Factored Language Model based on Recurrent Neural Network

Youzheng Wu Xugang Lu Hitoshi Yamamoto Shigeki Matsuda
Chiori Hori Hideki Kashioka
National Institute of Information and Communications Technology (NiCT)
3-5 Hikari-dai, Seika-cho, Soraku-gun, Kyoto, Japan, 619-0289
{youzheng.wu,xugang.lu,hitoshi.yamamoto,shigeki.matsuda}@nict.go.jp
{chiori.hori,hideki.kashioka}@nict.go.jp

Abstract

Among various neural network language models (NNLMs), recurrent neural network-based language models (RNNLMs) are very competitive in many cases. Most current RNNLMs only use one single feature stream, i.e., surface words. However, previous studies proved that language models with additional linguistic information achieve better performance. In this study, we extend RNNLM by explicitly integrating additional linguistic information, including morphological, syntactic, or semantic factors. Our proposed RNNLM is called a factored RNNLM that is expected to enhance RNNLMs. A number of experiments are carried out that show the factored RNNLM improves the performance for all considered tasks: consistent perplexity and word error rate (WER) reductions. In the Penn Treebank corpus, the relative improvements over n-gram LM and RNNLM are 29.0% and 13.0%, respectively. In the IWSLT-2011 TED ASR test set, absolute WER reductions over RNNLM and n-gram LM reach 0.63 and 0.73 points.

Title and Abstract in another language (Chinese)

基于递归神经网络的因素语言模型

在多种神经网络语言模型（NNLM）中，递归神经网络语言模型（RNNLM）在很多场合都表现出优异的性能。目前，大多数的RNNLM只使用词汇这一种特征。然而，以往的研究表明利用了语言学特征的语言模型可进一步提高模型性能。本文通过引入语言学特征，包括形态，语法，或语义特征，对RNNLM进行扩展。我们称之为基于递归神经网络的因素语言模型（fRNNLM）。一系列的实验表明：本文提出的fRNNLM在多个任务中均显著地提高了RNNLM的性能：即一致地降低了困惑度和单词错误率（WER）。在宾州树库上，fRNNLM的困惑度相对相对于n元语言模型和RNNLM分别降低了29.0%和13.0%。在IWSLT-2011 TED语音识别测试集上，fRNNLM的单词错误率分别提高了0.63（相对于n元语言模型）和0.73（相对于RNNLM）。

Keywords: Factored Language Model, Recurrent Neural Network, Large Vocabulary Continuous Speech Recognition.

Keywords in Chinese: 因素语言模型, 递归神经网络, 大词汇量语音识别.

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. In n-gram language models, words are represented in a discrete space: the vocabulary. Standard back-off n-gram language models predict the following word based on the previous n-1 words, which can be expressed as,

$$p(w_i|w_{i-1},...,w_1) \approx p(w_i|w_{i-n+1},...,w_{i-2},w_{i-1}) \quad (1)$$

Even though n is usually limited to three or four, the number of parameters in a back-off n-gram LM is still enormous. Assuming the vocabulary size is $64K$, a 4-gram language model needs to estimate $64K^2$ bigrams, $64K^3$ trigrams and $64K^4$ 4-grams. Due to data sparseness, many are not observed during the training phase. This means that n-gram LMs have poor generalization to low-frequency and unseen n-grams. This problem becomes more severe as the vocabulary size increases. Many interesting approaches have been proposed to overcome it in large vocabulary continuous speech recognition (LVCSR) and statistical machine translation systems, especially smoothing techniques (Chen and Goodman, 1996), class n-gram language models (Brown et al., 1992), topic language models (Gildea and Hofmann, 1999; Hsu and Glass, 2006), structured language models (Chelba and Jelinek, 2000), maximum entropy language models (Rosenfeld, 1996) and random forests language models (Xu and Jelinek, 2004).

Among these techniques, one of the most successful schemes is the neural network language model (NNLM), such as the feed-forward NNLM (Bengio et al., 2003; Schwenk, 2007; Kuo et al., 2012), the recurrent NNLM (RNNLM) (Mikolov et al., 2010, 2011b) and the deep NNLM (Arisoy et al., 2012). Compared to other LMs, recurrent NNLMs, which are state-of-the art (Mikolov et al., 2011a; Arisoy et al., 2012), embed words in a continuous space in which probability estimation is performed using artificial neural networks consisting of input layer, single or multiple hidden layers, 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 surface word RNNLMs, such words as “prices" and “price" and “developed" and “developing" are completely independent training instances. In this paper, we integrate additional linguistic information into a RNNLM, called a factored RNNLM, which can further improve the generalization of RNNLM using multiple factors (or features) of words (stems, lemmas, parts-of-speech, etc.) instead of surface forms of words as input to recurrent neural networks. 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. In this paper, we only evaluate it on English, and our experiments show that it significantly enhances performance measured in perplexity and WER.

