Pun Detection, Location, and Interpretation in English Language

Manush Patel,Parmeet Singh Saluja,Shaival Patel
Email: {patel.man,saluja.pa,patel.shaiv}@husky.neu.edu

I. Introduction

A pun is a form of wordplay in which a word suggests two or more meanings by exploiting polysemy, homonymy, or phonological similarity to another word, for an intended humorous or rhetorical effect.(Aarons, 2017; Hempelmann and Miller, 2017). For example, the first of the following two contexts exploits the sound similarity between the word knows and nose, while the second exploits contrasting meanings of the word sting.
(1) Follow your knows.
(2) Can honeybee abuse lead to sting operation?
Puns where the two meanings share the same spelling are known as homographic while those relying on similar- but not identical spelling signs are known as heterographic. In this project, we would like to experiment on three major techniques:1. Word Sense Disambiguation,2: Phonetic Similarity and 3: Machine learning based approach for classification.

There are important real-world applications which involves computational interpretation of puns. For example: It has often been argued that humor can enhance humancomputer interaction (HCI) (Hempelmann, 2008).So, an interactive system that is able to recognize and produce contextually appropriate responses to users puns could further enhance the HCI experience. Recognizing humorous ambiguity is also important in machine translation, particularly for sitcoms and other comedic works, which feature puns and other forms of wordplay as a recurrent and expected feature (Schrter, 2005). Future automatic translation aids could scan source texts, flagging potential puns for special attention, and perhaps even proposing ambiguity-preserving translations that best match the original puns double meaning.

II. Related Work

Detection and Interpretation of humor have always been one of the main research topics in natural language processing field. Though these researches were not totally focused on puns but they have always shadowed on them and showed us that puns have a semantic tint. Some of the prior work which was done related to our area is summarized below. Computationally recognizing wordplay in jokes (Taylor and Mazlack et al.,2004) devised an n-gram methodology for recognizing imperfect puns in a particular small class of English jokes. As this approach was majorly focused on imperfect puns their applicability on perfect puns was very less. Towards the automatic detection and identification of English puns (Tristan Miller and Mladen Turkovi et al., 2016) and Automatic disambiguation of English puns (Tristan Miller and Iryna Gurevych et al., 2015) have presented various computational methods for the detection of puns in text. A computational approach using word association for puns (Sevgili et al., 2017) found that sentences containing a pun have a distinctive word that has a high semantic/phonetic association with the pun. In accordance with this observation, all the pairwise word associations were calculated to determine the most correlated word pair. If an element of this most correlated word pair had a second sense/spelling, it was an evidence of being a pun. The models that we were inspired were the top 3 models from the SamEval Task 7, which were Duluth Fermi and Idiom Savant .Duluth(Pedersen, 2017) used a word sense Disambiguation approach where he uses a WordNet::SenseRelate::AllWords algorithm and check for change in word senses based on different contexts. The short coming of his models were that it was not able to tell whether the results would give the same results on larger sentences or not. Also, the accuracy of this method completely depends on the WSD algorithm we use. The model Idiom Savant(Doogan et. al,2017) uses phonetics based approach and is good fit for finding Heterographic puns but is not reliable enough for Homographic puns.The model Fermi(Indurthi and Oota,2017) used the machine learning based approach to tackle the classification problem and also tried to come up with a similarity algorithm for location, though it was the best at classification task, it did not do well over other task. So we have decided to use mixed approach to tackle all these problems.

III. Dataset

The dataset we are using is provided by SemEval-2017 task 7 (Miller et al.,2017). The first dataset which is about homographic puns set contains 2250 context (jokes, slogans, aphorisms, etc) in this data set we have sentences with a single pun word. The second data set which is related to heterographic puns has 1780 context sentences. Both of them are in XML format. We first converted the XML format to a plain text file. The corpus was pretty much clean all we had to do was remove punctuation and case folding on the dataset which we extracted.Dataset contains punning and non-punning jokes, aphorisms, and other short, self-contained contexts sourced from professional humorists and online collections.Each context contains a maximum of one pun.Each pun (and its target) has a lexical entry in WordNet 3.1[11].The homographic data set contains 2250 contexts, of which 1607 contains a pun.The heterographic data set contains

1780 contexts, of which 1271 contains a pun. For natural language processing based model we have given entire datasets as inputs and for machine learning based approach we have used (80%) training(20% validation) and testing(20%) split.

