FAST OOV WORDS INCORPORATION USING STRUCTURED WORD EMBEDDINGS FOR NEURAL NETWORK LANGUAGE MODEL

Ruinian Chen and Kai Yu

Key Laboratory of Shanghai Education Commission for Intelligent Interaction and Cognitive Engineering
SpeechLab, Department of Computer Science and Engineering
Shanghai Jiao Tong University, Shanghai, China
{ruinian.chen, kai.yu}@sjtu.edu.cn

ABSTRACT

Recently, deep learning approaches have been widely used in language modeling and achieved great success. However, the out-of-vocabulary (OOV) words are often estimated in a rather crude way using only one special symbol, which ignores the linguistic information. In this paper we present an LSTM language model with structured word embeddings to tackle this problem. In our model, both input and output embeddings of LSTM language model are deployed with structured word embeddings. Utilizing syntactic-level and morphological-level parameters sharing, OOV words can be incorporated into the proposed model without retraining. The LSTM language model with structured word embeddings is instantiated for Chinese. Experiments show that the proposed model achieves PPL improvement on OOV words, and can be further integrated into automatic speech recognition systems for fast vocabulary updating.

Index Terms— OOV words, language modeling, syntactic embeddings, morphological embeddings

1. INTRODUCTION

Language model (LM) aim to judge the fluency and reasonability of a sentence, which is widely used in machine translation, automatic speech recognition (ASR) and other tasks. Although the conventional back-off n-gram language model has been the dominating model in this field for a long time, it suffers from the data sparsity problem caused by huge number of possible word combinations in real text. Recently, neural network language models can significantly outperform conventional n-gram models [1, 2]. After that, the Long Short-Term Memory (LSTM) network was introduced into language modeling, and achieves promising results [3, 4], due to its ability to model long-term contexts.

One of the main problems for most statistic language models is that they can not predict out-of-vocabulary (OOV) words. Consequently, OOV words lead to serious problems in many language models related tasks such as language model adaptation [5, 6] and automatic speech recognition (ASR) [7]. Especially in ASR, OOV words usually cause high word-error-rate (WER), because not only OOV words are recognized as in-vocabulary (IV) words but also the error will affect other nearby words.

The corresponding author is Kai Yu. This work has been supported by the National Key Research and Development Program of China under Grant No.2017YFB1002102, and the China NSFC projects (No. 61573241). Experiments have been carried out on the PI supercomputer at Shanghai Jiao Tong University.

A widely-used approach is to use a special $< unk >$ to represent all OOV words. But this approach has many shortcomings [8]: 1) OOV words can not be predicted, 2) the frequency of OOV words is mismatch between the training and test set, and 3) the approach ignores their types or linguistic information. In fact, OOV words can be categorized into two types [7]. First, words occur in both training and test sets. Those words are treated as OOV words because of their low-frequency occurrences. In this case, we can update the vocabulary and retrain the model. But this is time-consuming and the probability for those new words is unreliable due to data insufficiency. The second type of OOV words is the words which do not occur in the training set at all but only occur in the test set. In this situation, additional information is required.

In [9], a class-based n-gram language model is used to handle OOV words. The OOV words are assigned to the in-vocabulary (IV) classes consisting of IV words with similar semantic meanings. Web data is used to find the relation between IV words and OOV words. Morphological features are also used for OOV words clustering in [10, 11]. Subword-level language models exploited in [12, 13, 14] have also been used to handle OOV words in ASR systems. The main problem of subword-level language models is that they can not calculate the probability of words, hence the traditional perplexity measurement can not be applied.

In this paper, we propose an LSTM language model with structured word embeddings which has the ability to handle OOV words without retraining. The structured word embeddings are composed of two parts: syntactic embeddings and morphological embeddings. Part-of-speech (POS) classes are used to build syntactic embeddings. Morphological embeddings are generated by the character-level fixed-size ordinally-forgetting encoding (FOFE) method [15]. The LSTM language model with structured word embeddings is evaluated on a Chinese short message service (SMS) dataset. The proposed model has competitive performances to the baseline LSTM language models, and the number of parameters that need to be estimated is significantly fewer. Moreover, LSTM language model with structured embeddings can be integrated into ASR systems for fast vocabulary updating. Experimental results show improvements on both PPL and WER.