To the best of our knowledge, few studies have been done on this perspective. Our approach provides the following advantages:

- It predicts the following word based on the entire history (due to a recurrent connection between input and hidden layers) in the low-dimensional representation (due to the neural network architecture).
- It integrates the additional rich information of words in particular morphological and syntactic features to overcome the data sparseness problem caused by limited in-domain training data, such as in academic lecture ASR and MT tasks.
- It simultaneously interpolates all possible factors and the entire history in stead of backing-off to fewer factors and shorter context, which can address the optimization problem well in factored n-gram language models.
- Since it converges faster than RNNLM due to the integration of additional features, it can save several days of training if the training data are large.

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

2 Related work

Recently, deep neural networks are experiencing significant improvements in the fields of image processing, acoustic modeling (Seide et al., 2011), language modeling, etc. Neural network language models to LVCSR were first presented in (Bengio et al., 2003), which was a feed-forward NNLM with a fixed-length context consisting of projection, input, hidden, and output layers. Arisoy et al. (2012) 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 (Schwenk, 2007). 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) (Mikolov et al., 2010, 2011b), 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 NNLMs reduce computational complexity and have relatively fast training due to the factorization of the output layer. Other experiments (Mikolov et al., 2011a; Arisoy et al., 2012; Kuo et al., 2012) 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, Duh and Kirchhoff (2004) presented

a factored back-off n-gram LM (FLM) that assumes each word is equivalent to a fixed number (K) of factors, i.e., $W \equiv f^{1:K}$, and produces a statistic model of the following form: $p(f_i^{1:K}|f_{i-n+1:i-1}^{1:K})$. 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 (2004) 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, Emami and Jelinek (2004); Alexandrescu and Kirchhoff (2006); Kuo et al. (2009); Collins et al. (2005) introduced various syntactic features into their feed-forward NNLMs and discriminative language models. Table 1 summarizes FLM, RNNLM, and our approach from three points of view.

	Conditioning variables	History	Pros and Cons
FLM	Word and its linguistic features	n-1 preceding history	Better than n-gram LM due to linguistic features; Creating a large space of models that cannot be searched exhaustively.
RNNLM	Word	Entire history	Further enhancing FLM due to RNN architecture; Conditioning variables are only words, no morphological or syntactic linguistic features are used.
factored RNNLM	Word and its linguistic features	Entire history	Combining the above merits, but more parameters and computation complexity, which actually does not cause problems, as described in Section 4.4.

Table 1: Comparison of FLM, RNNLM, and factored RNNLM

Koehn and Hoang (2007) introduced various features from linguistic tools or word classes into phrase-based MT models for better translation performance.

3 Factored RNNLM

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 matrixes U and W. Unlike RNNLM, which predicts probability $P(w_i|w_{i-1}, s_{i-1})$, our factored RNNLM predicts probability $P(w_i|F(w_{i-1}), s_{i-1})$ of generating following word w_i and is explicitly conditioned on a collection or bundle of K factors of one preceding word. It is implicitly conditioned on the factors of the entire history by the delay copy of hidden layer s_{i-1}. Here, $F(w_{i-1})$ is the vector concatenated from K factor vectors f_{i-1}^k ($k = 1, ..., K$), f_{i-1}^k stands for the k-th factor vector encoded from the k-th factor of preceding word w_{i-1}, and the functions of factor extraction $f^k(\cdot)$ are used to extract the corresponding factors. A word's factors can be anything, including the word itself, its morphological class, its root, and any other linguistic features. An example is shown in Table 2.

In the input layer, the extracted factors are encoded into the factor vectors using the 1-of-n coding.

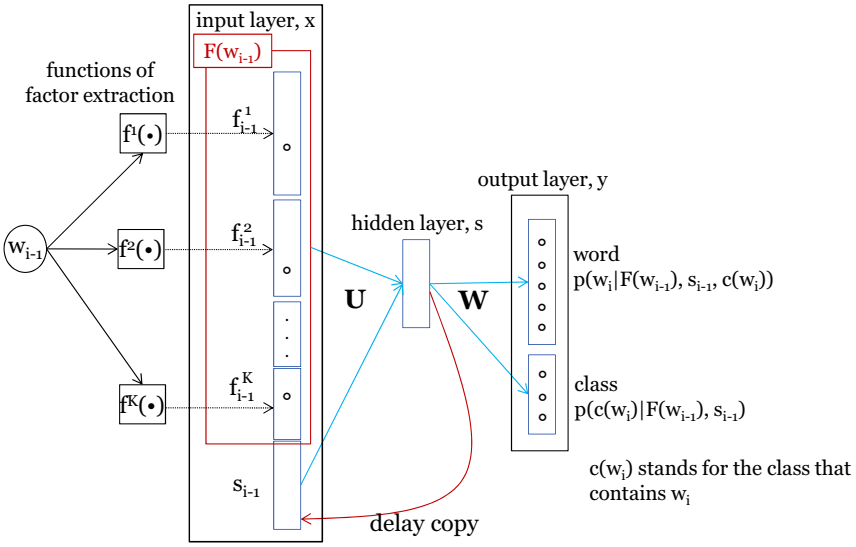


Figure 1: Architecture of factored recurrent NNLM.

