A Convolutional Architecture for Word Sequence Prediction

Mingxuan Wang[1] **Zhengdong Lu**[2] **Hang Li**[2] **Wenbin Jiang**[1] **Qun Liu**[3,1]
[1]Key Laboratory of Intelligent Information Processing,
Institute of Computing Technology, Chinese Academy of Sciences
{wangmingxuan,jiangwenbin,liuqun}@ict.ac.cn
[2]Noah's Ark Lab, Huawei Technologies
{Lu.Zhengdong,HangLi.HL}@huawei.com
[3]ADAPT Centre, School of Computing, Dublin City University

Abstract

We propose a convolutional neural network, named genCNN, for word sequence prediction. Different from previous work on neural network-based language modeling and generation (e.g., RNN or LSTM), we choose not to greedily summarize the history of words as a fixed length vector. Instead, we use a convolutional neural network to predict the next word with the history of words of variable length. Also different from the existing feedforward networks for language modeling, our model can effectively fuse the local correlation and global correlation in the word sequence, with a convolution-gating strategy specifically designed for the task. We argue that our model can give adequate representation of the history, and therefore can naturally exploit both the short and long range dependencies. Our model is fast, easy to train, and readily parallelized. Our extensive experiments on text generation and n-best re-ranking in machine translation show that genCNN outperforms the state-of-the-arts with big margins.

1 Introduction

Both language modeling (Wu and Khudanpur, 2003; Mikolov et al., 2010; Bengio et al., 2003) and text generation (Axelrod et al., 2011) boil down to modeling the conditional probability of a word given the proceeding words. Previously, it is mostly done through purely memory-based approaches, such as n-grams, which cannot deal with long sequences and has to use some heuristics (called smoothing) for rare ones. Another family of methods are based on distributed representations of words, which is usually tied with a neural-network (NN) architecture for estimating the conditional probabilities of words.

Two categories of neural networks have been used for language modeling: 1) recurrent neural networks (RNN), and 2) feedfoward network (FFN):

- The RNN-based models, including its variants like LSTM, enjoy more popularity, mainly due to their flexible structures for processing word sequences of arbitrary lengths, and their recent empirical success(Sutskever et al., 2014; Graves, 2013). We however argue that RNNs, with their power built on the recursive use of a relatively simple computation units, are forced to make greedy summarization of the history and consequently not efficient on modeling word sequences, which clearly have a bottom-up structures.

- The FFN-based models, on the other hand, avoid this difficulty by feeding directly on the history. However, the FFNs are built on fully-connected networks, rendering them inefficient on capturing local structures of languages. Moreover their "rigid" architectures make it futile to handle the great variety of patterns in long range correlations of words.

We propose a novel convolutional architecture, named genCNN, as a model that can efficiently combine local and long range structures of language for the purpose of modeling conditional probabilities. genCNN can be directly used in generating a word sequence (i.e.,

Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing, pages 1567–1576,
Beijing, China, July 26-31, 2015.

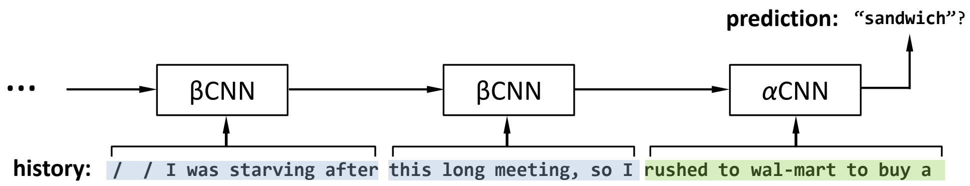

Figure 1: The overall diagram of a genCNN. Here “/” stands for a zero padding. In this example, each CNN component covers 6 words, while in practice the coverage is 30-40 words.

text generation) or evaluating the likelihood of word sequences (i.e., language modeling). We also show the empirical superiority of genCNN on both tasks over traditional n-grams and its RNN or FFN counterparts.

Notations: We will use $\mathcal{V}$ to denote the vocabulary, $\mathbf{e}_t$ ($\in \{1, \cdots, |\mathcal{V}|\}$) to denote the t^{th} word in a sequence $\mathbf{e}_{1:t} \stackrel{\text{def}}{=} [\mathbf{e}_1, \cdots, \mathbf{e}_t]$, and $\mathbf{e}_t^{(n)}$ if the sequence is further indexed by n.

2 Overview

As shown in Figure 1, genCNN is overall recursive, consisting of CNN-based processing units of two types:

- αCNN as the “front-end”, dealing with the history that is closest to the prediction;
- βCNNs (which can repeat), in charge of more “ancient” history.

