Deep Neural Networks in Machine Translation: An Overview

Jiajun Zhang and Chengqing Zong, *Institute of Automation, Chinese Academy of Sciences*

Deep neural networks (DNNs) are increasingly popular in machine translation.

Due to the powerful capacity of feature learning and representation, deep neural networks (DNNs) have made big breakthroughs in speech recognition and image processing. Following recent success in signal variable processing, researchers want to figure out whether DNNs can achieve similar progress in symbol variable processing, such as natural language processing (NLP). As one of the more challenging NLP tasks, machine translation (MT) has become a testing ground for researchers who want to evaluate various kinds of DNNs.

MT aims to find for the source language sentence the most probable target language sentence that shares the most similar meaning. Essentially, MT is a sequence-to-sequence prediction task. This article gives a comprehensive overview of applications of DNNs in MT from two views: indirect application, which attempts to improve standard MT systems, and direct application, which adopts DNNs to design a purely neural MT model. We can elaborate further:

- Indirect application designs new features with DNNs in the framework of standard MT systems, which consist of multiple submodels (such as translation selection and language models). For example, DNNs can be leveraged to represent the source language context's semantics and better predict translation candidates.
- Direct application regards MT as a sequence-to-sequence prediction task and, without using any information from standard MT systems, designs two deep neural networks—an encoder, which learns continuous representations of source language sentences, and a decoder, which generates the target language sentence with source sentence representation.

Let's start by examining DNNs themselves.

Deep Neural Networks

Researchers have designed many kinds of DNNs, including deep belief networks (DBNs), deep stack networks (DSNs), convolutional neural networks (CNNs), and recurrent neural networks (RNNs). In NLP, all these DNNs aim to learn the syntactic and semantic representations for the discrete words, phrases, structures, and sentences in the real-valued continuous space so

1541-1672/15/$31.00 © 2015 IEEE
Published by the IEEE Computer Society

that similar words (phrases or structures) are near each other. We briefly introduce five popular neural networks after giving some notations—namely, we use w_i to denote the ith word of a T-word sentence, and x_i as the corresponding distributed real-valued vector. Vectors of all the words in the vocabulary form the embedding matrix $L \in R^{k\times|V|}$, where k is the embedding dimension and $|V|$ is the vocabulary size. Additionally, U and W are parameter matrices of a neural network, and b is the bias; f and e indicate the source and target sentence, respectively.

Feed-Forward Neural Network

The feed-forward neural network (FNN) is one of the simplest multilayer networks.[1] Figure 1 shows an FNN architecture with hidden layers as well as input and output layers. Taking the language model as an example, the FNN attempts to predict the conditional probability of the next word given the fixed-window history words. Suppose we have a T-word sentence, $w_1, w_2, \ldots, w_t, \ldots, w_T$; our task is to estimate the four-gram conditional probability of w_t given the trigram history $w_{t-3}, w_{t-2}, w_{t-1}$. The FNN first maps each history word into a real-valued vector $x_{t-3}, x_{t-2}, x_{t-1}$ using embedding matrix $L \in R^{k\times|V|}$; $x_{t-3}, x_{t-2}, x_{t-1}$ are then concatenated to form a single input vector $x_{t\_history}$. The hidden layers are followed to extract the abstract representation of the history words through a linear transformation $W \times x_{t\_history}$ and a nonlinear projection $f(W \times x_{t\_history} + b)$, such as $f = tanh(x)$). The softmax layer is usually adopted in the output to predict each word's probability in the vocabulary.

Recurrent Neural Network

The recurrent neural network (RecurrentNN)[2] is theoretically more powerful than FNN in language modeling due to its capability of representing all the history words rather than a fixed-length context as in FNN. Figure 2 depicts the RecurrentNN architecture. Given the history representation h_{t-1} encoding all the preceding words, we can obtain the new history representation h_t with the formula $h_t = Ux_t + Wh_{t-1}$. With h_t, we can calculate the probability of next word using the softmax function:

$$p(y_t) = \frac{e^{y_t}}{\sum_i e^{y_i}}, \tag{1}$$

where i traverses all the words in the vocabulary. Similarly, the new history representation h_t and the next word will be utilized to get the history representation h_{t+1} at time $t + 1$.

Recursive Auto-encoder

The recursive auto-encoder (RAE)[3] provides a good way to embed a phrase or a sentence in continuous space with an unsupervised or semisupervised method. Figure 3 shows an RAE architecture that learns a vector representation of a four-word phrase by recursively combining two children vectors in a bottom-up manner. By convention, the four words w_1, w_2, w_3, and w_4 are first projected into real-valued vectors x_1, x_2, x_3, and x_4.

In RAE, a standard auto-encoder (in box) is reused at each node. For two children $c_1 = x_1$ and $c_2 = x_2$, the auto-encoder computes the parent vector y_1 as follows:

$$y_1 = f(W^{(1)} [c_1; c_2] + b^{(1)}). \tag{2}$$

To assess how well the parent's vector represents its children, the standard auto-encoder reconstructs the children in a reconstruction layer:

$$[c'_1; c'_2] = f'(W^{(2)} y_1 + b^{(2)}). \tag{3}$$

The standard auto-encoder tries to minimize the reconstruction errors between the inputs and the reconstructions during training:

$$E_{rec}([c_1; c_2]) = ½ \|[c_1; c_2] - [c'_1; c'_2]\|^2. \tag{4}$$

The same auto-encoder is reused until the whole phrase's vector is generated. For unsupervised learning, the objective is to minimize the sum of reconstruction errors at each node in the optimal binary tree:

$$RAE_\theta(x) = \underset{y\in A(x)}{\operatorname{argmin}} \sum_{s\in y} E_{rec}([c_1;c_2])_s. \tag{5}$$

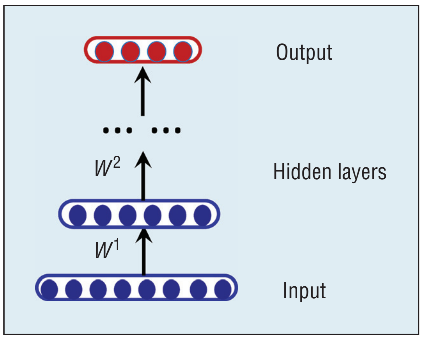


Figure 1. The feed-forward neural network (FNN) architecture. Taking the language model as an example, the FNN attempts to predict the conditional probability of the next word given the fixed-window history words.

