PunFields at SemEval-2018 Task 3: Detecting Irony by Tools of Humor Analysis

Elena Mikhalkova, Yuri Karyakin, Dmitry Grigoriev,
Alexander Voronov, and Artem Leoznov
Tyumen State University, Tyumen, Russia
(e.v.mikhalkova, y.e.karyakin)@utmn.ru

Abstract

The paper describes our search for a universal algorithm of detecting intentional lexical ambiguity in different forms of creative language. At SemEval-2018 Task 3, we used PunFields, the system of automatic analysis of English puns that we introduced at SemEval-2017, to detect irony in tweets. Preliminary tests showed that it can reach the score of F1=0.596. However, at the competition, its result was F1=0.549.

1 Introduction

It requires no proof that usually language users can detect lexical ambiguity in figurative speech without classifying it into types like metaphor, pun, humor, irony, sarcasm, etc. At the same time, such terminology is necessary in studyng these phenomena as well as solving problems of automatic classification. In our opinion, programs dealing with intentional lexical ambiguity should be more like natural language users, i.e. they should have one general mechanism of its detection optimized to a certain extent to different types of creative language. In Task 3 "Irony Detection in English Tweets" of SemEval-2018 (Van Hee et al., 2018), we attempted to employ PunFields, our previously introduced classifier of English puns (Mikhalkova and Karyakin, 2017b) [1], in the task of irony classification in tweets as the constrained submission. I.e. PunFields trained only on the Gold dataset released by the Organizers.

As for the unconstrained submission, we used a bag-of-words model with an SVM classifier. The training dataset was enriched with tweets from other SemEval competitions. We will discuss it in a separate paragraph later in detail.

The results of PunFields in this competition were above the chance value (cf.: Table 1) demonstrating higher scores of Recall and F1 score, but lower Acuracy and Precision compared to the enriched bag-of-words (unconstrained submission). In analysis of puns, PunFields is more efficient reaching F1=0.765.

[1]https://github.com/evrog/PunFields

	Ac.	Prec.	Rec.	F1 s.
Con.	0.5765	0.4753	0.6495	0.5489
Unc.	0.6033	0.5000	0.5563	0.5266
Hom.	**0.6782**	**0.7993**	**0.7337**	**0.7651**
Het.	0.5747	0.7580	0.5940	0.6661

Table 1: Results of PunFields at SemEval 2018 Task 3 and SemEval 2017 Task 7. Ac. - acuracy, Prec. - precision, Rec. - recall, F1 s. - F1 score. SemEval-2018: Con. - Constrained submission, Unc. - Unconstrained submission. SemEval-2017: Hom. - homographic puns, Het. - heterographic puns.

In this paper we will briefly outline the state-of-the-art system and possible reasons for its competition results.

2 Lexical Ambiguity in Irony and Puns

We will demonstrate similarity between a pun and irony by the two following examples.

Christmas shopping on 2 hours sleep is going to be **fun**.

In this ironic tweet from the Gold dataset of the competition, the word **fun** is used simultaneously in the meaning "joy, amusement", in the context of "Christmas shopping", and in the meaning, opposite to it, "problem, trouble", in the context of "2 hours sleep." This *opposition* of meanings put into one word or phrase consitutes the essence of what many researchers call irony (Van Hee, 2017; Barbieri and Saggion, 2014).

I could tell you a chemistry joke, but I know I wouldn't get a **reaction**.

In the pun above from the Gold dataset of SemEval2017 (Miller et al., 2017), the word **reaction** is used simultaneously in the meaning "chemical reaction", in the context of "chemistry", and in the meaning "emotional response", in the context of "joke". The only difference between the pun and irony in these examples is that in puns the opposition of meanings is more pragmatic. I.e. the two meanings of **reaction** belong to different spheres of human activity. In irony, the opposition is binary: fun=not fun.

