N-Hance at SemEval-2017 Task 7: A Computational Approach using Word Association for Puns

Özge Sevgili
İzmir Institute of Technology,
Computer Engineering, Urla,
İzmir, Turkey
ozgesevgili@iyte.edu.tr

Nima Ghotbi
İzmir Institute of Technology,
Computer Engineering, Urla,
İzmir, Turkey
nimaghotbi
@iyte.edu.tr

Selma Tekir
İzmir Institute of Technology,
Computer Engineering, Urla,
İzmir, Turkey
selmatekir@iyte.edu.tr

Abstract

This paper presents a system developed for SemEval-2017 Task 7, Detection and Interpretation of English Puns consisting of three subtasks; pun detection, pun location, and pun interpretation, respectively. The system stands on recognizing a distinctive word which has a high association with the pun in the given sentence. The intended humorous meaning of pun is identified through the use of this word. Our official results confirm the potential of this approach.

1 Introduction

Word Sense Disambiguation (WSD) (Navigli, 2009) is the task of determining the sense of a word in a specific context computationally. In the general case, a polysemous word's a single, specific sense is meant in a given context. However, a type of wordplay called pun suggests two or more meanings, by exploiting multiple meanings of words, or of similar-sounding words. Understanding the linguistic realization of puns and automatically detecting and disambiguating them are important for computational linguistics. Machine learning-based approaches are too expensive and impractical for this area as these humour constructs employ lexical-semantic anomalies and they are hard to be found in regular training sets. Thus, knowledge-based and unsupervised methods are prevalent. There are three tasks in the detection and interpretation of English puns: pun detection-the given context contains a pun or not, pun identification-location of the pun word, and pun disambiguation-identifying the two meanings referred by the pun in the given context.

Current approaches to pun interpretation rely on Lesk (1986), which considers the highest overlap between the context and gloss in order to return the target sense (two senses) for the pun. Miller and Gurevych (2015) performs automatic pun disambiguation in a comprehensive experimental setup for the first time using three Lesk variants along with Random and MFS (Most Frequent Sense) baselines. The general conclusion is that pun disambiguation results are poorer when compared with traditional WSD and traditional WSD must be extended with pun-specific features to increase accuracy in this area.

Our consideration for the task of detection and interpretation of English puns is mainly based on the assumption that pun containing contexts include a distinctive word that can be paired with the pun. This word has a central role in the wordplay such that the intended humorous meaning of pun is in connection with this word. This claim implies that this word's association with a pun is considerably greater than the other words' association scores with the pun. As the threshold to be used here is determined using all the sentences (either pun or not), the unsupervised nature of our method is preserved. In the computation of word association scores; we used the information-theoretic measure, Pointwise Mutual Information (PMI) (Church and Hanks, 1990). Our experiments support this claim computationally. The results both in homographic and heterographic puns are promising in detecting whether the given context is a pun or not.

In the remainder of this paper, we describe our system including the components for pun detection, pun location, and pun interpretation. In Section 3 we present our performance results. Finally, we summarize our findings and give comment on possible future extensions.

2 System Description

First of all, we performed exploratory analysis on both homographic and heterographic trial datasets. We observed that sentences containing a pun have a distinctive word that has a high semantic/phonetic association with the pun. In accordance with this observation, for a sentence, all the pairwise word associations should be calculated to determine the most correlated word pair. If an element of this most correlated word pair has a second sense/spelling, it's an evidence of being a pun. To illustrate; in the following sentence, "banker-interest" word pair has the highest correlation and the word "interest" of this pair has a second sense, making it a good candidate for the pun.

I used to be a banker but I lost interest.

We used PMI to measure the association between words. PMI distinguishes the relevant word pairs (e.g. banker-interest) from the irrelevant ones (e.g. used-interest) because word co-occurrence frequencies are normalized by the individual word frequencies, as seen in 1.

$$pmi(w1, w2) = log_2 \frac{p(w1, w2)}{p(w1)p(w2)} \quad (1)$$

To calculate the PMI scores, we used a subset of Wikipedia data[1]. Because pun words seldom appear in Wikipedia, we added test datasets to guarantee words co-occur at least once and thus the system is able to compute PMI scores for each word pair. We calculated PMI scores using nltk library (Bird et al., 2009)[2] using bigram collocation with a window size of 20.

