ADDRESSING THE POLYSEMY PROBLEM IN LANGUAGE MODELING WITH ATTENTIONAL MULTI-SENSE EMBEDDINGS

Rao Ma, Lesheng Jin, Qi Liu, Lu Chen, Kai Yu

SpeechLab, Department of Computer Science and Engineering
MoE Key Lab of Artificial Intelligence
Shanghai Jiao Tong University, Shanghai, China
{rm1031, king18817550378, liuq901, chenlusz, kai.yu}@sjtu.edu.cn

ABSTRACT

Neural network language models have gained considerable popularity due to their promising performance. Distributed word embeddings are utilized to represent semantic information. However, each word is associated with a single vector in the embedding layer, disabling the model from capturing the meanings of polysemous words. In this work, we address this problem by assigning multiple fine-grained sense embeddings to each word in the embedding layers. The proposed model discriminates among different senses of a word with attention mechanism in an unsupervised manner. Experiments demonstrate the benefits of our approach in language modeling and ASR rescoring. Investigations are also made on standard word similarity tasks. The results indicate that our proposed method is efficient in modeling polysemy and therefore obtains better word representations.

***Index Terms*—** language modeling, multi-sense embeddings, polysemy, attention models, distributed representation

1. INTRODUCTION

Neural network language models (NNLMs) have contributed towards a great amount of progress in automatic speech recognition (ASR) and natural language processing (NLP). Compared to backoff N-gram LMs [1, 2], NNLMs utilize compact representations for both words and contexts and generalize better to unseen data [3]. Recurrent neural network (RNN) LMs [4] have another strength that all of the predecessor words are taken into account to estimate probabilities. Due to the exploding or vanishing gradient problem of RNN [5], long short-term memory (LSTM) is proposed to overcome the error back-flow problems [6, 7].

In general, NNLMs consist of three parts: input embedding layer, hidden layer, and output embedding layer. Each embedding layer maps words to dense real-valued vectors, which are also called "distributed representations" [8]. Despite the success of word embeddings in capturing semantic properties, they are unable to deal with polysemy by nature. Nevertheless, polysemy is a common phenomenon in natural languages, especially for frequent words [9]. For instance, in the sentence "No one could believe how much *produce* our garden could *produce*", the word *produce* refers to two different senses as noun and verb. However, each word is associated with a single vector, ignoring the possible lexical ambiguity among different senses. Moreover, the embeddings of the polysemous words would be trained to approximate the average of its different semantic meanings, leading to other critical problems. According to the triangle inequality, $d(x, y) \leq d(x, z) + d(y, z)$ with distance measure d. Therefore, word pairs that are synonymous with different meanings of the same word will be pulled towards each other mistakenly in the vector space [10]. For instance, the distance of *grain* and *create* would be no more than the sum of the distances $d(\textit{grain}, \textit{produce})$ and $d(\textit{create}, \textit{produce})$.

Several works have tried to address the problem of learning multi-sense word embeddings. The task could be decomposed into the word sense disambiguation (WSD) step that determines word senses in the training corpus and the embedding learning step that updates specific sense embeddings. Some works adopt a two-stage approach that first conducts WSD with a pre-trained model and then performs embedding learning. [11, 12] cluster the contexts that a word appears to relabel word types and retrain sense embeddings. Recent work of [13] computes the average contextual representations of all words based on SemCor. These methods are either time-consuming or rely on external knowledge bases. Other works jointly perform WSD and embedding learning in the Skip-gram model [14, 15, 16]. However, in most work, the context words are not disambiguated along with the central word.

In this work, we develop a simple yet effective language model that is able to capture ambiguous senses for polysemous words. The model parameters are updated in a fully unsupervised fashion, not limited by the lack of large sense-

This study was supported by the National Key Research and Development Program of China (Grant No.2017YFB1002102), the Major Program of National Social Science Foundation of China (No.18ZDA293) and the Major Program of Science and Technology Commission of Shanghai Municipality (STCSM) 17JC1404104.

annotated corpus. Our model is trained on text-only data and has the ability to learn WSD. Within the output layer, each word is assigned with multiple fine-grained embeddings representative of different senses. We adopt the attention mechanism [17, 18] to calculate the weighted sum of sense specific word embeddings depending on the context. In addition, we could feed the disambiguated embedding to the model input and achieve further performance gain. We give both qualitative and quantitative analysis in the experiments to demonstrate the effectiveness of our method.

