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 (NNLMs), 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 matrixes U and W. Unlike RNNLM, which pre-

[1] http://www.ted.com/

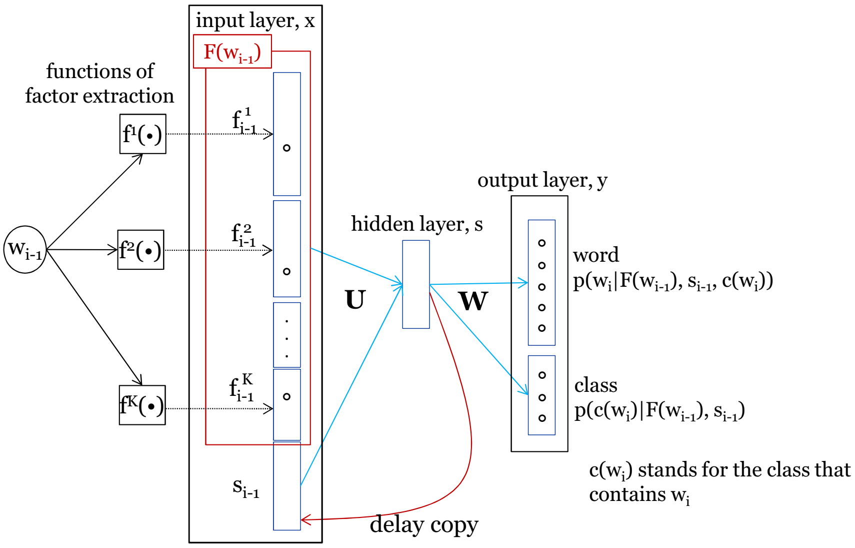

Figure 1: Architecture of factored recurrent NNLM.

Table 1: An example of factor sequences.

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:	NN	IN	JJ	NNS	CC	VBG	NNS

dicts 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^k_{i-1} $(k = 1, \ldots, K)$, f^k_{i-1} 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 1[2].

In the input layer, the extracted factors are encoded into the factor vectors using the 1-of-n coding. 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. (1). 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. (2).

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

$$x_i = [F(w_{i-1}),\ s_{i-1}] \tag{2}$$

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^1_{i-1} is a surface word factor vector and f^k_{i-1} $(k = 2, \ldots, K)$ are dropped, our proposed factored RNNLM goes back to the RNNLM.

The hidden layer employs a sigmoid activation function:

$$\begin{aligned} s_i^m &= f(\sum_j (x_i^j \times u_{mj})) \quad \forall m \in [1, H] \\ f(z) &= \frac{1}{1+e^{-z}} \end{aligned} \tag{3}$$

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 [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,

$$y_c^l = g(\sum_j (s_i^j \times w_{lj})) \quad \forall l \in [1, C] \tag{4}$$

where C is the number of predefined classes. The second computes the posterior probability distribution over the

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

words that belong to class $c(w_i)$, the one that contains predicted word w_i:

$$y_w^o = g(\sum_j (s_i^j \times w_{oj})) \quad \forall o \in [1, nc(w_i)] \tag{5}$$

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:

$$g(z_d) = \frac{e^{z_d}}{\sum_x e^{z_x}} \tag{6}$$

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

$$\begin{aligned} P(w_i|F(w_{i-1}), s_{i-1}) &= P(c(w_i)|F(w_{i-1}), s_{i-1}) \times \\ &\quad P(w_i|F(w_{i-1}), s_{i-1}, c(w_i)) \\ &= y_c^l|_{l=classid(c(w_i))} \times y_w^o|_{o=wordid(w_i)} \end{aligned} \tag{7}$$

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).

$$L = \frac{1}{2} \times \sum_{i=1}^{N} (t_i - p_i)^2 + \gamma \times (\sum_{lk} u_{lk}^2 + \sum_{tl} w_{tl}^2) \tag{8}$$

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. 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. (9) over all the training instances in several iterations. In Eq. (9), 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.

$$\psi^{new} = \psi^{previous} - \eta \times \frac{\partial L}{\partial \psi} \tag{9}$$

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 dictionary[3], 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 2: Summary of the IWSLT test data sets