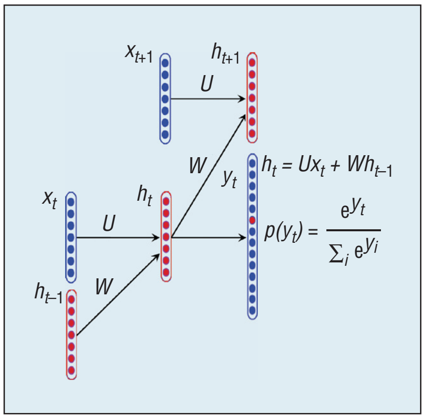


Figure 2. The recurrent neural network (RecurrentNN) architecture. Theoretically, it's more powerful than FNN in language modeling due to its capability of representing all the history words rather than a fixed-length context.

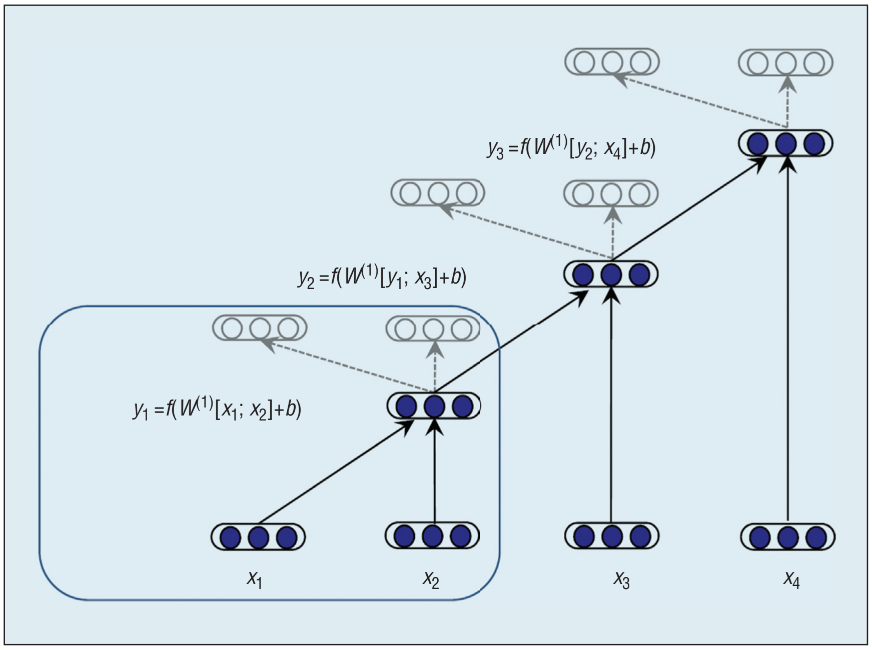

$y_3 = f(W^{(1)}[y_2; x_4]+b)$
$y_2 = f(W^{(1)}[y_1; x_3]+b)$
$y_1 = f(W^{(1)}[x_1; x_2]+b)$
x_1
x_2
x_3
x_4

Figure 3. The recursive auto-encoder (RAE) architecture. It learns a vector representation of a four-word phrase by recursively combining two children vectors in a bottom-up manner.

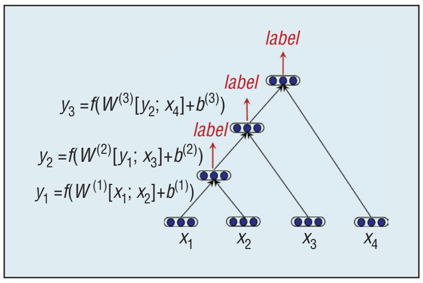

label
label
label
$y_3 = f(W^{(3)}[y_2; x_4]+b^{(3)})$
$y_2 = f(W^{(2)}[y_1; x_3]+b^{(2)})$
$y_1 = f(W^{(1)}[x_1; x_2]+b^{(1)})$
x_1
x_2
x_3
x_4

Figure 4. Recursive neural network (RecursiveNN) architecture. The structure, representation, and parameter matrices $W^{(1)}$, $W^{(2)}$, and $W^{(3)}$ have been learned to optimize the label-related supervised objective function.

$A(x)$ denotes all the possible binary trees that can be built from input x.

Recursive Neural Network

The recursive neural network (RecursiveNN)[4] performs structure prediction and representation learning using a bottom-up fashion similar to that of RAE. However, RecursiveNN differs from RAE in four points: RecursiveNN is optimized with supervised learning; the tree structure is usually fixed before training; RecursiveNN doesn't have to reconstruct the inputs; and different matrices can be used at different nodes. Figure 4 illustrates an example that applies three different matrices. The structure, representation, and parameter matrices $W^{(1)}$, $W^{(2)}$, and $W^{(3)}$ have been learned to optimize the label-related supervised objective function.

Convolutional Neural Network

The convolutional neural network (CNN)[5] consists of the convolution and pooling layers and provides a standard architecture that maps variable-length sentences into fixed-size distributed vectors. Figure 5 shows the architecture. The CNN model takes as input the sequence of word embeddings, summarizes the sentence meaning by convolving the sliding window and pooling the saliency through the sentence, and yields the fixed-length distributed vector with other layers, such as dropout and fully connected layers.

Given a sentence $w_1, w_2, \ldots, w_t, \ldots, w_T$, each word w_t is first projected into a vector x_t. Then, we concatenate all the vectors to form the input $X = [x_1, x_2, \ldots, x_t, \ldots, x_T]$.

The convolution layer involves several filters $W \in R^{h \times k}$ that summarize the information of an h-word window and produce a new feature. For the window of h words $X_{t:t+h-1}$, a filter F_l $(1 \le l \le L)$ generates the feature y_t^l as follows:

$$y_t^l = f\left(WX_{t:t+h-1} + b\right). \tag{6}$$

When a filter traverses each window from $X_{1:h-1}$ to $X_{T-h+1:T}$, we get the feature map's output: $\left[y_1^l, y_2^l, \ldots, y_{T-t+1}^l\right]$ $(y^l \in R^{T-h+1})$. Note that the sentences differ from each other in length T, and y^l has different dimensions for different sentences. A key question becomes how do we transform the variable-length vector y^l into a fixed-size vector?

The pooling layer is designed to perform this task. In most cases, we apply a standard max-over-time pooling operation over y^l and choose the maximum value $\hat{y}^l = max\left\{y^l\right\}$. With L filters, the dimension of the pooling layer output will be L. Using other layers, such as fully connected linear layers, we can finally obtain a fixed-length output representation.

