Sense-Aware Neural Models for Pun Location in Texts

Yitao Cai and **Yin Li** and **Xiaojun Wan**
Institute of Computer Science and Technology, Peking University
The MOE Key Laboratory of Computational Linguistics, Peking University
{caiyitao,xyz1305121,wanxiaojun}@pku.edu.cn

Abstract

A homographic pun is a form of wordplay in which one signifier (usually a word) suggests two or more meanings by exploiting polysemy for an intended humorous or rhetorical effect. In this paper, we focus on the task of pun location, which aims to identify the pun word in a given short text. We propose a sense-aware neural model to address this challenging task. Our model first obtains several WSD results for the text, and then leverages a bidirectional LSTM network to model each sequence of word senses. The outputs at each time step for different LSTM networks are then concatenated for prediction. Evaluation results on the benchmark SemEval 2017 dataset demonstrate the efficacy of our proposed model.

1 Introduction

There exists a class of language constructs known as puns in natural language utterances and texts, and the speaker or writer intends for a certain word or other lexical item to be interpreted as simultaneously carrying two or more separate meanings. Though puns are an important feature in many discourse types, they have attracted relatively little attention in the area of natural language processing.

A pun is a form of wordplay in which a word suggests two or more meanings by exploiting polysemy, homonymy, or phonological similarity to another word, for an intended humorous or rhetorical effect (Miller et al., 2017). Puns where the two meanings share the same pronunciation are known as homographic puns, which are the focus of this study. For example, the following punning joke exploits contrasting meanings of the word "interest" and it is a homographic pun.

I used to be a banker but I lost interest.

Since the pun word plays the key role in forming a pun, it is very important and meaningful to identify the pun word in a given text. The task of identifying the pun word is known as pun location, which is defined in SemEval 2017 Task 7[1]. In order to address this special task, various approaches have been attempted, including rule based approach (Vechtomova, 2017), knowledge-based approach (Indurthi and Oota, 2017; Xiu et al., 2017) and supervised approach (Pramanick and Das, 2017; Mikhalkova and Karyakin, 2017). However, these approaches do not achieve good results, and the best F1 score for homographic pun location is just 0.6631, which is achieved by the Idiom Savant system with a knowledge based approach (Doogan et al., 2017). The results demonstrate that pun location is a very challenging task.

In order to address this challenging task and improve the state-of-the-art results, we propose a sense-aware neural model in this study. Our model first obtains several WSD (Word Sense Disambiguation) results for the text, and leverages a bidirectional LSTM network to model each sequence of word senses. The outputs at each time step for different LSTM networks are then concatenated for pun word prediction. Evaluation results of cross-validation on the benchmark SemEval 2017 dataset demonstrate the efficacy of our proposed model.

The contributions of this paper are summarized as follows:

- We propose a novel sense-aware neural model to address the pun location task.
- Our proposed model outperforms several baseline neural models and achieves the state-of-the-art performance.

[1]http://alt.qcri.org/semeval2017/task7/

2 Related Work

Pun detection aims to determine whether a given short text contains a pun (Miller et al., 2017). Various methods have been proposed to address this task, including WSD based methods (Pedersen, 2017), PMI-based methods (Sevgili et al., 2017) supervised methods (Xiu et al., 2017; Indurthi and Oota, 2017; Pramanick and Das, 2017; Mikhalkova and Karyakin, 2017; Vadehra, 2017). More specifically, the bi-directional RNN has been used in (Indurthi and Oota, 2017), and vote-based ensemble classifier is used by (Vadehra, 2017).