Together, genCNN takes history $\mathbf{e}_{1:t}$ of arbitrary length to predict the next word $\mathbf{e}_{t+1}$ with probability

$$p(\mathbf{e}_{t+1} \,|\mathbf{e}_{1:t}; \bar{\Theta}), \tag{1}$$

based on a representation $\phi(\mathbf{e}_{1:t}; \bar{\Theta})$ produced by the CNN, and a $|\mathcal{V}|$-class soft-max:

$$p(\mathbf{e}_{t+1}|\mathbf{e}_{1:t}; \bar{\Theta}) \propto e^{\mu_{\mathbf{e}_{t+1}}^{\top} \phi(\mathbf{e}_{1:t}) + b_{\mathbf{e}_{t+1}}}. \tag{2}$$

genCNN is devised (tailored) fully for modeling the sequential structure in natural language, notably different from conventional CNN (Lawrence et al., 1997; Hu et al., 2014) in 1) its specifically designed weights-sharing strategy (in αCNN), 2) its gating design, and 3) certainly its recursive architectures. Also distinct from RNN, genCNN gains most of its processing power from the heavy-duty processing units (i.e.,αCNN and βCNNs), which follow a bottom-up information flow and yet can adequately capture the temporal structure in word sequence with its convolutional-gating architecture.

3 genCNN: Architecture

We start with discussing the convolutional architecture of αCNN as a stand-alone sentence model, and then proceed to the recursive structure. After that we give a comparative analysis on the mechanism of genCNN.

αCNN, just like a normal CNN, has fixed architecture with predefined maximum words (denoted as L_α). History shorter than L_α will filled with zero paddings, and history longer than that will be folded to feed to βCNN after it, as will be elaborated in Section 3.3. Similar to most other CNNs, αCNN alternates between convolution layers and pooling layers, and finally a fully connected layer to reach the representation before soft-max, as illustrated by Figure 2. Unlike the toyish example in Figure 2, in practice we use a larger and deeper αCNN with L_α = 30 or 40, and two or three convolution layers (see Section 4.1). Different from conventional CNN, genCNN has 1) weight sharing strategy for convolution, and 2)“external” gating networks to replace the normal pooling mechanism, both of which are specifically designed for word sequence prediction.

3.1 αCNN: Convolution

Different from conventional CNN, the weights of convolution units in αCNN is only partially shared. More specifically, in the convolution units there are two types feature-maps: TIME-FLOW and the TIME-ARROW, illustrated re-

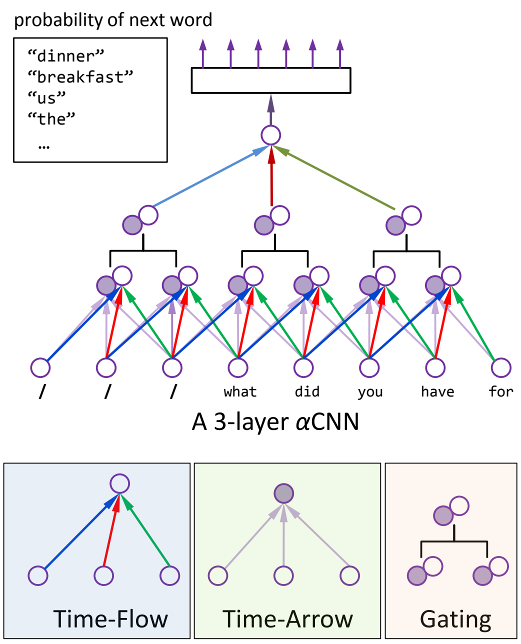

Figure 2: Illustration of a 3-layer αCNN. Here the shadowed nodes stand for the TIME-ARROW feature-maps and the unfilled nodes for the TIME-FLOW.

spectively with the unfilled nodes and filled nodes in Figure 2. The parameters for TIME-FLOW are shared among different convolution units, while for TIME-ARROW the parameters are location-dependent. Intuitively, TIME-FLOW acts more like a conventional CNN (e.g., that in (Hu et al., 2014)), aiming to understand the overall temporal structure in the word sequences; TIME-ARROW, on the other hand, works more like a traditional NN-based language model (Vaswani et al., 2013; Bengio et al., 2003): with its location-dependent parameters, it focuses on capturing the direction of time and prediction task.

