UWAV at SemEval-2017 Task 7: Automated feature-based system for locating puns

Ankit Vadehra
University of Waterloo
avadehra@uwaterloo.ca

Abstract

In this paper we describe our system created for SemEval-2017 Task 7: Detection and Interpretation of English Puns(Miller et al., 2017). We tackle subtask 1, pun detection, by leveraging features selected from sentences to design a classifier that can disambiguate between the presence or absence of a pun. We address subtask 2, pun location, by utilizing a decision flow structure that uses presence or absence of certain features to decide the next action. The results obtained by our system are encouraging, considering the simplicity of the system. We consider this system as a precursor for deeper exploration on efficient feature selection for pun detection.

1 Introduction

Puns are ambiguous word pairs in language that play on the different meaning of the word (polysemy), or utilize similarly pronounced sounds (phonology) often for a humorous effect(Miller and Turković, 2016). They are widely used in written and spoken literature, intended as jokes. The task tries to detect whether a sentence is a pun or not (Subtask 1). If a pun is detected, we try to detect which word in the sentence was meant as a pun (Subtask2). For example, the sentence, *"I'd tell you a chemistry joke, but, I know I wouldn't get a reaction."* is a play on the word *reaction*, which is a common phrase. But, including it after *chemistry*, which deals with chemical reactions; gives it a humorous intent. Another example, the sentence, *"I used to meditate a lot, but now I only do it every now and zen."* plays on the similar pronunciation of the words zen and then. Puns often require knowledge about syntactic similarity between words and phrases like meditation and zen, chemistry and reaction, etc.

Recently, a definite trend is noticeable in developing a system that can automatically detect puns efficiently so that the humorous nature of the pun can be captured. Pun detection is of high importance for language modeling, machine translation and sentiment analysis task so that the actual meaning of terms can be understood. This allows effectively utilizing the intended meaning rather than just the words themselves.

The absence of any unified global word knowledge base makes detections of puns difficult since automatic selection of intended meaning is hard. There exist many Word Sense Disambiguation (WSD) approaches, but they are inefficient in capturing the word play(Miller and Gurevych, 2015). This makes pun detection a very interesting area to work on.

Due to the ambiguous nature of the words used, sentences containing puns are often wrongly classified. Due to an absence of a globally used pun dataset, all work in this area utilize self-accumulated and tagged datasets. This makes it difficult to compare the performance of different approaches.

We use the SemEval2017 Task7 dataset and design a classifier system that can detect the presence and location of puns. We explain the features selected and extracted that can efficiently detect puns in Section 2.1.1. In Section 2.2.1, we propose our decision flow based algorithm that utilizes features to locate the word in the sentence that was intended as the pun. We show and explain our results in Section 3 .

2 System Overview

We use this section to explain details of the system we designed for the task. The task was distributed in different subtasks and each subtask had two

Proceedings of the 11th International Workshop on Semantic Evaluations (SemEval-2017), pages 449–452,
Vancouver, Canada, August 3 - 4, 2017.

tasks for Homographic puns and Heterographic puns. Homographic puns play on the two distinct meanings of a word, whereas, Heterographic puns deal with the similar phonological pronunciation between words. For each subtask we design a single system, to tackle both Heterographic and Homographic puns, since in real life, we don't find a distinction between the two types. We try to present a system that could identify puns, irrespective of their type.

2.1 Subtask 1: Pun Detection

In this section we present the description of the classifier system used for detecting whether a sentence contains a pun or not.

2.1.1 Feature Set

For an efficient classifier we extract the following binary features from the sentence. The value for each feature is either 1 or 0, denoting the presence or absence of that feature in that sentence. We select the features that were generally found in puns.

• **Homophone:** Homophones are a set of words that have the same pronunciation but different meaning like "knew" and "new". We utilize two sources[1,2] from the Internet to create a list of homophones. This list is in no way exhaustive but covers the most frequent homophones.

• **Antonym:** Antonyms are pair of words that are opposite in meaning, like "good" and "bad". We utilize the WordNet (Miller, 1995) library to check whether the sentence contains any antonym pairs.

