Fermi at SemEval-2017 Task 7: Detection and Interpretation of Homographic puns in English Language

Vijayasaradhi Indurthi
SIEL, IIIT Hyderabad
vijaya.saradhi@research.iiit.ac.in

Oota Subba Reddy
SIEL, IIIT Hyderabad
oota.subba@students.iiit.ac.in

Abstract

This paper describes our system for detection and interpretation of English puns. We participated in 2 subtasks related to homographic puns achieve comparable results for these tasks. Through the paper we provide detailed description of the approach, as well as the results obtained in the task.

Our models achieved an F1-score of 77.65% for Subtask 1 and 52.15% for Subtask 2.

1 Introduction

The pun, also called paronomasia, is a form of word play that suggests two or more meanings, by exploiting multiple meanings of words, or of similar-sounding words, for an intended humorous or rhetorical effect. A pun is also a special form of ambiguity (mostly lexical) that is consciously used to create statements with ambiguous-distinct-meanings. These ambiguities can arise from the intentional use of homophonic, homographic, metonymic, or figurative language. A homographic pun exploits distinct meanings of the same written word, and a homophonic pun exploits distinct meanings of the same spoken word.

Examples of homographic puns.

- *"I used to be a banker but I lost interest"*,
- *"Tires are fixed for a flat rate"*
- *"Getting rid of your boat for another could cause a whole raft of problems"*

In the first example, the word interest is the pun denoting interest as willingness and also as a fixed charge for borrowing money. In the second example, the word flat is the pun denoting flat as in a flat tyre and flat as in flat rate. The third example, the word raft is the pun denoting raft as a batch and raft as a type of boat.

In the present work, we focus on homographic puns. We present methods to a) identify a pun sentence, b) identify the pun word given a pun sentence and c) interpret the different word senses of the pun word. The details of the shared task are available at (Tristan Miller and Hempelmann, 2017)

2 Subtask 1 - Pun detection

In this task, for each sentence, the system must decide whether it contains a pun or not. While this task is mostly unsupervised, we cast this problem as a supervised learning classification problem. We have randomly selected part of the dataset and manually annotated them as a pun sentence or a non-pun. We decided to leverage on models that try to model sequences of word vectors. We can view the each sentence as a sequence where we only have one label at the end. This many-to-one mapping lends itself nicely to Recurrent Neural Network. Due to this reason, we used a recurrent neural network to train the classifier and generate the model. Using this model we classify the remaining dataset.

We used two settings for the train and test split. In setting 1, we used 70% of the dataset for training and 30% of the data for testing. In Setting 2, we used 30% of the dataset for training and the remaining 70% of the data for testing.

Recent research has shown that deep learning methods can minimize the reliance on feature engineering by automatically extracting meaningful features from raw text (Collobert, 2012). Thus, we propose to use distributed word embeddings which capture lexical and semantic features as input features to our neural network model.

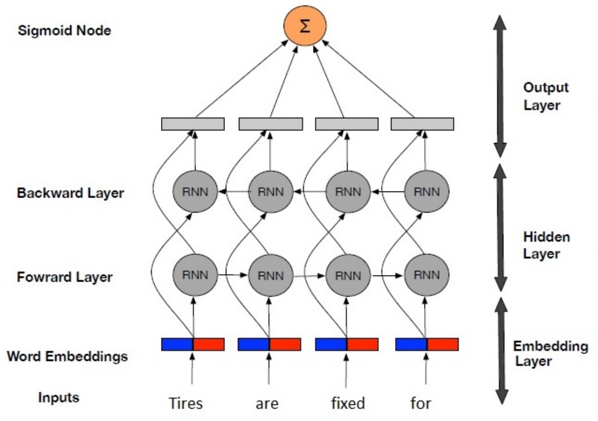

Figure 1: BiDirectional RNN architecture for detecting puns

Distributed word embeddings map words in a language to high dimensional real-valued vectors in order to capture hidden semantic and syntactic properties of words. These embeddings are typically learned from large unlabeled text corpora. In our work, we use the pre-trained 50 dimensional GloVe embeddings (Pennington et al., 2014) which were trained on about 6B words from the twitter using the Continuous Bag of Words architecture.

