Towards Recurrent Neural Networks Language Modeling with Succeeding Words

Conference Paper · January 2013

CITATIONS
0

READS
33

4 authors, including:

Yangyang Shi
Microsoft, Sunnyvale, Unites States
26 PUBLICATIONS **333** CITATIONS
SEE PROFILE

Pascal Wiggers
Amsterdam University of Applied Sciences/Centre for Applied Research on Education
17 PUBLICATIONS **129** CITATIONS
SEE PROFILE

Catholijn M. Jonker
Delft University of Technology
543 PUBLICATIONS **6,371** CITATIONS
SEE PROFILE

Towards Recurrent Neural Networks Language Modeling with Succeeding Words

Yangyang Shi[1], *Martha, A Larson*[1], *Pascal Wiggers*[2], *Catholijn M. Jonker*[1]

[1]Delft University of Technology, Intelligent System Department, Mekelweg 4, 2628 CD Delft, NL
[2]CREATE-IT Applied Research, Amsterdam University of Applied Sciences (HvA)
Duivendrechtsekade 36-38, 1096 AH Amsterdam, The Netherlands

Abstract

A statistical language model suffers from long distance dependency insufficiency. Most conventional language models only make use of the preceding word information. The recurrent Neural networks help language models to capture long history information. However, they still don't make use of the succeeding words. In order to fill this gap, in the language model training, this paper proposes to use a forward and backward strategy and maximum entropy method to integrate recurrent neural network language models (RNNLM) with the succeeding words. In speech decoding, a two pass alternative rescoring strategy is proposed to further optimize the combination of the RNNLM with other models. The application of the RNNLM with succeeding words in speech recognition is based on the fact that the succeeding words are available to the conventional RNNLM, as in speech recognition the RNNLM in most cases can only use N-best rescoring strategy. The experiments show that the forward and backward RNNLM, the maximum entropy extension recurrent neural networks (RNNME) with succeeding words, and their combination improve the RNNLM in terms of both the perplexity and the word error rate(WER). The two pass alternative rescoring also helps to reduce the WER. The RNNME using the succeeding three words achieves the best performance, when it is combined with a forward and backward strategy using word level geometric interpolation. It reduces the perplexity of RNNLM by almost 30% and reduce the WER from 16.83% to 14.62%. It gets even better WER 14.44%, when the two pass alternative rescoring is applied.

Index Terms: Neural networks, N-best rescoring, language models, recurrent neural networks language models

1. Introduction

A statistical language model is a critical component in automatic speech recognition, machine translation and optical character recognition. In order to capture the regularities of a language, it assigns a probability over each word sequence. Most statistical language models decompose the probability of a word sequence into a product of conditional probabilities of one word given the preceding words history. The conventional n-gram language models use the previous $n-1$ words as history. The state of the art recurrent neural networks language models (RNNLM) [1, 2] theoretically can use word information from arbitrarily long distances.

However, all these language models are based on the assumption that current word w_i only depends on the previous words. In fact, it is apparent from looking at a language that this assumption is often not true. In fact, there are significant dependencies within each sentence. For example, in the sentence "the dog barks", "dog" crucially depends on "barks". The conditional probability of 'dog' given 'the' is much smaller than the conditional probability of 'dog' given the two words "the" and "barks". The succeeding word "barks" helps more in predicting the word "dog" than the preceding word "the".

In order to extensively exploit the global dependencies within a sentence, this study investigates the methods of integrating the succeeding words of a sentence into RNNLM. Recent studies [1, 2, 3] demonstrated that RNNLM outperforms the conventional n-gram language models. However, because of the highly computational complexity, in its most applications in automatic speech recognition, the RNNLM can only be used for N-best rescoring. In other words, the RNNLM in fact is used to deal with sentences. The availability of succeeding words in speech decoding process makes it possible and necessary to integrate the RNNLM with these information.