2.1 Pun Detection

This subtask requires deciding whether a given sentence contains a pun or not. In order to accomplish this, we followed a process that consists of the following steps:

- Converting each sentence into tokens.
- Stopword removal.
- Generating word pairs preserving word order in the sentence (leaving out the reverse of each ordered pair).
- Calculating PMI scores for each pair and sorting the list of scores for each sentence.

[1]https://dumps.wikimedia.org/enwiki/latest/enwiki-latest-pages-articles.xml.bz2

[2]http://www.nltk.org/howto/collocations.html

For instance, if the input of the system is the sentence below:

They threw a party for the inventor of the toaster. And he was toasted.

The following sorted PMI scores of word pairs are given as output (Table 1).

Pair of words	PMI score
toaster, toasted	11.6896
threw, toasted	7.9549
threw, toaster	7.8618
inventor, toasted	7.4851
inventor, toaster	7.3920
threw, inventor	3.6572
party, toasted	3.1461
party, toaster	3.0530
threw, party	1.6402
party, inventor	-1.1516

Table 1: Sorted PMI scores for each pair in the sentence.

In the realization of this subtask, we ask whether the highest PMI score is distinctively higher than the others. In order to answer this, we need a global threshold value to be used for the whole set of sentences (either pun or not). To determine this threshold value, we used the interquartile range (IQR) of the set of sorted PMI scores of every sentence. IQR is preferred because it is able to eliminate outliers.

To sum up, we had an IQR value for each sentence and we took their median as the threshold value because median and IQR are consistent with each other. Our threshold value for homographic sentences was 2.458 and for heterographic sentences was 2.940. Thus, if the difference between the highest PMI score and the succeeding one in a sentence was higher than the threshold value, the sentence was marked as containing a pun.

2.2 Pun Location

The main aim of the subtask is to find the location of pun in pun containing sentences. Here we assumed that the second element of the pair with the highest PMI score in a sentence is the pun using the observation that pun word usually is located at the end of the sentence.

2.3 Pun Interpretation

In this subtask, we are given the location of the pun word and we are required to find the two relevant senses of it. We provided a solution only for the homographic dataset for this subtask.

In order to identify the first relevant sense in the context, we used lesk method which predicts the answer on the basis of the overlap between the given sentence and the dictionary entries. It is appropriate because we need a context-based prediction. As an implementation detail, we used pywsd (Tan, 2014)[3] library that includes simple lesk algorithm implementation.

As for the second sense of the pun word, we again rely on our assumption that the given sentence has a target word that can be paired with the pun by its highest correlation. In order to determine this target word, we get help from pairwise associations using PMI scores. Then, the second sense of the pun word can be predicted by finding the highest similar sense to this target word. This final step requires to find all senses of the pun word and choose the highest similar sense among them, respectively. Here, we represented each individual sense of pun by its synonym. We used wordnet (Miller, 1995)[4] dictionary to extract all senses and synonyms for a given word.

To measure similarity, we used word2vec (Mikolov et al., 2013)[5] method in which words are represented by local vectors to be used in a computational manner. The similarity of two words are calculated by taking cosine of vectors of those words. We used the gensim library (Řehůřek and Sojka, 2010)[6] to calculate word2vec for each word. To feed word2vec, we used, again, Wikipedia data. Our vector size was set as 128-dimensional and our window size was 10.

To sum up, we selected pun's best word pair with respect to the PMI score. Then, we took every sense of the pun and retrieved the synonymous words for every sense. After that, we calculated cosine similarity between each sense and pun's word pair. As a result, the sense with the highest similarity is recognized as the second suggested meaning of pun. To illustrate, in the following sentence, the word pair "room - admitted" is the most correlated with the PMI score of 0.6130:

"There's room for one more," Tom admitted.

All senses of the pun word, namely "admitted", and their synonymous words were taken. Then, using their word vectors, cosine similarities between the synonyms and the word, "room" were calculated, as seen in Table 2.

word and synonym	cosine similarity
room, admit	-0.0113
room, accommodate	0.2292
room, accept	-0.0972

Table 2: Cosine similarity values between synonymous words of the senses of pun and its highest correlated word in the sentence.

According to the cosine similarity values, as the second sense of pun "accommodate" ("accommodate.v.04") which means "have room for; hold without crowding" was chosen. As we said before, the first sense was identified using the simple lesk algorithm. For the above example sentence, the algorithm returned the result "admit.v.08" which means "serve as a means of entrance".