2. LANGUAGE MODEL WITH ATTENTIONAL MULTI-SENSE EMBEDDINGS

2.1. LSTM Language Model

Given a sequence of words $(w_1, w_2, \cdots, w_T)$, its joint probability could be decomposed with the chain rule:

$$P(w_1, w_2 \cdots, w_T) = P(w_1) \prod_{t=1}^{T-1} P(w_{t+1}|w_1, w_2, \cdots, w_t), \tag{1}$$

where w_1 and w_T are special symbols to denote the start and the end of the sentence. Therefore, $P(w_1) = 1$.

Let V denote the vocabulary of words. Assume the embedding size and the hidden size of the model to be d. The input embedding layer $\mathbf{W}_{in} \in \mathbb{R}^{|V| \times d}$ maps each word w_t into a d-dimensional embedding vector $\mathbf{x}_t$. Given $\mathbf{h}_{t-1}, \mathbf{c}_{t-1}, \mathbf{x}_t$ as input, the LSTM transformation computes hidden state $\mathbf{h}_t$ and cell state $\mathbf{c}_t$ at each time step with the formula:

$$\begin{bmatrix} \mathbf{i}_t \\ \mathbf{o}_t \\ \mathbf{f}_t \\ \mathbf{g}_t \end{bmatrix} = \begin{bmatrix} \sigma \\ \sigma \\ \sigma \\ \tanh \end{bmatrix} \left(\mathbf{W}^\top \begin{bmatrix} \mathbf{x}_t \\ \mathbf{h}_{t-1} \end{bmatrix} + \mathbf{b} \right) \tag{2}$$
$$\mathbf{c}_t = \mathbf{f}_t \odot \mathbf{c}_{t-1} + \mathbf{i}_t \odot \mathbf{g}_t$$
$$\mathbf{h}_t = \mathbf{o}_t \odot \tanh(\mathbf{c}_t),$$

where $\mathbf{W}$ is the parameter matrix and $\mathbf{b}$ is the bias.

The output layer is composed of an embedding matrix $\mathbf{W}_{out} \in \mathbb{R}^{|V| \times d}$ and a bias vector $\mathbf{b}_{out} \in \mathbb{R}^{|V|}$. Let $\mathbf{e}_w$ and b_w denote the output embedding and the bias entry of w for $w \in V$. The probability of $P(w|w_1, w_2, \cdots, w_t)$ in Equation (1) could be approximated as $P_\theta(w|\mathbf{h}_{t-1}, \mathbf{c}_{t-1}, \mathbf{x}_t)$,

$$P_\theta(w|\mathbf{h}_{t-1}, \mathbf{c}_{t-1}, \mathbf{x}_t) = \frac{\exp(\mathbf{h}_t^\top \mathbf{e}_w + b_w)}{\sum_{w' \in V} \exp(\mathbf{h}_t^\top \mathbf{e}_{w'} + b_{w'})}. \tag{3}$$

Parameters of the language model θ are optimized by minimizing the cross entropy loss between the predicted probability distribution and the ground truth word w_{t+1},

$$\mathcal{L}_{CE} = \sum_{t=1}^{T-1} -\log(P_\theta(w_{t+1}|\mathbf{h}_{t-1}, \mathbf{c}_{t-1}, \mathbf{x}_t)). \tag{4}$$

Previous work has shown that sharing weights between the input embedding $\mathbf{W}_{in}$ and the output projection matrix $\mathbf{W}_{out}$ in the language model leads to better performance [19]. Weight tying not only reduces the total number of model parameters but also spares the model from learning a one-to-one correspondence between the input and output embeddings. In a tied LSTMLM, we have $\mathbf{W}_{in} = \mathbf{W}_{out}$.