In the both cases, the two meanings are envoked by two contexts, scripts, or themes. Therefore, if a computer program detects coexistence of two topics within an utterance, it is likely to classify correctly both pun and irony, but would be unable to tell which type it deals with. But can ordinary sentences without an intentionally ambiguous word or phrase also contain two scripts? Let us demonstrate what happens to a pun and irony if we transform them into sentences without ambiguity:

> Christmas shopping on 2 hours sleep is **not** going to be fun / is going to be a **problem**.
>
> I could tell you a chemistry joke, but I know I wouldn't get an **emotional** reaction / a **chemical** reaction.

When we transform these sentences, one of the scripts in a pun starts to dominate, for example the three word unity "joke + emotional reaction" would outnumber the one-word script of "chemistry", and vice versa "chemistry + chemical reaction" would outnumber the script of "joke".

However, in case of irony, this ambiguity is not so obvious. On the one hand, if we change "fun" to "problem", words "sleep", "hours" and "problem" *can* form a script. Although this script is not so easily recognizable and requires knowledge that two hours of sleep is too little (for example in the known semantic vector representations of words or documents sleep *can* be found close to lack, fatigue, and time). On the other hand, if we leave the word "fun" in the setence and just add "not" to it ("not going to be fun"), our algorithm will need to process auxiliary words and other variants of negation: never, neither, don't, etc. But these stop-words often create noise in searching for the main topics. If we ignore "not", the script "Christmas shopping + fun" would still form a union and dominate in the utterance.

In sum, intentional lexical ambiguity in irony is not easy for recognition compared to more explicit speech genres like puns. Another feature of irony that supports this statement is that otherwise there would be no need to mark irony with indicators of ambiguity like hashtags and emoji. [2]

3 PunFields

PunFields is a program that turns an utterance into a vector of length 39 based on 39 semantic classes of Roget's Thesaurus called Sections. [3] A thesaurus is a kind of dictionary that unites words into classes that build a hierarchy. Unlike WordNet, the hierarchy is designed by the author/-s of a thesaurus beforehand (top-down approach). Roget's Thesaurus places all lexemes at the lowest level on the basis of their semantic proximity. The clusters of words then unite into Sections, Sections into Divisions, Divisions into Classes. The 39 Sections of Roget's Thesaurus are in a way similar to dimensions of the word2vec algorithm (Mikolov et al., 2013), but word2vec has a fine tuning of meaning proximities.

Data Preprocessing. Before submitting data to PunFields, tweets are put into a preprocessing pipeline. The first step in the pipeline is separating symbols from words in a way that mentions (start with "@") and hashtags (start with "#") are left intact. Emoji are processed as word combiations (":ok_hand_sign:" is simply "ok hand sign"). Words in hashtags are often separated with a "_". Hence, we add a space before and after each "_". For eaxmple, the line "@kennychesney, :ok_hand_sign: #New#color#new#beginning" becomes "@kennychesney , : ok _ hand _ sign : #New #color #new #beginning".

The next step is treatment of emphatic devices in words that are not mentions and hashtags. We judge words with one letter repeated more than

[2]There are cases when it is impossible to recognize ambiguity without such indicators. For example, "Luv this" or "Sitting in this hall is fun" can be expressions of the real state of things unless the user adds a hashtag "#not", an emoji like "unamused face", etc. Note that these indicators quite often come at a particular place - end of the utterance. This fact also brings irony and puns together: in puns, usually the target word (the word used in the two senses) also occurs in the end.

[3]In (Mikhalkova and Karyakin, 2017a), we tried to join some of these classes together for a better performance, but it appeared that such enlargement does not improve it. And vice versa when we reach a certain number of general classes - approximately 4 - the performance starts to decrease significantly.

two times as emphasized. The words are "de-emphasized". For example, the word "everrrrrrr" becomes "ever".

We then proceed to mentions and hashtags. As there can be capitalized real names or titles, we decapitalize them starting with the second letter and add a space before the capital letter and after the word, for example, "#ElektrikBLOOM" becomes "Elektrik Bloom ". If there are (successions of) numbers, a space is added on the both sides of the number. If there are names spelt as one word like "@WhoopiGoldberg", we add a space after every succession staring with a capital letter: "Whoopi Goldberg ".