2. OOV WORDS TREATMENT IN LSTM LANGUAGE MODEL

2.1. LSTM Language Model

Recently, deep learning approaches are widely used in LM and have achieved great successes. The long short-term memory (LSTM) net-

work is a type of recurrent neural network (RNN) architecture particularly suited for sequence. Let $\mathcal{V}$ be vocabulary set. At each time stamp t, the input word w_t is represented by a one-hot vector $\boldsymbol{e}_t$, and then the word embedding can be obtained as $\boldsymbol{x}_t$:

$$\boldsymbol{x}_t = \boldsymbol{E}_{\mathtt{i}} \boldsymbol{e}_t \tag{1}$$

where $\boldsymbol{E}_{\mathtt{i}} \in \mathbb{R}^{m \times |\mathcal{V}|}$ is the input word embedding matrix, m denotes the dim of input word embedding. Concretely, one step of the LSTM takes $\boldsymbol{x}_t, \boldsymbol{h}_{t-1}, \boldsymbol{c}_{t-1}$ as inputs, and produces $\boldsymbol{h}_t, \boldsymbol{c}_t$. Computation details are omitted in this paper. The probability distribution of the next word is calculated at the output layer by applying an affine transformation to the hidden layer followed by a softmax function:

$$P(w_{t+1} = j | w_{1:t}) = \frac{\exp(\boldsymbol{h}_t^{\mathrm{T}} \boldsymbol{E}_{\mathtt{o}}^j + b^j)}{\sum_{k \in \mathcal{V}} \exp(\boldsymbol{h}_t^{\mathrm{T}} \boldsymbol{E}_{\mathtt{o}}^k + b^k)} \tag{2}$$

Where $\boldsymbol{E}_{\mathtt{o}}^j$ is the j-th column of $\boldsymbol{E}_{\mathtt{o}} \in \mathbb{R}^{m \times |\mathcal{V}|}$, also referred as output embeddings, and $\boldsymbol{b}^j$ is the bias term. We found that the bias term of output layer plays an important role and highly related to the frequency of words.

Since most computational cost lies in propagation on the output layer, the factorized softmax output layer was proposed for language model speed up [16]. This method is based on the assumption that words can be mapped to classes. Let $\mathcal{S}$ be the class set. Different from equation (2), the probability distribution of the next word of the factorized output layer is calculated as below:

$$P(w_{t+1} = j | w_{1:t}) = P(s_{t+1} = \mathtt{s}_j | \boldsymbol{h}_t) P(w_{t+1} = j | \mathtt{s}_j, \boldsymbol{h}_t) \tag{3}$$

$$P(s_{t+1} = \mathtt{s}_j | \boldsymbol{h}_t) = \frac{\exp(\boldsymbol{h}_t^{\mathrm{T}} \boldsymbol{E}_{\mathtt{c}}^{\mathtt{s}_j} + b_{\mathtt{c}}^{\mathtt{s}_j})}{\sum_{n \in \mathcal{S}} \exp(\boldsymbol{h}_t^{\mathrm{T}} \boldsymbol{E}_{\mathtt{c}}^n + b_{\mathtt{c}}^n)} \tag{4}$$

$$P(w_{t+1} = j | \mathtt{s}_j, \boldsymbol{h}_t) = \frac{\exp(\boldsymbol{h}_t^{\mathrm{T}} \boldsymbol{E}_{\mathtt{o}}^j + b^j)}{\sum_{k \in \mathcal{V}_{\mathtt{s}_j}} \exp(\boldsymbol{h}_t^{\mathrm{T}} \boldsymbol{E}_{\mathtt{o}}^k + b^k)} \tag{5}$$

Where $\mathtt{s}_j$ denotes the class of word w_{t+1}, and $\mathcal{V}_{\mathtt{s}_j}$ is the set of all words belong to class $\mathtt{s}_j$. Here the probability calculation of words are divided into two stages: we first estimate the probability distribution over the classes and then compute the probability of a particular word from the desired class. Actually a word can belong to multiple classes. But in this paper, each word is mapped to a distinct class i.e. all classes are mutually exclusive. Commonly used word classes are frequency-based classes or those obtained from data-driven methods [16].