In this paper we investigate an extension of the RNNLM that takes into account the succeeding words. We look at two approaches, namely the implicit method and the explicit method. In the implicit method, the current word w_t and the succeeding words w_{t+k} ($k \geq 2$) are used as input to the RNNLM to predict the next word w_{t+1}. However, the recurrent hidden layer in the RNNLM cannot be used to store the information from the succeeding words. Because the succeeding words information decreases as the RNNLM moves from left to right. Alternatively, a maximum entropy model is applied to combine the succeeding words by directly connecting the input to the output. In the explicit method, we interpolate the conventional forward RNNLM with a backward RNNLM on the sentence level. In contrast to the forward RNNLM, in which the probability of a word is conditioned on the preceding words, the backwards RNNLM predicts one word based on the succeeding words in the given sentence. In addition to these modification on the RNNLM, this paper also propose an two pass alternative rescoring strategy to make use of these modified models in speech decoding.

The rest of the paper is organized as follows. The next section describes some related works. Section 3 introduce the RNNLM, forward and backward RNNLM, RNNME with succeeding words and the two pass alternative rescoring. Section 4 shows the experiment results of these models in term of perplexity and word error rate. Based on the experiment results, Section 5 presents the conclusions.

2. Related works

In 2003, Bengio *et al.* [4] proposed a feed-forward neural networks language model, in which they projected the vocabulary to a continuous feature vector space and directly applied the previous $n-1$ word feature vectors as the input. The recurrent

neural network language models [1, 2] avoid this explicit modeling of the word history, by copying the previous hidden layer into the current input layer. This equips the language model with a short memory to store previous long term history information. At the same time, the RNNLM maps the discrete vocabulary to a continuous space, which helps a language model to relieve the data sparseness influence. In this paper, based on the fact that whole sentence information is available when RNNLM is applied for rescoring in speech recognition, we propose two ways of modeling the succeeding words in a sentence to improve the performance of the RNNLM.

One way to use the succeeding words in language models is to apply the forward and backward modeling strategy. This strategy has been investigated before on n-gram language models. Duchateau *et al.* [5] showed that both the forward and backward smoothed n-gram language models got almost the same perplexity. However, they did not give the word error rate (WER) performance of backwards n-gram language models in an automatic speech recognition task. They illustrated the performance of backwards n-gram confidence measure using the metric described by [6]. When a confidence measure is combined with the score from a backward language model, it got significantly increase in terms of cross entropy. The practical usage of the backward n-gram language models was achieved by *et al.* [7], in which the scores from the forward decoder and the backward decoder are linearly interpolated with equal weights. Their bidirectional decoding got dominant improvement over 272 different language pairs from 17 languages. The improvement of statistical machine translation by the backward n-gram is further confirmed by Deyi Xiong *et al.* [8], in which they proposed to use backward n-gram language model and mutual information trigger models to enhance language models in phrase-based statistical machine translation. Different from previous work, they applied the backward n-gram information into online decoding rather than processing the backward information after decoding.

The other way to integrate a language model with succeeding words is to apply maximum entropy language models. These models have been applied to model the whole sentence in speech recognition [9], in which each sentence is represented as a "bag of features". In fact, maximum entropy language models can be viewed as neural networks with no hidden layer, which use a weight matrix directly connecting the input to output layer [10]. Mikolov *et al.* [10] proposed to apply the maximum entropy model on the history n-gram features to update output from the RNNLM. However, they do not take advantage of the succeeding words information. In this article, we extend the usage of the maximum entropy language models in the RNNLM with succeeding words.

To incorporate these language models with succeeding information into a speech recognizer, we apply the N-best rescoring. As is discussed in [11], it becomes computationally challenging when incorporating the language models which captures long distance dependencies into a speech recognizer. The two pass N-best rescoring currently is the mostly applied way to use RNNLM in speech recognizer. The standard rescoring approach was proposed in [12], in which one system computed the N-best hypotheses, a second system rescored these hypotheses and all the scores were combined to improve the overall performance. [13] modified the standard approach by optimizing the combination weight with regards to average WER. This paper proposes an two pass alternative rescoring for the N-best list. It helps to avoid the combination weights for different models falling into a local optima.

3. RNNLM with succeeding words information

Based on state of the art language model–RNNLM, we propose a forward and backward RNNLM, an extension of RNNME with succeeding words and an additional two pass alternative rescoring in speech decoding to integrate the succeeding words into sentence hypotheses scoring. These models will be explained in turn in this section.