LM training data			
corpora	#sentences		#words
in-domain	$142K$		$2{,}402K$
general-domain	$123.4M$		$2{,}726.6M$
Test sets			
data sets	#talks	#utterances	#words
dev2010	8	934	$17.5K$
test2010	11	1664	$27.0K$
test2011	8	818	$12.4K$
test2012	11	1124	$21.9K$

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

[3] http://www.speech.cs.cmu.edu/cgi-bin/cmudict

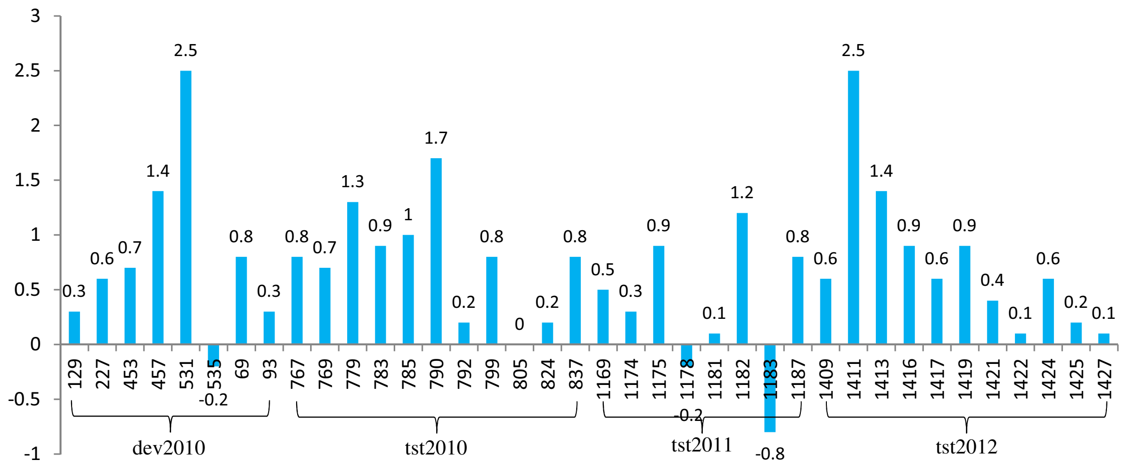

Figure 2: Absolute WER improvement on each talk.

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 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.

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[4]. In the fRNNLM, we investigate three commonly used types of factors: word, stem[5] and part-of-speech (POS). 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 480 and 300.

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$_{wp}$ 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. 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.

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

	dev2010	test2010	test2011	test2012
4-gram LM	16.5	13.8	12.3	13.9
RNNLM	$16.3_{0.2}$	$14.0_{-0.2}$	$12.2_{0.1}$	$13.9_{0.0}$
fRNNLM$_{wp}$	15.8	13.1	11.9	13.4
fRNNLM$_{wsp}$	$15.7_{0.8}$	$13.2_{0.6}$	$11.8_{0.5}$	$13.3_{0.6}$

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 $(|f^1|+...+|f^K|+H)\times H+H\times(C+|V|)$, respectively. That means, our factored RNNLM has $(|f^1|+...+|f^K|-|V|)\times H$ addi-

[4] http://www.nactem.ac.uk/tsujii/software.html
[5] http://tartarus.org/~martin/PorterStemmer/

Table 4: Re-scoring results sampled from RNNLM and fRNNLM$_{wp}$. * denotes deletion errors, capitalized words denote substitution errors, and underlined words show their differences. #e stands for the number of errors.

model	#e	**result**
Reference		or we'll be here all day with my childhood stories
RNNLM	5	THE WORLD we * * ARE all day with my childhood stories
fRNNLM$_{wp}$	1	or * be here all day with my childhood stories
Reference		she's painting here a mural of his horrible final weeks in the hospital
RNNLM	2	she's painting * HERO mural of his horrible final weeks in the hospital
fRNNLM$_{wp}$	0	she's painting here a mural of his horrible final weeks in the hospital
Reference		and so you are standing there and everything else is dark but there's this portal that you wanna jump in
RNNLM	7	and so you are * STAYING IN ANYTHING else is dark but there's THE SPORT ALL that you WANT TO jump in
fRNNLM$_{wp}$	5	and so you are * STAYING IN ANYTHING else is dark but there's this HORRIBLE that you WANT TO jump in
Reference		my worlds of words and numbers blur with color emotion and personality
RNNLM	4	my WORLD SO FLOATS and numbers BELAIR with color emotion and personality
fRNNLM$_{wp}$	1	my worlds of words and numbers BELAIR with color emotion and personality

tional free parameters. If the factored RNNLM only employs word factor (f^1) and POS factors (f^2), then, it has 39 $\times$ H additional free parameters. In experiments, H is usually set to $300 - 1000$, $|V|$, the word vocabulary's size, is usually set to several hundreds of thousands.

