Multi-view LSTM Language Model with Word-synchronized Auxiliary Feature for LVCSR

Yue Wu, Tianxing He, Zhehuai Chen, Yanmin Qian, and Kai Yu

Key Laboratory of Shanghai Education Commission for Intelligent Interaction and Cognitive Engineering
SpeechLab, Department of Computer Science and Engineering
Brain Science and Technology Research Center
Shanghai Jiao Tong University, Shanghai, China
{yuewu619, cloudygoose, chenzhehuai, yanminqian, kai.yu}@sjtu.edu.cn

Abstract. Recently long short-term memory language model (LSTM LM) has received tremendous interests from both language and speech communities, due to its superiorty on modelling long-term dependency. Moreover, integrating auxiliary information, such as context feature, into the LSTM LM has shown improved performance in perplexity (PPL). However, improper feed of auxiliary information won't give consistent gain on word error rate (WER) in a large vocabulary continuous speech recognition (LVCSR) task. To solve this problem, a multi-view LSTM LM architecture combining a tagging model is proposed in this paper. Firstly an on-line unidirectional LSTM-RNN is built as a tagging model, which can generate word-synchronized auxiliary feature. Then the auxiliary feature from the tagging model is combined with the word sequence to train a multi-view unidirectional LSTM LM. Different training modes for the tagging model and language model are explored and compared. The new architecture is evaluated on PTB, Fisher English and SMS Chinese data sets, and the results show that not only LM PPL promotion is observed, but also the improvements can be well transferred to WER reduction in ASR-rescore task.

Keywords: LSTM language model, speech recognition, multi-view, auxiliary feature, tagging model

1 Introduction

A language model judges the fluency and reasonability of a sentence. It is widely used in natural language processing, machine translation, automatic speech recognition (ASR) and other tasks. Statistical language models were developed in 1987, aiming to model the probability of the next word in a sentence given the preceding words [1].

The smoothed n-gram model has been the dominating model in the field for a long time. Nevertheless, it has been shown that recurrent neural network language models (RNN LM) can significantly outperform traditional n-gram models [2][3], because of its ability to memorize previous history. After that,

the Long Short-Term Memory (LSTM) [4] RNN was proposed, being able to connect long time lags between the relevant input and target output, thereby incorporating long-range contexts. Nowadays, this structure has been widely applied to language modeling, achieving promising results [5].

In order to further promote the capability of a LSTM based language model, many works have been focused on adding additional word and sentence level information in the language model [6]. The additional information includes part of speech (POS), named entity recognition (NER), chunking [7], environment of a sentence, grammatical parser, etc. This auxiliary information is combined into traditional multi-view models as well as joint frameworks [8]. Although works have been proven to be effective regarding their perplexity performance, they fail to show corresponding WER and SER reduction in LVCSR task.

We argue that the key reason behind this performance inconsistency is that standard auxiliary features inference mechanisms (such as maximum entropy and BLSTM tagging model) utilize not only history words but also future word sequence, which should not be fed into a LM for next word prediction. It's like cheating, future information is given when predicting future words, which will easily decrease PPL, but won't help reduce WER and SER in n-best rescoring [6][8].

To address this problem, we propose to use word-synchronized features, by which we mean that the feature extraction process will only use history words and information. More details about are given below.

In our work, an unidirectional LSTM tagging model which differs from those traditional feature extractor, is utilized to produce auxiliary feature. Next, the tagging model is connected with a LSTM LM, forming a multi-view structure. For training our model, five methods are experimented to determine the most suitable training process. Finally, we compare our new multi-view LSTM LM with not only basic LSTM LM baseline but also models of related works on both PPL and ASR-rescoring.

The rest of the paper is organized as follows: Section 2 gives the detail of the related work of language model. We show how to connect the tagging model with the language model in Section 3. Section 4 shows the experimental setup and results. Finally, the conclusions can be found in Section 5.

2 Related Work

As LSTM RNN model has been proven to be a promising structure, many researches have been focusing on applying it in tasks like LM, tagging etc. In order to achieve further improvement, more useful and complex structures based on LSTM RNN model have been explored. Figure 1 shows three extensions of the basic LSTM language model, and all these models are reproduced and compared to proposed model. Concrete results and analysis are shown in Section 4.

