Idiom Savant at Semeval-2017 Task 7: Detection and Interpretation of English Puns

Samuel Doogan Aniruddha Ghosh Hanyang Chen Tony Veale
Department of Computer Science and Informatics
University College Dublin
Dublin, Ireland
{samuel.doogan, aniruddha.ghosh, hanyang.chen}@ucdconnect.ie
tony.veale@ucd.ie

Abstract

This paper describes our system, entitled *Idiom Savant*, for the 7th Task of the Semeval 2017 workshop, "Detection and interpretation of English Puns". Our system consists of two probabilistic models for each type of puns using Google n-grams and Word2Vec. Our system achieved f-score of 0.84, 0.663, and 0.07 in homographic puns and 0.8439, 0.6631, and 0.0806 in heterographic puns in task 1, task 2, and task 3 respectively.

1 Introduction

A pun is a form of wordplay, which is often profiled by exploiting polysemy of a word or by replacing a phonetically similar sounding word for an intended humorous effect. From Shakespeare's works to modern advertisement catchphrases (Tanaka, 1992), puns have been widely used as a humorous and rhetorical device. For a polysemous word, the non-literal meaning is addressed when contextual information has low accordance with it's primary or most prominent meaning (Giora, 1997). A pun can be seen as a democratic form of literal and non-literal meaning. In using puns, the author alternates an idiomatic expression to a certain extent or provides enough context for a polysemous word to evoke non-literal meaning without attenuating literal meaning completely (Giora, 2002).

Task 7 of the 2017 SemEval workshop (Miller et al., 2017) involves three subtasks. The first subtask requires the system to classify a given context into two binary categories: puns and non-puns. The second subtask concerns itself with finding the word producing the punning effect in a given context. The third and final subtask involves annotating puns with the dual senses with which the punning effect is being driven.

In a written context, puns are classified into 2 categories. Homographic puns shown in example 1, exploits polysemy of the language by using a word or phrase which has multiple coherent meanings given its context; And heterographic puns shown in example 2, humorous effect is often induced by adding incongruity by replacing a phonetically similar word which is semantically distant from the context.

(1) Tires are fixed for a *flat* rate.

(2) A dentist hates having a bad day at the *orifice*.

The rest of the paper is organized as follows. Section 2 give a general description of our approach. Section 3 and 4 illustrate the detailed methodologies used for detecting and locating Heterographic and Homographic puns respectively. In section 5, we provided an analysis of the system along with experimental results and finally section 6 contains some closing remarks and conclusion.

2 General Approach

We argue that the detection of heterographic puns rests on two assumptions. Firstly, the word being used to introduce the punning effect is phonetically similar to the intended word, so that the reader can infer the desired meaning behind the pun. Secondly, the context in which the pun takes place is a subversion of frequent or idiomatic language, once again so that the inference appropriately facilitated. This introduces two computational tasks - designing a model which ranks pairs of words based on their phonetic similarity, and introducing a means by which we can determine the normativeness of the context in question. The sys-

Proceedings of the 11th International Workshop on Semantic Evaluations (SemEval-2017), pages 103–108,
Vancouver, Canada, August 3 - 4, 2017.

tem is attempting to recreate how a human mind might recognize a pun. Take this example:

(3) "Acupuncture is a *jab* well done"

It is immediately noticeable that this sentence is not a normative use of language. However, we can easily recognize the familiar idiom "a job well done", and it is easy to make this substitution due to the phonetic overlap between the words "job" and "jab". Our system is therefore trying to mimic two things: the detection of an infrequent (or even semantically incoherent) use of language, and the detection of the intended idiom by means of phonetic substitution. To model the detection of subverted uses of idioms, we use the Google n-gram corpus (Brants and Franz, 2006). We assume that the normativeness of a context is represented by the n-gram frequency provided by this corpus. The system then replaces phonetically similar words in the non-normative context in an attempt to produce an idiomatic use of language. We determine an idiomatic use of language to be one that has an adequately high frequency in the Google n-gram corpus. We argue that if, by replacing a word in an infrequent use of language with a phonetically similar word, we arrive at a very frequent use of language, we have derived an indicator for the usage of puns. For example, the quadgram "a jab well done" occurs 890 times in the corpus. By replacing the word "jab" with "job", the new quadgram occurs 203575 times. This increase in frequency suggests that a pun is taking place. The system uses several methods to examine such changes in frequency, and outputs a "score", or the estimated likelihood that a pun is being used. The way in which these scores are computed is detailed below.

