JU_CSE_NLP at SemEval 2017 Task 7: Employing Rules to Detect and Interpret English Puns

Dipankar Das and **Aniket Pramanick**
Department of Computer Science and Engineering
Jadavpur University, Kolkata, India
dipankar.dipnil2005@gmail.com
abcdefgh.ABCD65@gmail.com

Abstract

The problem of detection and interpretation of English puns falls under the area of word sense disambiguation in natural language processing, which deals with the sense of a word used in a sentence from the readers' perspective. We have tried to design a system to identify puns from a sentence by developing a cyclic dependency–based system which is implemented based on some rules which are actually statistical inferences taken from a set of random data collected from the Web.

1 Introduction

Extensive research has been done in the field of modelling and detecting puns (Hempelmann, 2003; Miller and Gurevych, 2015). The context or the sense depends largely on the perspective and knowledge of the reader about a particular language. For example, in the sentence, 'I was a banker but I lost *interest*,' the word in italics conveys two different meanings or 'senses' in the sentence. So, the word 'interest' could be called a pun. A *pun* is the exploitation of the various meanings of a word or words with phonetic similarity but different meanings.

Our system is a rule-based implementation of a dependency network and a hidden Markov model.

2 Dataset and Preprocessing

In the present shared task (Miller et al., 2017), participants are provided with a trial dataset and a test dataset. No training data was supplied due to the large cardinality of such words or contexts in general. In case of the subtask on pun detection (Subtask 1), the test data was subdivided into two sets: a *homographic* set containing 2250 contexts, and a *heterographic* set containing 1780 contexts. On the other hand, in case of Subtask 2, another set of data was provided and that set was also subdivided into homographic and heterographic sets.

For training purposes, or rather to analyze the contexts statistically, a dataset was collected from random sources, mostly form *Project Gutenberg* and used in our present experiments. This dataset contains 413 sentences and is not subdivided into homographic and heterographic subsets.

The given data containing English contexts is preprocessed and each word of a sentence is tagged with its part of speech using NLTK, an open-source package for NLP written in Python. For example, the sentence, 'I was a banker but I lost interest.' is tagged using the Stanford NLP parser as follows:

> [('I', 'PRP'), ('was', 'VBD'), ('a', 'DT'), ('banker', 'NN'), ('but', 'CC'), ('I', 'PRP'), ('lost', 'VBD'), ('interest', 'NN'), ('.', '.')]

We also generate the parse tree for the sentence, which looks as in Figure 1. Using such parse trees, the clauses are identified and used for our further tasks.

3 System Framework

We have used a hidden Markov model (Ghahramani, 2001) and incorporated cyclic dependency (Toutanova et al., 2003) in order to detect points of pun occurrences in English sentences. The probability has been calculated with respect to each word being pun in a sentence. To calculate the probability, the parts of speech of the words immediately surrounding the target word are considered and the probability is increased accordingly.

3.1 Features

To train the system, 413 sentences, each containing a pun, has been analyzed. The probability of stop

Proceedings of the 11th International Workshop on Semantic Evaluations (SemEval-2017), pages 432–435,
Vancouver, Canada, August 3 - 4, 2017.

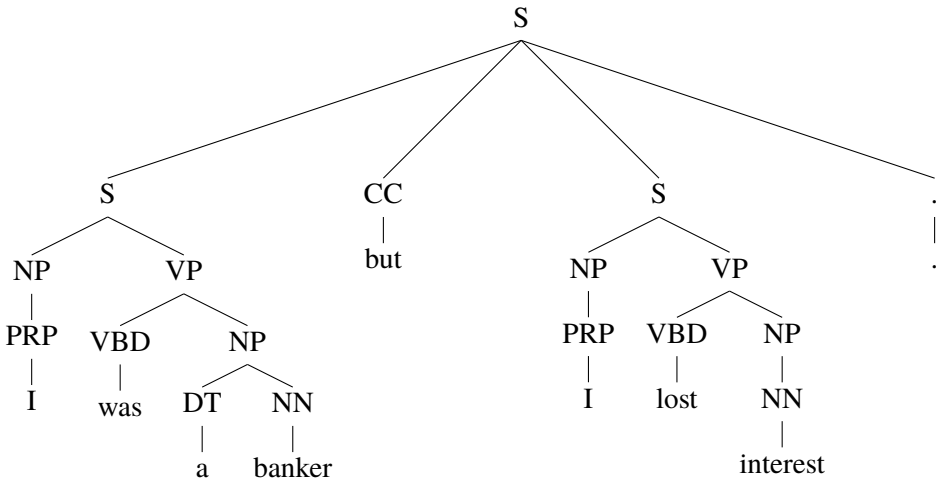

Figure 1: Parse tree for 'I was a banker but I lost interest.'

words such as articles, prepositions, 'be'-verbs, conjunctions, and infinitives are assumed to be 0 as they cannot be a pun. The probability of each part of speech being a pun is calculated from the training data and is given in Table 1.

Part of Speech	**Probability**
Noun	0.1538
Adjective	0.1111
Verb	0.0806
Others	1/(Sentence Length)

Table 1: Pun probability by part of speech.

Since in the task it is explicitly mentioned that each sentence will contain at most one pun, *sentence length* in this context has been defined as the number of words in the sentence. Each word of a smaller sentence will have higher probability of being pun.

The concept of clauses has been used in order to modify the probability. The clauses are extracted from the parse tree generated by the Stanford Parser. And clauses are classified as follows:

Type 1: Any clause that doesn't contain any other clauses falls under this category. A word which belongs to this type of clause has a pretty low probability of being pun.

Type 2: Any clause that contains only Type 1 clauses falls under this category. A word which belongs to this type of clause has a medium probability of being pun.