Machine Translation

Statistical models dominate the MT community today. Given a source language sentence f, statistical machine translation (SMT) searches through all the target language sentences e and finds the one with the highest probability:

$$e' = \arg\max_e p(e \mid f). \tag{7}$$

Usually, $p(e|f)$ is decomposed using the log-linear model[6]:

$$e' = \arg\max_e p(e \mid f)$$
$$= \arg\max_e \frac{exp\left(\sum_i \lambda_i h_i(f, e)\right)}{\sum_{e'} exp\left(\sum_i \lambda_i h_i(f, e')\right)}, \tag{8}$$

where $h_i(f, e)$ can be any translation feature and λ_i is the corresponding weight.

The translation process can be divided into three steps: partition the source sentence (or syntactic tree) into a sequence of words or phrases (or set of subtrees), perform word/phrase or subtree translation; and composite the fragment translations to obtain the final target sentence. If the translation unit is *word*, it's the word-based model. If *phrase* is the basic translation unit, it's the popular phrase-based model. In this article, we mainly take the phrase-based SMT[7] as an example.

In the training stage, we first perform word alignment to find word correspondence between the bilingual sentences. Then, based on the word-aligned bilingual sentences, we extract phrase-based translation rules (such as the *a*–*e* translation rules in Figure 6) and learn their probabilities. Meanwhile, the phrase reordering model can be trained from the word-aligned bilingual text. In addition, the language model can be trained with the large-scale target monolingual data.

During decoding, the phrase-based model finds the best phrase partition of the source sentence, searches for the best phrase translations, and figures out the best composition of the target phrases. Figure 6 shows an example for a Chinese-to-English translation. Phrasal rules (*a*–*e*) are first utilized to get the partial translations, and then reordering rules (*f*–*i*) are employed to arrange the translation positions. Rule *g* denotes that “the two countries” and “the relations between” should be swapped. Rules *f*, *g*, and *i* just composite the target phrases monotonously. Finally, the language model measures which translation is more accurate.

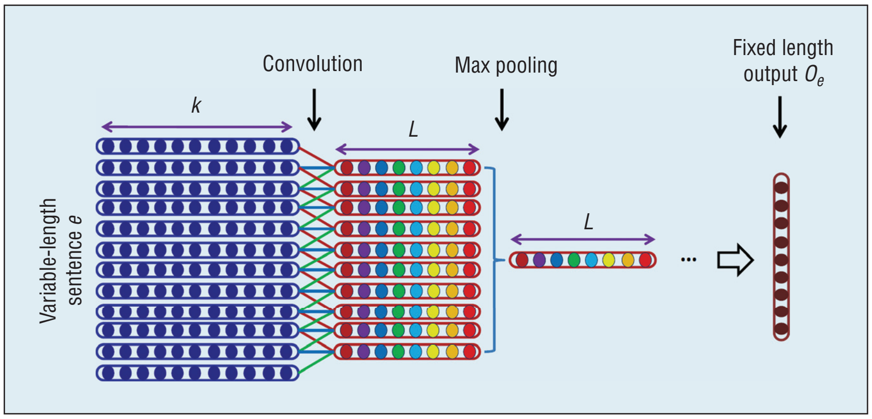
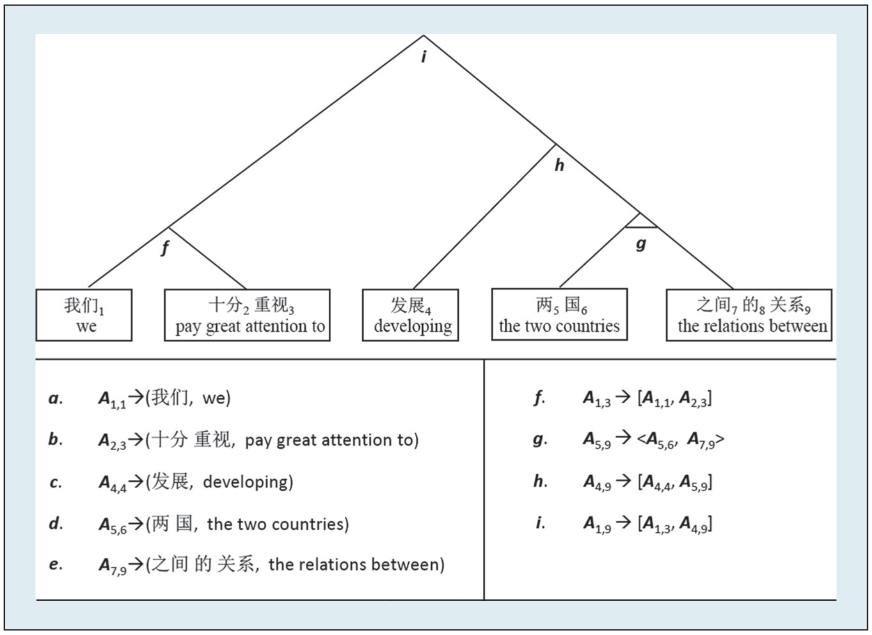


Figure 5. The convolutional neural network (CNN) architecture. The CNN model takes as input the sequence of word embeddings, summarizes the sentence meaning by convolving the sliding window and pooling the saliency through the sentence, and yields the fixed-length distributed vector with other layers, such as dropout and fully connected layers.


i
h
f
g
我们$_1$ we
十分$_2$ 重视$_3$ pay great attention to
发展$_4$ developing
两$_5$ 国$_6$ the two countries
之间$_7$ 的$_8$ 关系$_9$ the relations between

a. $A_{1,1}$→(我们, we)
b. $A_{2,3}$→(十分 重视, pay great attention to)
c. $A_{4,4}$→(发展, developing)
d. $A_{5,6}$→(两 国, the two countries)
e. $A_{7,9}$→(之间 的 关系, the relations between)
f. $A_{1,3} \rightarrow [A_{1,1}, A_{2,3}]$
g. $A_{5,9} \rightarrow \langle A_{5,6}, A_{7,9} \rangle$
h. $A_{4,9} \rightarrow [A_{4,4}, A_{5,9}]$
i. $A_{1,9} \rightarrow [A_{1,3}, A_{4,9}]$

Figure 6. An example of translation derivation structure prediction. Phrasal rules (*a*–*e*) are first utilized to get the partial translations, and then reordering rules (*f*–*i*) are employed to arrange the translation positions. Rule *g* denotes that “the two countries” and “the relations between” should be swapped. Rules *f*, *g*, and *i* just composite the target phrases monotonously. Finally, the language model measures which translation is more accurate.

