Factored Recurrent Neural Network Language Model in TED Lecture Transcription

Youzheng Wu, Hitoshi Yamamoto, Xugang Lu, Shigeki Matsuda, Chiori Hori, Hideki Kashioka

Spoken Language Communication Laboratory,
National Institute of Information and Communications Technology,
Kyoto, Japan

youzheng.wu@nict.go.jp

Abstract

In this study, we extend recurrent neural network-based language models (RNNLMs) by explicitly integrating morphological and syntactic factors (or features). Our proposed RNNLM is called a factored RNNLM that is expected to enhance RNNLMs. A number of experiments are carried out on top of state-of-the-art LVCSR system that show the factored RNNLM improves the performance measured by perplexity and word error rate. In the IWSLT TED test data sets, absolute word error rate reductions over RNNLM and n-gram LM are 0.4~0.8 points.

1. Introduction

Language models (LM) are a critical component of many application systems such as automatic speech recognition (ASR), machine translation (MT) and optical character recognition (OCR). In the past, statistical back-off n-gram language models with sophisticated smoothing techniques have gained great popularity because of their simplicity and good performance. Recently, neural network based language models (NNLNs), such as the feed-forward NNLM [3, 19], the recurrent NNLM (RNNLM) [15, 16] and the deep NNLM [2], have been continuously reported to perform well amongst other language modeling techniques. Among them, RNNLMs are state-of-the-art [2, 14], which embed words in a continuous space in which probability estimation is performed using artificial neural networks consisting of input layer, hidden layer, and output layer. Due to consistent improvement in terms of perplexity and word error rate and their inherently strong generalization, they have become an increasingly popular choice for LVCSR and statistical MT tasks.

Many of these RNNLMs only use one single feature stream, i.e., surface words, which are limited to generalize over words without using linguistic information, including morphological, syntactic, or semantic. In this paper, we extend word-based RNNLMs by explicitly integrating morphological and syntactic factors (or features), called a factored RNNLM (fRNNLM), and show its performance in a LVCSR system. The experimental results of our state-of-the-art recognizer on transcribing TED lectures\(^1\) demonstrate that it significantly enhances performance measured in perplexity and word error rate (WER).

This paper is organized as follows: In Section 2, we describe our proposed factored RNNLM in detail. Section 3 shows the performance of our model as measured by both perplexity and WER. We introduce related studies in Section 4. We finally summarize our findings and outline future plans in Section 5.

2. Proposed method

The purpose of this paper is to integrate additional linguistic information into a RNNLM, called a factored RNNLM, which can improve the generalization of RNNLM using multiple factors of words (stems, lemmas, parts-of-speech, etc.) instead of surface forms of words as input to recurrent neural networks. First of all, let us use an example to illustrate the shortcomings of surface word RNNLM. In extreme cases, the training data might only contain the following sentence: “difference between developed countries and developing countries”. During training in the RNNLM that treats each word as a token in itself, the bi-gram “developing countries” is a completely unseen instance. However, for our factored RNNLM that incorporates stem features, “developing countries” belongs to seen instances in a sense because it shares the same stem bi-gram “develop countri” with the previous bi-gram “developed countries.” This coincides with our intuition; “developed” and “developing” should add knowledge to each other during training. Our factored RNNLM may be more effective for such morphologically rich languages as Czech, Arabic, or Russian. This paper however, only evaluates it on English.

2.1. fRNNLM

The architecture of our factored RNNLM is illustrated in Fig. 1. It consists of input layer \(x\), hidden layer \(s\) (state layer), and output layer \(y\). The connection weights among layers are denoted by matrices \(U\) and \(W\). Unlike RNNLM, which pre-
\(^1\)http://www.ted.com/
functions of factor extraction

\[ F(w_i) \]

\[ f_{i-1}^1, f_{i-1}^2, \ldots, f_{i-1}^K \]

\[ [F(w_{i-1}), s_{i-1}] \]

\[ x_i = [F(w_{i-1}), s_{i-1}] \]  

(2)

Using the concatenation operation, our factored RNNLM can simultaneously integrate all factors and the entire history in stead of backing-off to fewer factors and a shorter context. The weight of each factor is represented in connection weight matrix \( U \). Therefore, it can address the optimization problem well in factored n-gram LM [4, 7]. In the special case that \( f_{i-1}^1 \) is a surface word factor vector and \( f_{i-1}^k (k = 2, \ldots, K) \) are dropped, our proposed factored RNNLM goes back to the RNNLM.