Pun location is a more challenging task than pun detection, because it aims to find the actual pun word in the given text. Previous works find some clues about puns in the texts. For example, pun is more likely appeared towards the end of sentences (Pedersen, 2017; Miller and Turković, 2016). Many puns have a particularly strong associations with other words in the contexts (Sevgili et al., 2017). A variety of methods have been proposed to locate the pun words. For example, UWaterloo system constructs a rule-based pun locator that scores candidate words according to eleven simple heuristics (Vechtomova, 2017). BuzzSaw system attempts to locate the pun in a sentence by selecting the polysemous word with the two most dissimilar senses (Oele and Evang, 2017). Duluth system identifies the last word which changed senses between different word sense disambiguation results (Pedersen, 2017). Fermi system uses Bi-directional RNN to learn a classification model (Indurthi and Oota, 2017). Idiom Savant system uses n-grams features, and only content words including nouns, verbs, adverbs and adjectives are considered as candidate words (Doogan et al., 2017). Pun interpretation is considered a subsequent step for pun location, and it aims to annotate the two meanings of the given pun by reference to WordNet sense keys. In the work of (Miller and Gurevych, 2015), traditional language-agnostic WSD approaches are adapted to "disambiguate" puns, and rather to identify their double meanings.

Word Sense Disambiguation (WSD) is also related to our work. Some prior works compute overlaps of glosses between the target word and its context (Lesk, 1986). These approaches derive information from some lexicon thesauruses for WSD, including WordNet (Fellbaum, 1998) and BabelNet (Navigli and Ponzetto, 2012). Supervised models, including neural models, have been successfully applied to WSD (Yuan et al., 2016; Raganato et al., 2017).

3 Baseline Neural Model (BM)

The task of pun location needs to locate the exact pun word in each short text or sentence. We regard pun location as a word-level classification task, and attempt to train a model that can predict whether a word in a sentence is a pun or not. A word will be regarded as a pun word with high probability when it is a noun, verb, adjective or adverb, therefore, we only try to make prediction of one word when it has one of the four kinds of parts of speech tags.

Our baseline model for pun word prediction is similar to (Indurthi and Oota, 2017) and it adopts a bi-directional LSTM network to accomplish this task. The neural network architecture of the model is shown in Figure 1. The input to the network is the embeddings of words, and we use the pre-trained word embeddings by using word2vec (Mikolov et al., 2013) on the Wikipedia corpus whose size is over 11G. The hidden state of the forward LSTM and the hidden state of the backward LSTM are concatenated at each time step (word), and we get the concatenated hidden vectors for all words: $h_1, ..., h_n$. The vector h_i for each word having one of the four kinds of POS tags is then sent to a two-layer feed-forward neural network with $tanh$ as activation function, and the output is a real number o_i. We then use the sigmoid function σ on o_i to make prediction. Since in the experimental data there is only one pun word in each sentence, we will take the k-th word as pun word if and only if o_k is the largest number out of all $o_i, i = 1, ..., n$, and $\sigma(o_k) > 0.5$. We use the cross-entropy loss in this model.

4 Sense-Aware Neural Model (SAM)

The baseline neural model is built on the word level and the word senses can only be implicitly captured by the model. Moreover, a pun word usually has two senses in the sentence, while the baseline neural model cannot disambiguate them. In order to improve the prediction performance, we propose a sense-award neural model which is built on WSD results. Two or more WSD results are obtained by using different WSD algorithms or different configurations, and the WSD results may be different. The sequence of word senses corresponding to each WSD result is modeled by a

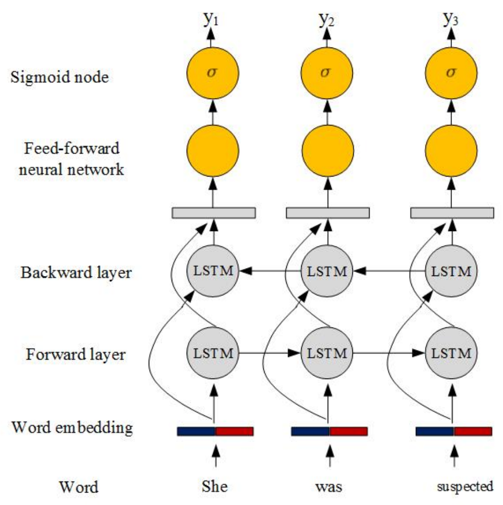

Figure 1: Baseline neural model with bidirectional LSTMs

bi-directional LSTM network and the outputs of different LSTM networks are then concatenated for prediction. In this way, the different senses of words can be well captured by our model.