Obviously, from training and decoding, we can see the difficulties in SMT:

- It's difficult to obtain accurate word alignment because we have no knowledge besides the parallel data.
- It's difficult to determine which target phrase is the best candidate for a source phrase because a source phrase can have many translations, and different contexts lead to different translations.
- It's tough work to predict the translation derivation structure because phrase partition and phrase reordering for a source sentence can be arbitrary.
- It's difficult to learn a good language model due to the data sparseness problem.

Table 1. Statistical machine translation difficulties and their corresponding deep neural network solutions.

Word alignment	FNN, RecurrentNN
Translation rule selection	FNN, RAE, CNN
Reordering and structure prediction	RAE, RecurrentNN, RecursiveNN
Language model	FNN, RecurrentNN
Joint translation prediction	FNN, RecurrentNN, CNN

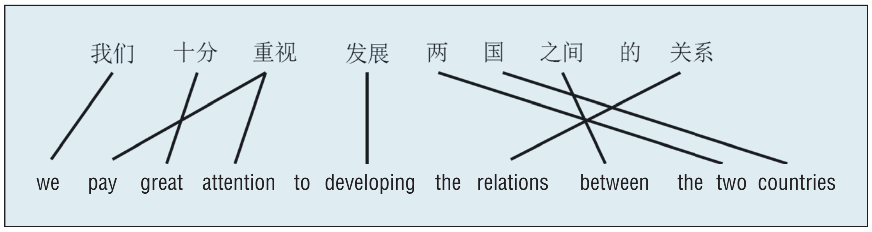

我们 十分 重视 发展 两 国 之间 的 关系
we pay great attention to developing the relations between the two countries

Figure 7. Word alignment example. Each line connecting a Chinese word to an English word indicates they are translation pairs.

The core issues lie in two areas: data sparseness (when considering additional contexts) and the lack of semantic modeling of words (phrases and sentences). Fortunately, DNNs are good at learning semantic representations and modeling wide context without severe data sparseness.

DNNs in Standard SMT Frameworks

The indirect application of DNNs in SMT aims to solve one difficult problem in an SMT system with more accurate context modeling and syntactic/semantic representation. Table 1 gives an overview of SMT problems and their various DNN solutions.

DNNs for Word Alignment

Word alignment attempts to identify the word correspondence between parallel sentence pairs. Given a source sentence $f = f_1, f_2, ..., f_t, ..., f_T$ and its target translation $e = e_1, e_2, ..., e_t, ..., e_{T'}$, the word alignment is to find the set $A = \{(i, j), 1 \le i \le T, 1 \le j \le T'\}$, in which (i, j) denotes that f_i and e_j are translations of each other. Figure 7 shows an example.

In SMT, the generative model is a popular solution for word alignment. Generative approaches use the statistics of word occurrences and learn their parameters to maximize the likelihood of the bilingual training data. They have two disadvantages: discrete symbol representation can't capture the similarity between words, and contextual information surrounding the word isn't fully explored.

Nan Yang and colleagues[8] extended the HMM word alignment model and adapted each subcomponent with an FNN. The HMM word alignment takes the following form:

$$p(a, e \mid f) = \prod_{j=1}^{T'} p_{lex}(e_j \mid f_{a_j}) p_d(a_j - a_{j-1}), \quad (9)$$

where p_{lex} is the lexical translation probability and p_d is the distortion probability. Both components are modeled with an FNN. For the lexical translation score, the authors employed the following formula:

$$s_{\text{lex}}(e_j|f_i, e, f) = f^3 \circ f^2 \circ f^1 \circ L \, (window(e_j), window\,(f_i)). \quad (10)$$

The FNN-based approach considers the bilingual contexts ($window(e_j)$ and $window\,(f_i)$). All the source and target words in the window are mapped into vectors using $L \in R^{k \times |V|}$ and concatenated to feed to hidden layers f^1 and f^2. Finally, the output layer f^3 generates a translation score. A similar FNN is applied to model the distortion score $s_d(a_j - a_{j-1})$. This DNN-based method not only can learn the bilingual word embedding that captures the similarity between words, but can also make use of wide contextual information.

Akihiro Tamura and colleagues[9] adopted RecurrentNN to extend the FNN-based model. Because the FNN-based approach can only explore the context in a window, the RecurrentNN predicts the jth alignment a_j by conditioning on all the preceding alignments a_1^{j-1}.

The reported experimental results indicate that RecurrentNN outperforms FNN in word alignment quality on the same test set. It also implies that RecurrentNN can capture long dependency by trying to memorize all the history.

DNNs for Translation Rule Selection

With word-aligned bilingual text, we can extract a huge number of translation rules. In phrase-based SMT, we can extract many phrase translation rules for a given source phrase. It becomes a key issue to choose the most appropriate translation rules during decoding. Traditionally, translation rule selection is usually performed according to co-occurrence statistics in the bilingual training data rather than by exploring the large context and its semantics.

Will Zou and colleagues[10] used two FNNs (one for source language and the other for target language) to learn bilingual word embeddings so as to make sure that a source word is close to its correct translation in the joint embedding space. The FNN used for source or target language takes as input the concatenation of the context words, applies one hidden layer, and finally generates a score in the output layer. For source language embedding, the objective function is as follows:

$$J_{\text{src}} + \lambda J_{\text{tgt}\rightarrow\text{src}}, \quad (11)$$

where J_{src} is the contrastive objective function with monolingual data, and $J_{\text{tgt}\rightarrow\text{src}}$ is a bilingual constraint:

$$J_{\text{tgt}\rightarrow\text{src}} = \| L_{\text{src}} - A_{\text{tgt}\rightarrow\text{src}} L_{\text{tgt}} \|^2. \quad (12)$$

Equation 12 says that after word alignment projection $A_{\text{tgt}\rightarrow\text{src}}$, the target word embeddings L_{tgt} should be close to the source embeddings L_{src}. This method has shown that it can cluster bilingual words with similar meanings. The bilingual word embeddings are adopted to calculate the semantic similarity between the source and target phrases in a phrasal rule, effectively improving the performance of translation rule selection.