Assume, for example, that the factor extracted by function $f^k(w_{i-1})$ is the m-th element in the k-th factor vocabulary, which is then encoded to $|f^k|$-dimension vector f^k_{i-1} by setting the m-th element of the vector to 1 and all the other elements to 0. Here, $|f^k|$ stands for the size of the k-th factor vocabulary. The K factor vectors are concatenated into $F(w_{i-1})$ as expressed in Eq. (2). Finally, the input layer is formed by concatenating factor vectors $F(w_{i-1})$ of the preceding word w_{i-1} and hidden layer s_{i-1} at the preceding time step, as shown in Eq. (3).

Word:	difference	between	developed	countries	and	developing	countries
Lemma:	difference	between	developed	country	and	developing	country
Stem:	differ	between	develop	countri	and	develop	countri
Part-of-speech[1]:	NN	IN	JJ	NNS	CC	VBG	NNS

Table 2: An example of factor sequences.

$$F(w_{i-1}) = [f^1_{i-1},\ f^2_{i-1},\ ...,\ f^K_{i-1}] \tag{2}$$

$$x_i = [F(w_{i-1}),\ s_{i-1}] \tag{3}$$

Using the concatenation vector, our proposed 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 (Duh and Kirchhoff, 2004). In the special case that f^1_{i-1} is a surface word factor vector and f^k_{i-1} $(k = 2, ..., K)$ are dropped, the factored RNNLM goes back to the RNNLM.

[1] `http://www.cis.upenn.edu/~treebank/`

The hidden layer employs a sigmoid activation function:

$$\begin{aligned} s_i^m &= f(\sum_j (x_i^j \times u_{mj})) \quad \forall m \in [1, H] \\ f(z) &= \frac{1}{1+e^{-z}} \end{aligned} \tag{4}$$

where H is the number of hidden neurons in the hidden layer and u_{mj} is an element in matrix U denoting the corresponding connection weight.

Like (Goodman, 2001; Mikolov et al., 2011b), 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,

$$y_c^l = g(\sum_j (s_i^j \times w_{lj})) \quad \forall l \in [1, C] \tag{5}$$

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:

$$y_w^o = g(\sum_j (s_i^j \times w_{oj})) \quad \forall o \in [1, nc(w_i)] \tag{6}$$

where $nc(w_i)$ is the number of words belonging to class $c(w_i)$ and w_{lj} and w_{oj} 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:

$$g(z_d) = \frac{e^{z_d}}{\sum_x e^{z_x}} \tag{7}$$

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

$$\begin{aligned} P(w_i|F(w_{i-1}), s_{i-1}) &= P(c(w_i)|F(w_{i-1}), s_{i-1}) \times P(w_i|F(w_{i-1}), s_{i-1}, c(w_i)) \\ &= y_c^l|_{l=classid(c(w_i))} \times y_w^o|_{o=wordid(w_i)} \end{aligned} \tag{8}$$

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 (Mikolov et al., 2011b), which proportionally assigns words to classes based on their frequencies. The pseudo codes are shown in Fig. 2.

3.1 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 (Boden, 2002) by minimizing an error function defined in Eq. (9).

$$L = \frac{1}{2} \times \sum_{i=1}^{N} (t_i - p_i)^2 + \gamma \times (\sum_{lk} u_{lk}^2 + \sum_{tl} w_{tl}^2) \tag{9}$$

```
#vocab[i].cn denotes the number of the i-th word that occurs
#vocab[i].classid denotes the class index of the i-th word
#nclass is the number of classes predefined
 double df=0, a=0, b=0;
 for (i=0; i<|V|; i++) b+=vocab[i].cn;
 for (i=0; i<|V|; i++) {
    df+=vocab[i].cn/b;
    if (df>1) df=1;
    if (df>(a+1)/nclass) {
       vocab[i].classid=a;
       if (a<nclass-1) a++;
    }
    else {
       vocab[i].classid=a;
    }
 }
```

Figure 2: Frequency binning. $|V|$ is the word vocabulary's size.

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. γ is the regularization term's weight, which is determined experimentally using a validation set.

The training algorithm randomly initializes the matrixes and updates them with Eq. (10) over all the training instances in several iterations. In Eq. (10), ψ stands for one of the connection weights in the neural network and η 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.

$$\psi^{new} = \psi^{previous} - \eta \times \frac{\partial L}{\partial \psi} \tag{10}$$

3.2 Free parameter & time complexity

To better understand the differences between RNNLM and our factored RNNLM, we compare them in terms of the number of free parameters and computational complexity of one training step in Table 3. τ is the amount of steps used in BPTT.

	Free Parameter	Computational Complexity
RNNLM (1)	$(\|V\|+H)\times H + H\times(C+\|V\|)$	$(1+H)\times H\times\tau + H\times\|V\|$
fRNNLM (2)	$(\|f^1\|+...+\|f^K\|+H)\times H + H\times(C+\|V\|)$	$(K+H)\times H\times\tau + H\times\|V\|$
Difference (2)-(1)	$(\|f^1\|+...+\|f^K\|-\|V\|)\times H$	$(K-1)\times H\times\tau$

Table 3: RNNLM vs. factored RNNLM (fRNNLM).