Some mentions and hashtags are problematic to process as words are written without any indication that they should be separate, e.g. "kennychesney". We check in the NLTK names dictionary (Bird et al., 2009) if the glued hashtag starts with any name of length more than 5 characters. If yes, we unglue the word at the name borderline capitalizing the fisrt letters: "Kenny Chesney" (as luck would have it, most names consist of just two parts). Otherwise, we consider the word to be something other than a name. To separate words with spaces, we use Wordninja [4]. This library has problems with names and titles, so we have to check the names before using it. Words of more than two characters are also checked in a list of slang abbreviations. For example, "bbq" becomes "barbeque". [5]

Tweet to Vector. Preprocessed tweets go into the classifier. PunFields collects Section numbers for every word and collocation in a sentence, removing duplicates and excluding stop words. Then, it builds a semantic vector of the sentence weighing how many of its elements belong to each Section. For example, the tweet "Christmas shopping on 2 hours sleep is going to be fun" has 7 words belogning to different Sections: shopping - 33; 2 - 4; sleep - 11, 27, 14, 34; go - 11, 0, 14, 35, 7; christmas - 38, 5; fun - 35; hour - 5. Its vector will be as follows: {1, 0, 0, 0, 1, 2, 0, 1, 0, 0, 0, 2, 0, 0, 2, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 1, 2, 0, 0, 1}.

Classification. When the vectors are ready, we split them into a training and test set. The classification is conducted using different Scikit-learn (Pedregosa et al., 2011) algorithms. At the present competition, we used SVM with the linear kernel, because preliminary tests using the Gold dataset with a 5-fold cross validation showed that it has the highest result among the tested algorithms: Precision=0.556, Recall=0.646, F1-score=0.596 (766 samples in a training set). However, when we used all the 3834 samples for training during the competition, they obviously created a too noisy feature space for a good generalization. Hence, the decrease in the result. At SemEval2017 (Miller et al., 2017), when we tried larger datasets for training, the result grew better. Tweets generally seem to be more "noisy" than puns due to slang, contractions, glued words, etc. although there are no strict ways of calculating this "noisyness".

Instead of SVM, we also tested a deep learning network with different architectures implemented in the Python library Keras (Chollet et al., 2015) with TensorFlow (TensorFlow Development Team, 2015) and Theano (Theano Development Team, 2016) backends, but we only managed to reach as much as 0.56 for F1-score. [6] Furthermore, we tried to replace Roget's semantic classes with a vector model realization via the Gensim library (Řehůřek and Sojka, 2010) summing every score in the 300 long word2vec representations similarly to what PunFields does with the 39 Sections. However, it did not bring any significant result either.

All in all, PunFields appears to better classify such cases where there are two evidently different topics in one utterance that are expressed by two groups of words that outnumber other (noisy) groups. In the toy examples from the previous paragraph, the "chemistry" pun belongs to the so-called homographic puns, i.e. such puns where one word is used in two meanings. Unlike them, heterographic puns are such puns where the word used in one meaning resembles in form/sounding another word. That creates a disbalance between the two groups of words representing the two clashing topics: one of the groups gets the mentioned word and the other one does not (as the word is only implied). For example, in the heterographic pun "I relish the fact that you have mustard the strength to ketchup to me" the group of "mastered" and "catch up" is hidden. Similar to heterographic puns, ironic tweets have one word

[4]https://pypi.python.org/pypi/wordninja

[5]The pipeline is available at https://github.com/evrog/PunFields.

[6]Due to limitations in the size of papers, we will not describe the architecture here. Some of the code we used for the neural network is available at our PunFields repository.

expressed in full and its opposite implied. Consequently, PunFields processes such cases worse than homographic puns.