2.2. Structured Attentional Multi-Sense Embeddings

Note that in Equation (2) and Equation (3), each word is associated with a single embedding vector, limiting neural networks from making estimations based on different word senses. In this work, we propose a language model that learns multi-sense word embeddings. For the untied model, we assign $N (N > 1)$ sense embeddings to each word in the output layer and leave the input layer unchanged. Therefore, the output embedding matrix becomes $\mathbf{W}'_{out} \in \mathbb{R}^{N \times |V| \times d}$. In the tied model, we have $\mathbf{W}'_{in} = \mathbf{W}'_{out}$ additionally.

To efficiently train the multi-sense embeddings, we incorporate the attention mechanism to compute the disambiguated word embedding. At each time step, the proposed model automatically searches for sense embeddings that are relevant to the given context for each word, as shown in Figure 1. This process can also be considered as the network performing word sense discrimination based on context representations.

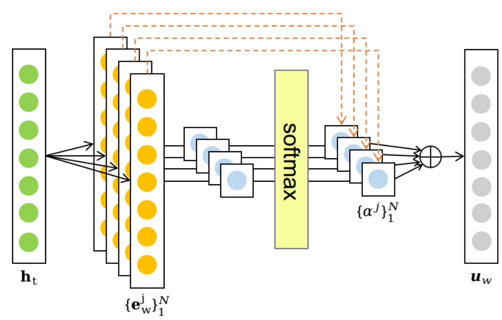


Fig. 1. We feed the hidden state $\mathbf{h}_t$ as the query vector to compute the disambiguated embedding $\mathbf{u}_w$ for word w.

Assume $\mathbf{e}_w^1, \mathbf{e}_w^2, \ldots, \mathbf{e}_w^N$ to be the output multi-sense embeddings for word w. The disambiguated embedding $\mathbf{u}_w$ is calculated as a weighted sum of these sense embeddings,

$$\mathbf{u}_w = \sum_{j=1}^{N} \alpha^j \mathbf{e}_w^j. \tag{5}$$

The weight α^j of each sense embedding $\mathbf{e}_w^j$ is computed by

$$\alpha^j = \frac{\exp(\mathbf{h}_t^\top \mathbf{e}_w^j)}{\sum_{j'=1}^{N} \exp(\mathbf{h}_t^\top \mathbf{e}_w^{j'})}. \tag{6}$$

The process to compute disambiguated embedding could run in parallel for all words in the vocabulary. Consequently, the predicted distribution $P_\theta(w|\mathbf{h}_{t-1}, \mathbf{c}_{t-1}, \mathbf{x}_t)$ is calculated as in Equation (3) with $\mathbf{e}_w$ replaced by $\mathbf{u}_w$ for $w \in V$.

Denote the disambiguated embedding for target word w_t as $\mathbf{u}_t$ at time step $t-1$, $\mathbf{u}_t$ can also serve as the next input embedding for word w_t. Therefore, the network can utilize word sense information in the input representations and model the sentence in a superior way. For instance, assume that the model has seen the word sequence "No one could believe how much ..." up to step $t-1$ and the next token for prediction is "produce". The computed $\mathbf{u}_t$ would be closer to the noun sense embedding of "produce" than its verb sense embedding. By inputting $\mathbf{u}_t$ in the next time step, the model could absorb the information that the input word is likely to be a noun and therefore make more accurate predictions in the following estimations. In this work, we set the input embedding at step t as $\mathbf{x}_t$ in the untied model and set the input embedding as $\mathbf{u}_t$ in the tied model.

3. EXPERIMENTS

3.1. Experimental Setup

To evaluate our algorithm, we train our proposed model and the baseline LSTMLM on three standard datasets. Penn TreeBank (PTB) contains one million words of 1989 Wall Street Journal material. Text8 is a collection of Wikipedia articles published by Google. Short Message Service (SMS) dataset is a Chinese conversation corpus. The detailed description of these datasets is listed in Table 1.