From this table we can observe that the factored RNNLM has more free parameters and larger computational complexity. If the factored RNNLM only employs word factor (f^1) and POS factors

(f^2), then, it has $39 \times H$ additional free parameters. The additional computational complexity is $(K-1) \times H \times \tau$. In experiments, H is usually set to $300-1000$, τ is usually set to 4, $|V|$ is usually set to several hundreds of thousands. This means that $H \times |V| \gg (K-1) \times H \times \tau$, and the increased complexity can be neglected. Owing to the additional free parameters, our factored RNNLM converges faster and reduces training time. Section 4.4 shows the exact running time spent on experiments.

4 Experiments

In this section, we show the performance of our factored RNNLM as measured by perplexity. After analyzing these results, we present the performance measured by word error rate when the factored RNNLM is used in a LVCSR system. In our experiments, we mainly compare our factored RNNLMs with a 4-gram LM with modified Kneser-Ney smoothing (Chen and Goodman, 1996) and RNNLM (Mikolov et al., 2011b). In the factored RNNLM, we investigate four commonly used types of factors: word, stem[2], lemma[3] and part-of-speech (POS).

For perplexity results, we use the WSJ portion of Penn Treebank (LDC99T42). The WSJ portion is divided into training (sections 00-20), heldout (sections 21-22), and test (sections 23- 24) sets containing 930K, 101K, and 97K words respectively. The vocabulary is limited to 10K words. This setting is the same as that used by other studies (Xu and Jelinek, 2004; Mikolov et al., 2011b). The sizes of the factor vocabularies in the training set are shown in Table 4[4]. Note that the word vocabulary (10001 in Table 4) contains 10K words and one special token "<unk>" denoting words not in the vocabulary.

Factors	Word	Lemma	Stem	POS
Sizes	10001	7356	6892	37

Table 4: Statistics of factor vocabularies.

4.1 Impacts of factors

This experiment analyzes the contribution from each factor to the factored RNNLM in terms of the perplexities on the heldout and test sets. We set the number of hidden neurons in the hidden layer and the number of classes in the output layer for both the RNNLM and factored RNNLM to 320 and 300. Table 5 shows the experimental results. fRNNLM$_{wslp}$ denotes the factored RNNLM incorporating the word, stem, lemma, and POS factors, and so forth, the ratio is computed using $\frac{|U|_{factored\ RNNLM}-|U|_{RNNLM}}{|U|_{RNNLM}}$ that indicates the percentage of additional parameters in matrix U against the RNNLM. Subscript numbers are the relative improvements over RNNLM.

From this table, we observe the following: (1) All of the factored RNNLMs significantly improve their performances. For example, the improvement of fRNNLM$_{wsp}$ against the RNNLM on the test set reaches 14.4%. (2) No significant differences are found among the factored RNNLMs with various combinations of factors. The contributions from stem and lemma factors are less than 1.0%. In particular, it is not necessary to use both stem and lemma because they are very similar and obviously do not complement each other. (3) Although the size of the parts-of-speech is the

[2] `http://tartarus.org/~martin/PorterStemmer/`

[3] `http://lemmatizer.org/turglem-english-description`

[4] We directly use manually tagged parts-of-speech in the Penn Treebank corpus. Section 4.6 investigates automatically tagged parts-of-speech.

	Ratio	Heldout	Test
4-gramLM		156.26	156.41
RNNLM	-	146.94	145.63
fRNNLM$_{wp}$	0.4%	$128.14_{12.8\%}$	$126.47_{13.1\%}$
fRNNLM$_{wsp}$	67.5%	$127.09_{13.4\%}$	$124.63_{14.4\%}$
fRNNLM$_{wlp}$	72.0%	$126.81_{13.7\%}$	$124.76_{14.3\%}$
fRNNLM$_{wslp}$	138.8%	$126.06_{14.2\%}$	$124.76_{14.3\%}$

Table 5: Impacts of factors measured by perplexities.

smallest (only 37, Table 4), 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. Therefore, in the following experiments we only use word, stem, and POS in our factored RNNLM.