• **Idioms:** Idioms are group of words that are used together as an expression having a specific meaning. For example the phrase "blue moon" means something rare. We create a list of the most common idioms from the Englishclub website[3].

• **Homonym:** Homonyms are words that have more than one definitions. For example the word 'fine'. We create a list of the most common homonyms from Wikipedia[4] and Alphalink[5].

• **word2vec similarity:** We utilize the word2vec word representations(Mikolov et al., 2013) pre-trained on the 100 billion Google News words[6]. We construct a set of unique pair of words from the sentence and check the similarity score between them. If a score higher than the threshold (set to 0.30) is obtained for any pair, we assume that a word2vec similar word pair exists.

• **WordNet similarity:** We can utilize the WordNet (Miller, 1995) tree to calculate the distance between two words. Similar words are relatively closer and have high similarity score. If any pair of words has score higher than threshold (set to 0.30) we assume that a WordNet similar pair exists.

The threshold similarity value was set by manually calculating the similarity score for some similar words like *ketchup* and *mustard*.

2.1.2 Classification

Extracting the six binary features, described in the previous section, from our training dataset (explained in Section 3.1.1) we train three classifiers.

• **Support Vector Machine(SVM):** We use an SVM with a linear kernel, C=1, a squared-hinge loss function, and L2 loss penalty (Pedregosa et al., 2011).

• **Naive Bayes(NB):** We use Binomial variant with Laplace smoothing parameter = 1 (Pedregosa et al., 2011).

• **Logistic Regression(LR):** We use Logistic Regression with L2 loss penalty, and C=1 (Pedregosa et al., 2011).

The classifier gives an output of 1 or 0, denoting the presence or absence of a pun. We use the best-of-three approach, pooling the results of all three classifiers and selecting the highest occurring value (1 or 0).

2.2 Subtask 2: Pun Location

In this section we describe the decision flow based algorithm to expose the word in the sentence that represents the pun.

2.2.1 Algorithm Design

We present the decision flow algorithm used to detect the pun word in the sentence in Algorithm 1.

We start by utilizing the list of homophones and homonyms as described in Section 2.1.1. We utilize the word2vec pre-trained vectors (Mikolov et al., 2013) and the WordNet library (Miller,

[1] http://www.zyvra.org/lafarr/hom.htm

[2] http://www.singularis.ltd.uk/bifroest/misc/homophones-list.html

[3] https://www.englishclub.com/ref/Idioms/

[4] https://en.wikipedia.org/wiki/List_of_true_homonyms

[5] http://home.alphalink.com.au/~umbidas/homonym_main.htm

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

1995) to calculate similarity score using *calcScore* function.

Algorithm 1 Algorithm to detect location of pun word in sentence.

Require: hph = list of homophones; hom = list of homonym; wn = WordNet Library; w2v = word2vec pre-trained vectors; S = Sentence to test;

```
sHalf ← Second Half of words in S
sHalf ← Reverse words in sHalf
flag = 0
for word ∈ sHalf do
    if length(word) > 3 then
        if word ∈ (hph ∪ hom) then
            pun ← word
            flag = 1; break;
        end if
        if antonym(word) ∈ S then
            pun ← word
            flag = 1; break;
        end if
    end if
end for
if flag = 0 then
    prevScore = 0
    revS ← Reverse of words in S
    wP ← Pairs of words ∈ revS
    for p ∈ wP do                ▷ p = (word1, word2)
        score ← calcScore(word1, word2)
        if score > prevScore then
            prevScore ← score
            pun ← word1
        end if
    end for
end if
if prevScore = 0 then
    pun ← Last word of S
end if
```

We break the sentence into its constituent words, and check whether any word in the second half of the sentence is a homophone/homonym. If any such word is found, it is considered as the pun word. If no such match is found, we check whether any word in the second half has an antonym (Section 2.1.1) in the sentence. If an antonym pair is found, that word is considered as the pun.