Corpus	Word Counts			OOV	Vocab
	Train	Valid	Test		
PTB	887K	70K	78K	6.09	10,000
text8	15M	1M	700K	3.55	44,475
SMS	2M	105K	16K	0.02	40,695

Table 1. Number of running words, OOV rate [%] on the test set and the vocabulary size for three datasets.

On all the datasets, we train language models with one hidden LSTM layer. In order to perform weight tying, we select the embedding size and the hidden size equal to 256. Sentences are not concatenated in the training and evaluation of PTB and SMS. The BPTT parameter is set to 35 for text8 corpus. We use the SGD optimizer with momentum for training. The initial learning rate is set to 2.0 and halves when the perplexity (PPL) on the validation set is not improved. The early stopping method is adopted to prevent overfitting.

In our experiments, we calculate perplexity results on all the datasets. Perplexities are given without crossing sentence boundaries for PTB and SMS, which is consistent with the ASR settings. We also evaluate the character error rate (CER) of our proposed model on SMS eval set (about 25 hours, 3K utterances) by performing 50-best hypotheses rescoring. On text8, we further investigate the quality of multi-sense embeddings on standard word similarity tasks.

3.2. Experiments on Language Modeling and Rescoring

We train both the tied and untied language models and show the perplexity results on PTB and text8 in Table 2. The first line denotes the baseline LSTMLM that associates each word with a single embedding, which could be seen as a special case of our model with $N = 1$. Since polysemous words with more than four senses are rare, we train the proposed models with $N = 2$ and $N = 3$. Results on untied models show that by assigning multi-sense embeddings in the output layer, the proposed models learn to discriminate among different word senses and thus outperform baseline models. By using the disambiguated embeddings as model input, additional performance gains could be observed on the tied models. Models associated with three sense embeddings produce best performance in most cases, whereas perplexity increases by a small margin when N increases from 2 to 3 in the untied model on text8. Assigning too many senses to each word might make the model difficult to optimize. According to statistics, about 80% of words in WordNet 3.0 are unambiguous, and less than 5% words have more than three senses [10]. Therefore, setting N to 2 would cover most polysemy in practice.

Model	PTB		text8	
	Untied	Tied	Untied	Tied
$N = 1$ (Baseline)	93.3	91.7	167.0	161.1
$N = 2$ (Ours)	92.1	89.9	**158.1**	157.3
$N = 3$ (Ours)	**91.7**	**87.2**	162.4	**155.7**

Table 2. Word level perplexity results on PTB and text8.

We also test our method on the n-best rescoring task. Here we only train the tied models which have shown superiority over the untied models. Table 3 shows the word level perplexity and CER results. Since a large number of Chinese words are polysemous, increasing the number of embeddings per word produces better perplexity results. Furthermore, 3.7% relative improvement in CER is obtained at $N = 2$.

Model	PPL	CER
$N = 1$ (Baseline)	94.5	10.7
$N = 2$ (Ours)	**91.9**	**10.3**
$N = 3$ (Ours)	92.3	10.5

Table 3. Perplexity and CER [%] results on SMS.

3.3. Experiments on Word Similarity

We evaluate the quality of trained embeddings on three standard word similarity datasets: the WordSim-353, the Mturk-771 and the RG-65 dataset. Each of the datasets contains a list of word pairs along with human-assigned similarity scores in the range from 1 to 10. All the models are trained on the text8 corpus to extract specific word embeddings.

For each dataset, we present the Spearman's rank correlation between human judgment score and model's sim-

Model	WordSim-353			Mturk-771			RG-65		
	In	Out	Tied	In	Out	Tied	In	Out	Tied
$N = 1$ (Baseline)	0.402	0.575	0.607	0.365	0.476	0.500	0.327	0.546	0.536
$N = 2$ (Ours)	0.424	0.595	**0.612**	**0.367**	**0.494**	**0.517**	0.351	**0.572**	0.555
$N = 3$ (Ours)	**0.461**	**0.604**	0.609	0.351	0.479	0.492	**0.366**	0.545	**0.565**

Table 4. Word similarity results for embeddings trained on the text8 corpus. Spearman's correlation ρ is reported for different embeddings: input / output embeddings of the untied model and the embeddings of the tied model.

