yu, George E Dahl, Abdel-rahman Mohamed, Navdeep Jaitly, Andrew Senior, Vincent Vanhoucke, Patrick Nguyen, Tara N Sainath, et al. 2012. Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups. *Signal Processing Magazine, IEEE*, 29(6):82–97.

Mark Hopkins and Jonathan May. 2011. Tuning as ranking. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, EMNLP '11, pages 1352–1362, Stroudsburg, PA, USA. Association for Computational Linguistics.

Liang Huang and David Chiang. 2007. Forest rescoring: Faster decoding with integrated language models. In *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics*, pages 144–151, Prague, Czech Republic, June. Association for Computational Linguistics.

Philipp Koehn, Franz Josef Och, and Daniel Marcu. 2003. Statistical phrase-based translation. In *HLT-NAACL*.

Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. 2012. Imagenet classification with deep convolutional neural networks. In F. Pereira, C.J.C. Burges, L. Bottou, and K.Q. Weinberger, editors, *Advances in Neural Information Processing Systems 25*, pages 1097–1105. Curran Associates, Inc.

Quoc V. Le, Jiquan Ngiam, Adam Coates, Ahbik Lahiri, Bobby Prochnow, and Andrew Y. Ng. 2011. On optimization methods for deep learning. In *Proceedings of the 28th International Conference on Machine Learning, ICML 2011, Bellevue, Washington, USA, June 28 - July 2, 2011*, pages 265–272.

Yang Liu, Qun Liu, and Shouxun Lin. 2006. Tree-to-string alignment template for statistical machine translation. In *Proceedings of the 44th Annual Meeting of the Association of Computational Linguistics*. The Association for Computer 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, ACL 2013, 4-9 August 2013, Sofia, Bulgaria, Volume 1: Long Papers*, pages 791–801.

Shixiang Lu, Zhenbiao Chen, and Bo Xu. 2014. Learning new semi-supervised deep auto-encoder features for statistical machine translation. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 122–132, Baltimore, Maryland, June. Association for Computational Linguistics.

Sameer Maskey and Bowen Zhou. 2012. Unsupervised deep belief features for speech translation. In *INTERSPEECH 2012, 13th Annual Conference of the International Speech Communication Association, Portland, Oregon, USA, September 9-13, 2012*.

Franz Josef Och and Hermann Ney. 2002. Discriminative training and maximum entropy models for statistical machine translation. pages 295–302.

Franz Josef Och. 2003. Minimum error rate training in statistical machine translation. In *ACL '03: Proceedings of the 41st Annual Meeting on Association for Computational Linguistics*, pages 160–167, Morristown, NJ, USA. Association for Computational Linguistics.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *ACL '02: Proceedings of the 40th Annual Meeting on Association for*

Computational Linguistics, pages 311–318, Morristown, NJ, USA. Association for Computational Linguistics.

Robert E. Schapire. 1999. A brief introduction to boosting. In *Proceedings of the 16th International Joint Conference on Artificial Intelligence - Volume 2*, IJCAI'99, pages 1401–1406, San Francisco, CA, USA. Morgan Kaufmann Publishers Inc.

Ashish Vaswani, Yinggong Zhao, Victoria Fossum, and David Chiang. 2013. Decoding with large-scale neural language models improves translation. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing, EMNLP 2013, 18-21 October 2013, Grand Hyatt Seattle, Seattle, Washington, USA, A meeting of SIGDAT, a Special Interest Group of the ACL*, pages 1387–1392.

Taro Watanabe. 2012. Optimized online rank learning for machine translation. In *Proceedings of the 2012 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, NAACL HLT '12, pages 253–262, Stroudsburg, PA, USA. Association for Computational Linguistics.

Kenji Yamada and Kevin Knight. 2001. A syntax-based statistical translation model. In *Proceedings of the 39th Annual Meeting of the Association of Computational Linguistics*, pages 523–530.