IV. Methodology

As our goal consists of 3 different subtasks,so we defined a baseline model for each subtask.

Pun Detection: The baseline defined for this subtask is a random classifier which classifies each context with pun or non pun with equal probability and with no prior assumption.

Pun Location: There are two baselines defined for this subtask, first is baseline is to select a random word from the context words and classify it as a pun and the second baseline is to select the last word of the context as a pun.

Pun Interpretation: The baseline defined for this subtask is to select the most frequent sense from the all the senses that we get from wordnet for the word located as pun word in pun location task.

These baselines(Miller et al.,2017) are purely based on randomized models, that is why they have very low precision and recall. So we have tried different approaches for Homographic and Heterographic puns.

A. *Data Preprocessing*

The data was provided in XML format. We converted the data into text format. First we tried lowercasing the data and also removing punctuation . But it created problem as for example Fishermans was changed to fisherman s in the sentence and that created context problems and tagging problems. So what we did was just expanded the contractions, removed some punctuation mark which does not affect the context of the sentence. The two data set we used are provided by SemEval-2017 task 7 (Miller et al.,2017). The first dataset which is about homographic puns set contains 2248 context (jokes, slogans, aphorisms, etc) in this data set we have sentences with a single pun word. The second data set which is related to heterographic puns has 1780 context sentences. For later phase of the project we implemented Recurrent Neural Network and its input require labeled dataset. So to satisfy the requirements we manually annotated the data set. The methodology we used for manual annotation is as follows. Each three of us manually annotated 33% of the dataset and cross verified one other person's annotations and for verification we used Internet resources and standard language rules to identify pun.

Original :
```
<text id="hom_1">
  <word id="hom_1_1">They</word>
  <word id="hom_1_2">hid</word>
  <word id="hom_1_3">from</word>
  <word id="hom_1_4">the</word>
  <word id="hom_1_5">gunman</word>
  <word id="hom_1_6">in</word>
  <word id="hom_1_7">a</word>
  <word id="hom_1_8">sauna</word>
  <word id="hom_1_9">where</word>
  <word id="hom_1_10">they</word>
  <word id="hom_1_11">could</word>
  <word id="hom_1_12">sweat</word>
  <word id="hom_1_13">it</word>
  <word id="hom_1_14">out</word>
  <word id="hom_1_15">.</word>
</text>
```

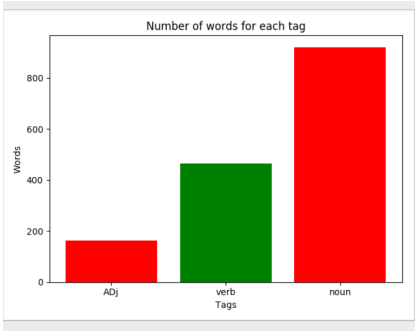

Fig. 1. Number of pun words per Part of speech type

Cleaned: they hid from the gunman in a sauna where they could sweat it out

The following is the distribution of pun words with various POS tags, that we generated in order to analyze what sort of words create the pun effect in general sentences.

B. *Word Sense Disambiguation based Model-Homographic Puns*

In this model for Pun Detection we are using the fact that this type of puns are created by words which have similar sounds but different contexts and meaning.So in this model we are try to find word senses based on three different context windows and for sentences with expanded contractions. What we did was we tried only on the sentences where we expanded the contractions, but what we did was that we assigned context window based on the length of the sentences. We divided our context sizes into 3 categories, trigrams , four-grams and five grams.

