UWaterloo at SemEval-2017 Task 7: Locating the Pun Using Syntactic Characteristics and Corpus-based Metrics

Olga Vechtomova
University of Waterloo
Waterloo, ON, Canada
ovechtom@uwaterloo.ca

Abstract

The paper presents a system for locating a pun word. The developed method calculates a score for each word in a pun, using a number of components, including its Inverse Document Frequency (IDF), Normalized Pointwise Mutual Information (NPMI) with other words in the pun text, its position in the text, part-of-speech and some syntactic features. The method achieved the best performance in the Heterographic category and the second best in the Homographic. Further analysis showed that IDF is the most useful characteristic, whereas the count of words with which the given word has high NPMI has a negative effect on performance.

1 Introduction

The pun is defined as "A joke exploiting the different possible meanings of a word or the fact that there are words which sound alike but have different meanings" (Oxford University Press, 2017). When a pun is a spoken utterance, two types of puns are commonly distinguished: homophonic puns, which exploit different meanings of the same word, and heterophonic puns, in which one or more words have similar but not identical pronunciations to some other word or phrase that is alluded to in the pun. The SemEval Task 7 (Miller et al., 2017) focused on the identification of puns as written texts, rather than spoken utterances, and hence distinguished between homographic and heterographic puns.

We participated in Subtask 2: Pun location, which required participating systems to identify which word is the pun. Only the cases which contain exactly one pun word were given to the participants in each of the two categories: homographic and heterographic puns.

Our approach to identifying the pun word is to rank words in the pun text by a score calculated as the sum of values of eleven features. The feature values are calculated using a combination of corpus statistics and rule-based methods. The word with the highest score is considered to be the pun word. The method is described in detail in Section 2. In developing the word ranking method, we were guided by a number of intuitions, outlined below.

The punchline in a pun or a joke is almost always close to the end, since it is at the end that the reader is expected to uncover the second hidden (non-obvious) meaning of the pun. This intuition is consistent with Ruskin's Script-based Semantic Theory of humour (Ruskin, 1985). The system therefore only assigns scores to words located in the second half of the pun text.

What makes a homographic pun humorous is the simultaneous perception by a reader of two conflicting meanings of the same pun word. The pun author can achieve this by using words that are associated with (or evoke) different senses of the pun word. For example in "Why don't programmers like nature? It has too many bugs" The word "programmers" is associated with one sense of "bugs", but the word "nature" is associated with another sense. We operationalize this intuition by calculating Normalized Pointwise Mutual Information (NPMI) between pairs of words to find words that are semantically associated with each other.

Heterographic puns often contain one or more words that are associated with either the pun word itself or its similarly sounding word. In the case of "What did the grape say when it got stepped on? Nothing - but it let out a little whine." The pun word "whine" has a similarly sounding word "wine", which is associated with the preceding

Proceedings of the 11th International Workshop on Semantic Evaluations (SemEval-2017), pages 412–416,
Vancouver, Canada, August 3 - 4, 2017.

word "grape". To operationalize this intuition, we used a dictionary of similarly sounding words. If for a given word in the pun text there exists a similarly sounding word (or words), we calculate NPMI between it and each other word in the text. We also calculate NPMI between the original word as it appears in the pun and each other word. We hypothesize that if a similarly sounding word is more strongly associated (i.e. has higher NPMI) with other words in the text, compared to the original word, it is likely to be the pun word, and receives an additional weight.

The pun word has to stand out from the rest of the text and attract the reader's attention, as it is the realization of the joke's punchline. One possible reason why it stands out is because it is a more rare word compared to the surrounding words. Inverse Document Frequency (IDF) is a measure of how rare the word is in a corpus. The less frequent the word is in a corpus, the higher is its IDF. We hypothesize that a word, which has the highest IDF in the second half of the text is more likely to be the pun word than words with lower IDFs. We thus assign an additional weight to such a word. Furthermore, only nouns, adjectives, adverbs and verbs are assigned scores by our system.

Sometimes, a pun word is a made up word, e.g. "velcrows" in "There is a special species of bird that is really good at holding stuff together. They are called velcrows." We assign an additional weight to words that have zero frequency in a large corpus.

A number of intuitions were guided by the syntactic structure of the text. Thus, we hypothesize that if the pun text consists of two sentences, the pun word is located in the second sentence, as it is most likely to contain the punchline. Therefore, all words in the second sentence receive an additional weight. In a similar vein, if the text contains a comma or the words "then" or "but", all words following them receive additional weights. These clues can signal a pause, a shift in the narrative or a juxtaposition, which all precede the punchline.

2 Methodology

Each test case is tokenized and POS-tagged using Stanford CoreNLP toolkit (Manning et al., 2014). For each word w that is either a noun, an adjective, an adverb or a verb (henceforth referred to as *content words*), the IDF is calculated as $IDF_w = log(N/n_w)$, where n_w is the number of documents in the corpus containing w, and N is the total number of documents in the corpus. For calculating IDF we used ClueWeb09 TREC Category B corpus (Language Technologies Institute, 2009), consisting of 50 million English webpages. To obtain term frequencies, the corpus was indexed and queried using the Wumpus Search Engine (Buettcher, 2007).