Homographic puns are generally figurative in nature. Due to identical spelling, interpretation of literal and non-literal meaning is solely dependent on the context information. Literal and non-literal meaning of a polysemous word are referred by different slices of context, which is termed as "double grounding" by Feyaerts and Brône (2002). Considering example 1, it is easily noticeable that two coherent meanings of 'flat', 'a deflated pneumatic tire' and 'commercially inactive', have been referred by 'Tires' and 'rate' respectively. Thus detection of homographic puns involves establishing links between concepts present in context with meanings of polysemous word.

From the general description of different types of puns, it is evident that detection of pun is complex and challenging. To keep the complexity at its minimum, *Idiom Savant* contains two distinct models to handle homographic and heterographic pun tasks.

3 Heterographic Puns

Idiom Savant calculates scores for all possible ngram pairs for a given context. To generate pairs, the system first separates the context into n-grams. For each of these original n-grams, the corpus is searched for n-grams that are at most one word different. The pairs are then scored using the metric described below. The scores for these pairs are then used to tackle each subtask, which is covered below. Since heterographic puns are fabricated by replacing phonetically similar words, classification and identification requires a phonetic knowledge of the language. To obtain phonetic representation of a word, CMU pronouncing dictionary[1] was used. We have ignored the lexical stresses in the pronunciation, as experimentation showed that coarser definitions led to better results. To measure the phonetic distance between a phoneme representation of a pair of words, we have employed three different strategies which use Levenshtein distances. The first distance formula, d_{ph}, calculates Levenshtein distance between two words by considering each CMU phoneme of a word as a single unit. Take the pun word and intended word from example 2:

$$d_{ph}(\{\text{AO, F, AH, S}\}, \{\text{AO, R, AH, F, AH, S}\}) = 2$$

Our second strategy treats the phonetic representation as a concatenated string and calculates Levenshtein distance d_{phs}.

$$d_{phs}(\text{"AOFAHS"}, \text{"AORAHFAHS"}) = 3$$

With this metric, the distance reflects the similarity between phonemes such as "AH" and "AA", which begin with the same vowel sounds. The fallback method for out-of-vocabulary words uses the original Levenshtein string distance.

$$d_{ch}(\text{"office"}, \text{"orifice"}) = 2$$

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

The system normalizes these distances with respect to the length of the phonetic representation of the target words to reduce the penalty caused by word length. By converting distance measures into similarity ratios, longer words remain candidates for possible puns, even though Levenshtein distances will be greater than the shorter counterparts. The system chooses the maximum positive ratio from all possible phonetic representations. If no positive ratio exists, the target word is discarded as a possible candidate.

$$ratio_f(w_1, w_2) = \frac{\min_{w \in w_1, w_2} ||w||_f - d_f(w_1, w_2)}{\min_{w \in w_1, w_2} ||w||_f}$$

where $f \in \{ph, phs, ch\}$

$$ratio = max(ratio_{ph}, ratio_{phs}, ratio_{ch})$$

We choose the maximum ratio in order to minimize the drawbacks inherent in each metric. The assumption is that the maximum ratio between all three methods is the most reflective of the real phonetic similarity between a pair of words. The final score is calculated as the inverted ratio subtracted from the difference between the ngram pair's frequency.

$$score = (freq_{ngram'} - freq_{ngram}) - \frac{1}{ratio^n}$$

Deducting the original n-gram's frequency from the new frequency effectively ignores normal uses of language which do not relate to pun language. The value of the exponent introduces a trade off between phonetic similarity and frequency. The frequencies of certain n-grams are so high that if n is too low, even words with very little phonetic similarity will score high using this method. In our experiments, an optimal value of 10 was found for this trade off.

3.1 Binary Classification

Tto classify a context as a pun or non pun, *Idiom Savant* finds the maximum score from all possible n-gram pairs. If the maximum score found exceeds a threshold value, the context is classified as a pun.