For sentence input $\mathbf{x} = \{\mathbf{x}_1, \cdots, \mathbf{x}_T\}$, the feature-map of type-f on Layer-ℓ is
if $f \in$ TIME-FLOW:

$$z_i^{(\ell,f)}(\mathbf{x}) = \sigma(\mathbf{w}_{\mathsf{TF}}^{(\ell,f)} \hat{\mathbf{z}}_i^{(\ell-1)} + b_{\mathsf{TF}}^{(\ell,f)}), \quad (3)$$

if $f \in$ TIME-ARROW:

$$z_i^{(\ell,f)}(\mathbf{x}) = \sigma(\mathbf{w}_{\mathsf{TA}}^{(\ell,f,i)} \hat{\mathbf{z}}_i^{(\ell-1)} + b_{\mathsf{TA}}^{(\ell,f,i)}), \quad (4)$$

where

- $z_i^{(\ell,f)}(\mathbf{x})$ gives the output of feature-map of type-f for location i in Layer-ℓ;
- $\sigma(\cdot)$ is the activation function, e.g., Sigmoid or Relu (Dahl et al., 2013)
- $\mathbf{w}_{\mathsf{TF}}^{(\ell,f)}$ denotes the location-independent parameters for $f \in$ TIME-FLOW on Layer-ℓ, while $\mathbf{w}_{\mathsf{TA}}^{(\ell,f,i)}$ stands for that for $f \in$ TIME-ARROW and location i on Layer-ℓ;
- $\hat{\mathbf{z}}_i^{(\ell-1)}$ denotes the segment of Layer-$\ell-1$ for the convolution at location i , while

  $$\hat{\mathbf{z}}_i^{(0)} \stackrel{\text{def}}{=} [\mathbf{x}_i^\top, \ \mathbf{x}_{i+1}^\top, \ \cdots, \ \mathbf{x}_{i+k_1-1}^\top]^\top$$

  concatenates the vectors for k_1 words from sentence input $\mathbf{x}$.

3.2 Gating Network

Previous CNNs, including those for NLP tasks (Hu et al., 2014; Kalchbrenner et al., 2014), take a straightforward convolution-pooling strategy, in which the "fusion" decisions (e.g., selecting the largest one in max-pooling) are based on the values of feature-maps. This is essentially a soft template matching, which works for tasks like classification, but undesired for maintaining the composition functionality of convolution. In this paper, we propose to use separate gating networks to release the scoring duty from the convolution, and let it focus on composition. Similar idea has been proposed by (Socher et al., 2011) for recursive neural networks on parsing, but never been combined with a convolutional structure.

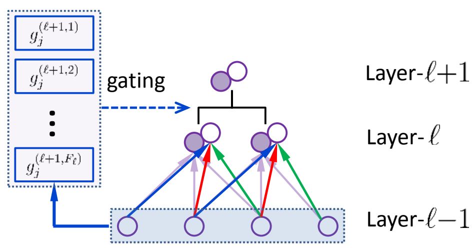

Figure 3: Illustration for gating network.

Suppose we have convolution feature-maps on Layer-ℓ and gating (with window size = 2) on Layer-$\ell+1$. For the j^{th} gating window $(2j-1, 2j)$, we merge $\hat{\mathbf{z}}_{2j-1}^{(\ell-1)}$ and $\hat{\mathbf{z}}_{2j}^{(\ell-1)}$ as the input (denoted as $\bar{\mathbf{z}}_j^{(\ell)}$) for gating network, as illustrated in Figure 3. We use a separate gate for each feature-map, but follow a different parametrization strategy for TIME-FLOW and TIME-ARROW. With window size = 2, the gating is binary, we use a logistic regressor to determine the weights of two candidates. For $f \in$ TIME-ARROW, with location-dependent $\mathbf{w}_{\text{gate}}^{(\ell,f,j)}$, the normalized weight for *left* side is

$$g_j^{(\ell+1,f)} = 1/(1 + e^{-\mathbf{w}_{\text{gate}}^{(\ell,f,j)} \bar{\mathbf{z}}_j^{(\ell)}}),$$

while for For $f \in$TIME-FLOW, the parameters for the corresponding gating network, denoted as $\mathbf{w}_{\text{gate}}^{(\ell,f)}$, are shared. The gated feature map is then a weighted sum to feature-maps from the two windows:

$$z_j^{(\ell+1,f)} = g_j^{(\ell+1,f)} z_{2j-1}^{(\ell,f)} + (1 - g_j^{(\ell+1,f)}) z_{2j}^{(\ell,f)}. \quad (5)$$

We find that this gating strategy works significantly better than pooling directly over feature-maps, and slightly better than a hard gate version of Equation 5

3.3 Recursive Architecture

As suggested early on in Section 2 and Figure 1, we use extra CNNs with conventional weight-sharing, named βCNN, to summarize the history out of scope of αCNN. More specifically, the output of βCNN (with the same dimension of word-embedding) is put before the first word as the input to the αCNN, as illustrated in Figure 4. Different from αCNN, βCNN is designed just to summarize the history, with weight shared across its convolution units. In a sense, βCNN has only TIME-FLOW feature-maps. All βCNN are identical and recursively aligned, enabling *gen*CNN to handle sentences with arbitrary length. We put a special switch after each βCNN to turn it off (replacing a pading vector shown as "/" in Figure 4) when there is no history assigned to it. As the result, when the history is shorter than L_α, the recursive structure reduces to αCNN.

