PunFields at SemEval-2017 Task 7: Employing Roget's Thesaurus in Automatic Pun Recognition and Interpretation

Elena Mikhalkova
Institute of Philology
and Journalism
Tyumen State University
Tyumen, Russia, 625003
e.v.mikhalkova@utmn.ru

Yuri Karyakin
Institute of Mathematics
and Computer Science
Tyumen State University
Tyumen, Russia, 625003
y.e.karyakin@utmn.ru

Abstract

The article describes a model of automatic interpretation of English puns, based on Roget's Thesaurus, and its implementation, PunFields. In a pun, the algorithm discovers two groups of words that belong to two main semantic fields. The fields become a semantic vector based on which an SVM classifier learns to recognize puns. A rule-based model is then applied for recognition of intentionally ambiguous (target) words and their definitions. In SemEval Task 7 PunFields shows a considerably good result in pun classification, but requires improvement in searching for the target word and its definition.

1 Introduction

The following terminology is basic in our research of puns. **A pun** is a) a short humorous genre where a word or phrase is intentionally used in two meanings, b) a means of expression the essence of which is to use a word or phrase so that in the given context the word or phrase can be understood in two meanings simultaneously. **A target word** is a word that appears in two meanings. **A homographic pun** is a pun that "exploits distinct meanings of the same written word" (Miller and Gurevych, 2015) (these can be meanings of a polysemantic word or homonyms, including homonymic word forms). **A heterographic pun** is a pun in which the target word resembles another word or phrase in spelling; we will call the latter **the second target word**. Consider the following example (the Banker joke):

> "I used to be a banker, but I lost interest."

The Banker joke is a homographic pun; *interest* is the target word. Unlike it, the Church joke below is a heterographic pun; *propane* is the target word, *profane* is the second target word:

> "When the church bought gas for their annual barbecue, proceeds went from the sacred to the propane."

Our model of automatic pun analysis is based on the following premise: in a pun, there are two groups of words and their meanings that indicate the two meanings of the target word. These groups can overlap, i.e. contain the same polysemantic words used in different meanings.

In the Banker joke, words and collocations *banker*, *lost interest* point at the professional status of the narrator and his/her career failure. At the same time, *used to*, *lost interest* tell a story of losing emotional attachment to the profession: the narrator became disinterested. The algorithm of pun recognition that we suggest discovers these two groups of words based on common semes[1] (Subtask 1), finds the words that belong to the both groups, and chooses the target word (Subtask 2), and, based on the common semes, picks up the best suitable meaning which the target word exploits (Subtask 3). In case of heterographic puns, in Subtask 2, the algorithm looks for the word or phrase that appears in one group and *not* in the other.

2 Subtask 1: Mining Semantic Fields

We will call a semantic field a group of words and collocations that share a common seme. In taxonomies like WordNet (Kilgarriff and Fellbaum, 2000) and Roget's Thesaurus (Roget, 2004) (further referred to as Thesaurus) semes appear as hierarchies of word meanings. Top-levels attract

[1]Bits of meaning. Semes are some parts of meaning present both in the word and in its hypernym. Moving up the taxonomy like Thesaurus or WordNet, hypernyms become more general, and the seme connecting them to the word becomes more general, too.

words with more general meanings (hypernyms). For example, Thesaurus has six top-level Classes that divide into Divisions that divide into Sections and so on, down to the fifth lowest level[2]. Applying such dictionaries to get semantic fields (the mentioned common groups of words) in a pun is, therefore, the task of finding two most general hypernyms in WordNet or two relevant Classes among the six Classes in Thesaurus. We chose Thesaurus, as its structure is only five levels deep, Classes labels are not lemmas themselves, but arbitrary names (we used numbers instead). Also, it allows parsing on a certain level and insert corrections (adding lemmas, merging subsections, etc.[3]). After some experimentation, instead of Classes, we chose to search for relevant Sections that are 34 subdivisions of the six Classes[4].

After normalization (including change to lowercase; part-of-speech tagging, tokenization, and lemmatization with NLTK tools (Bird et al., 2009); collocation extraction[5]; stop-words removal[6]), the algorithm collects Section numbers for every word and collocation and removes duplicates (in Thesaurus, homonyms proper can belong to different subdivisions in the same or different Sections). Table 1 shows what Sections words of the Banker joke belong to.

Then, the semantic vector of a pun is calculated. Every pun p is a vector in a 34-dimensional space:

$$p_i = p_i(s_{1i}, s_{2i}, ..., s_{34i})$$

The value of every element s_{ki} equals the number of words and collocations in a pun that belong to a Section S_k. The algorithm passes from a Section to a Section each time checking every word and collocation w_{ji} in the bunch of extracted items l_i. If a word or collocation belongs to a Section, the value of s_{ki} increases by 1:

$$s_{ki} = \sum_{j=1}^{l_i} \{1 | w_{ji} \in S_k\},$$

$$k = 1, 2, ..., 34, i = 1, 2, 3...$$

For example, the semantic vector of the Banker joke looks as follows: see Table 2.