Finding the correct threshold value to accurately classify contexts is discussed below in the Experiments and Results section.

3.2 Locating Pun Word

By maximizing the score when replacing all potential lexical units, the system also produces a candidate word. Whichever replacement word used to produce the top n-gram pair is returned as the candidate word. The system only examines the last two ngrams. Those grams, the system annotates the POS tag and only the content words — nouns, verbs, adverbs and adjectives— are considered as candidate words. The system uses a fall back by choosing the last word in the context when no adequate substitution is found.

3.3 Annotating senses for pun meanings

Subtask 3 introduces an added difficulty with regards to heterographic puns. The system needs to correctly identify the two senses involved of pun, which is based on the accuracy of selecting target words. The system produce a ranked list of n-gram pairs using single word substitution. The highest ranked pair then contains the new or replaced word with which we search for a sense in WordNet (Fellbaum, 1998). For this particular task, the pun word are already given, so the system chooses only the n-grams which contain this word, and only needs to replace this word in order to produce pairs.

Once both words are found, we apply the semantic similarity measure akin to the one used in our systems approach to homographic puns described in Section 4. Both the original and target word is compared to a list of wordnet glosses corresponding to the senses available for each word. *Idiom Savant* uses Word2Vec cosine similarity between the words and their sense glosses to choose the best sense key.

3.4 Tom Swifties

"Tom Swifty" (Lessard and Levison, 1992) is one type of pun often found in the test data set. An example found is " It's March 14th, Tom said piously". Such puns frequently use adverbs to introduce the contextual ties inherent in heterographic puns. Despite that, most of these adverbs occurred in the test data set show little connection with their contexts, rather they are specifically used for ironic purpose. As such, our system did not adequately recognize these instances, so we designed a separate procedure for these cases. To flag whether a pun might be of a Tom Swifty type, the system uses a Part of Speech tagger from

NLTK (Bird, 2006) and also analyses the suffixes of the last word in the context (for example, words ending in "ly").

With relation to tasks 1, an amalgamation of this approach and the original is performed. If the highest score does not exceed the threshold, we check to see if the pun is of type Tom Swifty. If this is the case, then we mark the context as a pun. Task 2 operates similarly - if the pun is flagged as a Tom Swifty, then the last adverb is returned as a candidate. For task 3 however, we need to transform the adverb into the intended word in order to get the appropriate sense entry in WordNet.

To do so we build two prefix trees: one is a phonetic prefix tree based on CMU pronunciation dictionary; the other is a string prefix tree, to cover the exception cases where the adverb is not present in the CMU. If the word is in the phonetic prefix tree, the program returns all words which share at least two common prefix phonemes. For example, given the adverb "punctually", the words "puncture", "punk", "pun" and so on will be returned as candidates. If the string prefix tree is used, the program returns words which share at least the first three characters found in th input word. For the word "dogmatically", "dogmatic", "dogma", and "dog" will be returned as candidates. The list of such candidates is then used to replace the ngrams in which they occur, and the new ngram pairs are ranked according to the metric described at the beginning of 3. The highest scoring prefix is then used to search the appropriate WordNet sense tags.

4 Homographic Puns

Since polysemous words have identical spelling but different meanings, detecting homographic puns is solely dependent on context information. Following double grounding theory, if the i^{th} word of input sentence $W = w_{1:n}$, has a higher possibility to be the punning word, two senses of w_i should infer a higher similarity score with two different components in its context $c_i = w_{1:i-1,i+1:n}$. In the baseline model we design, the pun potential score of a word w_i is computed as the sum of cosine similarities between the word w_i and every word in context $w_j \in c_i$, using distributed representation *Word2Vec* (Mikolov et al., 2013). The word with highest score is returned as the punning word.

