BuzzSaw at SemEval-2017 Task 7: Global vs. Local Context for Interpreting and Locating Homographic English Puns with Sense Embeddings

Dieke Oele
CLCG
University of Groningen
The Netherlands
d.oele@rug.nl

Kilian Evang
CLCG
University of Groningen
The Netherlands
k.evang@rug.nl

Abstract

This paper describes our system participating in the SemEval-2017 Task 7, for the subtasks of homographic pun location and homographic pun interpretation. For pun interpretation, we use a knowledge-based Word Sense Disambiguation (WSD) method based on sense embeddings. Pun-based jokes can be divided into two parts, each containing information about the two distinct senses of the pun. To exploit this structure we split the context that is input to the WSD system into two local contexts and find the best sense for each of them. We use the output of pun interpretation for pun location. As we expect the two meanings of a pun to be very dissimilar, we compute sense embedding cosine distances for each sense-pair and select the word that has the highest distance. We describe experiments on different methods of splitting the context and compare our method to several baselines. We find evidence supporting our hypotheses and obtain competitive results for pun interpretation.

1 Introduction

A pun is a word used in a context to evoke two or more distinct senses for humorous effect. For example, in the 1987 movie "The Running Man", Arnold Schwarzenegger's character cuts his enemy Buzzsaw in half with a chainsaw, then announces: "He had to split." The verb *split* is the pun here, evoking two senses in the context: that of leaving, and that of disintegrating into two parts.

Recognizing and appreciating puns requires sophisticated feats of intelligence currently unique to humans. A recently proposed set of artificial intelligence tasks (Miller et al., 2017) challenges computers to try their hand at it: *pun detection* (tell whether or not a text contains a pun), *pun location* (given a text with a pun, tell which word is the pun) and *pun interpretation* (given a pun in context, tell which senses it evokes).

Pun interpretation is closely related to the task of Word Sense Disambiguation. A typical WSD system chooses that sense of a word which fits best in the context the word appears in. A pun interpretation system, however, should return not one but two different senses of a word. Miller and Turković (2016) suggest a straightforward extension of the WSD approach to pun interpretation: choose the best scoring sense and second-best scoring sense for the word in its context. However, this approach does not take into account the specific structure of pun-based jokes. In most cases, such jokes can be divided into two parts, where in the first part cues for one sense are concentrated, and in the second part, cues for another sense. Figure 1 shows examples of such cases.

A pun interpretation system could exploit this two-part structure by splitting the global context of the entire joke into two local contexts and performing WSD separately for each local context, choosing the best sense for each of the two. As this process makes each context more informative for the respective sense, we hypothesize that it leads to more accurate pun interpretation than the simple approach which uses the top-scoring two senses according to the global context.

Additionally, we believe that we can use the output of our pun interpretation system for pun location. We hypothesize that the two senses of a pun are typically very dissimilar, as this is important for the joke to be recognizable. We therefore attempt to locate puns by selecting the polysemous word with the most dissimilar two senses.

Proceedings of the 11th International Workshop on Semantic Evaluations (SemEval-2017), pages 444–448,
Vancouver, Canada, August 3 - 4, 2017.

The first time he put the horses on the carriage, it went without a **hitch**.
If a priest is called a white collar worker, a nun would be a creature of **habit**.
Television sets in Britain have to cross the English **Channel**.
Old math teachers never die, they just become **irrational**.
In the winter my dog wears his coat, but in the summer he wears his coat and **pants**.

Figure 1: Examples of pun-based jokes. The pun is typeset in boldface. Words that we judge to be cues for one sense are marked with a dashed underline, words and n-grams that we judge to be cues for the other sense are marked with a dotted underline. Note that the cues for the two senses tend to divide the jokes into two non-overlapping parts.

2 Method

Similar to Miller and Gurevych (2015) we use a knowledge-based WSD system and apply it to pun annotation. Our method is loosely based on the Lesk algorithm exploiting both the context of the words and the definitions (hereafter referred to as glosses) of the senses (Oele and van Noord, 2017). Given a word, Lesk selects the sense whose definition has the highest number of words in common with the context. In our method, which we call Lesk++, instead of counting the number of words that overlap between the gloss of a sense and its context, word and sense embeddings are used to compute the similarity between the gloss of a sense and the context.

2.1 Word Sense Disambiguation

Our WSD method takes sentences as input and outputs a preferred sense for each polysemous word. Given a sentence $w_1 \dots w_i$ of i words, we retrieve a set of word senses from the sense inventory for each word w and sort them in ascending order. Then, for each sense s of each word w, we consider the similarity of its lexeme (the combination of a word and one of its senses (Rothe and Schütze, 2015)) with the context and the similarity of the gloss with the context.

For each potential sense s of word w, the cosine similarity is computed between its gloss vector G_s and its context vector C_w and between the context vector C_w and the lexeme vector $L_{s,w}$. The score of a given word w and sense s is thus defined as follows:

$$\text{Score}(s, w) = cos(G_s, C_w) + cos(L_{s,w}, C_w) \quad (1)$$

The sense with the highest score is chosen. When no gloss is found for a given sense, only the second part of the equation is used.