Jianfeng Gao and colleagues[11] attempted to predict the similarity between a source and a target phrase using two FNNs with the objective of maximizing translation quality on a validation set. For a phrasal rule $(f_{1,\ldots,i}, e_{1,\ldots,j})$, the FNN (for source or target language) is employed first to abstract the vector representation for $f_{1,\ldots,i}$ and $e_{1,\ldots,j}$, respectively. The similarity score will be *score* $\left(f_{1,\ldots,i}, e_{1,\ldots,j}\right) = y_{f_{1,\ldots,i}}{}^{T} y_{e_{1,\ldots,j}}$. The FNN parameters are trained to optimize the score of the phrase pairs that can lead to better translation quality in the validation set.

Because word order isn't considered in the above approach, Jianjun Zhang and colleagues[12] proposed a bilingually constrained RAE (BRAE) to learn semantic phrase embeddings. As shown in Figure 3, unsupervised RAE can get the vector representation for each phrase. In contrast, the BRAE model not only tries to minimize the reconstruction error but also attempts to minimize the semantic distance between phrasal translation equivalents. By fine-tuning BRAE's parameters, the model can learn the semantic vector representation for each source and target phrase. Using BRAE, each phrase translation rule can be associated with a semantic similarity. With the help of semantic similarities, translation rule selection is much more accurate.

Lei Cui and colleagues[13] applied the auto-encoder to learn the topic representation for each sentence in the parallel training data. By associating each translation rule with topic information, topic-related rules can be selected according to the distributed similarity with the source language text.

Although these methods adopt different DNNs, they all achieve better rule prediction by addressing different aspects such as phrase similarity and topic similarity. FNN as used in the first two approaches is simple and learns much of the semantics of words and phrases with bilingual or BLEU (Bilingual Evaluation Understudy) objectives. In contrast, RAE is capable of capturing a phrase's word order information.

DNNs for Reordering and Structure Prediction

After translation rule selection, we can obtain the partial translation candidates for the source phrases (see the branches in Figure 6). The next task is to perform derivation structure prediction, which includes two subtasks: determining which two neighboring candidates should be composed first and deciding how to compose the two candidates. The first subtask hasn't been explicitly modeled to date. The second subtask is usually done via the reordering model. In SMT, the phrase reordering model is formalized as a classification problem, and discrete word features are employed, although data sparseness is a big issue and similar phrases can't share similar reordering patterns with each other.

Peng Li and colleagues[14,15] adopted the semisupervised RAE to learn the phrase representations that are sensitive to reordering patterns. For two neighboring translation candidates (f_1, e_1) and (f_2, e_2), the objective function is

$$E = \alpha E_{\text{rec}}(f_1, e_1, f_2, e_2) + (1 - a)\, E_{\text{reorder}}((f_1, e_1), (f_2, e_2)), \quad (13)$$

where $E_{\text{rec}}(f_1, e_1, f_2, e_2)$ is the sum of reconstruction errors, and $E_{\text{reorder}}((f_1, e_1), (f_2, e_2))$ is the reordering loss computed with cross-entropy error function. The semisupervised RAE shows that it can group the phrases sharing similar reordering patterns.

Feifei Zhai and colleagues[16] and Jianjun Zhang and colleagues[17] explicitly modeled the translation process of the derivation structure prediction. A type-dependent RecursiveNN[17] jointly determines which two partial translation candidates should be composed together and how that should be done. Figure 8 shows a training example. For a parallel sentence pair (f, e), the correct derivation exactly leads to e, as Figure 8a illustrates. Meanwhile, we have other wrong derivation trees in the search space (Figure 8b gives one incorrect derivation). Using RecursiveNN, we can get scores $S_{\text{RecursiveNN}}(cTree)$ and $S_{\text{RecursiveNN}}(wTree)$ for the correct and incorrect derivations. We train the model by making sure that the score of the correct derivation is better than that of incorrect one:

$$S_{\text{RecursiveNN}}(cTree) \leq S_{\text{RecursiveNN}}(wTree) + \Delta(S_{\text{RecursiveNN}}(cTree), S_{\text{RecursiveNN}}(wTree)), \quad (14)$$

where, $\Delta(S_{\text{RecursiveNN}}(cTree), S_{\text{RecursiveNN}}(wTree))$ is a structure margin.

As RecursiveNN can only explore the children information, Shujie Liu and colleagues[18] designed a model

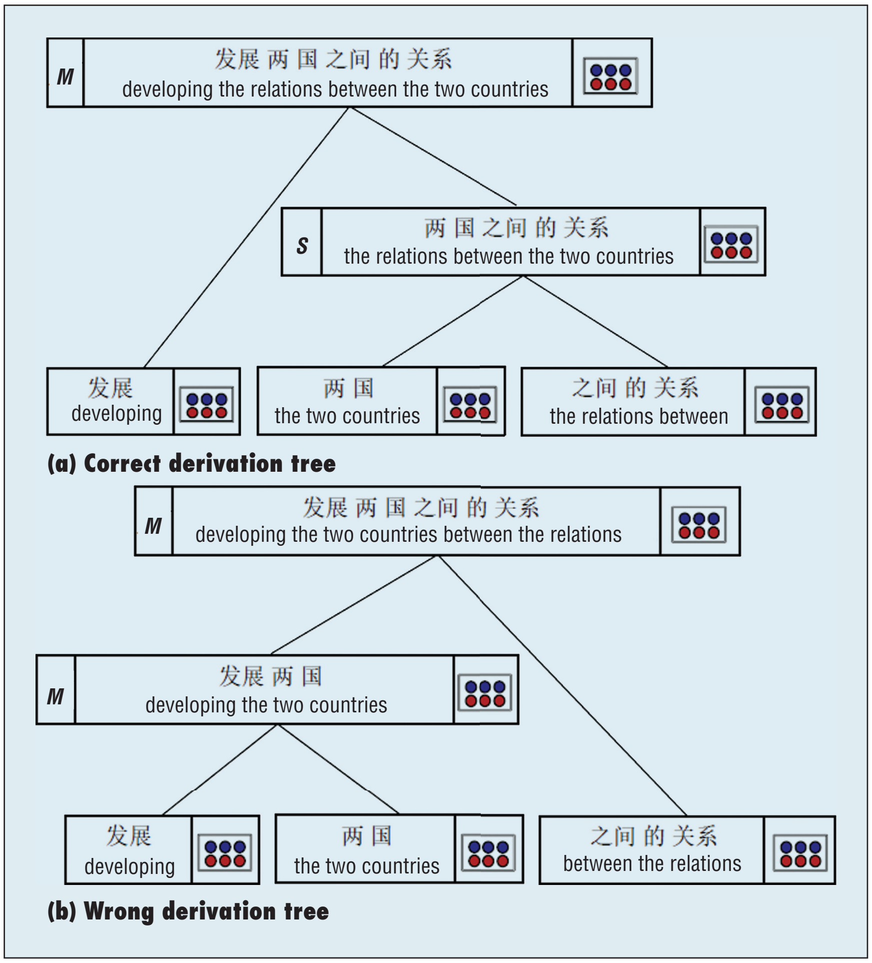

M
发展 两 国 之间 的 关系
developing the relations between the two countries
S
两 国 之间 的 关系
the relations between the two countries
发展
developing
两 国
the two countries
之间 的 关系
the relations between
(a) Correct derivation tree
M
发展 两 国 之间 的 关系
developing the two countries between the relations
M
发展 两 国
developing the two countries
发展
developing
两 国
the two countries
之间 的 关系
between the relations
(b) Wrong derivation tree