We did this because lets say that the sentences is of 3 words then there is no point in checking 4 grams and 5 grams of the sentences. We compared the word sense that we got from this context with context that takes the whole sentence as the context and if the word sense conflicts then there is a pun in the sentence.

For Pun Detection we used the previous algorithm but what we did was whenever we found a conflict in word senses for a word we would add it to the list if puns that we found in that sentence. Now for many sentences many of the words in the list were not puns and words with frequency of 10 or more, so we considered such words as stop words and ignored them, and even after this context if there are multiple words in the list, according to the research by authors of Duluth, we considered the latest conflict in the sentence as the pun in the sentence.

For interpretation we simply used lesk algorithm on that word which we said was a pun and found the word sense with the whole sentence as the context.

C. *Context Normativeness and Phonetic Similarity-Heterographic Pun*

In this model for Pun Detection we are using the fact that this type of puns are created by words which do not

have similar sounds but have similar spelling. By observation we have found that in heterographic puns there is use of n grams which occur very less frequently in normal use of language.But the candidate n-gram occurs frequently. By our current observation we found that pun words are generally Noun, Verb, Adverb or Adjective. We tag the text using NLTK[4] POS tagger and select the words which are tagged as Noun, Verb, Adverb or Adjective. Sentences are divided into n-grams to capture the context information. We have used trigrams and quadgrams so that all context information can be correctly captured even in cases of shorter sentences. Then we try to find the candidate n-gram which is more frequent in google n-grams then the given trigram and is only one word different from the given trigram. The candidate trigrams we get from this process are again filtered on the basis of word they differ from. Two words on which the trigram differ should be rhyming to each other. Rhyming words are checked using CMU pronouncing dictionary[5]. All candidate trigrams which are one rhyming word different from given trigram are ranked on the basis of a score. We have used three techniques to calculate the phonetic distance where first is the minimum edit distance between words(ch), second is Levenshtein distance between the phonetic representation of the word (phs) and third is Levenshtein distance between the phonetic representation of the word without spaces (ph). Now we normalize this distance as we do not want to penalize the large words and get the resultant ratio.

$$ratio_f = \frac{(w\_len\_f - dis\_f)}{w\_len\_f}$$

where $f \in ch, ph, phs$ and w_len is length of the word and dis is corresponding phonetic distance.

We capture the minimum phonetic similarity of the two words by taking maximum value from above three techniques. Now we use another measure which is the frequency of trigram is occurring in the large corpus. We calculate the difference in frequency of new and old trigram and subtract the inverse of nth power of ratio of the two words which are different in trigram where n is the word length.Selecting only the trigrams which has positive score and then choosing finally with the maximum score we can easily detect and locate the pun.

D. *Machine learning based classification*

We tried to convert this unsupervised problem into a supervised one by manually annotating some of the dataset by the method described above. We can treat each sentence as a sequence and we can give each sentence single label at the end, for sequencing the sentences we first thought of using word2vec approach but since our dataset is relatively small we used Glove Pretrained vectors instead. Glove pretrained vectors are trained over 6B words from twitter and are easy to use to represent a sentence as a sequence.

The Neural Network configuration is described below:

Embedding layer: Transforms words into embedded features and acts as input to hidden layer.

Hidden layer:

1. LSTM with 100 neurons with return sequences i.e. getting the last output feed into current input feed.
2. Conv1d Network with relu activation
3. Global max pooling 1d with relu activation
4. Final fully connected neural netwrok with two output nodes with softmax activation

optimizer- adam optimizer of keras

loss function - binary cross entropy epochs - 40 Following are the graphs for training and validation accuracy vs number of epochs and training and validation loss with number of epochs.