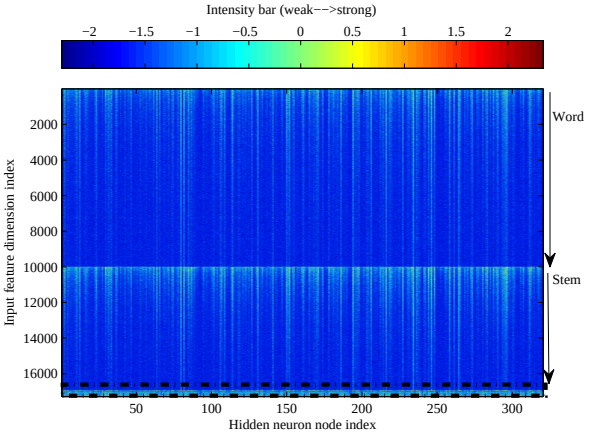
For a better understanding of the contribution of each factor to the factored RNNLM, we do a quantitative analysis of the connection weight values. The basic assumption in this analysis is that if one feature has a strong correlation or contribution to the factored RNNLM, the connections between the input features to the hidden neurons have large values (either positive or negative corresponding to positive or negative correlations). We show connection weight matrix U (corresponding to the logs of the absolute values of neural connection weights) in Figs. 3 (a) and (b). The horizontal and vertical axis denote the hidden neurons and the input feature dimensions. Since feature streams (word, stem, POS and history) are organized in blocks in matrix U, we mark each feature stream in blocks on the right vertical axis. In these figures, the connection intensity is marked by color, the brighter the color, the stronger the connection. From these figures, we can see that the POS feature stream shows the strongest connection intensity among all feature streams. The POS feature stream contributes the most to the factored RNNLM. However, RNNLM (Mikolov et al., 2011b) does not use it. In addition, the feature stream of the history also shows relatively strong intensity that confirms that the entire history is important.

4.2 Hidden neurons

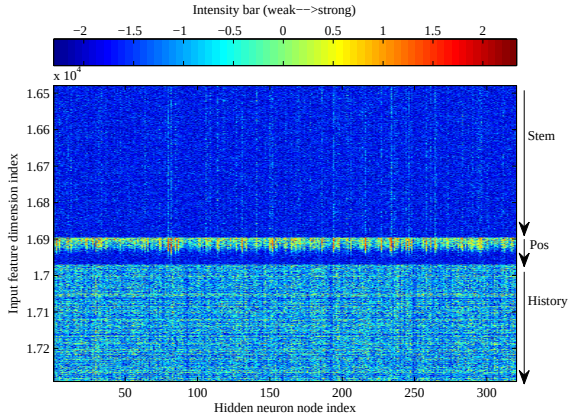
In this subsection, we evaluate the impacts from various numbers of hidden neurons in the hidden layer. Table 6 shows the results of the heldout set and the relative gains over the RNNLM. The experiments prove that factored RNNLMs consistently reduce perplexity. With increasing hidden neurons, both RNNLM and fRNNLM$_{wsp}$ enhance performance. The biggest improvement over RNNLM is 13.4%. The convergence column denotes the difference of the fRNNLM and RNNLM iterations, showing that factored RNNLM converges using fewer iterations. For example, RNNLM converges after 15 iterations, while fRNNLM$_{wsp}$ takes 12 iterations.

4.3 Convergence study

Figure 4 demonstrates the training progress of RNNLM and fRNNLM$_{wsp}$. In the same way, the number of hidden neurons in the hidden layer and the number of classes are set to 320 and 300, respectively. From this figure, we can observe that fRNNLM$_{wsp}$ significantly outperforms RNNLM at all iterations, especially at iterations 1-4 where the improvements exceed 20.0% and iterations



(a) Connection weight intensities of word, stem, POS and history



(b) Locally amplified view in rectangle marked position.

Figure 3: Neural connection weight intensity: between input feature and hidden neural nodes.

#Hidden neurons	RNNLM	fRNNLM$_{wsp}$	Gain	Convergence
60	163.71	147.00	10.2%	-3
120	152.33	133.07	12.6%	-2
240	147.74	128.75	12.8%	-2
320	146.94	127.09	13.4%	-1
480	143.18	126.70	11.5%	-2
640	142.22	126.04	11.4%	-1
1000	141.91	125.76	11.4%	0

Table 6: Impact from hidden layer on heldout data set.

5-10 where they exceed 15.0%, the final improvement reaches 13.5%. In other words, the relative

improvements decrease with increasing iterations.

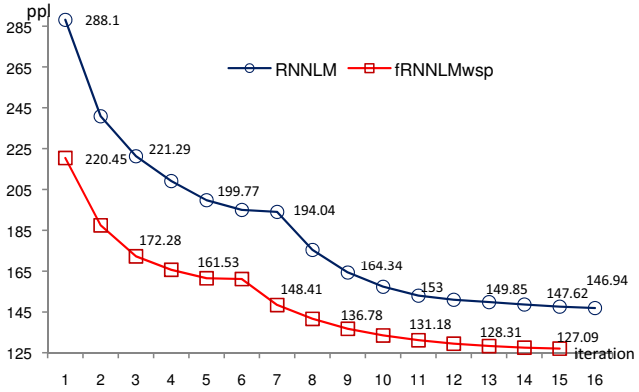


Figure 4: Convergence curve.

4.4 Running-time analysis

This subsection analyzes the time complexity of the two RNNLMs. Table 7 shows the training time of an iteration, the training time of all iterations, and the test time on a PC with 1006G of memory and 24 2.66Ghz CPUs with 144 cores. We observe the following: (1) No significant difference of elapsed time is found between RNNLM and fRNNLM$_{wsp}$ during an iteration of training and test stage. (2) RNNLM requires more time than fRNNLM$_{wsp}$ because it takes 18 iterations to reach a convergence and fRNNLM$_{wsp}$ uses 16 iterations. This experiment shows that although fRNNLM$_{wsp}$ has more free parameters and time complexities (shown in Table 3), it saves time owing to its fast convergence.

