ECNU at SemEval-2017 Task 7: Using Supervised and Unsupervised Methods to Detect and Locate English Puns

Yuhuan Xiu[1], **Man Lan**[1,2*], **Yuanbin Wu**[1,2]
[1]Department of Computer Science and Technology,
East China Normal University, Shanghai, P.R.China
[2]Shanghai Key Laboratory of Multidimensional Information Processing
51164500032@stu.ecnu.edu.cn
mlan,ybwu@cs.ecnu.edu.cn

Abstract

This paper describes our submissions to task 7 in SemEval 2017, i.e., Detection and Interpretation of English Puns. We participated in the first two subtasks, which are to detect and locate English puns respectively. For subtask 1, we presented a supervised system to determine whether or not a sentence contains a pun using similarity features calculated on sense vectors or cluster center vectors. For subtask 2, we established an unsupervised system to locate the pun by scoring each word in the sentence and we assumed that the word with the smallest score is the pun.

1 Introduction

A pun is a form of wordplay in which one signifier (e.g., a word or phrase) suggests two or more meanings by exploiting polysemy, or phonological similarity to another signifier, for an intended humorous or rhetorical effect. The study of puns can be seen as a respectable research topic in traditional linguistics and the cognitive sciences.

Semeval 2017 task 7(Miller et al., 2017) contains three subtasks, i.e., pun detection, pun location, and pun interpretation. And we participated in the first two subtasks. The detection and location of English puns are to determine whether or not a sentence contains a pun and which word is a pun respectively, which differ from traditional word sense disambiguation (WSD). WSD is to determine an exact meaning of the target word in the given context. However, WSD algorithms could provide the lexical-semantic understanding for pun detection and location. And we adopted a knowledge-based WSD algorithm to obtain possible *senses*[1] for each word in the sentence.

There are two types of puns: some are homographic puns and the others are heterographic puns. A homographic pun exploits distinct meanings of the same written word, and a heterographic pun exploits distinct meanings of the similar but not exactly the same spoken word. The organizers provided two test datasets about homographic puns and heterographic puns respectively for each subtask. Since they did not provide official training datasets, we collected our own positive samples(each sentence contains a pun) from the *Pun of the Day website*[2], which conclude 60 homographic puns and 60 heterographic puns. Besides, we also assembled a raw dataset of 120 negative samples(sentences that do not contain puns) from the Internet. Then, we combined 120 negative samples with 60 homographic puns or 60 heterographic puns into homographic or heterographic training dataset. Then, we did the same data preprocessing for both training and test datasets. Firstly, we performed part-of-speech(POS) tagging using *Stanford CoreNLP tools*(Manning et al., 2014). Secondly, we removed the stop words in sentences. The words produced after this series of processing are denoted as *target words* for each sentence.

Since there are two different puns, we adopted different methods for them to detect and locate English puns. For homographic puns, we calculated the semantic similarity between sense vectors of each target word in the sentence to obtain a vector representation of a sentence and score each target word in the sentence, and for heterographic puns, we computed the semantic similarity between cluster center vectors of each sentence for the same purpose.

[1]The sense is the gloss provided by WordNet

[2]http://www.punoftheday.com/

Proceedings of the 11th International Workshop on Semantic Evaluations (SemEval-2017), pages 453–456,
Vancouver, Canada, August 3 - 4, 2017.

2 Homographic Puns Detection and Location

To detect and locate homographic puns, we performed exploratory analysis on training dataset. We found that the degree of semantic similarity between two meanings of a pun is supposed not to be high, where the semantic similarity between meanings is measured by calculating the distance between sense vectors.

(Miller and Turković, 2016) makes a case for research into computational methods for detection of puns in running context. Inspired by their work, for subtask 1, we presented a supervised system using similarity features which are calculated on sense vectors of each target word to create each target word vector, and to obtain the vector representation of the sentence. For subtask 2, we located the pun by scoring each target word in the sentence.

2.1 Pun Detection

This subtask is to determine whether or not a given sentence contains a pun. To address this subtask, we performed the process that consists of the following steps:

- For each *target word* in the sentence, we adopted Simplified Lesk algorithm(Kilgarriff and Rosenzweig, 2000) with respect to its POS to select the possible *senses*. Simplified Lesk disambiguates a word by examining the *definitions*[3] and selecting the single sense with the highest *overlap score*[4]. In our case, we selected the possible senses which *overlap scores* are higher than or equal to the second highest *overlap score*.