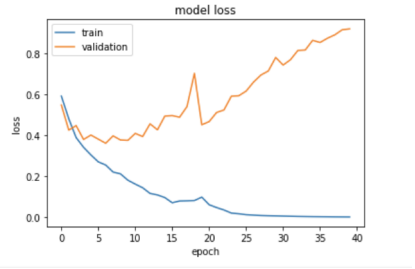

Fig. 2. loss vs. number of epochs

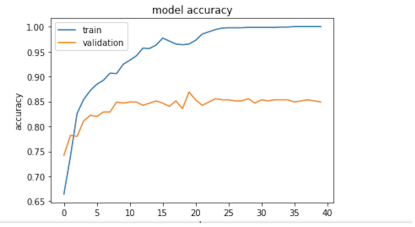

Fig. 3. accuracy vs. number of epochs

E. *Pun interpretation*

For the task of pun interpretation we used the data provided by location task and provide the sentence and ambiguous word to the Lesk algorithm which is a word sense disambiguation approach. Essentially the interpretation of pun is based on underlying WSD algorithm and there is so less variation that we could experiment with this approach.

V. Experiments

A. *Experiments on WSD based approach*

First we tried with 2 context lengths one is trigrams and one is the whole sentence is the context window , but there were some cases where the pun word and the context for that pun were really far away in the sentence. So what we did was we assigned context window sized based on the length of the sentence. We divided the context window sizes into 3 groups namely, trigrams, fourgrams and fivegrams based on the length of the sentences.

For location what we tried first was put up all the words which had conflict in word sense into a list and selected the latest conflict in the list as the pun. But the thing is that we were getting a lot words which are not pun and we noticed that these words have a very high frequency in the data set,

sow what we did was if the conflict is in the word with frequency greater than 10 we ignored it. The final list of conflicts reduced drastically, and the accuracy increased, we still got some results where there are more than one conflict in the sentence so what we did was selected the last conflict in the sentence as the pun, this intuition was based on the baseline model for location.

B. Experiment on Context Normativeness and Phonetic Similarity:

When we didnt have a constraint on the number of n-grams which we need to replace candidate n-grams, the calculation of the result took a very long time, and the the program became inefficient. So we decided on to reduce the number of n-grams we want to use, what we did was used on top 30 frequent n-grams were taken to replace the candidate n-grams.

Also, we added a constraint on the candidate n-grams and consider such n-grams whose frequency is less than 500, because puns are usually non -normative words and if we take words whose frequency is higher than 500 we are basically trying to detect words for normative words.

As we are using CMU dict to find rhyming words and CMU dict gives a large set of words as rhyming words for a given word so we used edit distance to filter those words out. The value of edit distance was decided based on experiments and we came to conclusions that if we chose a higher value of threshold for edit distance then the results were over fitted and if the threshold value was less the results were under fitted.

Cardinal number can also act as a pun word so we added them into P.O.S tags set.

Minimum ratio of the word-pair to considered as Candidate and Replacement word is 0.3, we chose this value because a higher ratio means that we were getting more false negatives as the result.

Following are the accuracy measures for our models.Essentially we have calculated Precision, Recall and Accuracy for our detection task and accuracy for our location task. For the task of pun interpretation we did not include any evaluation as it is just a WSD algorithm run and is completely based on implementation of Lesk Algorithm used in our case the accuracy measured at 9% and 10% for simple lesk and adaptive lesk respectively. Note that we have used the implementation provided by pyWsd[8].

C. Experiments on Pun Generation

The approach that we have tried is to find phonetically similar words to replace potential pun candidates and also replace them with homophones which suits better with the context. This is a naive approach and does not generate meaningful and humorous puns.

Example: **Input**: Job well done Output: Job welle done.

Input : the seed starves Output: the steed starved

VI. Results

Pun Detection

Input - Some people find fire drills quite alarming.

Output - 1

Input - Honeybee abuse can lead to a sting operation.

Output - 1

Input - Follow your knows.

Output - 1

Input- Acupuncture is a jab well done.

Output - 1

Input- A dentist hates having a bad day at the orifice.

Output - 1

Input - A busy barber is quite harried.

Output - 0

Input - Slow and steady wins the race.