3.1. Recurrent Neural Networks Language Models

The RNNLM has an input layer x, a hidden layer h and an output layer y. Each time t, the input vector $x(t)$ consists of the current word vector $w(t)$ as well as the one copy $h(t-1)$ from the previous hidden neurons.

The activation function on hidden layer is a sigmoid function. On the output layer, the activation function is softmax function. The weight matrix between layers is estimated by back-propagation-through-time [10, 14].

3.2. Forward and Backward RNNLM

Given a word sequence $W = w_1, w_2, ..., w_n$, the RNNLM assigns the probability to W as follows.

$$p(W) = \prod_i p(w_i|h_i), \tag{1}$$

where h_i is the words history of word w_i. As it is shown in the equation 1, the RNNLM predicts a word in a forward way. We notate the RNNLM as forward RNNLM.

Different from the forward RNNLM, the backward RNNLM assigns the probability to W in the reverse direction.

$$p(W) = \prod_i p(w_i|s_i), \tag{2}$$

where s_i is the set of succeeding words of w_i in the word sequence W. The training of the backward RNNLM use the same algorithm as the forward RNNLM, but the order of the words in the sentence is reversed during training.

There are many ways to combine the forward and backward RNNLM. This paper studies the following heuristic ways:

1. Word level linear interpolation (WI):

$$p(W) = \prod_i \{\lambda_b p(w_i|s_i) + \lambda_f p(w_i|h_i)\}. \tag{3}$$

   where λ_b is the weight for backward model, λ_f for forward model and $\lambda_b + \lambda_f = 1$.

2. Sentence level linear interpolateion (SI):

$$p(W) = \lambda_b \prod_i p(w_i|s_i) + \lambda_f \prod_i p(w_i|h_i). \tag{4}$$

3. Word level geometric interpolation (WG):

$$p(W) = \prod_i (p(w_i|s_i)^{\alpha_b} p(w_i|h_i)^{\alpha_f})^{\frac{1}{\alpha_b+\alpha_f}}, \tag{5}$$

   where $\alpha_b \geq 0$ and $\alpha_f \geq 0$.

4. Sentence level Maximization (SM):

$$p(W) = \max\{\prod_i p(w_i|s_i), \prod_i p(w_i|h_i)\}, \tag{6}$$

In method (1), we presume that each word is determined bidirectionally. The sentence probability is the product of the word probabilities which are the interpolations the conditional probabilities given the previous and suceeding information. The method (2) treats the sentence as the basic unit which is scored by the forward RNNLM as well as the backward RNNLM. The method (3) and (4) may not generate the proper probabilities. However, in perspective of N-best rescoring, way (3) simply is the linear interpolation of language model logrithm scores. The (4) actually is an extreme case of the (3) with one weight is zero in the interpolation.

3.3. Maximum Entropy Model Extension in RNNLM

In a maximum entropy model, the conditional probability of the current word given the history features works as follows:

$$p(w|h) = \frac{\exp \sum_{i=1}^{N} \lambda_i f_i(h, w)}{\sum_w \exp \sum_{i=1}^{N} \lambda_i f_i(h, w)}, \quad (7)$$

where f_j is one feature, λ_i is the weight for feature i and h is the history of features. In fact, it can be viewed as a neural network language model, with the difference that the features are continous valued and automatically learned from the history. Tomas [10] implemented the RNNME as the extension of RNNLM with maximum entropy model taking adavantage of the preceding n-gram features.

In this paper, we extend the RNNME with the succeeding words. It has the following form:

$$p(w|c) = \frac{\exp(\sum_i \lambda_i f_i(h, w) + \sum_j \lambda_j g_j(s, w))}{\sum_w \exp(\sum_i \lambda_i f_i(h, w) + \sum_j \lambda_j g_j(s, w))}, \quad (8)$$

where c contains the preceding and succeeding information. s are the succeeding word features.