In practice, 90+% sentences can be modeled by αCNN with $L_\alpha = 40$ and 99+% sentences can be contained with one extra βCNN. Our experiment shows that this recursive strategy yields better estimate of conditional density than neglecting the out-of-scope history (Section 6.1.2). In practice, we found that a larger (greater L_α) and deeper αCNN works

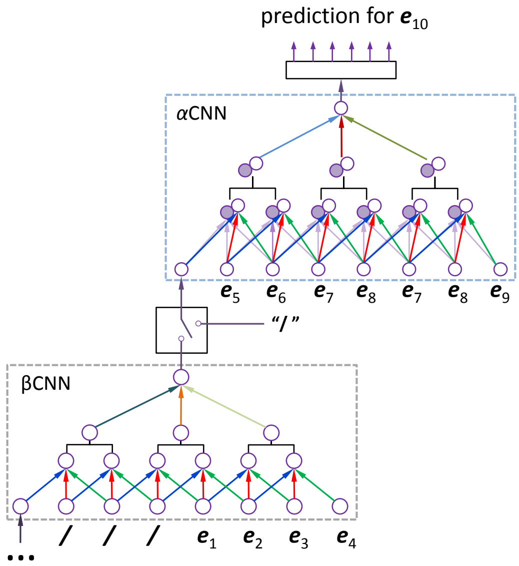

Figure 4: *gen*CNN with recursive structure.

better than small αCNN and more recursion, which is consistent with our intuition that the convolutional architecture is better suited for modeling the sequence.

3.4 Analysis

3.4.1 TIME-FLOW vs. TIME-ARROW

Both conceptually and systemically, *gen*CNN gives two interweaved treatments of word history. With the globally-shared parameters in the convolution units, TIME-FLOW summarizes *what has been said*. The hierarchical convolution+gating architecture in TIME-FLOW enables it to model the composition in language, yielding representation of segments at different intermediate layers. TIME-FLOW is aware of the sequential direction, inherited from the space-awareness of CNN, but it is not sensitive enough about the prediction task, due to the uniform weights in the convolution.

On the other hand, TIME-ARROW, living in location-dependent parameters of convolution units, acts like an arrow pin-pointing the prediction task. TIME-ARROW has predictive power all by itself, but it concentrates on capturing the direction of time and consequently short on modelling the long-range dependency.

TIME-FLOW and TIME-ARROW have to work together for optimal performance in predicting *what is going to be said*. This intuition has been empirically verified, as our experiments have demonstrated that TIME-FLOW or TIME-ARROW alone perform inferiorly. One can imagine, through the layer-by-layer convolution and gating, the TIME-ARROW gradually picks the most relevant part from the representation of TIME-FLOW for the prediction task, even if that part is long distance ahead.

3.4.2 genCNN vs. RNN-LM

Different from RNNs, which recursively applies a relatively simple processing units, genCNN gains its ability on sequence modeling mostly from its flexible and powerful bottom-up and convolution architecture. genCNN takes the "uncompressed" history, therefore avoids

- the difficulty in finding the representation for history, e.g., those end in the middle of a chunk (e.g.,"`the cat sat on the`"),
- the damping effort in RNN when the history-summarizing hidden state is updated at each time stamp, which renders the long-range memory rather difficult,

both of which can only be partially ameliorated with complicated design of gates (Hochreiter and Schmidhuber, 1997) and or more heavy processing units (essentially a fully connected DNN) (Sutskever et al., 2014).

4 genCNN: Training

The parameters of a genCNN $\bar{\Theta}$ consists of the parameters for CNN Θ_{nn}, word-embedding Θ_{embed}, and the parameters for soft-max $\Theta_{softmax}$. All the parameters are jointly learned by maximizing the likelihood of observed sentences. Formally the log-likelihood of sentence $\mathcal{S}_n$ ($\stackrel{\text{def}}{=} [\mathbf{e}_1^{(n)}, \mathbf{e}_2^{(n)}, \cdots, \mathbf{e}_{T_n}^{(n)}]$) is

$$\log p(\mathcal{S}_n; \bar{\Theta}) = \sum_{t=1}^{T_n} \log p(\mathbf{e}_t^{(n)} | \mathbf{e}_{1:t-1}^{(n)}; \bar{\Theta}),$$

which can be trivially split into T_n training instances during the optimization, in contrast to the training of RNN that requires unfolding through time due to the temporal-dependency of the hidden states.