Output - 0

Pun Detection	Precision	Recall	Accuracy
Baseline	0.52	0.52	0.52
WSD	0.78	0.86	0.70
CN & PS	0.8	0.647	0.65
RNN	-	-	0.84

Fig. 4. Pun detection evaluation

Pun Location	Accuracy
Baseline	0.57
WSD	0.28
CN & PS	0.66

Fig. 5. Pun location evaluation

Pun Location

Input - Some people find fire drills quite alarming.

Output - alarming

Input - Honeybee abuse can lead to a sting operation.

Output - sting

Input - Follow your knows.

Output - knows

Input- Acupuncture is a jab well done.

Output - jab

Input- A dentist hates having a bad day at the orifice.

Output - orifice

Input - Why was six afraid of seven because seven eight nine.

Output - seven

Pun Interpretation

Input- Old bakers never die , they just stop making lots of dough .

Output - ('dough', Synset('dough.n.01'))

Input- OLD gardeners never die they are just deflowered .

Output -('deflowered', Synset('deflower.v.01'))

There are obviously some false negatives and false positives in the model of Heterographic Puns. But the model of Homographic Puns was showing quite less false positives.

As we can see from above examples that the system is able to detect things correctly most of the time but in the

last example of Pun Location the pun was at eight and the actual word was ate. System fails here. It is happened due to very high frequency of trigram ”one eight nine”. To detect these special types of puns we have added Cardinal Number also to the set of tags of pun words.

VII. Conclusion

We have tackled this problem with word sense disambiguation based approaches and also leveraged the power of phonetic distances to find heterographic pun. The main point where these approaches are lagging is the reliance over underlying algorithms such as WSD approach is reliant on underlying Lesk algorithm or the Context Normativeness and Phonetic Similarity approach is dependent for context based information on Google n-grams using phrasefinder API[12], for POS tagging on NLTK POS tagger, and for finding rhyming words on CMU dict. We found that RNN based approach is better than other two when it comes to classification, but the other two approaches came through for location of puns as they were solely reliant on whether or not they could find the punning effects of words in the context. As we know that these approaches depend on number of other things so the efficiency also directly becomes dependent on these factors.All the detection models worked fine with Tom Swifty type of puns, so we did not introduce a separate method for these puns. We also tried to generate puns with rules such as phonetic similar substitution and results were quite interesting, with further research we plan to improve this method and generate meaningful and humorous puns.

References

[1] SemEval-2017 Task 7: Detection and interpretation of English puns by Tristan Miller and Christian F. Hempelmann and Iryna Gurevych.

[2] Feature-Rich Part-of-Speech Tagging with a Cyclic Dependency Network by Kristina Toutanova, Dan Klein, Christopher Manning, and Yoram Singer.

[3] Automatic sense disambiguation using machine readable dictionaries : How to tell a pine cone from an ice cream cone by M.E. Lesk.

[4] NLTK: The Natural Language Toolkit by Steven Bird.

[5] http://www.speech.cs.cmu.edu/cgi-bin/ cmudict

[6] Idiom Savant at Semeval-2017 Task 7: Detection and Interpretation of English Puns by Doogan, Ghosh and Chen.

[7] Duluth at SemEval-2017 Task 7: Puns Upon a Midnight Dreary, Lexical Semantics for the Weak and Weary by Pedersen.

[8] Pywsd: Python Implementations of Word Sense Disambiguation (WSD) Technologies [software].Liling Tan. 2014.

[9] GloVe: Global Vectors for Word Representation. Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014.

[10] Puns and tacit linguistic knowledge. In Salvatore Attardo, editor, The Routledge Handbook of Language and Humor, Routledge, New York, NY, Routledge Handbooks in Linguistics, pages 8094.Debra Aarons. 2017.

[11] Christiane Fellbaum (1998, ed.) WordNet: An Electronic Lexical Database. Cambridge, MA: MIT Press.

[12] Keras, Chollet, Francois and others,2015

[13] Detection and Interpretation of Homographic puns in English Language, Indurthi and Reddy,2017.