For each content word w, the system also calculates pairwise Normalized Pointwise Mutual Information (NPMI) (Bouma, 2009) with each other content word present in the text.

$$NPMI(x,y) = \left(\ln \frac{p(x,y)}{p(x)p(y)}\right) \Big/ -\ln p(x,y) \quad (1)$$

where $p(x,y)$ is calculated as $f(x,y)/N$, in which $f(x,y)$ is the number of times y occurs within the span of s words before or after x in the corpus, and N is the number of word occurrences (tokens) in the corpus; $p(x) = f(x)/N$; $p(y) = f(y)/N$. The co-occurrence span size s was set in our system to 20.

In some puns, the pun word may be hyphenated, where the string after the hyphen can be associated with other content words in the sentence, for example, in "The one who invented the door knocker got a *No-bell* prize." "bell" is associated with "knocker". To account for these cases, we check if a word has a hyphen, extract its second half, lemmatize it, and calculate its NPMI with all other content words present in the text. Given a word pair (x,y), where x is hyphenated and z is the string after the hyphen, calculate $NPMI(x,y)$ and $NPMI(z,y)$. If $NPMI(z,y) > NPMI(x,y)$, then assign the $NPMI(z,y)$ value to $NPMI(x,y)$. We did not experiment with calculating NPMI for the string before the hyphen.

In heterographic puns, a word that is spelled differently, but has similar pronunciation to a word present in the pun, may be associated with other words in the text. A list of 2167 similarly sounding words was compiled from two publicly available resources [1,2]. For each content word, the system checks if it has at least one similarly sounding word in the list, and if so, creates a set of

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

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

f1	Number of content words in the text of the pun that have a lower NPMI with the word x than with any of its similarly sounding words.
f2	Number of content words in the text of the pun that have a lower NPMI with the word x than with its substring following the hyphen (for hyphenated words).
f3	1 - word x has zero frequency in the ClueWeb09 corpus.
f4	1 - word x has a similarly sounding word.
f5	Number of content words y for which $NPMI(x, y) > m$.
f6	1 - word x is located in the third quarter of the text; 2 - in the fourth quarter.
f7	2 - word x is located in the second sentence.
f8	1- word x is located after the earliest occurrence of a comma.
f9	1- word x is located after the earliest occurrence of "then".
f10	1- word x is located after the earliest occurrence of "but".
f11	1 - word x has the highest IDF in the second half of the text.

Table 1: Components of the score calculated for every content word x in the text of the pun.

Method	Precision (rank)	Recall (rank)	F1 score (rank)	Coverage (rank)
Heterographic	0.7973 (1)	0.7954 (1)	0.7964 (1)	0.9976(2)
Homographic (submission 1)	0.6526 (2)	0.6521 (2)	0.6523 (2)	0.9994 (2)
Homographic (submission 2)	0.6519	0.6503	0.6511	0.9975

Table 2: Submission results

similarly sounding words H, including the original word. For each $h \in H$ it calculates its NPMI with each other content word in the text. Given a word pair (x, y), where $x \in H$, $NPMI(x, y) = \max_{h \in H} NPMI(h, y)$. For each content word x in the pun text the system counts the number of content words y for which $NPMI(x, y) > m$ (feature f5 in Table 1), where m is set to 0.3. The system also counts the number of content words y, which have lower NPMI with the original word x, than with any of its similarly sounding words (feature f1).

For every word in the second half of the text, the score is calculated as the sum of values of the features presented in Table 1. The word that has the highest score is selected to be the pun word. If there are ties, the word closer to the end is selected.

3 Results

We made one submission in the Heterographic category and two in the Homographic category (Table 2). Our submission in the Heterographic category achieved the best result among all submissions, exceeding the second-best one in F1 score by 16%. Our best submission in the Homographic category achieved the second best result, with F1 being only 0.02% lower than that of the best submission. Our submission in the Heterographic category and Submission 1 in the Homographic category use all features listed in Table 1. The system used to generate submission 2 in the Homographic category does not use the list of similarly sounding words, hence does not use features f1 and f4.

4 Extensions

After the submission, we noticed that puns may consist of more than two sentences, therefore, we modified feature f7 to assign one point to the last sentence, instead of the second. This resulted in slight improvement ("Submitted (corrected)" in Table 3).

Following the submission we developed another component (f12) to the system presented in Section 2. We were guided by the intuition that in heterographic puns, word x may have the strongest association with word y, however its similarly sounding word h may have the strongest association with a different word z, but the two words z and y are not associated. For example, in "A chicken farmer's favorite car is a coupe." the word "coupe" (x) is strongly associated with "car" (z), however its similarly sounding word "coop" is strongly associated with "chicken" (y). The words "chicken" and "car" however do not have a strong association. We operationalize it as follows. When a word x has a similarly sounding word h, the system finds a word z among all content words W in text with $\max_{z \in W} NPMI(h, z)$. Similarly, for the word x the system finds a word y among all content words W in text with $\max_{y \in W} NPMI(x, y)$. If $NPMI(z, y) < t$ the system adds one point to the score of the word x. Different t values (0.1, 0.2, 0.3, 0.4, 0.5) were evaluated, with $t = 0.2$ showing the best results. The addition of this new feature (row "f12 added" in Table 3 showed some improvement.