4.1 Implementation Details

Architectures: In all of our experiments (Section 5 and 6) we set the maximum words for αCNN to be 30 and that for βCNN to be 20. αCNN have two convolution layers (both containing TIME-FLOW and TIME-ARROW convolution) and two gating layers, followed by a fully connected layer (400 dimension) and then a soft-max layer. The numbers of feature-maps for TIME-FLOW are respectively 150 (1st convolution layer) and 100 (2nd convolution layer), while TIME-ARROW has the same feature-maps. βCNN is relatively simple, with two convolution layer containing only TIME-FLOW with 150 feature-maps, two gating layers and a fully connected layer. We use ReLU as the activation function for convolution layers and switch to Sigmoid for fully connected layers. We use word embedding with dimension 100.

Soft-max: Calculating a full soft-max is expensive since it has to enumerate all the words in vocabulary (in our case 40K words) in the denominator. Here we take a simple hierarchical approximation of it, following (Bahdanau et al., 2014). Basically we group the words into 200 clusters (indexed by c_m), and factorize (in an approximate sense) the conditional probability of a word $p(\mathbf{e}_t|\mathbf{e}_{1:t-1}; \bar{\Theta})$ into the probability of its cluster and the probability of $\mathbf{e}_t$ given its cluster

$$p(c_m|\mathbf{e}_{1:t-1}; \bar{\Theta})\, p(\mathbf{e}_t|c_m; \Theta_{softmax}).$$

We found that this simple heuristic can speed-up the optimization by 5 times with only slight loss of accuracy.

Optimization: We use stochastic gradient descent with mini-batch (size 500) for optimization, aided further by AdaGrad (Duchi et al., 2011). For initialization, we use Word2Vec (Mikolov et al., 2013) for the starting state of the word-embeddings (trained on the same dataset as the main task), and set all the other parameters by randomly sampling from uniform distribution in $[-0.1, 0.1]$. The optimization is done mainly on a Tesla K40 GPU, which takes about 2 days for the training on a dataset containing 1M sentences.

5 Experiments: Sentence Generation

In this experiment, we randomly generate sentences by recurrently sampling

$$\mathbf{e}^{\star}_{t+1} \sim p(\mathbf{e}_{t+1}|\mathbf{e}_{1:t};\bar{\Theta}),$$

and put the newly generated word into history, until EOS (end-of-sentence) is generated. We consider generating two types of sentences: 1) the plain sentences, and 2) sentences with dependency parsing, which will be covered respectively in Section 5.1 and 5.2.

5.1 Natural Sentences

We train genCNN on Wiki data with 112M words for one week, with some representative examples randomly generated given in Table 1 (upper and middle blocks). We try two settings, by letting genCNN generate a sentence 1)from the very beginning (middle block), or 2) starting with a few words given by human (upper block). It is fairly clear that most of the time genCNN can generate sentences that are syntactically grammatical and semantically meaningful. More specifically, most of the sentences can be aligned to a parse tree with reasonable structure. It is also worth noting that quotation marks (`` and '') are always generated in pairs and in the correct order, even across a relatively long distance, as exemplified by the first generated sentence in the upper block.

5.2 Sentences with Dependency Tags

For training, we first parse(Klein and Manning, 2002) the English sentences and feed sequences with dependency tags as follows

```
( I ⋆ like ( red ⋆ apple ) )
```

to genCNN in training, where 1) each paired parentheses contain a subtree, and 2) the symbol "$\star$" indicates that the word next to it is the dependency head in the corresponding subtree. Some representative examples generated by genCNN are given in Table 1 (bottom block). As it suggests, genCNN is fairly accurate on respecting the rules of parentheses, and probably more remarkably, it can get the dependency tree head right most of the time.

6 Experiments: Language Modeling

We evaluate our model as a language model in terms of both perplexity (Brown et al., 1992) and its efficacy in re-ranking the n-best candidates from state-of-the-art models in statistical machine translation, with comparison to the following competitor language models.

Competitor Models we compare genCNN to the following competitor models

- 5-gram: We use SRI Language Modeling Toolkit (Stolcke and others, 2002) to train a 5-gram language model with modified Kneser-Ney smoothing;
- FFN-LM: The neural language model based on feedfoward network (Vaswani et al., 2013). We vary the input window-size from 5 to 20, while the performance stops increasing after window size 20;
- RNN: we use the implementation[1] of RNN-based language model with hidden size 600;
- LSTM: we adopt the code in Groundhog[2], but vary the hyper-parameters, including the depth and word-embedding dimension, for best performance. LSTM (Hochreiter and Schmidhuber, 1997) is widely considered to be the state-of-the-art for sequence modeling.