2.2. OOV Words Treatment

As mentioned before, two approaches have been used to deal with OOV words problem in classic LSTM language models:

1. A special class $< unk >$ is used to substituted all OOV words. Following [9], another measurement, called adjusted perplexity, is used:

$$P(w_{\mathtt{OOV}}) = \frac{P(< unk >)}{|\mathcal{V}_{\mathtt{OOV}}|} \tag{6}$$

   where $\mathcal{V}_{OOV}$ is the vocabulary set of all OOV words. We refer this approach as "unk" in experiments.

2. Retraining the model with the updated vocabulary. Since the OOV words have no or very few positive examples in the training set, its probability will be assigned to a small value after training. This method can be analogous to the smoothing methods used in n-gram language model. We refer this approach as "retrain" in experiments.

The two approaches all have their shortcomings. In the unk LSTM language model, the probability of OOV words is misestimated because the frequency of OOV words is mismatch between the training and test data. Moreover, this approach ignores the linguistic information of OOV words. The main problem of the retrained LSTM language model is time consumption.

3. STRUCTURED WORD EMBEDDINGS FOR OOV WORDS INCORPORATION

In traditional LSTM language models, word embeddings for each word are independent, which causes two problems. First, new words can not utilize the trained word embeddings. Second, the embeddings for rare words are undertrained because of insufficient training data. The motivation of structured word embeddings is to utilize parameter sharing to solve those two problems. Unlike data-driven methods, the parameter sharing approach must be based on explicit rules. By exploiting both syntactic and morphological rules, we can easily find the shared parameters for OOV words to build their own structured word embeddings in our model.

3.1. Morpho-Syntactic Structured Embeddings

At syntactic-level, each word is assigned to a single part-of-speech (POS) class. All words in the same POS class share the same POS class embedding, referred as syntactic embedding.A part-of-speech is a category of words which have similar grammatical properties [17]. Hence, we assume that syntactic embeddings represent the basic syntactic function of words.

For each word, we use several example sentences to label its POS tags and choose the most frequent one as the class of the word (POS tags can also be obtained from dictionaries). The example sentences of in-vocabulary (IV) words are chosen from the training set. For OOV words, example sentences can be made up or chosen from other data sources, such as web data. Different from data-driven methods, the POS tag based syntactic embeddings can be easily generated for OOV words using rules.

Characters (or subwords) representation are widely used in many NLP tasks as an additional feature to improve the performance on low-frequency words, especially in morphologically-rich languages [18, 19, 20]. But for high-frequency words, the improvement is limited. In this paper, morphological embeddings are built for low-frequency words to further capture their semantic meaning. This is based on the assumption that the data sparsity problem of low-frequency words is less serious at the character level. For high-frequency words, the word embeddings are retained. Hence mixed embeddings, the morphological embeddings for low-frequency words and the word embeddings for high-frequency words, should be in the same dimension.

In previous works [18, 21], both word embeddings and subword-level feature are combined together to get the augmented embeddings for all words. That is to say, they can not handle OOV words. In contrast, the proposed morphological embeddings for low-frequency words in this paper only rely on character-level features. Hence, it has the ability to model OOV words.

The proposed morphological embeddings leverage character information through a character-level fixed-size ordinally-forgetting encoding (FOFE) [15]. In our model, all low-frequency words are represented by sequences of characters $\boldsymbol{e}_{1:T}$ [1], where $\boldsymbol{e}_t$ is the one-hot representation of the character at time stamp t. FOFE encodes

[1]The definition of sub-word may vary for different languages. In this paper, Chinese is used to instantiate the structured embedding framework