Prior to disambiguation itself, we sort the words by the number of senses, so that the word with the fewest senses will be considered first. The idea behind this is that words that have fewer senses are easier to disambiguate (Chen et al., 2014). As the algorithm relies on the words in the context which may themselves be ambiguous, if words in the context have been disambiguated already, this information can be used for the ambiguous words that follow. We therefore use the resulting sense of each word for the disambiguation of the following words, starting with the "easiest" words.

Our method requires lexeme embeddings $L_{s,w}$ for each sense s. For this we use AutoExtend (Rothe and Schütze, 2015) to create additional embeddings for senses from WordNet on the basis of word embeddings. AutoExtend is an auto-encoder that relies on the relations present in WordNet to learn embeddings for senses and lexemes. To create these embeddings, a neural network containing lexemes and sense layers is built, while the WordNet relations are used to create links between each layer. The advantage of their method is that it is flexible: it can take any set of word embeddings and any lexical database as input and produces embeddings of senses and lexemes, without requiring any extra training data.

Ultimately, for each word W we need a vector for the context C_w, and for each sense s of word w we need a gloss vector G_s. The context vector C_w is defined as the mean of all the content word representations in the sentence: if a word in the context has already been disambiguated, we use the corresponding sense embedding; otherwise we use the word embedding. For each sense s, we take its gloss as provided in WordNet. In line with Banerjee and Pedersen (2002), we expand this gloss with the glosses of related meanings, excluding antonyms. Similar to the creation of the context vectors, the gloss vector G_s is created by averaging the word embeddings of all the content words in the gloss.

2.2 Pun Interpretation: the Context Had to Split

The WSD method that we described above returns the sense with the highest score taking into account the whole context. For pun interpretation, we could simply adapt it to return the best and second-best sense. However, to exploit the two-part structure of pun-based jokes, we instead split the context into two local contexts, run WSD for each local context and then return the best sense according to each of the two.

Ideally, we want to split the context so that all cues for one sense are in one local context, and all cues for the other sense are in the other. We therefore split the context so as to maximize the semantic dissimilarity between both parts. For each possible split of the text into two contiguous parts, we create a vector for each part by taking the means of all content words, as described earlier, and compute the cosine distance between both vectors. The pair of parts with the highest distance are used as local contexts for the WSD system.

For each polysemous word in the sentence, the WSD system is applied twice, using a different part of the context. The highest scoring sense for each run is chosen. As both runs could assign the same sense to the word, the second best sense of the first run is chosen in this case.

2.3 Pun Location: Attracting Opposites

We attempt to locate the pun in a sentence by selecting the polysemous word with the two most dissimilar senses. In order to do this, we use the two senses as determined by the pun interpretation system for each ambiguous word in the sentence. We therefore retrieve the two best senses for each polysemous word in the sentence and compute the cosine distance between their embeddings. The word that has the maximum distance between its senses is chosen as the answer.

3 Experiments

We use the sense and lexeme embeddings from Rothe and Schütze (2015)[1]. They lie within the same vector space as the pre-trained word embeddings by Mikolov et al. (2013)[2]. This model contains 300-dimensional vectors for 3 million words and phrases from the Google News dataset. Our sense inventory is Princeton WordNet 3.1 (Fellbaum, 1998).

Although a pun can have two or more different part-of-speech tags, our method does not account for this. Instead, we use the POS that was assigned by the Stanford POS tagger (Toutanova et al., 2003).

3.1 Development Data

For the development of our system, we gathered and annotated a small dataset of 91 puns from the website "Pun of the Day"[3]. We used instances that have the same characteristics as in the data for the subtasks we consider (one pun per text, one content word per pun, target exists in WordNet 3.1, pun is homographic). From a small set of downloaded texts, both authors first independently selected the texts that meet all of these criteria. This was followed by a round of adjudication by discussion to determine the texts to use. We then used a similar process to annotate each pun for its two senses.

3.2 Pun Interpretation

We compare our pun interpretation method to three baselines: a random baseline, a most frequent sense baseline and a WSD system that does not use context splitting. The latter was modified to return the two senses with the highest score instead of one.

In addition, we compared different ways of splitting the context of the pun. Next to splitting on the basis of the maximal cosine distance between two possible parts of the context we also ran the WSD system with contexts that were split in half and with contexts that were split at the first punctuation symbol.

3.3 Pun Location

For pun location, we compared our system's performance to two baselines. One baseline randomly selects one content word from the text as the pun, and the other baseline always selects the *last* content word in the text as the pun. In addition, we used the output of all experimental setups of pun interpretation to assess the influence of the quality of assigned senses on pun location.

[1] http://www.cis.lmu.de/ sascha/AutoExtend/

[2] see https://code.google.com/p/word2vec/

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

4 Results

Results of the experiments for pun interpretation can be found in Table 1[4].