Figure 8. Type-dependent RecursiveNN: (a) correct derivation vs. (b) incorrect derivation. The correct derivation is obtained by performing forced decoding on the bilingual sentence pair; the derivation structure leads directly to the correct translation. The incorrect derivation is obtained by decoding the source sentence with the trained SMT model; it results in a wrong translation.

combining RecursiveNN and RecurrentNN together. This not only retains the capacity of RecursiveNN but also takes advantage of the history.

Compared to RAE, RecursiveNN applies different weight matrixes according to different composition types. Other work[17] has shown via experiments that RecursiveNN can outperform RAE on the same test data.

DNNs for Language Models in SMT

During derivation prediction, any composition of two partial translations leads to a bigger partial translation. The language model performs the task to measure whether the translation hypothesis is fluent. In SMT, the most popular language model is the count-based *n*-gram model. One big issue here is that data sparseness becomes severe as *n* grows. To alleviate this problem, researchers tried to design a neural network-based language model in the continuous vector space.

Yoshua Bengio and colleagues[1] designed an FNN as Figure 1 shows to learn the *n*-gram model in the continuous space. For an *n*-gram $e_1, \ldots, e_n$, each word in $e_1, \ldots, e_{n-1}$ is mapped onto a vector and concatenation of vectors feed into the input layer followed by one hidden layer and one softmax layer that outputs the probability $p(e_n|e_1, \ldots, e_{n-1})$. The network parameters are optimized to maximize the likelihood of the large-scale monolingual data. Ashish Vaswani and colleagues[19] employed two hidden layers in the FNN that's similar to Bengio's FNN.

The *n*-gram model assumes that the word depends on the previous $n - 1$ words. RecurrentNN doesn't use this assumption and models the probability of a sentence as follows:

$$p(e_1, \ldots, e_{T'}) = \prod_{j=1}^{T'} p(e_j \mid e_1, \ldots, e_{j-1}). \tag{15}$$

All the history words are applied to predict the next word.

Tomas Mikolov[20] designed the RecurrentNN (see Figure 2). A sentence start symbol <s> is first mapped to a real-valued vector as h_0 and then employed to predict the probability of e_1; h_0 and e_1 are used to form the new history h_1 to predict e_2, h_1 and e_2 generate h_2, and so on. When predicting $e_{T'}$, all the history $e_1, \ldots, e_{T'-1}$ can be used. The RecurrentNN language model is employed to rescore the *n*-best translation candidates. Michael Auli and Jianfeng Gao[21] integrated the RecurrentNN language model during the decoding stage, and further improvements can still be obtained than just rescoring the final *n*-best translation candidates.

DNNs for Joint Translation Prediction

The joint model predicts the target translation by using both of the source sentences' information and the target-side history.

Yuening Hu and colleagues[22] and Youzheng Wu and colleagues[23] cast the translation process as a language model prediction over the minimum translation units (smallest bilingual phrase pairs satisfying word alignments). They adopted RecurrentNN to model the process.

Michael Auli and colleagues[24] adapted the RecurrentNN language model and added a vector representation for the source sentence as the input along with the target history. Jacob Devlin and colleagues[25] proposed a neural network joint model (NNJM) that adapts FNN to take as input both the $n - 1$ target word history and h-window source context. They reported promising improvements over the strong baseline. Because no global information is employed in NNJM, Fandong Meng and colleagues[26] and Jiajun Zhang and colleagues[27] presented an augmented NNJM model: CNN is designed to learn the vector representation for each source sentence; then, the sentence representation augments the NNJM model's input to predict the target word generation. This approach further improves the translation quality over the NNJM model.

The RecurrentNN joint model just fits the phrase-based SMT due to the assumption that the translation is generated from left to right or right to left. In contrast, FNN and CNN can benefit all the translation models because they focus only on applying DNNs to learn the distributed representations of local and global contexts.

Purely Neural MT

Purely neural machine translation (NMT) is the new MT paradigm. The standard SMT system consists of several subcomponents that are separately optimized. In contrast, NMT employs only one neural network that's trained to maximize the conditional likelihood on the bilingual training data. The basic architecture includes two networks: one encodes the variable-length source sentence into a real-valued vector, and the other decodes the vector into a variable-length target sentence.

Kyunghyun Cho and colleagues,[28] Ilya Sutskever and colleagues,[29] and Dzmitry Bahdanau and colleagues[30] follow the similar RecurrentNN encoder-decoder architecture (see Figure 9). Given a source sentence in vector

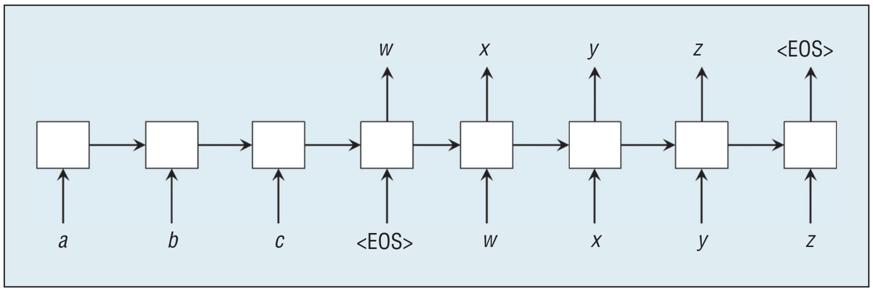


Figure 9. Neural machine translation (NMT) architecture. The model reads a source sentence *abc* and produces a target sentence *wxyz*.

sequence $X = (x_1, ..., x_T)$, the encoder applies RecurrentNN to obtain a vector $C = q(h_1, ..., h_T)$ in which h_t $(1 \le h_t \le T)$ is calculated as follows:

$$h_t = f(h_{t-1}, x_t), \tag{16}$$

where f and q are nonlinear functions. Sutskever and colleagues simplified the vector to be a fixed-length vector $C = q(h_1, ..., h_T) = h_T$, whereas Bahdanau and colleagues directly applied the variable-length vector $(h_1, ..., h_T)$ when predicting each target word.