The hidden layer employs a sigmoid activation function:

\[ s_i^m = f \left( \sum_j (x_i^j \times u_{m,j}) \right) \quad \forall m \in [1, H] \]

(3)

where \( H \) is the number of hidden neurons in the hidden layer and \( u_{m,j} \) is an element in matrix \( U \) denoting the corresponding connection weight.

Like [10, 16], we assume that each word belongs to exactly one class and divide the output layer into two parts: the first estimates the posterior probability distribution over all classes,

\[ g'_c = g \left( \sum_j (s_i^j \times w_{ij}) \right) \quad \forall l \in [1, C] \]

(4)

where \( C \) is the number of predefined classes. The second computes the posterior probability distribution over the
words that belong to class \( c(w_i) \), the one that contains predicted word \( w_i \) :

\[
y^o_w = g(\sum_j (s_j^i \times w_o)) \quad \forall o \in [1, nc(w_i)]
\]

(5)

where \( nc(w_i) \) is the number of words belonging to class \( c(w_i) \) and \( w_j \) and \( w_o \) are the corresponding connection weights.

To ensure that all outputs are between 0 and 1, and their sum equals to 1, the output layer employs a softmax activation function shown below:

\[
g(z_d) = \frac{e^{z_d}}{\sum_x e^{z_x}}
\]

(6)

Finally, probability \( P(w_i|F(w_{i-1}), s_{i-1}) \) is the product of two posterior probability distributions:

\[
P(w_i|F(w_{i-1}), s_{i-1}) = \frac{P(c(w_i)|F(w_{i-1}), s_{i-1}) \times \sum_l P(w_i|F(w_{i-1}), s_{i-1}, c(w_i))}{\sum_o \sum_l P(o|w_i, s_{i-1}) P(w_i|F(w_{i-1}), s_{i-1}, c(w_i))}
\]

(7)

The architecture of splitting the output layer into two parts can greatly speedup the training and the test processes of RNNLM without sacrificing much performance. Many word clustering techniques can be employed. In this paper, we map words into classes with frequency binning [16], which proportionally assigns words to classes based on their frequencies.

2.2. Training

To use the factored RNNLM, connection weight matrixes \( U \) and \( W \) must be learned. To learn them, training is performed with the back-propagation through time (BPTT) algorithm [5] by minimizing an error function defined in Eq. (8).

\[
L = \frac{1}{2} \times N \sum_{i=1} (t_i - p_i)^2 + \gamma \times (\sum_{lk} w_{lk}^2 + \sum_{tl} w_{tl}^2)
\]

(8)

where \( N \) is the number of training instances, \( t_i \) denotes the desired output; i.e., the probability should be 1.0 for the predicted word in the training sentence and 0.0 for all others. The first part of this equation is the summed squared error between the output and the desired probability distributions, and the second part is a regularization term that prevents RNNLM from over-fitting the training data. \( \gamma \) is the regularization term’s weight, which is determined experimentally using a validation set.

The training algorithm randomly initializes the matrices and updates them with Eq. (9) over all the training instances in several iterations. In Eq. (9), \( \psi \) stands for one of the connection weights in the neural network and \( \eta \) is the learning rate. After each iteration, it uses validation data for stopping and controlling the learning rate. Usually, the factored RNNLM needs 10 to 20 iterations.

\[
\psi_{\text{new}} = \psi_{\text{previous}} - \eta \times \frac{\partial L}{\partial \psi}
\]

(9)

3. Experiments

To evaluate our factored RNNLM in the context of large vocabulary speech recognition, we use the data sets for the IWSLT large vocabulary continuous speech recognition shared task [9] to recognize TED talks published on the TED website. TED talks touch on the environment, photography and psychology without adhering to a single genre. This task reflects the recent increase of interest in automatically transcribing lectures to make them either searchable or accessible.

The IWSLT evaluation campaign defines a closed set of publicly available English texts as training data for LM, including a small scale of in-domain corpus (TED transcriptions) and a large scale of general-domain corpora (English Gigaword Fifth Edition and News Commentary v7). All training data are preprocessed by a non-standard-word-expansion tool that converts non-standard words (such as CO2 or 95%) to their pronunciations (CO two, ninety five percent). The most frequent 32.6K words are extracted from the preprocessed in-domain corpora, which, with the CMU.v0.7a pronunciation dictionary3, are used as the LM vocabulary. Our vocabulary contains 156.3K entries with an OOV rate of 0.8% on the dev2010 data set. Additionally, the IWSLT data sets of tests 2010, 2011 and 2012 are used. Their statistics are shown in Table 2.