3.4. Additional Two Pass Alternative Rescoring

In additional to extending the RNNLM with succeeding words, we propose a two pass alternative rescoring to combine the conventional RNNLM with the RNNLM with succeeding words in speech recognition. In the N-best rescoring paradigm [12], N-best hypotheses are generated from one model, which are rescored by other models. The scores from these models are combined to improve the overall performance. In general, the combination weight is learned from the held out data, which is further applied to the evaluation data. In fact, it is an unconstrained nonlinear global optimization to find the optimal combination weight for each model. The resulting optimum, which is found on held out data, is typically a local optimum.

In order to avoid this situation, we propose a simple two pass alternative rescoring strategy to extend the standard strategy. In fact, it uses a filter method to solve an optimization problem. As it is discussed in [**?**], a global optimization problem is better to be treated as a filter problem, when the objective function can not be exactly evaluated.

The two pass alternative rescoring strategy works as follows: For two different language models m_1 and m_2, N-best hypotheses N and ratio $\alpha \in (0, 1)$.

1. The language model m_1 is used to rescore the N-best hypotheses N, of which a α portion of hypotheses are selected. $N := \alpha * N$
2. The hypotheses selected in previous step are rescored by m_2. The size of the N-best list is reduced again. $N := \alpha * N$. Continue to step (1) until the best hypothesis is obtained.

In fact, at each iteration, a language model provides an optimized search space for the next language model. One attractive point is that this strategy can be directly applied to the evaluation data, as it doesn't require predetermined weights which need to be learned from the held out data.

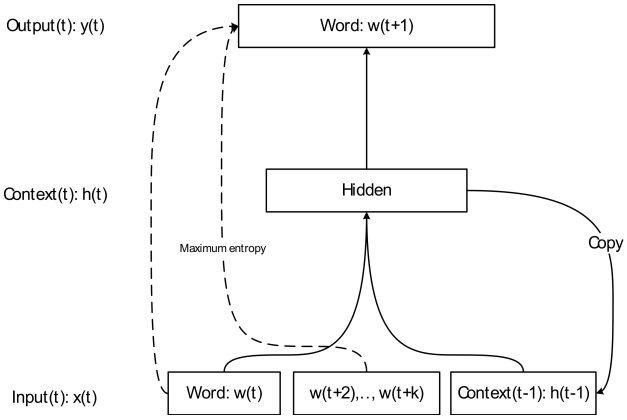

Figure 1: ENNLM

4. Experiments

The experiments are based on the Wall Street Journal (DARPA WSJ'92 and WSJ'93) data sets, which is the same data set used by Mikolov in [1]. The training corpus contains 37M words from the NYT section of the English Gigaword set with a vocabulary of 195K words. An independent set of 230K words is selected for testing (in terms of perplexity). A part of the N-best list rescoring data is used as development data for tuning the the weight for interpolation, language model score, acoustic model score and word insertion penalty. The rest of them are used for evaluation.

Table 1 shows the perplexity and WER results of the baseline models and the proposed models. The column 'PPL' denotes the perplexity results. 'WER' represents the WER for individual models. 'WER2' is the word error rate after applying the additional two pass alternative rescoring strategy, in which the two different models are the conventional RNNLM and the given model in that row.

There are four parts in the table. The top part is the results of the Kneser-Ney 5 order n-gram language and the forward (conventional) RNNLM. The second part gives the results from the backward RNNLM as well as four different combinations of the backward RNNLM with the forward RNNLM. The results from the RNNME and an extension of RNNME with succeeding words are given in the third part of the table. The bottom part of table shows the performance of the forward and backward strategy in the RNNME with succeeding words. All the models listed in the table contain 100 classes and 100 neurons in hidden layer. The weights of these models are trained by 4 times backpropagation through time (BPTT) with block size 10. In RNNME, all the models use 1 billion parameters to represent preceding n-gram and succeeding words.