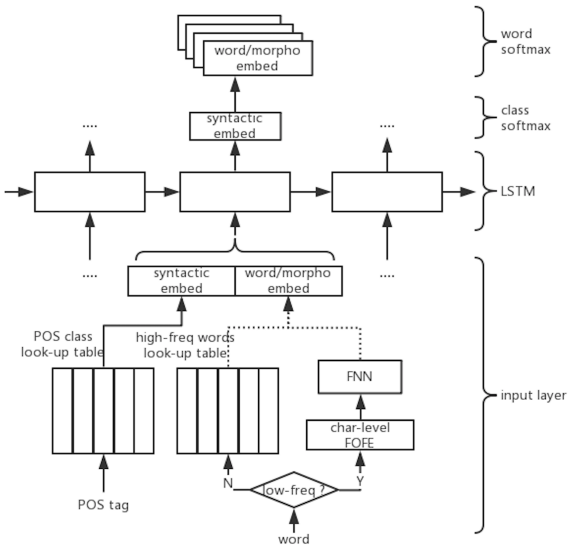

Fig. 1. The architecture of LSTM with structured embeddings

the whole sequence based on a simple recursive formula (with $\boldsymbol{z}_0 = \boldsymbol{0}$) as:

$$\boldsymbol{z}_t = \alpha \boldsymbol{z}_{t-1} + \boldsymbol{e}_t (1 \leq t \leq T) \tag{7}$$

here $0 < \alpha < 1$ is a constant forgetting factor to control the influence of the history on the final time step. Moreover, a feed-forward neural network (FNN) is used to convert the character-level FOFE encodes into the final morphological embeddings.

3.2. Incorporating Structured Embeddings with LSTM LM

The architecture of our model is shown in figure 1. At the input layer, the structured embedding of the input word is obtained by concatenating its syntactic embedding with word embedding (for high-frequency words) or morphological embedding (for low-frequency words). At the output layer, it is straightforward to utilize the factorized softmax structure mentioned in section 2.1. The output class embedding matrix $\boldsymbol{E}_c$ in equation (4) is replaced by syntactic embeddings, and the output embedding matrix $\boldsymbol{E}_o$ in equation (5) is replaced by word and morphological embeddings.

Once training is done, it is easy to get both syntactic and morphological embeddings for OOV words [2]. In order to calculate the probability for OOV words, we need to rebuild the output layer parameters: $\boldsymbol{E}_o, b$ in equation (5). All embeddings in $\boldsymbol{E}_o$ and bias term b are retained for IV words, and the embeddings for OOV words in $\boldsymbol{E}_o$ are filled with their morphological embeddings. In experiments, we found that the bias term is highly related to word frequency, which means the more frequent words have larger bias value. In this paper, we treat the bias term of OOV words as an empirical small constant value.

By leveraging structured word embeddings, OOV words can be incorporated into LSTM language model **without retraining**. As we mentioned before, the data sparsity problem of OOV words can also be relieved through parameter sharing in the proposed model during training.

and character is employed as the sub-word unit.

[2]We assume that all characters are covered in the training set character vocabulary

3.3. Parameter Compression

The proposed structured word embeddings can also be regarded as a parameter compression method. In LSTM language models, the word embeddings of low-frequency words occupy a large part of model parameters but are undertrained. By substituted low-frequency words with character representation, we can greatly reduce the number of parameters.

Let $\mathcal{V}$ be the vocabulary set of all words and H be the hidden layer size. In LSTM language models, the number of parameters for word embeddings is $2 \times |\mathcal{V}| \times H$. However, in the LSTM language model with structured embeddings, the total number of parameters is $(|\mathcal{V}_h| + |\mathcal{V}_{char}| + |\mathcal{S}|) \times H$ [3], where $\mathcal{V}_h$ denotes the high-frequency words and $\mathcal{V}_{char}$ denotes the character set. $\mathcal{S}$ denotes the POS tag set. In our experiments, we have $|\mathcal{V}| = 60000, |\mathcal{V}_h| = 8000, |\mathcal{V}_{char}| = 5000, |\mathcal{C}| = 32$, hence we can reduce amount of parameters by almost 90%.