- In order to obtain the sense vector for each sense, we used 300-dimensional word vectors which are pre-trained Google word vectors downloaded from Internet[5] to represent each word in the *sense* and the simple *min*, *max*, *average* pooling strategies were adopted to concatenate sense vector representations with dimensionality of 900.

- For each *target word* in the sentence, we calculated the similarity between its sense vectors using six kernel functions, i.e., cosine similarity, manhattan distance, euclidean distance, pearsonr distance, Spearman's rho distance and sigmoid function. Note that, the instruction of sigmoid function is : Firstly, compute the dot product of two vectors to obtain the value of K. Secondly, update K by K=tanh(K/D+1), where D is the dimension of the vector. Finally, we denote K as the similarity score calculated by sigmoid function.

- For each *target word* in the sentence, we combined each minimum score calculated by each kernel function into target word vector(6-dimensional).

- In order to obtain the sentence vector(18-dimensional), we simply adopted the *min*, *max*, *mean* pooling strategies on all target words in the sentence.

we explored two supervised machine learning algorithms to build the classifiers: AdaBoost(AB) and RandomForest(RF) both implemented in *scikit-learn tools*[6].

2.2 Pun Loaction

This subtask is to decide which word in the sentence is the pun. We scored each target word by averaging the elements of its target word vector described in section 2.1. Finally, we assumed that the word with the smallest score is a homographic pun.

3 Heterographic Puns Detection and Location

A heterographic pun corresponds to another word with similar spoken but distinct meaning. That is different from a homographic pun, which exploits distinct meanings of exactly one word. Therefore, we adopted different methods for heterographic puns. Through an artificial analysis on heterographic puns, we found that the original meaning of the heterographic pun differs greatly from the meanings of other words in the sentence. Therefore, we clustered all words in training and test datasets in order to cluster words with high degree of semantic similarity into the same cluster. Firstly, we used pre-trained Google word vectors

[3]In our implementation, the definitions are formed by concatenating the synonyms, gloss, and example sentences provided by WordNet

[4]Overlap score is the number of words in common with the context

[5]https://code.google.com/archive/p/word2vec

[6]http://scikit-learn.org/stable

to represent each word. Secondly, we clustered those word vectors into 100 clusters using k-means(k=100) clustering algorithm. Finally, we obtained a cluster center vector(300-dimensional) for each cluster by averaging the word vectors belonging to this cluster. Moreover, the semantic similarity between words in the sentence is measured by calculating the distance between cluster center vectors.

For subtask 1, we presented a supervised system using similarity features which are calculated on cluster center vectors to represent a sentence. For subtask 2, we selectively scored target word in the sentence.

3.1 Pun Detection

To address this subtask, we used the following two steps to implement our approach.

- We clustered the target words in a sentence into several clusters. If the number of clusters of all taget words in a sentence is exactly one, we assumed that this sentence does not contain a pun. If not, we calculated the similarity between those cluster center vectors using the six kernel functions adopted in section 2.1.
- We took the min, max, and mean scores calculated by each kernel function as a vector representation(18-dimensional) of the sentence.

we also explored two same classification algorithms as for homographic puns detection.

3.2 Pun Loaction

To locate the pun in the sentence, we split the location process into three steps.

- We calculated the similarity scores between a cluster center and other cluster centers using the six kernel functions to find the outlier cluster, then we computed the average value of those similarity scores as a score for each cluster center. We chose the cluster center with the smallest score as the outlier cluster.
- If there is only one word in the outlier cluster, we selected this word as a pun. If not, for each candidate word, we calculated the similarity scores between the *top-sense vector*[7] of it and every cluster center except the outlier one using the six kernel functions. Finally, We calculated the average value of these similarity scores as a score for each candidate word.
- We supposed that the word with the smallest score is a heterographic pun.

Particularly, if the number of clusters of all target words in a sentence is less than three, we could not find the outlier cluster. Therefore we calculated the similarity scores between *top-sense vectors* of target words in the sentence. We scored each target word by averaging all the similarity scores that are relevant to that target word.

4 Experiments

4.1 Datasets

Although organizers did not provide the training datasets, we collected our own training datasets. Table 1 shows the statistics of the datasets we used in our experiments.

Pun	Dataset	Positive	Negative	Total
homographic	training	60(33%)	120(67%)	180
	test	1,607(71%)	643(29%)	2,250
heterographic	training	60(33%)	120(67%)	180
	test	1,271(71%)	509(29%)	1,780

Table 1: Statistics of datasets in training and test data. The number in brackets are the percentages of different classes in each dataset.