Type 3: Any clause that contains Type 1, Type 2, or other Type 3 clauses falls under this category. A word which belongs to this type of clause has a very high probability of being pun.

Thus, the probability is increased according to the occurrence of the words within the aforementioned category of puns. Data collected from the trial set tells that a word that belongs to the Type 3 clause has 0.1 higher probability approximately than the words that belong to the clauses of other types.

Furthermore, it has been observed that if a word ends with '-ing' or '-ed', the probability of the word being a pun increases by 0.01.

It has also been observed that this probability depends on the parts of speech of the words adjacent to the target words. This is given in Tables 2 and 3.

POS of previous word	**Increase Factor**
Infinitive	0.02
Noun	0.01
Punctuations	0.014

Table 2: Pun probability by POS of previous word.

POS of next word	**Increase Factor**
Infinitive	0.02
Noun	0.01
Preposition	0.01
Punctuations	0.014

Table 3: Pun probability by POS of next word.

If the word is situated at the end of the sentence then its probability is increased by 0.02. The probability assigned to stop words and punctuation is 0.

3.2 Rule Definitions

Since this system is a rule-based implementation, a few rules are defined in order to identify the puns. These are as follows:

Rule 1: The probability of stop words (i.e., articles, prepositions, infinitives, and pronouns) being puns is trivially 0. 'Be'-verbs can also never be a pun in any English sentence. For example, in the sentence, 'I am a good boy,' the stop words words 'a' and 'am' cannot be puns in any case.

Rule 2: The probability of a word being a pun increases if it is a noun. It has been seen from the set of sentences collected from the Web that out of 195 words which are nouns, 30 are puns. This is highest among all parts of speech. For instance, in the sentence 'She thought it was a real horse, but it was a phony,' the pun 'phony' is a noun.

Rule 3: If a word belongs to a higher-level clause, then its probability of being pun increases. That is, if a word belongs to a Type 3 clause, its probability of being pun will be higher than a word that belongs to a Type 2 or Type 1 clause.

Rule 4: The probability of a word being a pun depends on the parts of speech of the words before and after it in a wrap-around fashion. For example, in the sentence 'She thought it was a real horse , but it was a phony,' the word just before the word 'phony' is 'a', an article, and thus 'phony' will have a greater probability of being pun than the other words in the sentence.

Rule 5: A probability of being a pun is associated to every part of speech, which is furnished in Table 1.

Rule 6: It has been observed that if a word ends with '-ed' or '-ing', it will have greater chance of being a pun.

3.3 Inference from Rules

In our system, we have used rules to determine the probability of each word in a sentence being a pun. We have used the basic statistical formula of determining the probability for dependent events in this respect: $P(A|B) = P(A \bigcap B)/P(B)$ and $P(A \bigcup B) = P(A) + P(B) - P(A \bigcap B)$.

Using standard formulas related to probability we have calculated the cumulative probability obtained by the cumulative effect of all the rules. Since in the task it is given that a given sentence in the test data could contain at most one word that could be determined a pun, we have only considered simple cases. For a random sentence that could potentially contain unknown number of puns, complex rules must be developed and the dependency among the rules to be determined.

3.4 Result

Assigning probability to each of the word of the sentence by the rules defined in the previous sections gives a vector whose length equals the length of the sentence. For example the sentence 'I was a banker but I lost interest.' gives the vector [0.000, 0.201, 0.000, 0.274, 0.000, 0.000, 0.261, 0.314, 0.000]. We denote this vector the *pun vector*.

It has been observed that if the maximum value among the components of this pun vector is less than or equal to 0.25, then the sentence does not contain any pun. On the other hand, if the maximum value among the components of the vector exceeds the value 0.25, then the sentence must contain a pun and the word corresponding to the maximum-valued component can be declared as a pun.

Applying this concept to the test data provided by the organizers, we obtained the coverage (C), precision (P), accuracy (A), recall (R), and F-score (F_1) shown in Tables 4 and 5.

Type	P	A	R	F_1
Homo	0.7251	0.6884	0.9079	0.8063
Hetero	0.7369	0.7174	0.9402	0.8261

Table 4: Results on Subtask 1.

Type	P	C	R	F_1
Homo	0.3348	1.0000	0.3348	0.3348
Hetero	0.3792	1.0000	0.3792	0.3792

Table 5: Results on Subtask 2.

4 Conclusion

In this paper we have presented a system based on a dependency probabilistic model to detect and identify puns from a given English sentence. Undoubtedly, the problem will become far more complex if the sentence contains more than one pun and the task is to identify each of those words individually.

In the test dataset, various non-pun words have been identified as puns because not only the context between pun and a non-pun is very narrow but the structural discrimination between the pun and the non-pun word in a sentence sometimes becomes very difficult even considering all aspects of the sentence including meanings and parts of speech of individual words.

Thus identifying a pun is pretty complex task and we believe that this model could be extended to identify complex cases. Apart from this, solutions to this problem will have tremendous effect in emotions recognition and analysis.

References

Zoubin Ghahramani. 2001. An introduction to hidden Markov models and Bayesian networks. *International Journal of Pattern Recognition and Artificial Intelligence* 15(1):9–42. https://doi.org/10.1142/S0218001401000836.

Christian F. Hempelmann. 2003. *Paronomasic Puns: Target Recoverability Towards Automatic Generation*. Ph.D. thesis, Purdue University, West Lafayette, IN.

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

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)*.

Kristina Toutanova, Dan Klein, Christopher D. Manning, and Yoram Singer. 2003. Feature-rich part-of-speech tagging with a cyclic dependency network. In *Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology*. volume 1, pages 173–180. https://doi.org/10.3115/1073445.1073478.