ilarity score computed for each word pair w and w'. For the input embeddings, the similarity measure is defined as $sim(w, w') = d(\mathbf{e}_w, \mathbf{e}_{w'})$ with cosine distance d. Since it is less trivial to deal with the multi-sense word embeddings for output embeddings and tied embeddings, we adopt the *weighted* similarity measure proposed in [20],

$$sim\text{-}w(w, w') = \sum_{i=1}^{N}\sum_{j=1}^{N} s(w, i)s(w', j)d(\mathbf{e}_w^i, \mathbf{e}_{w'}^j)^\alpha, \quad (7)$$

where $s(w, i) = \frac{freq(\mathbf{e}_w^i)}{\sum_{k=1}^{N} freq(\mathbf{e}_w^k)}$. $freq(\mathbf{e}_w^i)$ denotes the frequency of appearance that $\mathbf{e}_w^i$ dominates other senses of w in the training data. The constant $\alpha(\alpha > 1)$ bias the similarity computation towards closer senses of the two words.

The experimental results for several pre-trained language models are listed in Table 4. Here we set $\alpha = 5$. In general, the output embedding is superior to the input embedding, and the tied embedding yields comparable performance to the output embedding. With $N = 2$, our model significantly outperforms the baseline model on all the datasets with different embedding types. The model that assigns $N = 3$ sense embeddings obtains better performance in some cases. It is worth mentioning that the quality of the input embeddings also improves along with output embeddings in the untied model. The results indicate that our method mitigates the meaning conflation deficiency problem and therefore enhances the representation ability of the entire model.

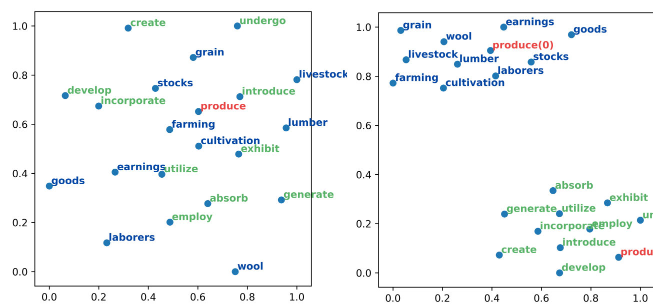
Produce		
$N = 1$	generate, create, utilize, incorporate, employ	
$N = 2$	cultivation, farming, laborers, goods, wool	
	generate, create, utilize, incorporate, exhibit	
Market		
$N = 1$	markets, enterprise, price, trade, commodity	
$N = 2$	shopping, dining, chinatown, mall, hotel	
	markets, trade, marketplace, price, enterprise	
Remains		
$N = 1$	continues, seems, appears, remained, survives	
$N = 2$	finds, graves, tombs, relics, skulls	
	continues, seems, remained, became, represents	
March		
$N = 1$	december, february, april, november, june	
$N = 2$	retreat, raid, surrender, mutiny, journey	
	february, december, april, june, november	

Table 5. Top-5 nearest neighbors computed by cosine similarity for the tied embeddings trained on text8.

3.4. Qualitative Analysis

The nearest neighbor results associated with several polysemous words are listed in Table 5. For the baseline model and the proposed model, we compute top five words that have the highest cosine similarity with word embedding or with each specific sense embedding of the given word. The results illustrate that different senses of ambiguous words are effectively captured by our model, whereas embeddings of the baseline model only capture the most commonly used meanings.

Figure 2 plots a subset of semantic space around the ambiguous word *produce*. Produce could be interpreted either as farm food or as bringing into existence. Words related to two senses are shown in different colors. Within the baseline model, *grain* and *create* that are both synonymous with *produce* are pulled close wrongly. Results obtained with our model are shown on the right, yielding better semantic space.



(a) $N = 1$ (Baseline) (b) $N = 2$ (Ours)

Fig. 2. Visualization of the nearest neighbors for *produce* in a two dimensional semantic space computed by t-SNE.