4. EXPERIMENTS

LSTM language model with structured word embeddings is evaluated on a Chinese short message service (SMS) dataset. Table 1 gives the information of datasets in details. Two different sized vocabularies are used for each dataset. The full vocabulary $\mathcal{V}_f$ covers all words appearing in the corpus. The small vocabulary $\mathcal{V}_s$ is a subset of $\mathcal{V}_f$. Here in-vocabulary (IV) is defined as words in $\mathcal{V}_s$ and out-of-vocabulary (OOV) means words in $\mathcal{V}_f$ but not in $\mathcal{V}_s$. The sms-30m dataset also as training set and a Mandarin spontaneous conversation test set (about 25 hours, 3K utterances) are used for ASR rescore task.

Dataset		#tokens	$\lvert\mathcal{V}_s\rvert$	$\lvert\mathcal{V}_f\rvert$	OOV rate on $\mathcal{V}_s$
sms	train	1.8M	33K	39K	0.8%
	valid	100K			1.0%
	test	1.5K			0.8%
sms-30m	train	5.6M	57K	67K	0.7%
	valid	166K			0.7%
	test	112K			0.6%

Table 1. Datasets Information

As indicated in section 2.2, two OOV words incorporation methods are used to build the baselines:

1. The LSTM language model is trained with the small vocabulary $\mathcal{V}_s$ and all OOV words are treated as a single $< unk >$ symbol, referred as "unk".
2. Retraining LSTM language model with the full vocabulary $\mathcal{V}_f$, referred as "retrain".

For the proposed LSTM language model with structured word embeddings, the small vocabulary $\mathcal{V}_s$ is used in the training phase, and the vocabulary of model is updated to $\mathcal{V}_f$ in the test phase.

In order to keep consistent with the size of the proposed model, the input embeddings size of the LSTM baselines is set to 600 and the output embeddings size is set to 300. In the LSTM language model with structured embeddings, the syntactic embeddings size is set to 300. And we use a 1 layer 5000-300 FNN for FOFE encoding, where 5000 is the size of character set $\mathcal{V}_c$ and 300 is the dim of morphological embeddings. The α of FOFE is set to 0.7, the bias term of new words is set to 0, those two empirical parameters are

[3]We do not times 2 because input and output embeddings are shared in our model.

fine-tuned in the valid set. The most frequent 8192 words are chosen to be the high-frequency words, and other words are all treated as low-frequency words in our model. All models are trained using stochastic gradient descent (SGD) with the same hyperparameters.

4.1. Perplexity Evaluation

The perplexity evaluation results are shown in table 2. Particularly, the PPL calculation of OOV words is replaced by equation (6) for the "unk" LSTM. It is shown that the proposed structure embeddings (SE) approach have similar performance with the unk LSTM. However, the retrained LSTM perform worse. For further investigation, we break down the PPL calculation for in-vocabulary (IV) and out-of-vocabulary (OOV) words separately for each model. The experimental results are shown in table 3. The unk LSTM perform best in IV words at the cost of sacrificing OOV words, because its PPL of OOV words is extremely high. Comparing to the unk LSTM, the retrained LSTM greatly improve the PPL of OOV words with some degradation in IV words. Our method further improves PPL in OOV words with a similar performance in IV words.

Dataset	OOV incorp.	#params	PPL
sms	unk	30M	117
	retrain	36M	121
	SE	4M	**114**
sms-30m	unk	51M	**185**
	retrain	60M	193
	SE	4M	**185**

Table 2. Perplexity Comparison Between Different OOV Incorporation Methods[5]

Dataset	OOV incorp.	PPL	
		IV	OOV
sms	unk	**109**	4809646
	retrain	116	836125
	SE	110	**548383**
sms-30m	unk	**175**	10747568
	retrain	185	2764168
	SE	179	**1023620**

Table 3. Perplexity Break Down on IV and OOV Words

4.2. Fast Vocabulary Updating in ASR

In automatic speech recognition (ASR) systems, backoff n-gram model is used as language model for generating lattice, from which n-best lists are generated. Neural network language models can then be used to rescore the n-best lists to get better performance. Usually, the n-gram and neural network language models sharing the same vocabulary. Hence when the vocabulary is updated, both n-gram and neural network language models need to be retrained. Comparing to neural network language models, the training time of n-gram language models can be neglected.