If no match is found then we consider the unique pair combinations of words in the sentence and calculate the word similarity. The word from the back of the sentence that has the highest similarity score with any other word in the sentence is considered as the pun word. If all decisions fail, the last word in the sentence is considered as the pun word.

3 Experiment and Results

3.1 Subtask 1: Pun Detection

We extract features and design a classifier for Subtask 1.

3.1.1 Dataset

We created the dataset for the classifier by gathering puns from the punoftheday[7] website. The website consists of user aggregated puns of all types. A collection of 5316 puns were accumulated.

For non-pun sentences multiple sources were used. Sentences containing homophones and homonyms were extracted from WordNet (Miller, 1995). We also utilized the SemEval2012 - Task 6[8] training dataset by taking a single sentence from each dataset item. The BBC dataset (Greene and Cunningham, 2006) of news articles was also used[9]. We extracted the first sentence for each news article. Compiling all these sources gave us a set of 4848 non-pun sentences.

3.1.2 Result

We tabulate the results achieved in the SemEval 2017 Task 7 in Table 1.

Table 1: Official SemEval Results for Subtask1.

PunTask	Precision	Recall	Accuracy	F1
Homographic	0.78	0.61	0.60	0.68
Heterographic	0.79	0.62	0.61	0.69

3.1.3 Limitations

On deeper analysis of the training dataset we found that the final testing dataset contained items from the punoftheday[7] website too. This made multiple items from the test dataset present in our training dataset. In total, 1145 items from the Homographic task and 715 items from the Heterographic task were present in our training set.

Due to this unfortunate coincidence, we recreated our training set, eliminating all the items that

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

[8] https://www.cs.york.ac.uk/semeval-2012/task6.html

[9] http://mlg.ucd.ie/datasets/bbc.html

were present in the SemEval Test set. This reduced our dataset to 3456 puns and 4848 non-puns. Retesting on this dataset provided us with the following results as mentioned in Table 2:

Table 2: Recalculated Results for Subtask1.

PunTask	Precision	Recall	Accuracy	F1
Homographic	0.68	0.47	0.47	0.56
Heterographic	0.65	0.42	0.43	0.51

3.2 Subtask2: Pun Location

For this task we run the testing dataset through our algorithm proposed in Section 2.2.1. We submitted two runs for this subtask, using WordNet Path Similarity for one run and word2vec word similarity for the other.

3.2.1 Result

The results achieved using WordNet as a similarity metric are tabulated in Table 3, whereas Table 4 tabulates the results obtained using word2vec as a similarity measure.

Table 3: SemEval Subtask2 results using WordNet as similarity measure.

PunTask	Precision	Recall	F1
Homographic	0.315	0.315	0.315
Heterographic	0.357	0.357	0.357

Table 4: SemEval Subtask2 results using word2vec as similarity measure.

PunTask	Precision	Recall	F1
Homographic	0.341	0.341	0.341
Heterographic	0.428	0.428	0.428

4 Conclusion

Based on our submissions and the results for SemEval2017 Task7, we believe that efficient feature selection may be a feasible approach for automatic detection of sentences containing puns. We propose to integrate language models and word substitution in future work to perform deeper analysis on the dataset.

Even though our system was not the best one, we do believe that the simplicity of its design is a highly attractive feature. We selected features for our system that are generally found in puns. We found that some of the features like homophones resulted in poor performance for Subtask 2. Future work on efficient feature selection might allow us to get much higher results.

References

Derek Greene and Pádraig Cunningham. 2006. Practical solutions to the problem of diagonal dominance in kernel document clustering. In *Proc. 23rd International Conference on Machine learning (ICML'06)*. ACM Press, pages 377–384.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*. pages 3111–3119.

George A Miller. 1995. Wordnet: a lexical database for english. *Communications of the ACM* 38(11):39–41.

Tristan Miller and Iryna Gurevych. 2015. Automatic disambiguation of english puns. In *ACL (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)*.

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.

Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, et al. 2011. Scikit-learn: Machine learning in python. *Journal of Machine Learning Research* 12(Oct):2825–2830.