Different from the Duluth system (Pedersen, 2017) which identifies the last word which changed senses between different runs of the WordNet::SenseRelate::AllWords disambiguation algorithm, we do not use the WordNet-based WSD results. Furthermore, we do not heuristically identify the pun word but propose a neural model to achieve this goal.

4.1 Word Sense Disambiguation and Sense Embedding

In order to obtain sense inventory and the sense embeddings for each word, we choose SenseGram (Pelevina et al., 2016). The SenseGram toolkit is available online[2], and it can take as an input the word embeddings and split different senses of the input words. For instance, the vector for the word "table" will be split into "table (data)" and "table (furniture)". SenseGram induces sense inventory from existing word embeddings via clustering of ego-networks of related words. In our work, the Wikipedia corpus is used to train word embeddings (together with contextual embeddings of words) by using word2vec and then the word embeddings are used by SenseGram for inducing sense inventory and sense embeddings. The word similarity graph used by SenseGram is built based on the similarity between word embeddings. Note that we do not use WordNet as sense inventory because the sense inventory is too fine-grained and many words are not included in WordNet.

Given each target word ω and its context words $C = \{c_1, ..., c_k\}$ in the sentence, we want to assign a sense vector to ω from the set of its sense vectors $S = \{s_1, ..., s_m\}$. We use two simple WSD methods for achieving this. The first WSD strategy is based on the sigmoid function. $\overline{c}_c$ is the mean of the contextual embeddings of words in C and the sense embedding of ω is chosen as

$$s^* = \arg\max_{s_i \in S} \frac{1}{1 + e^{-\overline{c}_c \cdot s_i}} \tag{1}$$

Let $\overline{c}_w$ be the mean of the word embeddings of words in C, which is different from $\overline{c}_c$. The second disambiguation strategy is based on the cosine similarity function.

$$s^* = \arg\max_{s_i \in S} \frac{\overline{c}_w \cdot s_i}{\|\overline{c}_w\| \cdot \|s_i\|} \tag{2}$$

For each WSD strategy, we can set different window sizes of 3 and 50 (the maximum sentence length in the corpus) as different configurations. By obtaining different WSD results, we expect to well capture the characteristics of homographic puns.

4.2 Neural Model Details

The proposed sense-aware neural model differs from the baseline neural model in that it models multiple sequences of word senses corresponding to different WSD results. In other words, the sense-aware model works on the sense level, but the baseline model works on the word level.

The architecture of the sense-aware model is illustrated in Figure 2, which contains several bi-directional LSTM networks. For each WSD result, the sequence of sense embeddings is taken as input for a bi-directional LSTM network. Assuming we have K WSD results, the outputs $h_i^j (j = 1, ..., K)$ by K different bi-directional LSTM networks for the same i-th word (i.e., the i-th time step) are then concatenated into one vector, and the vector is sent to a two-layer feed-forward neural network and a sigmoid function for prediction. The loss function is the same as that of the baseline model. Our sense-aware model can be considered as applying the baseline model on different WSD results and then combining the outputs for prediction.