2.1 Model

The network architecture of our model as illustrated in Figure 1 has the following structure

- Embedding Layer: This layer transforms each word into embedded features. The embedded features are a concatenation of the words Distributed word embeddings. The embedding layer acts as input to the hidden layer.
- Hidden Layer: The hidden layer consists of a Bi-Directional RNN. The output of the RNN is a fixed sized representation of its input.
- Output Layer: In the output layer, the representation learned from the RNN is passed through a fully connected neural network with a sigmoid output node that classifies the sentence as a pun or a non-pun.

2.2 Performance

We train the network for 25 epochs. The following table describes the results on the pun dataset. Table 1 shows the performance of our classifier on the two settings.

2.3 Discussion

While the pun classification was supposed to be an unsupervised classification problem, we cast the problem into a supervised classification problem by annotating the data partially. This might be a reason for very high precision for the Setting 1 as the classifier might have overfit the model to a wider range of training data than in the second setting. We used a dropout of 0.5 in Bi-Direction RNN to avoid the overfitting problem.

3 Subtask 2 - Pun location

In this task, for each sentence containing a pun, the system must identify the pun word. We introduce the algorithm which takes a sentence containing a pun as an input and returns the pun word.

3.1 Observations

Empirical observations of the puns show the following characteristics of the pun word

- The pun word usually appears towards the end of the pun (Miller, 2014)
- There exists a non-pun word whose similarity is more than a threshold, than any other word to word similarity in the sentence.
- Stopwords and non content words can not be puns
- The pun word will have atleast two word senses in the wordnet corpus (Miller, 1995)

3.2 Methodology

From the examples given in section 1, we see that the pun word interest is semantically close to the word banker and its root bank. In the next example, we observe that the word flat is semantically close to the word tires because tires can go flat. In the last example the word raft is semantically close to the word boat as both are used as transport objects in water.

Based on the above observations, we propose the following algorithm to identify the pun word in a sentence having a pun. The function FIND_PUN takes a sentence containing a pun as an input and returns the pun word discussed in Algorithm 1. First, a list of probable pun words is generated by removing the stopwords and punctuation in the

Setting	Train	Test	Precision	Recall	Accuracy.	F1
Setting 1	70%	30%	0.9697	0.7953	0.8360	0.8738
Setting 2	30%	70%	0.8918	0.6876	0.7173	0.7765

Table 1: Performance of the pun classifier using Bidirectional RNN in 2 settings for Subtask 1

sentence. Next, all words which have less than 2 entries in the wordnet are removed. If there is only 1 word left, we return this word as a pun word. In case multiple words are present in the probable puns list, then a list of pairs with all combination of words in the sentence with all the possible words in the probable puns is made. For every pair in the list of pairs, the similarity of the two words is calculated. The similarity of two words is also calculated by expanding the synsets of both the words and finding the most similar pair between the pairs of every synset in word 1 with every synset in word2.

Algorithm 1 PUN Detection Algorithm

```
1:  INPUT Sentence
2:  OUTPUT PUN word.
3:  Step 1: Tokenize the input sentence.
4:  Step 2: Remove punctuations and stop words
5:   from the tokens.
6:  Step 3: Remove tokens which have less than
7:   2 entries from the wordnet.
8:  if Only one Word in list then
9:      return Word(PUN word)
10: else
11:     for All pair of words in sentence do
12:         Bestpair=MAXSIM(Pair of words)
13:     end for
14:     return Word $\in$ Bestpair which occurs
15:      towards end of the sentence.
16: end if
```

The MAXSIM function takes two words as input and returns the maximum similarity between all pairs of words in their synsets. The algorithm is described in 2.