3 Evaluation Results

Table 3 presents the official scores[7] of our system. As results show, our main success comes from the task of pun detection. We have 0.7553 precision for homographic and 0.7725 for heterographic datasets and around 0.93 recall value for both of them. The relatively lower precision scores we get can be attributed to the general feature of unsupervised methods when compared with that of their supervised counterparts. In our implementation, there are minor errors while extracting pairs or calculating PMI scores. In those inconvenience cases, our system predicts randomly, for exactly 22 sentences in homographic dataset and 28 sentences in heterographic one which may cause a wrong decision.

For pun detection and pun location, our results for heterographic sentences are better than the homographic ones. Actually, it is because our pairwise association is more appropriate for heterographic sentences in which two words are distinctively correlated in one spelling than the others.

For pun interpretation, our results are lower than the other subtasks. To begin with; in identifying

[3]https://github.com/alvations/pywsd
[4]http://www.nltk.org/howto/wordnet.html
[5]https://code.google.com/p/word2vec/
[6]https://radimrehurek.com/gensim/models/word2vec.html
[7]system submitted after the official end of the evaluation

dataset	subtask	precision	recall	F1
homographic	`pun detection`	0.7553	0.9334	0.8350
	`pun location`	0.4269	0.4250	0.4259
	`pun interpretation`	0.0204	0.0200	0.0202
heterographic	`pun detection`	0.7725	0.9300	0.8440
	`pun location`	0.6592	0.6515	0.6553

Table 3: The performances of our system for three subtasks in each dataset.

the first sense of pun, we are limited by the performance of simple Lesk algorithm, $50 - 70\%$, as Lesk (1986) explained. Therefore, another algorithm for WSD may enable us to reach better results. As for disambiguating the second sense of the pun word, we utilize synonymous words for each sense of pun, however, there is sometimes none or limited synonymous words for an input sense. This may result in reaching a wrong decision.

4 Conclusion

We have described the system submitted to the SemEval-2017 Task 7, Detection and Interpretation of English Puns. For participated system descriptions and their highlighted ideas, please refer to the task description paper by Miller et al. (2017). Our system uses word association scores based on PMI to determine a target word that can be paired with the pun. The substantially high association score of this target word with pun is used as an indicator that the given sentence is a pun. Moreover, this word can be exploited to disambiguate the second meaning of a pun. The evaluation results show that the idea of this word pair association could reasonably accomplish the goal of the subtasks especially the task of pun detection.

This work suggests an interesting further direction to use PMI scores in conjunction with local vector similarities to identify pun-specific features in WSD.

References

Steven Bird, Ewan Klein, and Edward Loper. 2009. *Natural Language Processing with Python*. O'Reilly Media.

Kenneth Ward Church and Patrick Hanks. 1990. Word association norms, mutual information, and lexicography. *Comput. Linguist.* 16(1):22–29. http://dl.acm.org/citation.cfm?id=89086.89095.

Michael Lesk. 1986. Automatic sense disambiguation using machine readable dictionaries: How to tell a pine cone from an ice cream cone. In *Proceedings of the 5th Annual International Conference on Systems Documentation*. ACM, New York, NY, USA, SIGDOC '86, pages 24–26. https://doi.org/10.1145/318723.318728.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space. *CoRR* abs/1301.3781. http://arxiv.org/abs/1301.3781.

George A. Miller. 1995. Wordnet: A lexical database for english. *Commun. ACM* 38(11):39–41. https://doi.org/10.1145/219717.219748.

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 (ACL-IJCNLP 2015)*.

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)*. Association for Computational Linguistics, Vancouver, Canada, pages 58–68. http://www.aclweb.org/anthology/S17-2005.

Roberto Navigli. 2009. Word sense disambiguation: A survey. *ACM Comput. Surv.* 41(2):10:1–10:69. https://doi.org/10.1145/1459352.1459355.

Radim Řehůřek and Petr Sojka. 2010. Software Framework for Topic Modelling with Large Corpora. In *Proceedings of the LREC 2010 Workshop on New Challenges for NLP Frameworks*. ELRA, Valletta, Malta, pages 45–50. `http://is.muni.cz/publication/884893/en`.

Liling Tan. 2014. Pywsd: Python implementations of word sense disambiguation (wsd) technologies [software]. https://github.com/alvations/pywsd.