[2]https://github.com/tudarmstadt-lt/sensegram

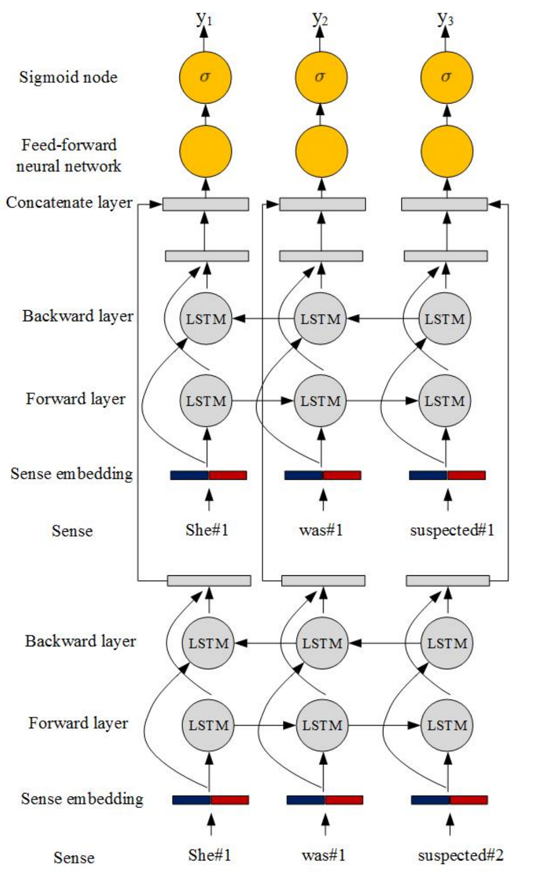

Figure 2: Sense-aware neural model with bidirectional LSTMs

4.3 Model Training

We use stochastic gradient descent to train the neural models with a batch size of 225 and 512 hidden units, and the learning rate is 0.0001 and the size of embeddings and hidden vectors is 300.

5 Experiments

5.1 Dataset

We use the benchmark dataset from SemEval 2017 Task 7. There are a total of 2250 sentences in the dataset, 1607 of which contain a pun. For Pun Location, we only use sentences with pun words for evaluation, the same as the task setting. Since no training data is provided, so we test our models with 10-fold cross validation. We combine the output results on each test set of all 10 folds and then calculate precision, recall and F-score on the combined set. Thus the scores are comparable to the official results based on the whole test set.

5.2 Word Sense Disambiguation

As is mentioned in section 4.1, we can obtain four WSD results with two different strategies and two different window sizes. We make three groups based on four WSD results: Group 1 (G1) contains two WSD results with the first WSD strategy with two window sizes of 3 and 50; Group 2 (G2) contains two WSD results with the second WSD strategy with two window sizes; Group 3 (G3) contains all results in Group 1 and Group 2.

5.3 Evaluation Results

We compared our proposed SAM model (w/ three groups of WSD results) with the baseline model BM. We also apply the bi-directional LSTM model on the sequence of senses for each single WSD result and thus get BiLSTM-WSD1 through BiLSTM-WSD4. Moreover, we apply SVM and CRF models with various features (e.g., ngram, POS tagging, word location, word similarity, etc.) on this task.

Table 1 shows the results. In the table, we also present the results of the best participating system Idiom Savant, and two official baselines (last word and max. polysemy). We can see that the baseline BM model does not perform well, while the CRF model performs very well. The results of BiLSTM-WSD1 through BiLSTM-WSD4 are much better than the BM model, which indicates that the sense-level prediction is more suitable than the word-level prediction. Our proposed SAM model with different groups of WSD results can further improve the performance, because different WSD results may provide complementary information for pun location. The SAM model with G1 performs the best, even outperforming the SAM model with more WSD results (G3), which indicates the necessity for choosing proper WSD results.

6 Conclusion

In this work, we apply the neural network models to the pun location task. We proposed a novel sense-aware neural model to leveraging multiple WSD results. Evaluation results on the benchmark SemEval 2017 dataset demonstrate the efficacy of our proposed model. In future work, we will test with more advanced WSD algorithms and try to address the pun interpretation task.

Method	Precision	Recall	F-Score
SVM	0.717	0.717	0.717
CRF	0.759	**0.759**	0.759
BM	0.751	0.617	0.677
BiLSTM-WSD1	0.751	0.742	0.746
BiLSTM-WSD2	0.754	0.745	0.750
BiLSTM-WSD3	0.735	0.726	0.730
BiLSTM-WSD4	0.742	0.732	0.737
SAM-G1	0.815	0.747	**0.780**
SAM-G2	**0.828**	0.731	0.776
SAM-G3	0.804	0.745	0.773
Idiom Savant	0.664	0.663	0.663
last word	0.470	0.470	0.470
max. polysemy	0.180	0.180	0.180

Table 1: Comparison results.