4 Unconstrained Submission

As for the unconstrained submission, we decided to use it to test another assumption that irony is close not only to puns, but to different kinds of humor. However, as in this competition we deal with irony in Twitter, we decided to check this assumption on humorous *tweets*. We took 1,000 humorous tweets from SemEval 2017 #HashtagWars: Learning a Sense of Humor (Potash et al., 2017) and 1,000 tweets with 0 to +2 positive polarity from SemEval 2016 Task 4: Sentiment Analysis in Twitter (Nakov et al., 2016) and combined them with 2,000 tweets from the Gold dataset. For the classifier, we used Bernoulli bag-of-words SVM model with the linear kernel given as the benchmark system by the competition Organizers. After reshuffling the dataset several times, we chose that result which had a more or less equal number of items in the both classes.

The result of this test shows a decrease in the efficiency of the benchmark system. However, to our surprise it was also slightly above the chance value.

5 Conclusions

With this research, we continue to elaborate on the universal mechanism of detecting intentional lexical ambiguity in creative language. We tried to demonstrate that irony and puns share some features which can be processed by a similar algorithm. However, the results of our system, PunFields, at this competition leave much to be desired.

We are planning to replace the core of the system, the semantic scheme of 39 classes, with a more elaborate system of word2vec representations and, maybe, a deep learning classifier if it provides a better result. So far, we were unable to get the system working with these elements, but the problem was, very likely, in the technical side.

All in all, PunFields shows the result that is higher than the chance operating on the data it was not meant to process. With due elaboration, we believe, it has a greater classifying potential than what was gained at the competition.

References

Francesco Barbieri and Horacio Saggion. 2014. Modelling irony in Twitter. In *Proceedings of the Student Research Workshop at the 14th Conference of the European Chapter of the Association for Computational Linguistics*, pages 56–64.

Steven Bird, Ewan Klein, and Edward Loper. 2009. *Natural language processing with Python: analyzing text with the natural language toolkit.* " O'Reilly Media, Inc.".

François Chollet et al. 2015. Keras. https://github.com/keras-team/keras.

Elena Mikhalkova and Yuri Karyakin. 2017a. Detecting intentional lexical ambiguity in English puns. In *23rd International Conference on Computational Linguistics and Intellectual Technologies*, volume 1, pages 167–179.

Elena Mikhalkova and Yuri Karyakin. 2017b. Punfields at Semeval-2017 Task 7: Employing Roget's Thesaurus in Automatic Pun Recognition and Interpretation. In *Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017)*, pages 426–431.

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, Christian 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)*, pages 58–68.

Preslav Nakov, Alan Ritter, Sara Rosenthal, Fabrizio Sebastiani, and Veselin Stoyanov. 2016. Semeval-2016 Task 4: Sentiment analysis in Twitter. In *Proceedings of the 10th International Workshop on Semantic Evaluation (SemEval-2016)*, pages 1–18.

F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. 2011. Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research*, 12:2825–2830.

Peter Potash, Alexey Romanov, and Anna Rumshisky. 2017. Semeval-2017 Task 6:# hashtagwars: Learning a sense of humor. In *Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017)*, pages 49–57.

Radim Řehůřek and Petr Sojka. 2010. Software Framework for Topic Modelling with Large Corpora. In *Proceedings of the LREC 2010 Workshop on New Challenges for NLP Frameworks*, pages 45–50, Valletta, Malta. ELRA. http://is.muni.cz/publication/884893/en.

TensorFlow Development Team. 2015. TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems. Software available from tensorflow.org.

Theano Development Team. 2016. Theano: A Python framework for fast computation of mathematical expressions. *arXiv e-prints*, abs/1605.02688.

Cynthia Van Hee. 2017. *Can machines sense irony? : exploring automatic irony detection on social media*. Ph.D. thesis, Ghent University.

Cynthia Van Hee, Els Lefever, and Véronique Hoste. 2018. SemEval-2018 Task 3: Irony Detection in English Tweets. In *Proceedings of the 12th International Workshop on Semantic Evaluation*, SemEval-2018, New Orleans, LA, USA. Association for Computational Linguistics.