2.1 Multi-view model

Considering that some extra linguistic features might contribute to language modeling, word and sentence level features were introduced in LM [6]. This

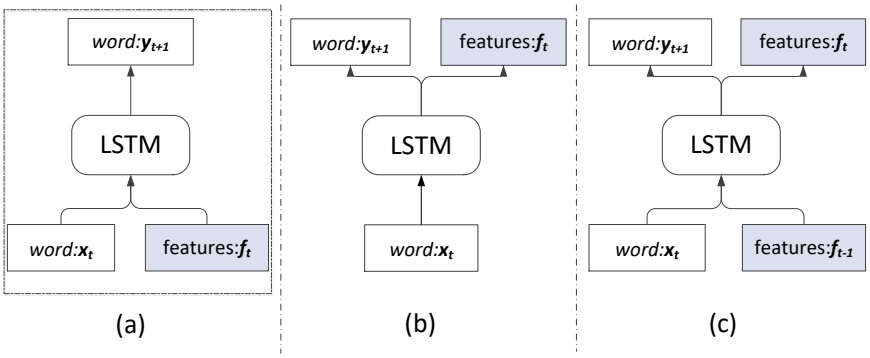

Fig. 1. *(a)Multi-view LSTM language model. (b)Multi-task LSTM model. (c)Multi-view combine with multi-task LSTM joint model.*

x

model have multi inputs, which contains different views of information. So it is called multi-view model (see Figure 1(a)).

As argued in Section 1, research on this kind of model shows improvements on perplexity and word prediction accuracy (WPA), but integrating this model with ASR did not lead to commensurate improvements [8]. That is to say, the straightforward combination of words and features as the inputs of a language model do not contribute to speech recognition.

Our proposed model is based on the multi-view structure, but is specially tailored for ASR task by using word-synchronized auxiliary feature.

2.2 Multi-task model

As shown in Figure 1(b), in a multi-task structure, a language model was designed to train with other tasks jointly [9], they share the same inputs and hidden layers. However most researches on multi-task structure show that usually performance gain is achieved in the cooperating task, instead of LM task itself.

2.3 Multi-task and multi-view joint model

Other works combined the multi-task structure with multi-view structure, which is shown in Figure 1(c). Not only multiple tasks were trained together, but also the inputs of this model were multi-view. LM was jointed with other spoken language understanding (SLU) or natural language process (NLP) task, some models of improved version are researched. Better than multi-task models, these works show slight improvement in PPL and ASR-rescoring, but more promotion is gained in the cooperating task [10].

3 Method

Our model is composed of two parts: The first one is a tagging model which produces word-synchronized auxiliary features, the other is a multi-view language model with auxiliary features propagated from the tagging model. For both of them, single layer unidirectional LSTMs are used. We connect the two LSTM in series on word-level. The structure and mathematical details are given below.

3.1 Uni-LSTM tagging model

Tagging is a classification task, where a tagging model is trained to determine the category of every element within the input sequence. Neural network tagging model utilizes the neural network to accomplish tagging task [11]. Much work has been proposed to improve the accuracy of the tagging model. Recently, unidirectional and bidirectional LSTM RNN with word embedding showed a significant improvement [12] compared to previous models.

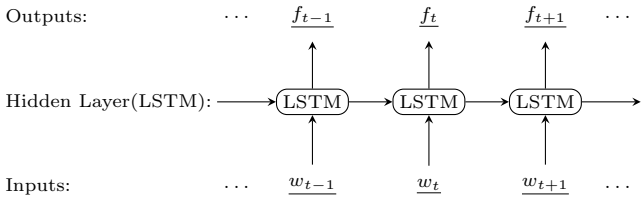

Fig. 2. *uni-directional LSTM tagging model*

As illustrated in Figure 2, in order to produce word-synchronized features, an unidirectional LSTM rather than bidirectional LSTM is utilized to build the tagging model. Moreover, in our preliminary experiments, bidirectional LSTM only give slight tagging accuracy gain over unidirectional ones.

An LSTM network is formed like the standard RNN except that the self-connected hidden units are replaced by specially designed units called memory blocks. In this paper, LSTM memory block is denoted as $\mathcal{L}$. To avoid confusion, the LSTM memory block of tagging model and language model are denoted as $\mathcal{L}_{\texttt{tag}}$ and $\mathcal{L}_{\texttt{LM}}$ severally.

The vector $\boldsymbol{w}_t$ uses one hot coding to represent the current word in the time step t, which is the input of both $\mathcal{L}_{\texttt{tag}}$ and $\mathcal{L}_{\texttt{LM}}$.