6.1 Perplexity

We test the performance of genCNN on PENN TREEBANK and FBIS, two public datasets with different sizes.

6.1.1 On PENN TREEBANK

Although a relatively small dataset [3], PENN TREEBANK is widely used as a language modelling benchmark (Graves, 2013; Mikolov et al., 2010). It has $930,000$ words in training set, $74,000$ words in validation set, and $82,000$ words in test set. We use exactly the same settings as in (Mikolov et al., 2010), with a $10,000$-words vocabulary (all out-of-vocabulary words are replaced with `unknown`)

[1] http://rnnlm.org/

[2] https://github.com/lisa-groundhog/GroundHog

[3] http://www.fit.vutbr.cz/~imikolov/rnnlm/simple-examples.tgz

`` we are in the building of china 's social development and the businessmen audience , '' he said .
clinton was born in DDDD , and was educated at the university of edinburgh.
bush 's first album , `` the man '' , was released on DD november DDDD .
it is one of the first section of the act in which one is covered in real place that recorded in norway .
this objective is brought to us the welfare of our country
russian president putin delivered a speech to the sponsored by the 15th asia pacific economic cooperation (apec) meeting in an historical arena on oct .
light and snow came in kuwait and became operational , but was rarely placed in houston .
johnson became a drama company in the DDDDs , a television broadcasting company owned by the broadcasting program .
((the two ⋆ sides) ⋆ should (⋆ assume (a strong ⋆ target))) .)
(it ⋆ is time (⋆ in (every ⋆ country) ⋆ signed (the ⋆ speech)) .)
((initial ⋆ investigations) ⋆ showed (⋆ that (spot ⋆ could (⋆ be (further ⋆ improved significantly)) .)
((a ⋆ book (to ⋆ northern (the 21 st ⋆ century))) .)

Table 1: Examples of sentences generated by *gen*CNN. In the upper block (row 1-4) the underline words are given by the human; In the middle block (row 5-8), all the sentences are generated without any hint. The bottom block (row 9-12) shows the sentences with dependency tag generated by *gen*CNN trained with parsed examples.

and end-of-sentence token (EOS) at the end of each sentence. In addition to the conventional testing strategy where the models are kept unchanged during testing, Mikolov et al. (2010) proposes to also update the parameters in an online fashion when seeing test sentences. This new way of testing, named "dynamic evaluation", is also adopted by Graves (2013).

From Table 2 *gen*CNN manages to give perplexity superior in both metrics, with about 25 point reduction over the widely used 5-gram, and over 10 point reduction from LSTM, the state-of-the-art and the second-best performer.

Model	**Perplexity**	**Dynamic**
5-gram, KN5	141.2	–
FFNN-LM	140.2	–
RNN	124.7	123.2
LSTM	126	117
*gen*CNN	**116.4**	**106.3**

Table 2: PENN TREEBANK results, where the 3rd column are the perplexity in dynamic evaluation, while the numbers for RNN and LSTM are taken as reported in the paper cited above. The numbers in boldface indicate that the result is significantly better than *all competitors* in the same setting.

6.1.2 On FBIS

The FBIS corpus (LDC2003E14) is relatively large, with 22.5K sentences and 8.6M English words. The validation set is NIST MT06 and test set is NIST MT08. For training the neural network, we limit the vocabulary to the most frequent 40,000 words, covering $\sim 99.4\%$ of the corpus. Similar to the first experiment, all out-of-vocabulary words are replaced with `unknown` and the EOS token is counted in the sequence loss.

From Table 3 (upper block), *gen*CNN clearly wins again in the comparison to competitors, with over 25 point margin over LSTM (in its optimal setting), the second best performer. Interestingly *gen*CNN outperforms its variants also quite significantly (bottom block): 1) with only TIME-ARROW (same number of feature-maps), the performance deteriorates considerably for losing the ability of capturing long range correlation reliably; 2) with only TIME-TIME the performance gets even worse,

Model	Perplexity
5-gram, KN5	278.6
FFN-LM(5-gram)	248.3
FFN-LM(20-gram)	228.2
RNN	223.4
LSTM	206.9
*gen*CNN	**181.2**
TIME-ARROW only	192
TIME-FLOW only	203
αCNN only	184.4

Table 3: FBIS results. The upper block (row 1-6) compares *gen*CNN and the competitor models, and the bottom block (row 7-9) compares different variants of *gen*CNN.

for partially losing the sensitivity to the prediction task. It is quite remarkable that, although αCNN (with $L_\alpha = 30$) can achieve good results, the recursive structure in full *gen*CNN can further decrease the perplexity by over 3 points, indicating that *gen*CNN can benefit from modeling the dependency over range as long as 30 words.