Method	Effect on performance	Precision	Recall	F1 score	Coverage
Submitted (corrected)		0.7981	0.7962	0.7971	0.9976
f12 added	+	0.8052	0.8033	0.8043	0.9976
f13 added	+	0.8368	0.8348	0.8358	0.9976
f1 removed	+	0.7744	0.7726	0.7735	0.9976
f2 removed	0	0.7981	0.7962	0.7971	0.9976
f3 removed	0	0.7981	0.7962	0.7971	0.9976
f4 removed	+	0.7926	0.7907	0.7916	0.9976
f5 removed	–	0.8407	0.8387	0.8397	0.9976
f6 removed	+	0.7926	0.7907	0.7916	0.9976
f7 removed	+	0.795	0.7931	0.794	0.9976
f8 removed	+	0.7926	0.7907	0.7916	0.9976
f9 removed	–	0.7989	0.797	0.7979	0.9976
f10 removed	+	0.7965	0.7946	0.7955	0.9976
f11 removed	+	0.6025	0.6011	0.6018	0.9976
f1+f4+f6+f7+f8+f10+f11+f12+f13		**0.8502**	**0.8482**	**0.8492**	**0.9976**

Table 3: Post-submission results with added/removed features (Heterographic puns)

Next, we evaluated component f13, which adds one point to the word's score if its IDF is above threshold i. The i values evaluated were 2, 3, 4 and 5, with $i = 3$ showing the best results. Addition of this feature ("f13 added" in Table 3) led to an improvement of 4.9% over the submitted result.

In order to determine which features contributed positively or negatively to performance, we removed each component one by one (Table 3). The second column in Table 3 shows the effect that the given feature has on the overall performance, e.g. if the removal of the feature causes drop in performance, the feature has a positive effect, indicated by a "+" sign. The component that has the strongest positive contribution to the system's performance is f11, which assigns one point to the word with the highest IDF in the second half of the text. The component that has the strongest negative impact is f5 (number of content words with which the given word has high NPMI). The number of words in the sentence that are more strongly related to the word's similarly sounding word (f1) is also a useful component. Based on this analysis, we modified the system to use only the positively contributing features (last row in Table 3, which outperformed our submitted method in all measures, achieving F1 score of 0.8492 (6.6% improvement).

5 Conclusions and future work

The paper described a method for identifying the location of a pun word using corpus-based characteristics of a word, such as its IDF and NPMI with other words in the pun text, as well its position in the text, part-of-speech and some syntactic features, such as presence of comma and words "but" and "then" prior to the given word's occurrence. The method achieved the best performance in the Heterographic category and the second best in the Homographic. Further analysis showed that IDF is the most useful characteristic, whereas the count of words with which the given word has high NPMI has a negative effect on performance.

Possible future improvements to the presented system are proposed below. In the Homographic pun category, some puns make use of idiomatic expressions. The joke exploits the dual interpretation of an idiomatic expression as, on the one hand, a combination of the literal meanings of its words, and on the other hand, its idiomatic meaning. For example, in "Luggage salespeople have to make a good case for you to buy." it would be useful if the system recognized the phrase "make a good case" as an idiomatic expression.

We used a rather limited list of similarly sounding words. A better way to find similarly sounding words and phrases would be useful, especially in those cases where a combination of words is pronounced similarly to one word, e.g. "There was a big paddle sale at the boat store. It was quite an *oar deal*."

Currently, the feature weights are selected empirically. A possible avenue for future work is to develop an automatic method for selecting the best feature weights.

References

Gerlof Bouma. 2009. Normalized (pointwise) mutual information in collocation extraction. In *Proceedings of the Biennial GSCL Conference*.

Stephan Buettcher. 2007. The Wumpus Infor-

mation Retrieval system. http://www.wumpus-search.org/docs/wumpus_tutorial.pdf. Last accessed: 2017-02-15.

CMU Language Technologies Institute. 2009. The ClueWeb09 dataset. http://lemurproject.org/clueweb09/. Last accessed: 2017-02-15.

Christopher D Manning, Mihai Surdeanu, John Bauer, Jenny Rose Finkel, Steven Bethard, and David McClosky. 2014. The Stanford CoreNLP Natural Language Processing Toolkit. In *ACL (System Demonstrations)*. pages 55–60.

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)*.

Oxford University Press. 2017. Oxford dictionary. https://en.oxforddictionaries.com/definition/pun. Last accessed: 2017-02-17.

Victor Ruskin. 1985. *Semantic Mechanisms of Humor*. D. Reidel Publishing Company.