And then the word embedding can be obtained as $\boldsymbol{x}_t$:

$$\boldsymbol{x}_t = E_{tag}\boldsymbol{w}_t \tag{1}$$

here E_{tag} is the word embedding matrix of tagging model,

The output of the LSTM hidden layer $\boldsymbol{h}_t$ in unidirectional tagging model is calculated as:

$$\boldsymbol{h}_t = \mathcal{L}_{\texttt{tag}}(\boldsymbol{x}_t, \boldsymbol{h}_{t-1}) \tag{2}$$

The computational details in a LSTM memory block $\mathcal{L}$ is formulated as the following composite function:

$$\begin{aligned} i_t &= \sigma(W_{xi}x_t + W_{hi}h_{t-1} + W_{ci}c_{t-1} + b_i) \\ f_t &= \sigma(W_{xf}x_t + W_{hf}h_{t-1} + W_{cf}c_{t-1} + b_f) \\ c_t &= f_t c_{t-1} + i_t \tanh(W_{xc}x_t + W_{hc}h_{t-1} + b_c) \\ o_t &= \sigma(W_{xo}x_t + W_{ho}h_{t-1} + W_{co}c_t + b_o) \\ h_t &= o_t \tanh(c_t) \end{aligned} \tag{3}$$

Where σ is the logistic (sigmoid) function, and i, f, o and c are respectively the *input gate*, *forget gate*, *output gate* and *cell* activation vectors.

$\boldsymbol{f}_t$ is the output of the tagging model, which is a probability distribution, and can be calculated from the LSTM memory block as:

$$\boldsymbol{f}_t = softmax(W_{ho}\boldsymbol{h}_t + \boldsymbol{b}_y) \tag{4}$$

Where softmax is a normalizing separator.

According to LSTM-RNN tagging model, the obtained probability distribution of each step is supposed to be independent with each other. However, in some tasks such as NER and chunking, tags are highly correlated with each other, for example, a few of types of tags can only follow specific types of tags.

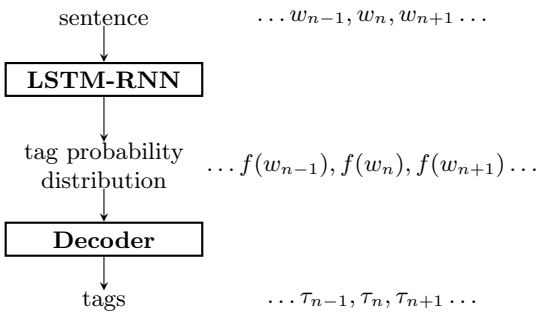

Fig. 3. LSTM tagging system with decoding

Adding these constrained relationships in prediction of tags can help a lot. As figure 3 shows, to make use of this kind of labeling constraints, a transition matrix between each step's output is utilized. This matrix is incorporated with probability distribution to generate a decoding process[12]. Which is a typical dynamic programming problem and can be solved with Viterbi algorithm[13].

In this paper, decoding process is denoted as $\mathcal{D}(\cdot)$, the output of decoder τ_t is a series of the final predicted tags, and they are also expressed as one hot vector:

$$\boldsymbol{\tau}_t = \mathcal{D}(\boldsymbol{f}_t) \tag{5}$$

3.2 Multi-view LSTM language model with word-synchronized auxiliary feature

Figure 4 shows the structure of the proposed multi-view LSTM language model with word-synchronized auxiliary feature.

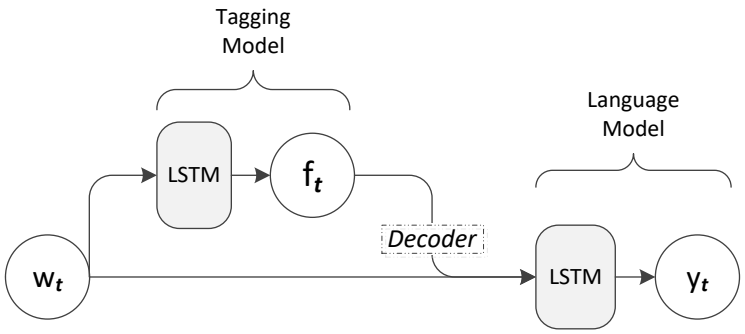

Fig. 4. *multi-view LSTM language model with word-synchronized auxiliary feature*

Our proposed model is a multi-view language model connected with a unidirectional tagging model. The first input of language model $\boldsymbol{w}_t$ is a one hot vector representing the current word, which is also feed into tagging model. The second input can be the output of the tagging model or decoder. Whether the decoding process is used or not, experimentation and comparison of them have been accomplished in Section 4.