For every possible pair between synsets1 and synsets2, calculate the word to word similarity using the word embeddings. Return the pair which has the maximum similarity.

For the pun example 3, the following table shows the similarity between all pairs of the words in the descending order of the similarity.

Algorithm 2 MAXSIM Model

```
1:  INPUT Words={Word1, Word2}
2:  Initialization MaxSimilarity=0
3:  OUTPUT Maximum similarity between all
    pairs of words in their synsets
4:  Step 1: synsets1 = get the synsets of word1
5:  Step 2: synsets2 = get the synsets of word2
6:  for pairword1 in synsets1 do
7:      for pairword2 in synsets2 do
8:          Step 3:WordSimilarity=
9:  similarity(pairword1,pairword2)
10:         if MaxSimilarity $\leq$ WordSimilarity
    then
11:             a:MaxSimilarity= WordSimilarity
12:         end if
13: end for
14: end for
15: Step 4:return MaxSimilarity
```

Word 1	Word 2	Max similarity
boat	**raft**	**0.68**
boat	whole	0.61
getting	whole	0.5
rid	whole	0.5
getting	problems	0.46
getting	raft	0.4
rid	raft	0.4
getting	boat	0.28
boat	problems	0.26
boat	whole	0.24
rid	raft	0.23
rid	whole	0.21
rid	problems	0.20

Table 2: All word similarity pairs for the example pun 3

Precision	Recall	Coverage.	F1
0.5215	0.5215	1	0.5215

Table 3: Performance of the pun word identification using MAXSIM for Subtask 2

In the first step, the following non content words are removed - of, your, for, could, cause. The remaining words are - rid, boat, getting, whole, problems. Table 2 shows the similarities between all pairs of the words. The highest pair boat-raft is returned by the maxsim function. Since the word raft appears later in the sentence, the word raft is identified as the pun word.

3.3 Performance

Table 2 describes the result of our teams run for Subtask 2.

3.4 Discussion

We have observed that our algorithm maxsim has performed decently for pun sentences which are short in length having up to 10 words in the sentence. We observed that as the length of the sentence increases, there are more number of words which are similar together and the accuracy of pun word identification decreased.

4 Subtask 3 - Pun interpretation

In this subtask, the pun word is given and the system has to annotate the word with the right WordNet sense keys. While we did not participate in this subtask, it is trivial to extend the work done for Subtask 2 to achieve the objective of Subtask 3.

5 Conclusion

Classifying an English sentence as a pun or not is a non trivial task. It is much more difficult and challenging to solve this problem as an unsupervised classification task. Experimentation can be done to use LSTMs, and BiDirectional LSTMs to improve the performance of the classifier. Character Level Embeddings can be used as features to capture orthographic and morphological features of a word. word2vec embeddings from the Google News dataset can be evaluated as alternative features for GloVe vectors. For pun identification task, cluster based approaches can be explored specifically for the long sentences. As the interpretation of most of the puns relies on specific domain knowledge, additional corpora can be used to augment our models for better performance.

References

R. Weston J. Bottou L. Karlen M. Kavukcuoglu K. Kuksa P Collobert. 2012. Natural language processing (almost) from scratch. journal of machine learning research.

George A Miller. 1995. Wordnet: A lexical database for english. communications of the acm vol. 38, no. 11: 39-41.

Tristan Miller. 2014. Towards the automatic detection and identification of english puns.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. Glove: Global vectors for word representation. In *Empirical Methods in Natural Language Processing (EMNLP)*. pages 1532–1543. http://www.aclweb.org/anthology/D14-1162.

Iryna Gurevych Tristan Miller and Christian F. Hempelmann. 2017. Systemdescription paper for semeval 2017 task 7. In *Proceedings of the 11th Annual Meeting of the SemEval)*. Association for Computational Linguistics.