4.2 Evaluation Metrics

For both subtask 1 and 2, the three widely-used evaluation measures *precision*(*P*), *recall*(*R*) and F_1 are adopted. Moreover, for subtask 1, *accuracy*(*Acc*) is also included and for subtask 2, *coverage*(*C*) is used. *Coverage* is defined as the ratio of sentence for which a location assignment was attempted.

4.3 Experiment on Training Data For Subtask 1

Table 2 and 3 show the results of different algorithms of subtask 1 on homographic and heterographic training datasets respectively. The 5-fold cross validation is performed for system development. From Table 2 and Table 3, we find that AdaBoost outperforms RandomForest algorithm and the ensemble method performed best on homographic pun. Therefore we chose the

[7]Top-sense is the definition of top-scoring synset returned by Simplified Lesk algorithm and we used the method described in Section 2.1 to obtain top-sense vector

ensemble classifier for homographic pun and the AdaBoost algorithm for heterographic pun.

Method	Algorithm	P	R	Acc	F_1
Single	AB	0.7945	0.6836	0.6667	0.7349
	RF	0.7814	0.6733	0.6346	0.7233
Ensemble	AB+RF	0.8013	0.6955	0.6835	0.7747

Table 2: Results of subtask 1 on homographic training dataset.

Method	Algorithm	P	R	Acc	F_1
Single	AB	0.8401	0.8233	0.8833	0.8316
	RF	0.7107	0.4833	0.77.22	0.57.59
Ensemble	AB+RF	0.8880	0.4500	0.8056	0.5973

Table 3: Results of subtask 1 on heterographic training dataset.

4.4 Results and Discussion on Test Data

Table 4 and Table 5 show the results of our systems and the top-ranked systems provided by organizers for subtask 1 and subtask 2 respectively. Compared with the top ranked systems, there is much room for improvement in our work. The reason for the poor performance may be that the constructing method of sense vectors is simple and straightforward, which neglects the word sequence and the sentence structure of the sense. We find that detecting puns at the sentence level is more effective than locating puns at the word level, and our systems performed better on heterographic puns.

Pun	System(rank)	P	R	Acc	F_1
Homographic	ECNU(6)	0.7127	0.6474	0.5628	0.6785
	PunFields(1)	0.8091	0.7785	0.7044	0.7900
	Duluth(2)	0.7832	0.8724	0.7364	0.8254
	UWAV(3)	0.7806	0.6067	0.5973	0.6828
Heterographic	ECNU(2)	0.7807	0.6761	0.6333	0.7247
	Idiom Savant(1)	0.8704	0.8190	0.7837	0.8439
	N-Hance(3)	0.7725	0.9300	0.7545	0.8440

Table 4: Performance of our systems and the top-ranked(ranked by P) systems. The numbers in the brackets are the official ranking

5 Conclusion

In this paper, we presented systems to detect and locate a pun in the sentence on both homographic and heterographic puns datasets. For homographic puns, we calculated the semantic similarity between sense vectors of each target word in the sentence to obtain its sentence vector and score each target word. And for heterographic puns, we computed the semantic similarity between cluster center vectors of each sentence for the same purpose. In the future, we will explore the requisite problem of pun interpretation, where the objection is to determine two senses of the pun.

Pun	System(rank)	C	P	R	F_1
Homographic	ECNU(8)	1.0000	0.3373	0.3373	0.3373
	Idiom Savant(1)	0.9988	0.6636	0.6627	0.6631
	UWaterloo(2)	0.9994	0.6526	0.6521	0.6523
	Fermi(3)	1.0000	0.5215	0.5215	0.5215
Heterographic	ECNU(4)	1.0000	0.5681	0.5681	0.5681
	UWaterloo(1)	0.9976	0.7973	0.7954	0.7964
	Idiom Savant(2)	1.0000	0.6845	0.6845	0.6845
	N-Hance(3)	0.9882	0.6592	0.6515	0.6553

Table 5: Performance of our systems and the top-ranked(ranked by P) systems. The numbers in the brackets are the official ranking

Acknowledgements

This research is supported by grants from Science and Technology Commission of Shanghai Municipality(14DZ2260800 and 15ZR1410700), Shanghai Collaborative Innovation Center of Trustworthy Software for Internet of Things (ZF1213) and NSFC (61402175).

References

Adam Kilgarriff and Joseph Rosenzweig. 2000. Framework and results for english senseval. *Computers and the Humanities* 34(1):15–48.

Christopher D Manning, Mihai Surdeanu, John Bauer, Jenny Rose Finkel, Steven Bethard, and David McClosky. 2014. The stanford corenlp natural language processing toolkit. In *ACL (System Demonstrations)*. pages 55–60.

Tristan Miller, Christian F. 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)*.

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.