The difference of formula between them is reflected in the input part. If using the decoder, the input of multi-view language model is:

$$\boldsymbol{\zeta}_t = W_{tag}\boldsymbol{\tau}_t + E_{word}\boldsymbol{w}_t \tag{6}$$

$\boldsymbol{\tau}_t$ is the output of decoder as one hot vector. Otherwise, the input of multi-view language model is:

$$\boldsymbol{\zeta}_t = W_{tag}\boldsymbol{f}_t + E_{word}\boldsymbol{w}_t \tag{7}$$

$\boldsymbol{f}_t$ is the output of tagging model as probability distribution. E_{word} is the Word embedding matrix, and W_{tag} is the parameter matrix connecting output of tagging model and hidden layer of language model. They are added together as the final input for LM at each time-step.

The output of the LSTM hidden layer of language model $\boldsymbol{h}_t$ is calculated as:

$$\boldsymbol{h}_t = \mathcal{L}_{\texttt{LM}}(\boldsymbol{\zeta}_t, \boldsymbol{h}_{t-1}) \tag{8}$$

The computational details in a LSTM memory block $\mathcal{L}$ is introduced in eq.3. $\boldsymbol{y}_t$ is the output of the language model, which is the probability distribution of the next word $P(\boldsymbol{x}_{t+1}|\boldsymbol{x}_1{:}\boldsymbol{x}_t)$, and can be calculated from output of the LSTM memory block as:

$$\boldsymbol{y}_t = softmax(W_{ho}\boldsymbol{h}_t + \boldsymbol{b}_y) \tag{9}$$

3.3 Training method

The proposed model consists of two parts: language model and tagging model. The language model training process follows standard convention, calculates the cross entropy of each word and then conducts back-propagation. We use mini-batch based stochastic gradient descent (SGD) as optimization method.

Since the training of tagging model can be varied, we propose five different training methods:

1) Train the tagging LSTM as an independent model in advance and fix it when training the multi-view language model. Only this method can utilize decoding process, because next methods need train tagging model but the decoding process doesn't support this. The advantage is that decoder is beneficial to promote the tagging model, but it has a disadvantage that it can't be updated when language model is training.

2) Train the tagging LSTM in advance, which will also be jointly trained during LM training, with the same learning rate of LM. The trouble is that the well-trained tagging model might be destroyed when the language model has just started training with a relatively big learning rate.

3) Randomize the parameters of the tagging part of our model and train the language model and tagging model jointly, with a same learning rate.

4) Because the choke point of the second and third methods is learning rates, Utilize stabilizer [14][15] in the training process based on the second method, in order to adjust learning rates in different parts of proposed model dynamically.

5) Similar to the forth method, this method adds beta stabilizer to the third method.

Since that which one is the best training method for our proposed model is unknown, these five methods are all tested and evaluated in experiments. The results are shown in section 4.

4 Experiments

4.1 Setting

We test our models on both English and Chinese data sets, including PTB, Swb-Fisher and Chinese SMS.

The first data set is the Penn Treebank Corpus (PTB) [16], which is a widely used data set to evaluate the effectiveness of a language model. It contains about 40K sentences of training, 3K validation sentences and 4K testing sentences.

Swb-Fisher data set is an English data set which contains ten millions words in training data. In the test stage, the switchboard subset of the NIST 2000

Hub5e set is used (referred to as hub5e, 1831 utterances). For the corresponding ASR-rescore task, the acoustic models are trained on English Switchboard task with 7-layer CD-DNN-HMM and 2048 neurons in each layer. Fourier transform based log filter-banks with 40 coefficients are used as feature. An English interpolated trigram language model trained on Swb-fisher data set is used in 1-pass decoding to generate N-best lists for ASR-rescore task.

SMS is a Chinese data set with two million words collected from the short message service (SMS). A mandarin spontaneous conversation test-set (about 25 hours) is taken for the decoding stage. A 5000 hours Mandarin corpus is used to train the DNN-HMM acoustic model. Lastly, a tri-gram language model trained on the SMS data set with forty thousand words is used in 1-pass decoding

At first, we wanted to investigate which kind of linguistic features contribute the most to our model. As the two LSTM structures of our model are combined on word-level, we only considered word-level information such as part of speech (POS), name entity recognition (NER) and chunking (CHUNK). The chunking label is produced by a method described in [7], which can transform a syntax parser tree into chunking on word-level. As we don't have real tagging label on all our training data, we used the popular Stanford Core-NLP tools[1] to generate labels on the data, and treat them as ground truth in our experiment.