<table>
<thead>
<tr>
<th>Test sets</th>
<th>#talks</th>
<th>#utterances</th>
<th>#words</th>
</tr>
</thead>
<tbody>
<tr>
<td>dev2010</td>
<td>8</td>
<td>934</td>
<td>17.5K</td>
</tr>
<tr>
<td>test2010</td>
<td>11</td>
<td>1664</td>
<td>27.0K</td>
</tr>
<tr>
<td>test2011</td>
<td>8</td>
<td>818</td>
<td>12.4K</td>
</tr>
<tr>
<td>test2012</td>
<td>11</td>
<td>1124</td>
<td>21.9K</td>
</tr>
</tbody>
</table>

For the in-domain and general-domain corpora, modified Kneser-Ney smoothed 3- and 4-gram LMs are constructed using SRILM [21], and interpolated to form a baseline of 3- and 4-gram LMs by optimizing the perplexity of the dev2010 data set.

Acoustic models are trained on 170h speech segmented from 788 TED talks that were published prior to 2011. We

3http://www.speech.cs.cmu.edu/cgi-bin/cmudict
employ two types of schemes, a Hidden Markov Model (HMM) and a Subspace Gaussian Mixture Model (SGMM) for each context-dependent phone and train them with the Kaldi toolkit [18]. HMM consists of 6.7K states and 240K Gaussians that are discriminatively trained using the boosted Maximum Mutual Information criterion. SGMM consists of 9.2K states. In addition, we apply speaker adaptive training with feature space maximum likelihood linear regression on top of the HMM and SGMM. The acoustic feature vectors are the largest dimension with maximum likelihood linear transform.

MFCCs, splice 9 adjacent frames, and apply LDA to reduce its dimension with maximum likelihood linear transform.

First, we employ a Kaldi speech recognizer [18] to decode each utterance using the trained AM and the 3-gram LM. Second, we use the 4-gram LM for lattice re-scoring and generate n-best lists. The n-best size is at most 100 for each utterance. Finally, we use RNNLM and factored RNNLM to re-score the n-best (n=100). Since it is very time consuming to train RNNLM and factored RNNLM on large data, the usual way is to train RNNLM on a small scale of in-domain corpus. This paper also employs this setting. The corpus is automatically tagged with parts-of-speech. In the fRNNLM, we investigate three commonly used types of factors: word, stem and part-of-speech (POS). We set the numbers of free parameters, i.e., the size of matrices $U$, $V$, and $W$ in Fig. 1, in the RNNLM and factored RNNLM are interpolated with 4-gram LM. The weight of fRNNLMs are interpolated with 4-gram LM. The weight of factored RNNLM has the smallest (only 37), they have the largest impact on our factored RNNLM. The main reason may lie in that syntactic factor (POS) has stronger complementariness to the surface word, while morphological factors (stem and lemma) are too similar to the word itself, limiting such complementariness. Table 4 demonstrates the re-scoring results sampled from RNNLM and fRNNLM_wsp. This table shows that the results of fRNNLM_wsp are more grammatically fluent. Fig. 2 illustrates the absolute improvements of fRNNLM_wsp over RNNLM for each talk in the sets of tests 2010 and 2011. Our approach improves most talks, expect talks 535, 1178 and 1183.

### 3.1. Overall results

The best re-scoring results measured by word error rate (WER) are demonstrated in Table 3. Note that RNNLM and fRNNLMs are interpolated with 4-gram LM. The weight of 4-gram LM is empirically set to 0.8 to optimize the performance on the dev2010 set.

The results show that fRNNLM_wsp and fRNNLM_wsp significantly improve upon 4-gram LM and RNNLM. The largest absolute improvements over the 4-gram LM and RNNLM are 0.8 points. However, no significant differences are found among the factored RNNLMs with various combinations of factors. Although the size of the parts-of-speech is the smallest (only 37), they have the largest impact on our factored RNNLM. The main reason may lie in that syntactic factor (POS) has stronger complementariness to the surface word factor, while morphological factors (stem and lemma) are too similar to the word itself, limiting such complementariness.