This experiment is divided into two phases. In the first phase, LSTM language model and the LSTM language model with structured embeddings (SE) are trained with the small vocabulary $\mathcal{V}_s$, denoted as unk LSTM and LSTM with SE respectively. An n-gram language model also trained with $\mathcal{V}_s$ is used to generate n-best lists. Then we do n-best lists rescore with the unk LSTM. In the second phase, the vocabulary $\mathcal{V}_s$ is extended the larger vocabulary $\mathcal{V}_f$. Since vocabulary is changed, we need to retrain the unk LSTM and n-gram model. But the vocabulary of LSTM with SE is rebuilt **without retraining**. After that, both retrained LSTM and LSTM with SE are used to rescore the n-best lists generated by the new n-gram model.

OOV incorp.	CER		
	ALL	IVS	OOVS
unk	11.76	10.88	39.80
retrain	11.38	10.91	26.20
SE	**11.18**	**10.74**	**25.07**

Table 4. Character Error Rate (CER, %) Comparison and Break Down on In-Vocabulary Sentences (IVS) and Out-of-Vocabulary Sentences (OOVS)

The experimental results are shown in table 4. Benefiting from the extension of vocabulary, the retrained LSTM gets an absolute 0.38% CER improvement over all sentences. The proposed LSTM model with structured embeddings (LSTM with SE) achieves the best performance. To investigate on what kind of sentences can get most gain from the proposed model, we split the rescore sentences into two categories depend on whether all words appear in $\mathcal{V}_s$ or not, referred as in-vocabulary sentences (IVS) and out-of-vocabulary sentences (OOVS) respectively. As shown in table 4, the unk LSTM trained with $\mathcal{V}_s$ has a much higher CER on out-of-vocabulary sentences, because the n-gram built by $\mathcal{V}_s$ can not generate those OOV words. By enlarging vocabulary, the retrained LSTM gets a significant improvement of CER on out-of-vocabulary sentences. Comparing to the retrained LSTM, the proposed model outperforms CER on both IV and OOV sentences. Moreover, the improvement of CER on OOV sentences (1.13% absolutely) is remarkably higher than the improvement of CER on IV sentences (0.13% absolutely), which means the LSTM with SE have a better ability to model OOV words. Note that by utilizing the proposed LSTM language model with structured word embeddings, we can save the time of model retraining in the traditional approach and achieve a better performance.

5. CONCLUSION AND FEATURE WORK

In this work, we propose a structured word embeddings which can be deployed in both input and output layer of LSTM language model. Utilizing syntactic-level and morphological-level parameters sharing, our model has the ability to handle out-of-vocabulary (OOV) words without retraining. The evaluation is done on a Chinese short message service dataset. The proposed model achieve PPL improvement on OOV words. LSTM language model with structured embeddings is further integrated into ASR systems for fast vocabulary updating. Without retraining, it has been shown that our model outperforms the retrained LSTM language model and the improvement on CER is most gained from out-of-vocabulary sentences.

For future work, the single part-of-speech (POS) tag for each word has its limitations: 1) words may have different POS tags in the different context, 2) except syntactic information, domain knowledge can also be used for word category. Therefore, a more robust multi-category classification way will be investigated in the future.

[5]We use the standard softmax as output layer for LSTM baselines, because it usually outperforms the factorized softmax on PPL evaluation.

6. REFERENCES

[1] Yoshua Bengio, Réjean Ducharme, Pascal Vincent, and Christian Jauvin, "A neural probabilistic language model," *Journal of machine learning research*, vol. 3, no. Feb, pp. 1137–1155, 2003.

[2] Tomas Mikolov, Martin Karafiát, Lukas Burget, Jan Cernockỳ, and Sanjeev Khudanpur, "Recurrent neural network based language model.," in *Interspeech*, 2010, vol. 2, p. 3.

[3] Sepp Hochreiter and Jürgen Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[4] Martin Sundermeyer, Ralf Schlüter, and Hermann Ney, "Lstm neural networks for language modeling," in *Thirteenth Annual Conference of the International Speech Communication Association*, 2012.