Regarding the training of our proposed tagging model, we try the five methods that mentioned in Section 3 on Chinese SMS. Then apply the best one to our following experiments with large scale.

Our general target is to compare our proposed multi-view LM with LSTM LM baseline and other LMs in Section 2. The multi-task model is a original single hidden layer LSTM model with two task. Multi-task and multi-view model is refer to the joint model in [10].

The hidden layers in all LSTM used in our experiments are of size 300, and the dropout rate is set to 0.5. SRILM [17] is used to produce n-gram LM.

4.2 Evaluation of the tagging model

POS, NER and CHUNK are the three different tag types used in our model. Table 1 shows the accuracy of unidirectional LSTM tagging model in proposed LM on all data sets. As we don't have real tagging label on all data sets, the training data sets are labeled by Standford tool and regarded as ground truth. So the accuracy is relative to Stanford label. The accuracy of these data-sets with POS are relatively high, with CHUNK and NER are not bad too. So it means we won't lose much information even if a unidirectional model is used.

4.3 Evaluation of training methods

Table 2 shows the perplexity comparison on five training methods of our proposed model, which are described in section 3.2. The data set is Chinese SMS.

[1] http://nlp.stanford.edu/software/

Table 1. *Accuracy of word-level features on all data-sets we used*

Data	Accuracy(decoding/no decoding)		
	POS	NER	CHUNK
ptb	96.36/96.32	80.32/82.06	84.23/81.77
fisher	94.24/94.24	79.93/81.97	85.72/83.47
sms	94.87/94.84	80.44/82.12	85.55/82.98

Table 2. *Perplexity comparison of different methods for training proposed model*

Model	Training Method	PPL
LSTM LM	-	102.11
Multi-view LM	1	98.02
	2	135.72
	3	148.65
	4	105.47
	5	107.21

The result (Table 2) shows that the first method is superior for training a tagging model. The second and third methods are obviously worse than the first one, because language model and tagging model are relatively independent as two difference task. Connecting two models in series and training them with identical learning rate will fail to adjust an appropriate learning rate for both, so can not lead to a win-win result. This conclusion is supported by the result of forth and fifth methods. When utilizing the beta stabilizer which can adjust learning rates in different LSTM dynamically, the PPL will be much less than second and third methods. However the last two methods did not exceed the first one as expected, because the beta stabilizer can only compensate the shortage of the imperfect learning rate, worse than training them respectively with their own appropriate learning rate. Moreover, the tagging LSTM in the first method is well trained enough. Therefore, the first method is used to train the proposed model. For all the following experiments, the tagging model will be trained independently and combined with language LSTM after training.

4.4 Evaluation of our multi-view language model

In this section, the data sets introduced above are used to evaluate the effectiveness of our multi-view language model. First we tried a wide range of models on the PTB data set and compared them over their respective perplexity, but note that PTB set cannot be tested for ASR-rescore task.

As we can see at Table 3, either POS by Stanford tool or our unidirectional LSTM tagging model, can significantly boost the performance.

Table 4 shows the perplexity, WER and SER on different added tags. The result shows a large improvement from POS feature, while improvement from NER and CHUNK are negligible. The reason might be that not only the tagging accuracy of these two types of feature are not very high, but also NER feature

Table 3. *Perplexity comparison of different LMs on PTB data set*

Model	Tagging	PPL
4-gram	-	141.46
LSTM LM	-	98.73
Multi-task+POS	-	100.88
Multi-view+Multi-task+POS	-	93.47
Multi-view+POS	Standford	91.69
	LSTM	**93.82**
Multi-view+NER	Standford	97.63
	LSTM	**97.92**
Multi-view+CHUNK	Standford	94.34
	LSTM	**95.63**

is sparse and CHUNK feature is less precise. Based on this result, we only use the POS feature for the large scale Fisher data experiments.

Table 5 shows the perplexity, WER and CER on Swb-Fisher set using the POS feature. The observations from this data set are similar as those from Chinese SMS.