Table 1: Results for pun interpretation on the shared task test data.

System	Coverage	Precision
Random baseline	98.92	7.24
MFS baseline	98.92	11.21
Lesk++, no splitting	98.23	15.45
Lesk++, split in half	98.23	16.39
Lesk++, split by punctuation	98.23	15.53
Lesk++, optimal split	98.23	15.53

Our system easily outperforms both the random baseline and the most frequent sense baseline. Also, if we split the context before disambiguating the target word, we gain higher scores as compared to a system that selects the two best scoring senses. We do not, however, gain higher scores when we split the contexts on the basis of maximum semantic dissimilarity. Instead we observe that a system that splits the context in half performs better.

Table 2 shows results for pun location using the output of our system for pun interpretation. Our system scores well above our random baseline. However, the baseline selecting the last content word is much stronger, as the pun often appears at the end of the joke in the data.

Table 2: Results for pun location on the shared task test data.

	Coverage	Precision
Random content word	100.00	13.20
Last content word	100.00	52.96
Lesk++, optimal split	100.00	27.69

Results of the experiments for pun location using the output of our system compared to all baselines for pun interpretation and different splitting setups are shown in Table 3. Using the output of our system, with or without context splitting, performs better compared to systems that use random or most frequent output. The output of systems that use splitting modules and the ones that do not seem to not make a big difference for pun location.

[4]Lesk++, optimal split, was the submitted system. Numbers differ slightly due to a fixed inconsistency in how words are handled for which only one sense could be found.

Table 3: Results for pun location on the shared task test data.

Pun interpretation system	Coverage	Precision
Random baseline	100.00	17.24
MFS baseline	100.00	18.48
Lesk++, no splitting	100.00	27.75
Lesk++, split in half	100.00	27.75
Lesk++, split by punctuation	100.00	28.44
Lesk++, optimal split	100.00	27.69

5 Discussion

Our method for pun interpretation does not yet deal with puns where each sense has a different part of speech. A solution to this would be the use of the senses of the word's second best option of a part-of-speech tagger as well. Also, our method does not deal with phrasal verbs and multi-word expressions.

Our method for pun location works much better than chance, but much worse than a simple heuristic exploiting the fact puns typically appear at the end in the data. It would be interesting to see if both methods can be combined, e.g. using confidence scores and the heuristic as a fallback. It would also be interesting to see if the heuristic can be applied to other types of data, such as movie scripts.

6 Conclusions

We hypothesized that the idea that pun-based jokes can be divided into two parts, each containing information about the two distinct senses of the pun, can be exploited for pun interpretation. Experiments were done splitting the context that is input to a WSD system into two parts, run WSD for each context and return the best sense for pun interpretation. Results of our experiments show that, on the pun interpretation task, systems that use such a module outperform a WSD system that returns the two best senses. Also, our system performs better compared to both the random and the most frequent baseline.

As we expected the two meanings of a pun to be very dissimilar, we used the output of pun interpretation for pun location. Computing cosine distances between each sense-pair and select the one that has the highest distance gains higher scores as compared to a system that randomly selects a content word to be the pun.

References

Satanjeev Banerjee and Ted Pedersen. 2002. An adapted Lesk algorithm for word sense disambiguation using WordNet. In *Proceedings of the Third International Conference on Computational Linguistics and Intelligent Text Processing*. Springer-Verlag, London, UK, UK, CICLing '02, pages 136–145. http://dl.acm.org/citation.cfm?id=647344.724142.

Xinxiong Chen, Zhiyuan Liu, and Maosong Sun. 2014. A Unified Model for Word Sense Representation and Disambiguation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing, EMNLP 2014, October 25-29, 2014, Doha, Qatar, A meeting of SIGDAT, a Special Interest Group of the ACL*. pages 1025–1035. http://aclweb.org/anthology/D/D14/D14-1110.pdf.

Christiane Fellbaum, editor. 1998. *WordNet An Electronic Lexical Database*. The MIT Press, Cambridge, MA ; London.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space. *CoRR* abs/1301.3781. http://dblp.uni-trier.de/db/journals/corr/corr1301.html.

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. http://www.aclweb.org/anthology/P15-1070.

Tristan Miller, Christian F. Hempelmann, and Irina 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. *European Journal of Humour Research* 4.

Dieke Oele and Gertjan van Noord. 2017. Distributional Lesk: Effective multilingual knowledge-based word sense disambiguation. Submitted for publication.

Sascha Rothe and Hinrich Schütze. 2015. Autoextend: Extending word embeddings to embeddings for synsets and lexemes. 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, pages 1793–1803. https://doi.org/10.3115/v1/P15-1173.

Kristina Toutanova, Dan Klein, Christopher D. Manning, and Yoram Singer. 2003. Feature-rich part-of-speech tagging with a cyclic dependency network. In *Proceedings of the 2003 Human Language Technology Conference of the North American Chapter of the Association for Computational Linguistics*. http://aclweb.org/anthology/N03-1033.