4. CONCLUSIONS AND FUTURE WORKS

In this work, we propose an extension to language model that learns multiple embeddings for each word in an unsupervised manner. The proposed model efficiently captures different word senses and outperforms traditional LSTMLM on language modeling, speech recognition and word similarity tasks. Furthermore, our approach could be easily adapted to other neuron network frameworks. We will further investigate the usage of multi-sense embeddings in other NLP tasks.

5. REFERENCES

[1] Stanley F Chen and Joshua Goodman, "An empirical study of smoothing techniques for language modeling," *Computer Speech & Language*, vol. 13, no. 4, pp. 359–394, 1999.

[2] 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, pp. 400–401, 1987.

[3] Ebru Arisoy, Tara N Sainath, Brian Kingsbury, and Bhuvana Ramabhadran, "Deep neural network language models," in *Proceedings of the NAACL-HLT 2012 Workshop: Will We Ever Really Replace the N-gram Model? On the Future of Language Modeling for HLT*. Association for Computational Linguistics, 2012, pp. 20–28.

[4] Tomáš Mikolov, Martin Karafiát, Lukáš Burget, Jan Černockỳ, and Sanjeev Khudanpur, "Recurrent neural network based language model," in *Eleventh annual conference of the international speech communication association*, 2010.

[5] Yoshua Bengio, Patrice Simard, Paolo Frasconi, et al., "Learning long-term dependencies with gradient descent is difficult," *IEEE transactions on neural networks*, vol. 5, no. 2, pp. 157–166, 1994.

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

[7] Martin Sundermeyer, Ralf Schlüter, and Hermann Ney, "Lstm neural networks for language modeling," in *Thirteenth annual conference of the international speech communication association*, 2012.

[8] David E Rumelhart, Geoffrey E Hinton, Ronald J Williams, et al., "Learning representations by back-propagating errors," *Cognitive modeling*, vol. 5, no. 3, pp. 1, 1988.

[9] George K Zipf, "Human behavior and the principle of least effort. an introduction to human ecology," 1950.

[10] Jose Camacho-Collados and Mohammad Taher Pilehvar, "From word to sense embeddings: A survey on vector representations of meaning," *Journal of Artificial Intelligence Research*, vol. 63, pp. 743–788, 2018.

[11] Joseph Reisinger and Raymond J Mooney, "Multi-prototype vector-space models of word meaning," in *Proceedings of the 2010 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. Association for Computational Linguistics, 2010, pp. 109–117.

[12] Eric H Huang, Richard Socher, Christopher D Manning, and Andrew Y Ng, "Improving word representations via global context and multiple word prototypes," in *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics*. Association for Computational Linguistics, 2012, pp. 873–882.

[13] Matthew Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer, "Deep contextualized word representations," in *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. Association for Computational Linguistics, 2018, pp. 2227–2237.

[14] Arvind Neelakantan, Jeevan Shankar, Alexandre Passos, and Andrew McCallum, "Efficient non-parametric estimation of multiple embeddings per word in vector space," in *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing*, 2014, pp. 1059–1069.

[15] Shobhit Jain, Sravan Babu Bodapati, Ramesh Nallapati, and Animashree Anandkumar, "Multi sense embeddings from topic models," in *Proceedings of the 3rd International Conference on Natural Language and Speech Processing*, 2019, pp. 34–41.

[16] Jiwei Li and Dan Jurafsky, "Do multi-sense embeddings improve natural language understanding?," in *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, 2015, pp. 1722–1732.

[17] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio, "Neural machine translation by jointly learning to align and translate," *arXiv preprint arXiv:1409.0473*, 2014.

[18] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Lukasz Kaiser, and Illia Polosukhin, "Attention is all you need," in *Advances in neural information processing systems*, 2017, pp. 5998–6008.

[19] Ofir Press and Lior Wolf, "Using the output embedding to improve language models," in *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics*. Association for Computational Linguistics, 2017, pp. 157–163.

[20] Ignacio Iacobacci, Mohammad Taher Pilehvar, and Roberto Navigli, "Sensembed: Learning sense embeddings for word and relational similarity," in *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing*. Association for Computational Linguistics, 2015, pp. 95–105.