	An iteration during training	All iterations during training	During test
RNNLM	48.92m	880m19s	29.18s
fRNNLM$_{wsp}$	49.58m	792m39s	29.35s

Table 7: Elapsed time during training and test. m=minute, s=second.

4.5 Hybrid LM

In the experiments described above, RNNLMs are compared to a 4-gram back-off n-gram language model with modified Kneser-Ney smoothing trained using the SRILM toolkit (Stolcke, 2002). It is also useful to interpolate the recurrent neural network with a back-off n-gram language model to reduce the perplexity and the word error rate. In the following this interpolated model will be denoted by a hybrid language model. Table 8 compares the hybrid RNNLMs in terms of perplexity.

This table demonstrates that the hybrid factored RNNLM also outperforms the hybrid of RNNLM, as we expected. For example, the perplexity reductions of n-gram+fRNNLM over n-gramLM+RNNLM on the heldout and test sets are 8.8% and 9.4%, respectively, and n-gramLM+fRNNLM largely improves the 4-gramLM on the heldout and test sets by 28.9% and 29.6%.

	Heldout	Test
4-gramLM	156.26	156.41
RNNLM	146.94	145.63
fRNNLM$_{wsp}$	127.09	124.63
4-gramLM+RNNLM	121.89	121.62
4-gramLM+fRNNLM$_{wsp}$	111.20$_{+8.8\%}$	110.19$_{+9.4\%}$

Table 8: Perplexities of hybrid language models.

4.6 N-best re-scoring

To evaluate the factored RNNLM in the context of large vocabulary speech recognition, we use the data sets for the IWSLT-2011 large vocabulary continuous speech recognition shared task (Federico et al., 2011) to recognize TED talks published on the TED website[5]. 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.

For LM, the IWSLT-2011 campaign defines a closed set of publicly available English texts, including a small collection of TED transcriptions (in-domain corpus) and a large collection of news sentences (general-domain). 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 100K words are extracted from the preprocessed corpora, which, with the CMU.v0.7a pronunciation dictionary[6], are used as the LM vocabulary. Our vocabulary contains 157K entries with an OOV rate of 0.78% on the test2010 data set. For the re-scoring test, we use the IWSLT data sets of tests 2010 and 2011. Their statistics are shown in Table 9.

LM training data			
	#sentences		#words
in-domain	124K		2,063K
general-domain	115,101K		2,458,626K
Test sets			
data	#talks	#utterances	#words
test2010	11	1664	27.0K
test2011	8	818	12.4K

Table 9: Summary of IWSLT2011 data sets

The acoustic models are trained on 170h speech segmented from 788 TED talks that were published prior to 2011. We 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 (Povey et al., 2011). HMM consists of 6.7K states and 240K Gaussians that are discriminatively trained using the boosted Maximum Mutual Information criterion. SGMM con-

[5] http://www.ted.com/
[6] http://www.speech.cs.cmu.edu/cgi-bin/cmudict

sists 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 have 40 dimensions. For each frame, we extract 13 static MFCCs, splice 9 adjacent frames, and apply LDA to reduce its dimension with maximum likelihood linear transform. For the in-domain and general-domain corpora, modified Kneser-Ney smoothed 3- and 4-gram LMs are constructed using SRILM (Stolcke, 2002), and interpolated to form a baseline of 3- and 4-gram LMs by optimizing the perplexity of the development data set.

First, we employ a Kaldi speech recognizer (Povey et al., 2011) 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. Note that since it is very time consuming to train RNNLM and factored RNNLM on large data, we only use the in-domain corpus for training them, and the corpus is automatically tagged with parts-of-speech[7] before training fRNNLM$_{wp}$ and fRNNLM$_{wsp}$. The best re-scoring results measured by word error rate are demonstrated in Table 10. We also conduct utterance-level significance tests.

	test2010(%)	**test2011(%)**
4-gram LM	14.34	15.32
4-gram+RNNLM	14.12	15.22
4-gram+fRNNLM$_{wp}$	$13.57^{\dagger}_{0.55}$	$14.64^{\dagger}_{0.58}$
4-gram+fRNNLM$_{wsp}$	$13.65^{\dagger}_{0.47}$	$14.59^{\dagger}_{0.63}$

Table 10: n-best re-scoring performance in word-error-rate. Subscript numbers are the absolute improvements over RNNLM. † indicates significantly better results than RNNLM at the p = 0.01 level using a two-sided t-test.

The experimental results show that fRNNLM$_{wp}$ and fRNNLM$_{wsp}$ significantly improves upon 4-gram LM and RNNLM. For example, the absolute improvements of fRNNLM$_{wsp}$ over the 4-gram LM on the sets of tests 2010 and 2011 are 0.69 and 0.73 points, respectively. However, fRNNLM$_{wsp}$ doest not significantly outperforms fRNNLM$_{wp}$. Table 11 demonstrates the re-scoring results sampled from RNNLM and fRNNLM$_{wp}$. This table shows that the results of fRNNLM$_{wp}$ are more grammatically fluent. Fig. 5 illustrates the absolute improvements of fRNNLM$_{wp}$ over RNNLM for each talk in the sets of tests 2010 and 2011. Our approach improves most talks, expect talks 824 and 1183.