Furthermore, as additional context information, w_i were replaced with set of gloss information extracted from its different senses, noted as g_i, obtained from WordNet. While calculating similarity between g_i and c_i, two different strategies were employed. In the first strategy, the system computes similarities between every combination of g_i and c_i, and sum of similarity scores is the score for w_i. In the second strategy, similarity score were calculated between g_i and g_j, the gloss of $w_j \in c_i$. In most of the cases, pun words and their grounding words in the context do not share the same part-of-speech (POS) tags. In the latter strategy, we added a POS damping factor, noted as p_{ij} of 0.2 if the POS tags of w_i and w_j are equal. Following Optimal Innovation hypothesis, the similarity of a punning word and its grounding word should neither be too high or too low in order to evoke the non-literal meaning. We applied following correction on computed similarities.

$$f_{ws}(x) = \begin{cases} 0 & x < 0.01 \\ 1 - x & x >= 0.01 \end{cases}$$

In puns, punning words and grounding words in context are often not adjacent. Thus the system does not consider the adjacent words of the candidate word. The system also ignored stopwords offered by *NLTK*. We noticed that words with high frequency other than stopwords overshadow low frequency words since every word with high frequency poses certain similarity score with every other phrases. Thus we added a frequency damping factor(f_{ij}) of 0.1 to the score for whose words have frequencies more than 100 in *Brown Corpus* (Francis and Kucera, 1979). The final scoring function is shown as follows.

$$score(W, i) = \sum_{j=1}^{n} p_{ij} f_{ij} \sum_{k=1}^{l} \sum_{m=1}^{q} f_{ws}(\frac{g_k g_m}{|g_k||g_m|})$$

n is the number of words in c_i and l and q is number of senses of w_i and w_j. g_k and g_m are gloss of the k^{th} sense and m^{th} sense of w_i and w_j respectively.

For task 3, in order to obtain the sense keys of intended meaning from *Wordnet*, we chose the top two glosses of the pun word based on similarity score between gloss and word in context. For adding more context, instead of comparing only with words in context, we performed similarity measurement among the glosses.

For subtask 1, for each word we calculated similarity with other words and we averaged the top

two similarity score. We have considered a word as a pun if the average score is more than threshold of 0.6, which we chose empirically after observing a number of examples. For subtask 3, we chose the top two senses of the word ranked by the gloss similarity as candidate senses of punned word.

5 Experiment results and analysis

5.1 Heterographic Puns Processing

ID	Method	P	R	F
1	Infrequent Quadgram	**0.90**	0.71	0.79
	Trigram Score	0.82	**0.87**	**0.84**
2	Last Word	0.55	0.55	0.55
	BestQuadGramPairs	**0.68**	**0.68**	**0.68**
3	TopSenses	**0.14**	**0.11**	**0.12**
	GlossSim	0.08	0.07	0.07

Table 1: The precision, recall, and F-score value of heterographic pun subtasks

The experiment results for the heterographic pun subtasks are shown in Table 5.1. For subtask 1, the baseline *infrequent quadgram* is created: if a pun contains no infrequent quadgrams, which have a frequency less than 150 in Ngram corpus, then it is labeled as a non pun. The system uses trigram in subtask 1 because it is computationally feasible to search the ngram space, whilst still being representative of typical uses of language. We set a balanced threshold value of -14 by observing the first 200 samples in the test set.

The high precision score indicates the underlying mechanism behind such puns: a mutation of a typical use of language needs to take place. However the recall for this baseline is poor. A large portion of puns de facto use frequent language usages as targets for linguistic perversion, which this baseline method fails.

Our system outperforms the baseline about five percentage of F-score. The largest factor regarding improper classifications of our model is false positives. Not all infrequent uses of language are heterographic puns. *Idiom Savant*'s technique would sometimes misread a context, modify an infrequent trigram that was not the source of a pun to produce a much more frequent trigram. These false positives are the result of the enormous amount of possible uses in the English language. Infrequent yet "normal" trigrams are an important caveat when using frequency based techniques such as *Idiom Savant*. Hence we see the difference between our model and the simple baseline: although the puns that were detected were very precise, the baseline failed to detect more subtle puns, where normal uses of language are still using phonetic translations to introduce ambiguity.

For subtask 2, *Idiom Savant* uses quadgrams to produce the scores. This is possible because the system employs a number of methods to reduce the search space created when attempting to replace quadgrams. Firstly, the system won't search the Tom Swifty puns in ngrams corpus. Analysing the first 200 samples in the test data, which is not Tom Swifty puns, we found that roughly half all pun words are the last words in the context. Using this method on the whole corpus produced the *L*astWord baseline seen above. When expanding that to quadgrams and thus enlarging the window, an even greater ratio presents itself. Of the same 200 samples, three fourth of punning words are present in the last quadgram. In the gold standard, ninety percent of pun words appear in the last quadgram. We apply the same scoring technique as described above and achieved the performance presented in the table. We find an increase of 13% as compared to the last word baseline across the board.