Table 4. *Perplexity, WER (%) and SER (%) comparison on Chinese SMS task*

Model	Tagging	PPL	WER	SER
4-gram	-	124.23	13.41	42.16
LSTM	-	102.11	11.30	41.59
Multi-task+POS	-	103.42	11.32	41.63
Multi-task+Multi-view+POS	-	98.24	10.91	40.89
Multi-view+POS	Standford	94.41	11.19	41.92
	LSTM	**98.02**	**10.83**	**40.71**
Multi-view+NER	Standford	101.88	11.28	41.59
	LSTM	**102.08**	**11.32**	**41.72**
Multi-view+CHUNK	Standford	96.71	11.25	41.63
	LSTM	**100.30**	**11.02**	**41.23**

When comparing our model with baseline, which is a ordinary LSTM LM, the perplexity, WER and SER are reduced with 4.0%, 4.0%, 2.0% relatively in both English and Chinese data set. Moreover, our proposed multi-view LSTM LM decreases the WER and SER with 4.4%, 2.2% and 3.2%, 2.8% than multi-view LSTM LM with Stanford POS. It needs to be emphasized that, our model doesn't perform better than multi-view LM with Stanford POS on PPL, because the multi-view LM with Stanford POS feature uses future information which shouldn't be fed to LM, As we argued in Section 1.

For all data sets, the multi-view LSTM language model with POS feature by Stanford as well as LSTM tagging model is significantly improving with regards to perplexity, by comparing to the original LSTM RNN language model. The

Table 5. *Perplexity, WER (%) and SER (%) comparison on English Fisher task*

Model	Tagging	PPL	WER	SER
4-gram	-	79.12	16.3	53.75
LSTM	-	65.42	15.62	53.42
Multi-task+POS	-	65.93	15.60	53.65
Multi-task+Multi-view+POS	-	92.77	15.20	52.23
Multi-view+POS	Standford	60.24	15.73	53.40
	LSTM	**62.71**	**15.01**	**52.19**

multi-view LSTM language model with Stanford POS works better in terms of perplexity, which is widely proved by previous works, but failed to show a commensurate increase in ASR-rescore task [8]. The stanford CoreNLP POS tagger model uses a maximum entropy algorithm [18] that utiliz future information, the future information only contributes to PPL task rather than the ASR-rescore task. A maximum entropy model uses contextual future information to calculate the posterior probability of the text. That is to say, adding this POS feature in language model training is equal to adding the future information in additional to the present word. This leads to a perplexity improvement, but no gain for WER and SER in the rescoring task is observed.

In addition, we want to confirm whether the POS tagging part contributed to the PPL or not, and thus add another contrasting experiment to validate our conclusion. In order to compare the influence of the tag information between the proposed and traditional model, we conduct experiments by removing and adding tag features to the model the input.

The comparison is shown in Table 6. "Without-tag" means that we deliberately remove the feature feed from the trained tagging part of proposed model. The results indicate that tests with tag feature are generally better than those without. And the latter even does not outperform an ordinary LSTM LM, which means tag feature plays an important role in the LM performance. Furthermore, the result confirms the contribution of our POS tagging model. Moreover, our model performs better than the original multi-view LM without tags, which confirms the contribution of our tagging model in the proposed multi-view structure.

Table 6. *Perplexity on different test-data*

Data-set	Tagging	PPL	
		without-tag	with-tag
PTB	Standford	98.77	91.69
	LSTM	98.64	93.82
Fisher	Standford	66.25	60.24
	LSTM	65.79	62.71
SMS	Standford	102.20	94.41
	LSTM	102.13	98.02

The proposed LSTM LM with word-synchronized auxiliary feature, performs better not only on ASR-rescore task but also PPL task. For this result, we consider the following two aspects. On one hand, the tagging model change the contextual information to word-synchronized auxiliary information, which is rather useful in ASR-rescore task. On the other hand, the proposed LM with unidirectional tagging process permits the decoding of LVCSR to proceed in real time.