The time complexities in the RNNLM and factored RNNLM are $(1 + H) \times H \times \tau + H \times |V|$ and $(K + H) \times H \times \tau + H \times |V|$, respectively. That means, the factored RNNLM has $(K - 1) \times H \times \tau$ additional computational complexity. τ is usually set to 4 or 5. This means that $H \times |V| \gg (K - 1) \times H \times \tau$, and the increased complexity can be neglected. On the contrary, our factored RNNLM converges faster and reduces training time due to the additional free parameters. Table 5 shows the training time of an iteration, the training time of all iterations, and the test time on a PC with 1006GB of memory and 24 2660MHz processors. From this table, 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) For the time of all iterations, RNNLM takes more time than fRNNLM$_{wsp}$ because it takes 16 iterations to reach a convergence and fRNNLM$_{wsp}$ uses 13 iterations. This experiment shows that although fRNNLM has more free parameters and time complexities, it saves time owing to its fast convergence.

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.

	#1	#2	time on testing tst2010
RNNLM	120m	1923m	35.7s
fRNNLM$_{wsp}$	141m	1843m	43.4s

Figure 3 demonstrates the convergence progress of RNNLM, fRNNLM$_{wp}$ 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.

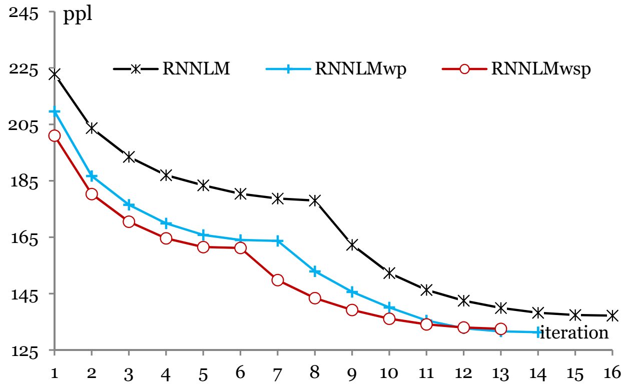

Figure 3: Convergence curve on the dev2010 set.

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

# of words in training data	dev2010		test2010		test2011		test2012	
	RNNLM	fRNNLM	RNNLM	fRNNLM	RNNLM	fRNNLM	RNNLM	fRNNLM
2.4M	16.3	15.7	14.0	13.2	12.2	11.8	13.9	13.3
9.0M	15.5	15.4	13.0	13.1	11.4	11.3	13.1	13.0
19.4M	15.4	15.3	12.9	12.9	11.3	11.2	13.0	13.0
30M	15.2	15.0	12.9	12.7	11.1	11.2	12.9	12.8

Table 6: Impact of training corpus size.

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 NNLMs 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 (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 [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

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

[2] 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.

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

[4] Bilmes, J. A. and Kirchhoff, K. (2003). Factored language models and generalized parallel backoff. In *Proceedings of NAACL 2003*, pages 4–6, USA.

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

[6] 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.

[7] Duh, K. and Kirchhoff, K. (2004). Automatic learning of language model structure. In *Proceedings of COLING 2004*, pages 148–154, Geneva, Switzerland.

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

[9] 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.

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

[11] 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.

[12] 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.

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

[14] 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.

[15] 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.

[16] 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.

[17] Moore, C., Lewis, W. (2010). Intelligent Selection of Language Model Training Data. In *Proceedings of the ACL 2010 Conference Short Papers*, pages 220C224, Hawaii, USA.

[18] 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.

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

[20] 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 workshop on the Future of Language Modeling*, pages 11–19, Canada.

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

[22] Xu, P., Chelba, C., and Jelinek, F. (2002). A study on richer syntactic dependencies for structured language modeling. In *Proceedings of ACL 2002*, pages 191–198, Philadelphia, USA.

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