[5] Jerome R Bellegarda, "Statistical language model adaptation: review and perspectives," *Speech communication*, vol. 42, no. 1, pp. 93–108, 2004.

[6] Wengong Jin, Tianxing He, Yanmin Qian, and Kai Yu, "Paragraph vector based topic model for language model adaptation," in *INTERSPEECH*, 2015.

[7] Welly Naptali, *Study on n-gram language models for topic and out-of-vocabulary words*, Ph.D. thesis, Toyohashi University of Technology, 2011.

[8] Florian Gallwitz, Elmar Noth, and Heinrich Niemann, "A category based approach for recognition of out-of-vocabulary words," in *Spoken Language, 1996. ICSLP 96. Proceedings., Fourth International Conference on*. IEEE, 1996, vol. 1, pp. 228–231.

[9] Welly Naptali, Masatoshi Tsuchiya, and Seiichi Nakagawa, "Class-based n-gram language model for new words using out-of-vocabulary to in-vocabulary similarity," *IEICE TRANSACTIONS on Information and Systems*, vol. 95, no. 9, pp. 2308–2317, 2012.

[10] Thomas Müller and Hinrich Schütze, "Improved modeling of out-of-vocabulary words using morphological classes," in *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies: short papers-Volume 2*. Association for Computational Linguistics, 2011, pp. 524–528.

[11] Mittul Singh, Clayton Greenberg, Youssef Oualil, and Dietrich Klakow, "Sub-word similarity based search for embeddings: Inducing rare-word embeddings for word similarity tasks and language modelling," in *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*, 2016, pp. 2061–2070.

[12] Tomáš Mikolov, Ilya Sutskever, Anoop Deoras, Hai-Son Le, Stefan Kombrink, and Jan Cernocky, "Subword language modeling with neural networks," *preprint (http://www. fit. vutbr. cz/imikolov/rnnlm/char. pdf)*, 2012.

[13] Carolina Parada, Mark Dredze, Abhinav Sethy, and Ariya Rastrow, "Learning sub-word units for open vocabulary speech recognition," in *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies-Volume 1*. Association for Computational Linguistics, 2011, pp. 712–721.

[14] M Ali Basha Shaik, Amr El-Desoky Mousa, Ralf Schlüter, and Hermann Ney, "Hybrid language models using mixed types of sub-lexical units for open vocabulary german lvcsr," in *Twelfth Annual Conference of the International Speech Communication Association*, 2011.

[15] Shiliang Zhang, Hui Jiang, Mingbin Xu, Junfeng Hou, and Li-Rong Dai, "The fixed-size ordinally-forgetting encoding method for neural network language models.," in *ACL*, 2015, pp. 495–500.

[16] Tomáš Mikolov, Stefan Kombrink, Lukáš Burget, Jan Černockỳ, and Sanjeev Khudanpur, "Extensions of recurrent neural network language model," in *Acoustics, Speech and Signal Processing (ICASSP), 2011 IEEE International Conference on*. IEEE, 2011, pp. 5528–5531.

[17] Wikipedia, "Part of speech — wikipedia, the free encyclopedia," 2017, [Online; accessed 23-October-2017].

[18] Thang Luong, Richard Socher, and Christopher D Manning, "Better word representations with recursive neural networks for morphology.," in *CoNLL*, 2013, pp. 104–113.

[19] Yoon Kim, Yacine Jernite, David Sontag, and Alexander M Rush, "Character-aware neural language models.," in *AAAI*, 2016, pp. 2741–2749.

[20] Ryan Cotterell, Hinrich Schütze, and Jason Eisner, "Morphological smoothing and extrapolation of word embeddings," in *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, 2016, vol. 1, pp. 1651–1660.

[21] Tianxing He, Xu Xiang, Yanmin Qian, and Kai Yu, "Recurrent neural network language model with structured word embeddings for speech recognition," in *Acoustics, Speech and Signal Processing (ICASSP), 2015 IEEE International Conference on*. IEEE, 2015, pp. 5396–5400.