In the second part of table, the results show that all the forward and backward RNNLM except using word level linear interpolation improve the RNNLM in terms of perplexity and

WER. The individual backward RNNLM gets similar perplexity as RNNLM, but it reduces the WER by 0.28% absolutely. Among the four forward and backward combination strategies, the word level geometric interpolation performs best. Individually, it achieves 1.14% absolute reduction in WER. The additional two pass alternative rescoring further enlarge the reduction to 1.39%. However, the table also shows that the word level linear interpolation doesn't improve RNNLM at all. It results in significant increase in WER, even it reduce the perplexity of RNNLM by almost 50%. This contradiction happens because the word level linear interpolation flattens the vocabulary distribution. Normally the perplexity reduction reflects the vocabulary distribution becoming sharp in the testing data area. However, considering about the Jensen's inequality, the perplexity reduction also happens, when the whole vocabulary distribution gets flat. Take an example of 2 words vocabulary. If the two words get the equal probability 0.5, the overall probability $0.5 * 0.5$ is much bigger than the case in which one word with probability 0.2, the other with 0.8. As we notice in the testing and N-best rescoring, the word level linear interpolation generally increases the probabilities of word squences and flats the whole vocabulary distribution, which obviously makes the vocabulary distribution less discriminative.

The third part of Table 1 gives the results from the RNNME and RNNME with succeeding words. The 'RNNME-P3' shows the conventional RNNME with preceding three words. The 'RNNME-S3' denotes the RNNME with succeeding three words. The 'RNNME-P3S3' denotes the RNNME with preceding three words and succeeding three words. The results indicate that using the maximum entrophy model to integrate succeeding words reduces the WER of RNNLM. The RNNME with preceding and succeeding words performs best, which achieves a reduction of almost 30% in perplexity and 1.86% absolute reduction in WER using the additional two pass alternative rescoring.

In this experiment, we use an n-gram for preceding words, but only use words instead of n-gram for succeeding words by taking into account of the memory problem. Because combination succeeding n-gram into RNNME would require even larger hash vector, which needs more memory and computations than the conventional RNNME. In this experiment, the hash vector size for RNNME is set to 1 billion, which takes almost 8GB memory in the training. In fact, it is still far less than enough. Both increasing features and reducing hash vector size will aggravate the hash value collision problem.

The bottom of table shows the results of combination the forward and backward strategy with the maximum entrophy model in the RNNLM. The RNNME with preceding and succeeding three words using word level geometric interpolation achieves the best results. It reduces the perplexity of RNNLM by almost 30% and reduce the WER from 16.83% to 14.62%.

To sum up, three approaches to combine the succeeding words with RNNLM all get improvement. In language model training, in terms of perplexity and WER, the maximum entropy modeling performs better than a forward and backward strategy. However, the forward and backward strategy is more effective than maximum entropy modeling. In forward and backward strategy, both forward RNNLM and backward RNNLM can be trained in parallel, as they do not share parameter with each other. In RNNME, both the succeeding words and preceding words are treated as features of the whole sentence, which share the same huge size of hash vector. As we see in the table, in the decoding, the additional two pass alternative rescoring also reduces the WER of all the models.

Table 1: Perplexity and word error rate results on WSJ with 100 hidden neurons

model	PPL	WER(%) rescore	WER(%) alternative rescore
KN5	174.5	17.30	-
RNNLM	163.4	16.83	-
RNNLM-B	158.2	16.55	16.03
RNNLM-WI	87.7	17.68	16.86
RNNLM-SI	118.2	16.38	16.18
RNNLM-WG	**159.2**	**15.69**	**15.44**
RNNLM-SM	145.2	16.56	16.27
RNNME-P3	115.3	15.23	14.98
RNNME-S1	164.9	16.60	16.21
RNNME-S2	165.2	16.55	16.18
RNNME-S3	165.4	16.47	16.18
RNNME-P3S3	**117.0**	**15.07**	**14.97**
RNNME-B-P3S3	115.0	15.53	15.30
RNNME-WI-P3S3	56.8	16.74	16.35
RNNME-SI-P3S3	87.2	15.24	15.04
RNNME-WG-P3S3	**116.2**	**14.62**	**14.44**
RNNME-SM-P3S3	105.5	15.48	15.24