6.2 Re-ranking for Machine Translation

In this experiment, we re-rank the 1000-best English translation candidates for Chinese sentences generated by statistical machine translation (SMT) system, and compare it with other language models in the same setting.

SMT setup The baseline hierarchical phrase-based SMT system (Chines→ English) was built using Moses, a widely accepted state-of-the-art, with default settings. The bilingual training data is from NIST MT2012 constrained track, with reduced size of 1.1M sentence pairs using selection strategy in (Axelrod et al., 2011). The baseline use conventional 5-gram language model (LM), estimated with modified Kneser-Ney smoothing (Chen and Goodman, 1996) on the English side of the 329M-word Xinhua portion of English Gigaword(LDC2011T07). We also try FFN-LM, as a much stronger language model in decoding. The weights of all the features are tuned via MERT (Och and Ney, 2002) on NIST MT05, and tested on NIST MT06 and MT08. Case-insensitive NIST BLEU[4] is used in evaluation.

Models	MT06	MT08	Ave.
Baseline	38.63	31.11	34.87
RNN rerank	39.03	31.50	35.26
LSTM rerank	39.20	31.90	35.55
FFN-LM rerank	38.93	31.41	35.14
*gen*CNN rerank	**39.90**	**32.50**	**36.20**
Base+FFN-LM	39.08	31.60	35.34
*gen*CNN rerank	**40.4**	**32.85**	**36.63**

Table 4: The results for re-ranking the 1000-best of Moses. Note that the two bottom rows are on a baseline with enhanced LM.

Re-ranking with *gen*CNN significantly improves the quality of the final translation. Indeed, it can increase the BLEU score by over 1.33 point over Moses baseline on average. This boosting force barely slacks up on translation with a enhanced language model in decoding: *gen*CNN re-ranker still achieves 1.29 point improvement on top of Moses with FFN-LM, which is 1.76 point over the Moses (default setting). To see the significance of this improvement, the state-of-the-art Neural Network Joint Model (Devlin et al., 2014) usually brings less than one point increase on this task.

7 Related Work

In addition to the long thread of work on neural network based language model (Auli et al., 2013; Mikolov et al., 2010; Graves, 2013; Bengio et al., 2003; Vaswani et al., 2013), our work is also related to the effort on modeling long range dependency in word sequence prediction(Wu and Khudanpur, 2003). Different from those work on hand-crafting features for incorporating long range dependency, our model can elegantly assimilate relevant information in an unified way, in both long and short range, with the bottom-up information flow and convolutional architecture.

CNN has been widely used in computer vision and speech (Lawrence et al., 1997; Krizhevsky et al., 2012; LeCun and Bengio, 1995; Abdel-Hamid et al., 2012), and lately in sentence representation(Kalchbrenner and

[4]ftp://jaguar.ncsl.nist.gov/mt/resources/mteval-v11b.pl

Blunsom, 2013), matching(Hu et al., 2014) and classification(Kalchbrenner et al., 2014). To our best knowledge, it is the first time this is used in word sequence prediction. Model-wise the previous work that is closest to genCNN is the convolution model for predicting moves in the Go game (Maddison et al., 2014), which, when applied recurrently, essentially generates a sequence. Different from the conventional CNN taken in (Maddison et al., 2014), genCNN has architectures designed for modeling the composition in natural language and the temporal structure of word sequence.

8 Conclusion

We propose a convolutional architecture for natural language generation and modeling. Our extensive experiments on sentence generation, perplexity, and n-best re-ranking for machine translation show that our model can significantly improve upon state-of-the-arts.

References

[Abdel-Hamid et al.2012] Ossama Abdel-Hamid, Abdel-rahman Mohamed, Hui Jiang, and Gerald Penn. 2012. Applying convolutional neural networks concepts to hybrid nn-hmm model for speech recognition. In *Acoustics, Speech and Signal Processing (ICASSP), 2012 IEEE International Conference on*, pages 4277–4280. IEEE.

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

[Axelrod et al.2011] Amittai Axelrod, Xiaodong He, and Jianfeng Gao. 2011. Domain adaptation via pseudo in-domain data selection. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 355–362. Association for Computational Linguistics.

[Bahdanau et al.2014] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*.