To create a baseline for subtask 3, we followed the approach described in (Miller and Gurevych, 2015). and choose the top WordNet senses for each word selected as pun word. As WordNet ranks each sense with their associated frequency of usage, the baseline simply selects the most frequent sense for the pun word and replaced word respectively. As the replaced word are produced by the system, the possibility of error even with the baseline approach is affected by the accuracy of previous steps. When an incorrect word is produced, the sense key attached is by default incorrect and thus the precision, recall, and F scores suffer. The baseline outperforms our system to choose the best sense keys by approximately 6 percentage points. Our method involves Word2Vec is insufficient for solving this subtask, which is evidently much more difficult than the previous subtasks.

5.2 Homographic Pun Processing

For homographic pun processing, we participated in subtask 2 and 3. We calculated scores of subtask 1 on test data after task. For subtask 1, our system achieves 0.84 F-score, which outperforms

the all positive baseline. For subtask 2, our system achieves 0.66 F-score. We observed that our system performed well on long sentences. However, for short sentences, most frequent word in the sentence were selected as pun word. This may be caused by lack of context.

Our system does not perform well on subtask 3 as it could not pick the apt sense intended in the pun. We noticed that the system can not pinpoint the apt senses whose glosses are not long enough.

Task	Method	P	R	F-score
Task 1	AllPositive	0.71	**1.00**	0.83
	WordPairSim	**0.73**	0.98	**0.84**
Task 2	WordSim	0.57	0.54	0.55
	WordGlossSim	**0.66**	**0.66**	**0.66**
Task 3	GlossSim	0.08	0.08	0.08

Table 2: The precision, recall, and F-score value of homographic pun processing subtasks

6 Concluding Remarks

We introduced *Idiom Savant*, a computational system that capable of classifying and analyzing heterographic and homographic puns. We show that using n-grams in combination with the CMU dictionary can accurately model heterographic pun.

There are however a number of drawbacks to this approach. We hypothesize that using a larger corpus would increase the performance of heterograhic pun processing. And we may combine different length grams to search for these idiomatic uses of language, which would more accurately model how human recognizes heterographic puns. Furthermore, the system has no means of checking whether the candidate words offered up by *Idiom Savant* are correlated to the rest of the context. Our system suffers intensely for short sentences and short gloss information, since *Word2Vec* doesn't offer context information.

References

Steven Bird. 2006. NLTK: The Natural Language Toolkit. In *Proceedings of the COLING/ACL 2006 Interactive Presentation Sessions*. Association for Computational Linguistics, Sydney, Australia, pages 69–72.

Thorsten Brants and Alex Franz. 2006. Web 1t 5-gram version 1. *Linguistic Data Consortium* .

Christiane Fellbaum, editor. 1998. *WordNet: An Electronic Lexical Database*. Massachusetts Institute of Technology.

Kurt Feyaerts and Geert Brône. 2002. Humor through double grounding: Structural interaction of optimality principles. *Odense Working Papers in Language and Communication* (23):312–336.

W. Nelson Francis and Henry Kucera. 1979. Brown corpus manual. *Brown University* .

Rachel Giora. 1997. Understanding figurative and literal language: The graded salience hypothesis. *Cognitive Linguistics (includes Cognitive Linguistic Bibliography)* 8(3):183–206.

Rachel Giora. 2002. Optimal innovation and pleasure. In *Stock, O., Strapparva, C. and A. Nijholt (eds.) Processing of The April Fools Day Workshop on Computational Humour*. Citeseer, pages 11–28.

Greg Lessard and Michael Levison. 1992. Computational modelling of linguistic humour: Tom swifties. In *ALLC/ACH Joint Annual Conference, Oxford*. pages 175–178.

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.

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 (Volume 1: Long Papers)*. Association for Computational Linguistics, Beijing, China, 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)*.

Keiko Tanaka. 1992. The pun in advertising: A pragmatic approach. *Lingua* 87(1):91 – 102.