The decoder also applies RecurrentNN to predict the target sentence $Y = (y_1, ..., y_{T'})$, where T' usually differs from T. Each target word y_t depends on the source context C and all the predicted target words $\{y_1, ..., y_{t-1}\}$; the probability of Y will be

$$p(Y) = \sum_{t=1}^{T'} p\left(y_t \mid \{y_1, \ldots, y_{t-1}\}, C_t\right). \tag{17}$$

Sutskever and colleagues chose $C_t = C = h_T$, and Bahdanau and colleagues set $C_t = \sum_{j=1}^{T} \alpha_{tj} h_j$.

All the network parameters are trained to maximize $\prod p(Y)$ in the bilingual training data. For a specific network structure, Sutskever and colleagues employed deep LSTM to calculate each hidden state, whereas Bahdanau and colleagues applied bidirectional RecurrentNN to compute the source-side hidden-state h_j. Both report similar or superior performance in English-to-French translation compared to the standard phrase-based SMT system.

The MT network architecture is simple, but it has many shortcomings. For example, it restricts tens of thousands of vocabulary words for both languages to make it workable in real applications, meaning that many unknown words appear. Furthermore, this architecture can't make use of the target large-scale monolingual data. Recently, Minh-Thang Luong and colleagues[31] and Sebastien Jean and colleagues[32] attempted to solve the vocabulary problem, but their approaches are heuristic. For example, they used a dictionary in the postprocessor to translate the unknown words.

Discussion and Future Directions

Applying DNNs to MT is a hot research topic. Indirect application is a relatively conservative attempt because it retains the standard SMT system's strength, and the log-linear SMT model facilitates the integration of DNN-based translation features that can employ different kinds of DNNs to deal with different tasks. However, indirect application makes the SMT system much more complicated.

In contrast, direct application is simple in terms of model architecture: a network encodes the source sentence and another network decodes to the target sentence. Translation quality is improving, but this new MT architecture is far from perfect. There's still an open question of how to efficiently cover most of the vocabulary, how to make use of the target large-scale monolingual data, and how to utilize more syntactic/semantic knowledge in addition to source sentences.

For both direct and indirect applications, DNNs boost translation performance. Naturally, we're interested in the following questions:

- Why can DNNs improve translation quality?
- Can DNNs lead to a big breakthrough?
- What aspects should DNNs improve if they're to become an MT panacea?

For the first question, DNNs represent and operate language units in the continuous vector space that facilitates the computation of semantic distance. For example, several algorithms such as Euclidean distance and cosine distance can be applied to calculate the similarity between phrases or sentences. But they also capture much more contextual information than standard SMT systems, and data sparseness isn't a big problem. For example, the RecurrentNN can utilize all the history information before the currently predicted target word; this is impossible with standard SMT systems.

For the second question, DNNs haven't achieved huge success with MT until recently. We've conducted some analysis and propose some key problems for SMT with DNNs:

- *Computational complexity.* Because the network structure is complicated, and normalization over the entire vocabulary is usually required, DNN training is a time-consuming task. Training a standard SMT system on millions of sentence pairs only requires about two or three days, whereas training a similar NMT system can take several weeks, even with powerful GPUs.
- *Error analysis.* Because the DNN-based subcomponent (or NMT) deals with variables in the real-valued continuous space and there are no effective approaches to show a meaningful and explainable trace from input to output, it's difficult to understand why it leads to better translation performance or why it fails.
- *Remembering and reasoning.* For current DNNs, the continuous vector representation (even using LSTM in RecurrentNN) can't remember full information for the source sentence. It's quite difficult to obtain correct target translation by decoding from this representation. Furthermore, unlike other sequence-to-sequence NLP tasks, MT is a more complicated problem that requires rich reasoning operations (such as coreference resolution). Current DNNs can't perform this kind of reasoning with simple vector or matrix operations.

These problems tell us that DNNs have a long way to go in MT. Nevertheless, due to their effective representations of languages, they could be a good solution eventually. To achieve this goal, we should pay attention to the path ahead.

First, DNNs are good at handling continuous variables, but natural language is composed of abstract discrete symbols. If they completely abandon discrete symbols, DNNs won't fully control the language generation process: sentences are discrete, not continuous. Representing and handling both discrete and continuous variables in DNNs is a big challenge.

Second, DNNs represent words, phrases, and sentences in continuous space, but what if they could mine deeper knowledge, such as parts of speech, syntactic parse trees, and knowledge graphs? What about exploring wider knowledge beyond the sentence, such as paragraphs and discourse? Unfortunately, representation, computation, and reasoning of such information in DNNs remain a difficult problem.

Third, effectively integrating DNNs into standard SMT is still worth trying. In the multicomponent system, we can study which subcomponent is indispensable and which can be completely replaced by DNN-based features. Instead of the log-linear model, we need a better mathematical model to combine multiple subcomponents.

Fourth, it's interesting and imperative to investigate more efficient algorithms for parameter learning of the complicated neural network architectures. Moreover, new network architectures can be explored in addition to existing neural networks. We believe that the best network architectures for MT must be equipped with representation, remembering, computation, and reasoning, simultaneously.

Acknowledgments

This research work was partially funded by the Natural Science Foundation of China under grant numbers 61333018 *and* 61303181, the International Science *and* Technology Cooperation Program of China under grant number 2014DFA11350, and the High New Technology Research and Development Program of Xinjiang Uyghur Autonomous Region, grant number 201312103.