[Bengio et al.2003] Yoshua Bengio, Rjean Ducharme, Pascal Vincent, and Christian Jauvin. 2003. A neural probabilistic language model. *Journal OF Machine Learning Research*, 3:1137–1155.

[Brown et al.1992] Peter F. Brown, Vincent J. Della Pietra, Robert L. Mercer, Stephen A. Della Pietra, and Jennifer C. Lai. 1992. An estimate of an upper bound for the entropy of english. *Comput. Linguist.*, 18(1):31–40, March.

[Chen and Goodman1996] Stanley F Chen and Joshua Goodman. 1996. An empirical study of smoothing techniques for language modeling. In *Proceedings of the 34th annual meeting on Association for Computational Linguistics*, pages 310–318. Association for Computational Linguistics.

[Dahl et al.2013] George E Dahl, Tara N Sainath, and Geoffrey E. Hinton. 2013. Improving deep neural networks for lvcsr using rectified linear units and dropout. In *Proceedings of ICASSP*.

[Devlin et al.2014] Jacob Devlin, Rabih Zbib, Zhongqiang Huang, Thomas Lamar, Richard 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*, pages 1370–1380.

[Duchi et al.2011] John Duchi, Elad Hazan, and Yoram Singer. 2011. Adaptive subgradient methods for online learning and stochastic optimization. *The Journal of Machine Learning Research*, 12:2121–2159.

[Graves2013] Alex Graves. 2013. Generating sequences with recurrent neural networks. *CoRR*, abs/1308.0850.

[Hochreiter and Schmidhuber1997] Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural Comput.*, 9(8):1735–1780, November.

[Hu et al.2014] Baotian Hu, Zhengdong Lu, Hang Li, and Qingcai Chen. 2014. Convolutional neural network architectures for matching natural language sentences. In *NIPS*.

[Kalchbrenner and Blunsom2013] Nal Kalchbrenner and Phil Blunsom. 2013. Recurrent continuous translation models. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1700–1709, Seattle, Washington, USA, October.

[Kalchbrenner et al.2014] Nal Kalchbrenner, Edward Grefenstette, and Phil Blunsom. 2014. A convolutional neural network for modelling sentences. *ACL*.

[Klein and Manning2002] Dan Klein and Christopher D Manning. 2002. Fast exact inference with a factored model for natural language parsing. In *Advances in neural information processing systems*, volume 15, pages 3–10.

[Krizhevsky et al.2012] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. 2012. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105.

[Lawrence et al.1997] Steve Lawrence, C Lee Giles, Ah Chung Tsoi, and Andrew D Back. 1997. Face recognition: A convolutional neural-network approach. *Neural Networks, IEEE Transactions on*, 8(1):98–113.

[LeCun and Bengio1995] Yann LeCun and Yoshua Bengio. 1995. Convolutional networks for images, speech, and time series. *The handbook of brain theory and neural networks*, 3361:310.

[Maddison et al.2014] Chris J. Maddison, Aja Huang, Ilya Sutskever, and David Silver. 2014. Move evaluation in go using deep convolutional neural networks. *CoRR*, abs/1412.6564.

[Mikolov et al.2010] Tomas Mikolov, Martin Karafit, Lukas Burget, Jan Cernocky, and Sanjeev Khudanpur. 2010. In *INTERSPEECH*, pages 1045–1048.

[Mikolov et al.2013] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space. *CoRR*, abs/1301.3781.

[Och and Ney2002] Franz Josef Och and Hermann Ney. 2002. Discriminative training and maximum entropy models for statistical machine translation. In *Proceedings of the 40th Annual Meeting on Association for Computational Linguistics*, pages 295–302.

[Socher et al.2011] Richard Socher, Cliff C. Lin, Andrew Y. Ng, and Christopher D. Manning. 2011. Parsing Natural Scenes and Natural Language with Recursive Neural Networks. In *Proceedings of the 26th International Conference on Machine Learning (ICML)*.

[Stolcke and others2002] Andreas Stolcke et al. 2002. Srilm-an extensible language modeling toolkit. In *Proceedings of the international conference on spoken language processing*, volume 2, pages 901–904.

[Sutskever et al.2014] Ilya Sutskever, Oriol Vinyals, and Quoc V Le. 2014. Sequence to sequence learning with neural networks. In *NIPS*.

[Vaswani et al.2013] Ashish Vaswani, Yinggong Zhao, Victoria Fossum, and David Chiang. 2013. Decoding with large-scale neural language models improves translation. In *EMNLP*, pages 1387–1392. Citeseer.

[Wu and Khudanpur2003] Jun Wu and Sanjeev Khudanpur. 2003. *Maximum entropy language modeling with non-local dependencies*. Ph.D. thesis, Johns Hopkins University.