5. Conclusion

Three different approaches to integrate recurrent neural network language models – that make use of both preceding an succeeding words to predict a word – in an automatic speech recognizer were proposed in this paper. These approaches were a forward and backward RNNLM and a RNNME with succeeding words in language model training, and the additional two pass alternative rescoring in speech decoding. In this paper, we implemented the word level linear interpolation, word level geometric interpolation, sentence level linear interpolation and sentence level maximum selection to combine the forward RNNLM with backward RNNLM. The word level geometric interpolation achieved the best results. Individually, it reduced the WER by 1.14%. It obtained extra 0.25% absolute reduction, when it was combined with the RNNLM by the additional two pass alternative rescoring. The integration of the RNNLM with succeeding words using maximum entropy achieved larger improvement than the forward and backward RNNLM. The RNNME with the succeeding three words reduced the perplexity of RNNLM by almost 30% relative and the WER by 11%. The combination of the three approaches obtained the highest performance over RNNLM. It reduced the WER from 16.83% to 14.44%.

6. Acknowledgement

Thank you to Tomas Mikolov for making the RNNLM Toolkit publicly available and for helpful discussion.

7. References

[1] T. Mikolov, M. Karafiát, L. Burget, J. Cernocký, and S. Khudanpur, "Recurrent neural network based language model," in *INTERSPEECH*, 2010, pp. 1045–1048.

[2] T. Mikolov, S. Kombrink, L. Burget, J. Cernocky, and S. Khudanpur, "Extensions of recurrent neural network language model," in *Acoustics, Speech and Signal Processing (ICASSP), 2011 IEEE International Conference on*, may 2011, pp. 5528 –5531.

[3] T. Mikolov, A. Deoras, S. Kombrink, L. Burget, and J. ernock, "Empirical evaluation and combination of advanced language modeling techniques," in *Proceedings of Interspeech 2011*. International Speech Communication Association, 2011, pp. 605–608.

[4] Y. Bengio, R. Ducharme, P. Vincent, and C. Janvin, "A neural probabilistic language model," *J. Mach. Learn. Res.*, vol. 3, pp. 1137–1155, Mar. 2003.

[5] J. Duchateau, K. Demuynck, and P. Wambacq, "Confidence scoring based on backward language models." in *ICASSP*. IEEE, 2002, pp. 221–224.

[6] M.-H. Siu and H. Gish, "Evaluation of word confidence for speech recognition systems," *Computer Speech & Language*, vol. 13, no. 4, pp. 299–319, 1999.

[7] A. Finch and E. Sumita, "Bidirectional phrase-based statistical machine translation," in *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing: Volume 3 - Volume 3*, ser. EMNLP '09, 2009, pp. 1124–1132.

[8] D. Xiong, M. Zhang, and H. Li, "Enhancing language models in statistical machine translation with backward n-grams and mutual information triggers," in *ACL*, 2011, pp. 1288–1297.

[9] R. Rosenfeld, S. F. Chen, and X. Zhu, "Whole-sentence exponential language models: A vehicle for linguistic-statistical integration," *Computers, Speech and Language*, vol. 15, p. 2001, 2001.

[10] T. Mikolov, A. Deoras, D. Povey, L. Burget, and J. Cernocký, "Strategies for training large scale neural network language models," in *ASRU*, 2011, pp. 196–201.

[11] T. M. Anoop Deoras and K. Church, "A fast re-scoring strategy to capture long-distance dependencies," in *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing*, 2011.

[12] M. Ostendorf, A. Kannan, S. Austin, O. Kimball, R. Schwartz, and J. R. Rohlicek, "Integration of diverse recognition methodologies through reevaluation of n-best sentence hypotheses," in *Proceedings of the workshop on Speech and Natural Language*, ser. HLT '91. Stroudsburg, PA, USA: Association for Computational Linguistics, 1991, pp. 83–87.

[13] A. Stolcke, Y. Knig, and M. Weintraub, "Explicit word error minimization in n-best list rescoring," in *Proc. EUROSPEECH*, 1997, pp. 163–166.

[14] P. J. Werbos, "Backpropagation through time: what it does and how to do it," *Proceedings of the IEEE*, vol. 78, no. 10, pp. 1550–1560, 1990.