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, syntactic, and/or semantic information into RNNLM. Our approach, which is a hybrid of factored n-gram LM and RNNLM, addresses the problems in them. In experiments, we investigate the influences of four commonly used types of features on our factored RNNLM: word, stem, lemma and part-of-speech. We carry out many experiments to evaluate the factored RNNLM performance and analyze the influencing factors. Our experimental results prove that factored RNNLM consistently outperforms n-gram LM and RNNLM for all considered tasks.

[7] `http://www.nactem.ac.uk/tsujii/software.html`

model	**result**
Reference	or we'll be here all day with my childhood stories
RNNLM	* THE WORLD WE'RE all day with my childhood stories
fRNNLM$_{wp}$	or WILL be here all day with my childhood stories
Reference	but don't worry if you can't see it so well
RNNLM	* * TILLER if you can't see it so well
fRNNLM$_{wp}$	* don't worry if you can't see it so well
Reference	and so you're standing there and everything else is dark but there's this portal that you wanna jump in
RNNLM	and so you're * STAYING IN ANYTHING else * TO START there's this portal that you WANT TO jump in
fRNNLM$_{wp}$	and so you're * STAYING IN ANYTHING else is dark but there's this portal that you WANT TO jump in
Reference	AND by the way here are four doctors in your part of the united states who offer it and their phone numbers
RNNLM	* by the way here are four doctors in your part of the united states who * OFFERED and their phone numbers
fRNNLM$_{wp}$	* by the way here are four doctors in your part of the united states who offer it and their phone numbers

Table 11: Re-scoring results sampled from RNNLM and fRNNLM$_{wp}$. * denotes deletion errors, capitalized words denote substitution errors, and underlined words show their differences.

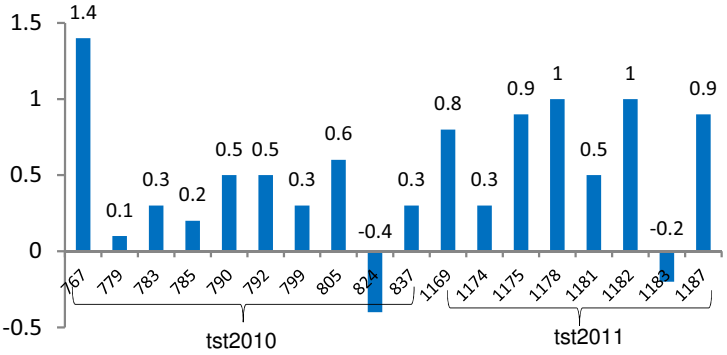


Figure 5: Absolute improvement on each talk.

Recently, syntactic parse trees are used in many advanced LMs (Chelba and Jelinek, 1998; Khudanpur and Wu, 2000; Xu et al., 2002; Collins et al., 2005; Rastrow et al., 2012). For future work, we intend to investigate topic information (Shi et al., 2012) and richer syntactic structure features into factored RNNLM, such as context-free rule productions, constituent/head features, and head-to-head dependencies that can be extracted using parser tools. Second, neural networks are notorious for being time consuming during training, future studies will also focus on speeding up the training of factored RNNLM using graphical processing units (Schwenk et al., 2012). Furthermore, factored RNNLMs need to be evaluated on other tasks like MT and with other languages such as Czech, Arabic, and Turkish.

References

Alexandrescu, A. and Kirchhoff, K. (2006). Factored neural language models. In *Proceedings of the NAACL 2006*, pages 1–4, New York, USA.

Arisoy, E., Sainath, T. N., Kingsbury, B., and Ramabhadran, B. (2012). Deep neural network language models. In *Proceedings of NAACL-HLT 2012 Workshop: Will We Ever Really Replace the N-gram Model? On the Future of Language Modeling for HLT*, pages 20–28, Montreal, Canada.

Bengio, Y., Ducharme, R., Vincent, P., and Jauvin, C. (2003). A neural probabilistic language model. *Journal of Machine Learning Research*, pages 1137–1155.

Boden, M. (2002). A guide to recurrent neural networks and backpropagation. In *The Dallas Project, Sics Technical Report*.

Brown, P. F., deSouza, P. V., Mercer, R. L., Pietra, V. J. D., and Lai, J. C. (1992). Class-based n-gram models of natural language. *Computational Linguistics*, 18(4):467–470.

Chelba, C. and Jelinek, F. (1998). Exploiting syntactic structure for language modeling. In *Proceedings of the 36th Annual Meeting of the Association for Computational Linguistics and 17th International Conference on Computational Linguistics*, pages 225–231, Montreal, Canada.

Chelba, C. and Jelinek, F. (2000). Structured language modeling. *Computer Speech and Language*, 13(4):283–332.

Chen, S. F. and Goodman, J. T. (1996). An empirical study of smoothing techniques for language modeling. In *Proceedings of ACL 1996*, pages 310–318, California, USA.