5 Conclusion

In this paper, we propose a multi-view LSTM LM with unidirectional tagging model, which produces word-synchronized auxiliary feature that only incorporate previous contextual information. This auxiliary feature is combined with the word sequence to train a multi-view LSTM LM. Five different training methods for this model are tested, and the best one is used in the large-scale ASR task. In the comparison experiments between our model and related works (N-gram LM, multi-task LM, multi-view LM with Stanford tagging and multi-task combined with multi-view LM), the used data sets are PTB, English Fisher and Chinese SMS and PPL, WER and SER are used as the evaluation criteria. Our proposed model shows significant improvements for all word-level features including POS, NER and chunking on WER and SER. Especially for the POS feature on English Fisher, our proposed model not only gives gian (4.0%) on PPL, but also shows significant WER and SER reduction (relative 4.0% and 2.0%) in ASR task, compared with the baseline (LSTM LM). Most importantly, comparing to the multi-view LM with POS feature produced by traditional model (Stanford tool), our model shows better result of WER and SER (4.4% and 2.2%) in ASR-rescore task. For more related models like multi-task and multi-task combined with multi-view models, our model all shows achievement in varying degrees.

References

1. Slava Katz: Estimation of probabilities from sparse data for the language model component of a speech recognizer. IEEE transactions on acoustics, speech, and signal processing, vol. 35, no. 3, 400-401 (1987)
2. Mikolov, Tomáš and Karafiát, Martin and Burget, Lukáš and Černockỳ, Jan and Khudanpur, Sanjeev: Recurrent neural network based language model. INTERSPEECH, Conference of the International Speech Communication Association, Makuhari, Chiba, Japan, September DBLP, 1045-1048 (2010)
3. Mikolov, Tomáš and Kombrink, Stefan and Burget, Lukáš and Černockỳ, Jan and Khudanpur, Sanjeev: Extensions of recurrent neural network language model. IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP). IEEE, 5528-5531 (2011)
4. Hochreiter, Sepp and Schmidhuber, Jrgen: Long short-term memory. Neural Computation, vol.9, col.8, 1735-1780 (1997)
5. Sundermeyer, Martin and Schlüter, Ralf and Ney, Hermann: Lstm neural networks for language modeling. INTERSPEECH, vol.31, 194-197 (2012)

6. Yangyang Shi, Pascal Wiggers, and Catholijn M Jonker: Towards recurrent neural networks language models with linguistic and contextual features. INTERSPEECH, vol.48 1664-1667 (2012)
7. Tjong Kim Sang, F Erik, Buchholz, and Sabine: Introduction to the CoNLL-2000 shared task: chunking. The Workshop on Learning Language in Logic and the Conference on Computational Natural Language Learning, 127-132 (2000)
8. Yangyang Shi, Martha Larson, Joris Pelemans, Catholijn M Jonker, Patrick Wambacq, Pascal Wiggers, and Kris Demuynck: Integrating meta-information into recurrent neural network language models. Speech Communication, vol.73, 64-80 (2015)
9. Ronan Collobert and Jason Weston: A unified architecture for natural language processing: deep neural networks with multitask learning. International Conference, 160-167 (2008)
10. Bing Liu and Ian Lane: Joint Online Spoken Language Understanding and Language Modeling with Recurrent Neural Networks. IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), 160-167 (2016)
11. Helmut Schmid: Part-of-speech tagging with neural networks. Proceedings of the 15th conference on Computational linguistics-Volume 1. Association for Computational Linguistics, 172-176 (1994)
12. Peilu Wang, Yao Qian, Frank K. Soong, Lei He, and Hai Zhao: A unified tagging solution: Bidirectional lstm recurrent neural network with word embedding. Computer Science (2015)
13. Andrew.J. Viterbi: Error bounds for convolutional codes and an asymptotically optimum decoding algorithm. Information Theory, IEEE Transactions , 260-269 (1967)
14. Pegah Ghahremani and Jasha Droppo. Self-stabilized deep neural network. IEEE International Conference on Acoustics, Speech and Signal Processing, 5450-5454 (2016)
15. Liu Qi, Tan Tian, and Yu Kai: An investigation on deep learning with beta stabilizer. IEEE International Conference on Signal Processing, 557-561 (2017)
16. Ann Taylor, Mitchell Marcus, and Beatrice Santorini: The Penn Treebank: An Overview. Springer Netherlands, vol.20, 5–22(2003)
17. Andreas Stolcke et al.: Srilm-an extensible language modeling toolkit. International Conference on Spoken Language Processing, 901–904 (2002)
18. Kristina Toutanova and Christopher D. Manning: Enriching the knowledge sources used in a maximum entropy part-of-speech tagger. Joint Sigdat Conference on Empirical Methods in Natural Language Processing and Very Large Corpora: Held in Conjunction with the Meeting of the Association for Computational Linguistics, 63-70 (2000)