References

1. Y. Bengio et al., "A Neural Probabilistic Language Model," *J. Machine Learning Research*, vol. 3, 2003, pp. 1137–1155; www.jmlr.org/papers/volume3/bengio03a/bengio03a.pdf.
2. J.L. Elman, "Distributed Representations, Simple Recurrent Networks, and Grammatical Structure," *Machine Learning*, vol. 7, 1991, pp. 195–225; http://crl.ucsd.edu/~elman/Papers/machine.learning.pdf.
3. R. Socher et al., "Semi-supervised Recursive Autoencoders for Predicting Sentiment Distributions," *Proc. Empirical Methods*

and Natural Language Process, 2011; http://nlp.stanford.edu/pubs/SocherPenningtonHuangNgManning_EMNLP2011.pdf.
4. J.B. Pollack, "Recursive Distributed Representations," *Artificial Intelligence*, vol. 46, no. 1, 1990, pp. 77–105.
5. Y. LeCun et al., "Gradient-Based Learning Applied to Document Recognition," *Proc. IEEE*, vol. 86, no. 11, 1998, pp. 2278–2324.
6. F.J. Och and H. Ney, "Discriminative Training and Maximum Entropy Models for Statistical Machine Translation," *Proc. ACL*, 2002, pp. 295–302.
7. D. Xiong, Q. Liu, and S. Lin, "Maximum Entropy Based Phrase Reordering Model for Statistical Machine Translation," *Proc. ACL*, 2006, pp. 521–528.
8. N. Yang et al., "Word Alignment Modeling with Context Dependent Deep Neural Network," *Proc. ACL*, 2013, pp. 41–46.
9. A. Tamura, T. Watanabe, and E. Sumita, "Recurrent Neural Networks for Word Alignment Model," to be published in *Proc. ACL*, 2015.
10. W.Y. Zou et al., "Bilingual Word Embeddings for Phrase-Based Machine Translation," *Proc. Empirical Methods and Natural Language Process*, 2013, pp. 1393–1398.
11. J. Gao et al., "Learning Continuous Phrase Representations for Translation Modeling," *Proc. ACL*, 2014; www.aclweb.org/anthology/P14-1066.pdf.
12. J. Zhang et al., "Bilingually-Constrained Phrase Embeddings for Machine Translation," *Proc. ACL*, 2014, pp. 111–121.
13. L. Cui et al., "Learning Topic Representation for SMT with Neural Networks," *Proc. ACL*, 2014; http://aclweb.org/anthology/P/P14/P14-1000.pdf.
14. P. Li, Y. Liu, and M. Sun, "Recursive Autoencoders for ITG-Based Translation," *Proc. Empirical Methods and Natural Language Process*, 2013, pp. 151–161.
15. P. Li et al., "A Neural Reordering Model for Phrase-Based Translation," *Proc. Conf. Computational Linguistics* (COLING), 2014, pp. 1897–1907.
16. F. Zhai et al., "RNN-Based Derivation Structure Prediction for SMT," *Proc. ACL*, 2014, pp. 779–784.
17. J. Zhang et al., "Mind the Gap: Machine Translation by Minimizing the Semantic Gap in Embedding Space," *Proc. AAAI*, 2014, pp. 1657–1664.
18. S. Liu et al., "A Recursive Recurrent Neural Network for Statistical Machine Translation," *Proc. ACL*, 2014, pp. 1491–1500.
19. A. Vaswani et al., "Decoding with Large-Scale Neural Language Models Improves Translation," *Proc. Empirical Methods and Natural Language Process*, 2013; https://aclweb.org/anthology/D/D13/D13-1.pdf.
20. T. Mikolov, "Statistical Language Models Based on Neural Networks," presentation at Google, 2012; www.fit.vutbr.cz/~imikolov/rnnlm/google.pdf.
21. M. Auli and J. Gao, "Decoder Integration and Expected BLEU Training for Recurrent Neural Network Language Models," *Proc. ACL*, 2014; http://research.microsoft.com/pubs/217163/acl2014_expbleu_rnn.pdf.
22. Y. Hu et al., "Minimum Translation Modeling with Recurrent Neural Networks," *Proc. European ACL*, 2014; www.cs.umd.edu/~ynhu/publications/eacl2014_rnn_mtu.pdf.
23. Y. Wu, T. Watanabe, and C. Hori, "Recurrent Neural Network-Based Tuple Sequence Model for Machine Translation," *Proc. Conf. Computational Linguistics* (COLING), 2014; http://anthology.aclweb.org/C/C14/C14-1180.pdf.
24. M. Auli et al., "Joint Language and Translation Modeling with Recurrent Neural Networks," *Proc. Empirical Methods and Natural Language Process*, 2013; http://research.microsoft.com/pubs/201107/emnlp2013rnnmt.pdf.
25. J. Devlin et al., "Fast and Robust Neural Network Joint Models for Statistical Machine Translation," *Proc. ACL*, 2014; http://aclweb.org/anthology/P/P14/P14-1000.pdf.
26. F. Meng, "Encoding Source Language with Convolutional Neural Network for Machine Translation," arXiv preprint arXiv:1503.01838, 2015.
27. J. Zhang, D. Zhang, and J. Hao, "Local Translation Prediction with Global Sentence Representation," to be published in *Proc. Int'l J. Conf. Artificial Intelligence*, 2015.
28. K. Cho et al., "Learning Phrase Representations Using RNN Encoder-Decoder for Statistical Machine Translation," *Proc. Empirical Methods and Natural Language Processing*, 2014, pp. 355–362.
29. I. Sutskever, O. Vinyals, and Q.V. Le, "Sequence to Sequence Learning with Neural Networks," *Proc. Neural Information Processing Systems* (NIPS), 2014; http://papers.nips.cc/paper/5346-sequence-to-sequence-learning-with-neural-networks.pdf.
30. D. Bahdanau, K. Cho, and Y. Bengio, "Neural Machine Translation by Jointly Learning to Align and Translate," arXiv preprint arXiv:1409.0473, 2014.
31. T. Luong et al., "Addressing the Rare Word Problem in Neural Machine Translation," arXiv preprint arXiv:1410.8206, 2014.
32. S. Jean et al., "On Using Very Large Target Vocabulary for Neural Machine Translation," arXiv preprint arXiv:1412.2007, 2014.

THE AUTHORS

Jiajun Zhang is an associate professor at the National Laboratory of Pattern Recognition at the Institute of Automation, Chinese Academy of Sciences. His research interests include machine translation, multilingual natural language processing, and statistical learning. Zhang has a PhD in computer science from the Institute of Automation, Chinese Academy of Sciences. Contact him at jjzhang@nlpr.ia.ac.cn.

Chengqing Zong is a professor at the National Laboratory of Pattern Recognition at the Institute of Automation, Chinese Academy of Sciences. His research interests include machine translation, natural language processing, and sentiment classification. Zong has a PhD in computer science from the Institute of Computing Technology, Chinese Academy of Sciences. He's a member of the International Committee on Computational Linguistics (ICCL). Contact him at cqzong@nlpr.ia.ac.cn.