Collins, M., Roark, B., and Saraclar, M. (2005). Discriminative syntactic language modeling for speech recognition. In *Proceedings of ACL 2005*, pages 507–514, Michigan, USA.

Duh, K. and Kirchhoff, K. (2004). Automatic learning of language model structure. In *Proceedings of the International Conference on Computational Linguistics (COLING) 2004*, pages 148–154, Geneva, Switzerland.

Emami, A. and Jelinek, F. (2004). Exact training of a neural syntactic language model. In *Proceedings of ICASSP 2004*, pages 245–248, Montreal, Canada.

Federico, M., Bentivogli, L., Paul, M., and Stuker, S. (2011). Overview of the iwslt 2011 evaluation campaign. In *Proceedings of IWSLT 2011*, pages 11–27, San Francisco, USA.

Gildea, D. and Hofmann, T. (1999). Topic-based language models using em. In *Proceedings of Eurospeech 1999*, pages 2167–2170.

Goodman, J. (2001). Classes for fast maximum entropy training. In *Proceedings of ICASSP 2001*, Utah, USA.

Hsu, B.-J. P. and Glass, J. (2006). Style and topic language model adaptation using hmm-lda. In *Proceedings of EMNLP 2006*, pages 373–381, Sydney, Australia.

Khudanpur, S. and Wu, J. (2000). Maximum entropy techniques for exploiting syntactic, semantic and collocational dependencies in language modeling. *Computer Speech and Language*, pages 355–372.

Koehn, P. and Hoang, H. (2007). Factored translation models. In *Proceedings of EMNLP 2007*, pages 868–876, Prague, Czech Republic.

Kuo, H.-K. J., Arisoy, E., Emami, A., and Vozila, P. (2012). Large scale hierarchical neural network language models. In *Proceedings of Interspeech 2012*.

Kuo, H.-K. J., Mangu, L., Emami, A., Zitouni, I., and Lee, Y.-S. (2009). Syntactic features for arabic speech recognition. In *Proceedings of Automatic Speech Recognition & Understanding (ASRU) 2009*, pages 327–332, Merano, Italy.

Mikolov, T., Anoop, D., Stefan, K., Burget, L., and Cernocky, J. (2011a). Empirical evaluation and combination of advanced language modeling techniques. In *Proceedings of INTERSPEECH 2011*, pages 605–608, Florence, Italy.

Mikolov, T., Karafiat, M., Burget, L., Cernocky, J. H., and Khudanpur, S. (2010). Recurrent neural network based language model. In *Proceedings of INTERSPEECH 2010*, pages 1045–1048, Makuhari, Japan.

Mikolov, T., Kombrink, S., Burget, L., Cernocky, J., and Khudanpur, S. (2011b). Extensions of recurrent neural network language model. In *Proceedings of ICASSP 2011*, pages 5528–5531, Prague, Czech Republic.

Povey, D., Ghoshal, A., Boulianne, G., Burget, L., Glembek, O., Goel, N., Hannemann, M., Motlicek, P., Qian, Y., Schwarz, P., Silovsky, J., Stemmer, G., and Vesely, K. (2011). The kaldi speech recognition toolkit. In *IEEE 2011 Workshop on Automatic Speech Recognition and Understanding*. IEEE Signal Processing Society. IEEE Catalog No.: CFP11SRW-USB.

Rastrow, A., Dredze, M., and Khudanpur, S. (2012). Fast syntactic analysis for statistical language modeling via substructure sharing and uptraining. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics*, pages 175–183, Jeju, Korea.

Rosenfeld, R. (1996). A maximum entropy approach to adaptive statistical language modeling. *Computational Linguistics*, 10(3):187–228.

Schwenk, H. (2007). Continuous space language models. *Computer Speech and Language*, 21(3):492–518.

Schwenk, H., Rousseau, A., and Attik, M. (2012). Large, pruned or continuous space language models on a gpu for statistical machine translation. In *Proceedings of NAACL-HLT 2012 Workshop: Will We Ever Really Replace the N-gram Model? On the Future of Language Modeling for HLT*, pages 11–19, Montreal, Canada.

Seide, F., Li, G., Chen, X., and Yu, D. (2011). Feature engineering in context-dependentdeep neural networks for conversational speech transcription. In *Proceedings of ASRU 2011*, Hawaii, USA.

Shi, Y., Wiggers, P., and Catholijn, M. J. (2012). Towards recurrent neural networks language models with linguistic and contextual features. In *Proceedings of Interspeech 2012*.

Stolcke, A. (2002). Srilm - an extensible language modeling toolkit. In *Proceedings of INTERSPEECH 2002*, pages 901–904, Colorado, USA.

Xu, P., Chelba, C., and Jelinek, F. (2002). A study on richer syntactic dependencies for structured language modeling. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, pages 191–198, Philadelphia, USA.

Xu, P. and Jelinek, F. (2004). Random forests in language modeling. In *Proceedings of EMNLP 2004*, pages 325–332, Barcelona, Spain.