Acknowledgment

This work was supported by National Natural Science Foundation of China (61772036, 61331011) and Key Laboratory of Science, Technology and Standard in Press Industry (Key Laboratory of Intelligent Press Media Technology). We thank the anonymous reviewers for their helpful comments. Xiaojun Wan is the corresponding author.

References

Samuel Doogan, Aniruddha Ghosh, Hanyang Chen, and Tony Veale. 2017. Idiom savant at semeval-2017 task 7: Detection and interpretation of english puns. In *Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017)*, pages 103–108.

Christiane Fellbaum. 1998. *WordNet: An Electronic Lexical Database.* Cambridge, MA: MIT Press, Cambridge, UK.

Vijayasaradhi Indurthi and Subba Reddy Oota. 2017. Fermi at semeval-2017 task 7: Detection and interpretation of homographic puns in english language. In *Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017)*, pages 457–460.

Michael Lesk. 1986. Automatic sense disambiguation using machine readable dictionaries:how to tell a pine cone from an ice cream cone. In *Acm Special Interest Group for Design of Communication*, pages 24–26.

Elena Mikhalkova and Yuri Karyakin. 2017. Punfields at semeval-2017 task 7: Employing roget's thesaurus in automatic pun recognition and interpretation. *arXiv preprint arXiv:1707.05479*.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space. *Computer Science*.

Tristan Miller and Iryna Gurevych. 2015. Automatic disambiguation of english puns. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, volume 1, pages 719–729.

Tristan Miller, Christian Hempelmann, and Iryna Gurevych. 2017. Semeval-2017 task 7: Detection and interpretation of english puns. In *Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017)*, pages 58–68.

Tristan Miller and Mladen Turković. 2016. Towards the automatic detection and identification of english puns. *The European Journal of Humour Research*, 4(1):59–75.

Roberto Navigli and Simone Paolo Ponzetto. 2012. Babelnet: The automatic construction, evaluation and application of a wide-coverage multilingual semantic network. *Artificial Intelligence*, 193(6):217–250.

Dieke Oele and Kilian Evang. 2017. Buzzsaw at semeval-2017 task 7: Global vs. local context for interpreting and locating homographic english puns with sense embeddings. In *Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017)*, pages 444–448.

Ted Pedersen. 2017. Duluth at semeval-2017 task 7 : Puns upon a midnight dreary, lexical semantics for the weak and weary. *CoRR*, abs/1704.08388.

Maria Pelevina, Nikolay Arefiev, Chris Biemann, and Alexander Panchenko. 2016. Making sense of word embeddings. In *The Workshop on Representation Learning for Nlp*, pages 174–183.

Aniket Pramanick and Dipankar Das. 2017. Ju-cse-nlp at semeval 2017 task 7: Employing rules to detect and interpret english puns. In *Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017)*, pages 432–435.

Alessandro Raganato, Claudio Delli Bovi, and Roberto Navigli. 2017. Neural sequence learning models for word sense disambiguation. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 1156–1167.

Özge Sevgili, Nima Ghotbi, and Selma Tekir. 2017. N-hance at semeval-2017 task 7: A computational approach using word association for puns. In *Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017)*, pages 436–439.

Ankit Vadehra. 2017. Uwav at semeval-2017 task 7: Automated feature-based system for locating puns. In *Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017)*, pages 449–452.

Olga Vechtomova. 2017. Uwaterloo at semeval-2017 task 7: Locating the pun using syntactic characteristics and corpus-based metrics. In *Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017)*, pages 421–425.

Yuhuan Xiu, Man Lan, and Yuanbin Wu. 2017. Ecnu at semeval-2017 task 7: Using supervised and unsupervised methods to detect and locate english puns. In *Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017)*, pages 453–456.

Dayu Yuan, Julian Richardson, Ryan Doherty, Colin Evans, and Eric Altendorf. 2016. Semi-supervised word sense disambiguation with neural models. *arXiv preprint arXiv:1603.07012*.