### 3.2. Free parameter & time complexity

The number of free parameters, i.e., the size of matrices $U$ and $W$ in Fig. 1, in the RNNLM and factored RNNLM are $(|V| + H) \times H + H \times (C + |V|)$ and $(|f_1| + \ldots + |f_K| + H) \times H + H \times (C + |V|)$, respectively. That means, our factored RNNLM has $((|f_1| + \ldots + |f_K| - |V|) \times H$ addi-

---

**Figure 2:** Absolute WER improvement on each talk.

**Table 3:** n-best re-scoring performance in WER. Subscript numbers are the absolute improvements over the 4-gram LM. $f$RNNLM$_{wsp}$ denotes the factored RNNLM incorporating the word, stem and POS.

<table>
<thead>
<tr>
<th></th>
<th>dev2010</th>
<th>test2010</th>
<th>test2011</th>
<th>test2012</th>
</tr>
</thead>
<tbody>
<tr>
<td>RNNLM</td>
<td>16.5</td>
<td>13.8</td>
<td>12.3</td>
<td>13.9</td>
</tr>
<tr>
<td>fRNNLM$_{wsp}$</td>
<td>16.3$_{0.2}$</td>
<td>14.0$_{-0.2}$</td>
<td>12.2$_{0.1}$</td>
<td>13.9$_{0.0}$</td>
</tr>
<tr>
<td>fRNNLM$_{wsp}$</td>
<td>15.8</td>
<td>13.1</td>
<td>11.9</td>
<td>13.4</td>
</tr>
<tr>
<td>fRNNLM$_{wsp}$</td>
<td>15.7$_{0.8}$</td>
<td>13.2$_{0.6}$</td>
<td>11.8$_{0.5}$</td>
<td>13.3$_{0.6}$</td>
</tr>
</tbody>
</table>
Table 4: Re-scoring results sampled from RNNLM and fRNNLM$_{wsp}$. * denotes deletion errors, capitalized words denote substitution errors, and underlined words show their differences. #e stands for the number of errors.

<table>
<thead>
<tr>
<th>model</th>
<th>result</th>
<th>#e</th>
</tr>
</thead>
<tbody>
<tr>
<td>Reference</td>
<td>or we’ll be here all day with my childhood stories</td>
<td></td>
</tr>
<tr>
<td>RNNLM</td>
<td>THE WORLD we * * ARE all day with my childhood stories</td>
<td></td>
</tr>
<tr>
<td>fRNNLM$_{wsp}$</td>
<td>1 or * * be here all day with my childhood stories</td>
<td></td>
</tr>
<tr>
<td>Reference</td>
<td>she’s painting here a mural of his horrible final weeks in the hospital</td>
<td></td>
</tr>
<tr>
<td>RNNLM</td>
<td>she’s painting * HERO mural of his horrible final weeks in the hospital</td>
<td></td>
</tr>
<tr>
<td>fRNNLM$_{wsp}$</td>
<td>0 she’s painting here a mural of his horrible final weeks in the hospital</td>
<td></td>
</tr>
<tr>
<td>Reference</td>
<td>and so you are standing there and everything else is dark but there’s this portal that you wanna jump in</td>
<td></td>
</tr>
<tr>
<td>RNNLM</td>
<td>and so you are * STAYING IN ANYTHING else is dark but there’s THE SPORT ALL that you WANT TO jump in</td>
<td></td>
</tr>
<tr>
<td>fRNNLM$_{wsp}$</td>
<td>5 and so you are * STAYING IN ANYTHING else is dark but there’s this HORRIBLE that you WANT TO jump in</td>
<td></td>
</tr>
<tr>
<td>Reference</td>
<td>my worlds of words and numbers BLAIR with color emotion and personality</td>
<td></td>
</tr>
<tr>
<td>RNNLM</td>
<td>my WORLD SO FLOATS and numbers BELAIR with color emotion and personality</td>
<td></td>
</tr>
<tr>
<td>fRNNLM$_{wsp}$</td>
<td>1 my worlds of words and numbers BELAIR with color emotion and personality</td>
<td></td>
</tr>
</tbody>
</table>

Table 5: Elapsed time during training and test. #1 and #2 denote time of an iteration and time of all iterations during training, m=minute, s=second.

<table>
<thead>
<tr>
<th></th>
<th>#1</th>
<th>#2</th>
</tr>
</thead>
<tbody>
<tr>
<td>RNNLM</td>
<td>120m</td>
<td>1923m</td>
</tr>
<tr>
<td>fRNNLM$_{wsp}$</td>
<td>141m</td>
<td>1843m</td>
</tr>
</tbody>
</table>

Figure 3 demonstrates the convergence progress of RNNLM, fRNNLM$_{wsp}$, and fRNNLM$_{wsp}$. From this figure, we can observe that fRNNLM$_{wsp}$ outperforms RNNLM at all iterations, however, the relative improvements decrease with increasing iterations.

![Figure 3: Convergence curve on the dev2010 set.](image)

3.3. Training corpus size

This subsection analyzes the influence of training corpus size to RNNLM and fRNNLM$_{wsp}$. The training corpus is gradually increased by selecting sentences from the general-domain corpus [17, 20]. Note that, we change the order of the training data as follows, the training starts with the sentences selected from the general-domain data, and ends with the in-domain data. The selected sentences are also sorted in descending order of perplexities.

The results are shown in Table 6. This experiment indicates that the impacts of morphological and syntactic information become smaller with increasing of training data. The largest improvement of fRNNLM$_{wsp}$ trained on the in-domain data (2.4M words) reaches 0.8 points. However, this improvement reduces to 0.2 points when the model is trained on the larger training data (30M words).

4. Related work

Neural network language models to LVCSR were first presented in [3], which was a feed-forward NNLM with a fixed-length context consisting of projection, input, hidden, and output layers. Arisoy et al. [2] proposed a deep NNLM that uses multiple hidden layers instead of single hidden layer in...
feed-forward NNLMs. Furthermore, several speedup techniques such as shortlists, regrouping and block models have been proposed [19]. Feed-forward NNLMs, which predict following word $w_i$ based on any possible context of length $n-1$ history, remain a kind of $n$-gram language model.

Recurrent NNLM (RNNLM) [15, 16], which has different architecture at the input and output layers, can be considered as a deep neural network LMs because of its recurrent connections between input and hidden layers, which enable RNNLMs to use their entire history. Compared with feed-forward NNLMs, recurrent RNNLMs reduce computational complexity and have relatively fast training due to the factorization of the output layer. Other experiments [2, 14, 13] demonstrated that RNNLM significantly outperforms feed-forward NNLM. Therefore, this paper uses RNNLM as a baseline and improves it by incorporating additional information other than surface words, such as morphological or syntactic features.

Although few studies incorporate morphological and syntactic features into RNNLM, using multiple features in language modeling is not novel. For example, Bilmes and Kirchhoff [4] presented a factored back-off $n$-gram LM (FLM) that assumes each word is equivalent to a fixed number of factors, i.e., $W \equiv f^{1:K}_{i-1}$, and produces a statistic model of the following form: $p(f_{i-1}^{1:K} | f_{i-1}^{K+1:K+1})$. The standard back-off in an $n$-gram LM first drops the most distant word $w_{i-n+1}$ in the case of Eq. (1), and then the second most distant word etc. until the unigram is reached. However, the factors in FLM occur simultaneously, i.e., without forming a temporal sequence, so the order in which they should be dropped is not immediately obvious. In this case, FLM creates a large space of back-off graphs that cannot be exhaustively searched. Duh and Kirchhoff [7] employed a genetic algorithm (GA) that, however, provides no guarantee of finding the optimal back-off graph. Our factored RNNLM addresses this optimization problem well, as described in Section 3. In addition, some studies [1, 2, 8, 12] introduced various syntactic features into their feed-forward NNLMs and discriminative language models.

### 5. Conclusion

In this paper we follow the architecture of a state-of-the-art recurrent neural network language model (RNNLM) and present a factored RNNLM by integrating additional morphological and syntactic information into RNNLM. In experiments, we investigate the impacts of three commonly used types of features on our factored RNNLM: word, stem and part-of-speech. We carry out extensive experiments to evaluate the factored RNNLM performance. Our experimental results prove that factored RNNLM consistently outperforms $n$-gram LM and RNNLM in terms of the IWSLT 2010–2012 development and test data sets.

### 6. References


<table>
<thead>
<tr>
<th># of words in training data</th>
<th>dev2010 RNNLM</th>
<th>test2010 RNNLM</th>
<th>test2011 RNNLM</th>
<th>test2012 RNNLM</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.4M</td>
<td>16.3</td>
<td>15.7</td>
<td>14.0</td>
<td>13.2</td>
</tr>
<tr>
<td>9.0M</td>
<td>15.5</td>
<td>15.4</td>
<td>13.0</td>
<td>13.1</td>
</tr>
<tr>
<td>19.4M</td>
<td>15.4</td>
<td>15.3</td>
<td>12.9</td>
<td>12.9</td>
</tr>
<tr>
<td>30M</td>
<td>15.2</td>
<td>15.0</td>
<td>12.9</td>
<td>12.7</td>
</tr>
</tbody>
</table>

Table 6: Impact of training corpus size.