To test the algorithm, we, first, collected 2,484 puns from different Internet resources and, second, built a corpus of 2,484 random sentences of length 5 to 25 words from different NLTK corpora (Bird et al., 2009) plus several hundred aphorisms and proverbs from different Internet sites. We shuffled and split the sentences into two equal groups, the first two forming a training set and the other two a test set. The classification was conducted, using different Scikit-learn algorithms (Pedregosa et al., 2011). We also singled out 191 homographic puns and 198 heterographic puns and tested them against the same number of random sentences.

In all the preliminary tests, the Scikit-learn algorithm of SVM with the Radial Basis Function (RBF) kernel produced the highest average F-measure results ($\bar{f} = \frac{f_{puns} + f_{random}}{2}$) for the class of puns. Table 3 illustrates results of different algorithms. The results were higher for the split selection, reaching 0.79 (homographic) and 0.78 (heterographic) scores of F-measure. The common selection got the maximum of 0.7 for average F-measure in several tests. We are inclined to think that higher results of split selection may be due to a larger training set.

3 Subtask 2: Hitting the Target Word

We suggest that, in a homographic pun, the target word is a word that immediately belongs to two semantic fields; in a heterographic pun, the target word belongs to at least one discovered semantic field and does not belong to the other. However, in reality, words in a sentence tend to belong to too many fields, and they create noise in the search. To reduce influence of noisy fields, we included such non-semantic features in the model as the

[2]The hierarchical structure of WordNet is not so transparent. According to WordNet documentation, it is rather a union of four nets: nouns, verbs, adjectives, and adverbs. Their grouping depends on the semantic mark-up. Thus, it resembles a folksonomy and its structure changes implicitly. At the same time, the documentation mentions "forty-five lexicographer files based on syntactic category and logical groupings" (wordnet.princeton.edu/wordnet/man/lexnames.5WN.html) which can count as "owner-prescribed" structural subdivisions. Hypernymic relations are more specific among verbs and nouns, whereas adjectives often relate antonymically. Also, all WordNet nouns finally relate to the synset "entity".

[3]We edited Thesaurus by adding words that were absent in it. If a word in a pun was missing in Thesaurus, the system checked up for its hypernyms in Wordnet and added the word to those Sections in Thesaurus that contained the hypernyms. We merged some small closely-related Sections, as well. Originally, there used to be 39 Sections. Editing was done after experimenting with training data.

[4]Sections are not always *immediate* subdivisions of a Class. Some Sections are grouped in Divisions.

[5]To extract collocations and search for them in Thesaurus, we applied our own procedure based on the part-of-speech analysis.

[6]After lemmatization, all words are analyzed in collocations, but only nouns, adjectives, and verbs compose a list of separate words.

Word	**Section No., Section name in Thesaurus**
I	-
use	24, Volition In General
	30, Possessive Relations
to	-
be	0, Existence
	19, Results Of Reasoning
a	-
banker	31, Affections In General
	30, Possessive Relations
but	-
lose	21, Nature Of Ideas Communicated
	26, Results Of Voluntary Action
	30, Possessive Relations
	19, Results Of Reasoning
interest	30, Possessive Relations
	25, Antagonism
	24, Volition In General
	7, Causation
	31, Affections In General
	16, Precursory Conditions And Operations
	1, Relation

Table 1: Semantic fields in the Banker joke

p_{Banker}	$\{1, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 2, 0, 1, 0, 0, 2, 1, 1, 0, 0, 0, 4, 2, 0, 0\}$

Table 2: Semantic vector of the Banker joke

Method	**Precision**	**Recall**	**F-measure**
Common selection			
SVM with linear kernel	0.67	0.68	0.67
SVM with polynomial kernel	0.65	0.79	0.72
SVM with Radial Basis Function (RBF) kernel	0.70	0.70	0.70
SVM with linear kernel, normalized data	0.62	0.74	0.67
Homographic puns			
SVM with RBF kernel	0.79	0.80	0.79
Multinomial Naive Bayes	0.71	0.80	0.76
Logistic Regression, standardized data	0.77	0.71	0.74
Heterographic puns			
SVM with RBF kernel	0.77	0.79	0.78
Logistic Regression	0.74	0.75	0.74

Table 3: Tests for pun recognition.

tendency of the target word to occur at the end of a sentence and part-of-speech distribution, given in (Miller and Gurevych, 2015). A-group (W_A) and B-group (W_B) are groups of words in a pun that belong to the two semantic fields sharing the target word. Thus, for some s_{ki}, k becomes A or B [7]. A-group attracts the maximum number of words in a pun:

$$s_{Ai} = \max_k s_{ki}, k = 1, 2, ..., 34$$

In the Banker joke, $s_{Ai} = 4, A = 30$ (Possessive Relations); words that belong to this group are *use*, *lose*, *banker*, *interest*. B-group is the second largest group in a pun:

$$s_{Bi} = \max_k (s_{ki}/s_{Ai}), k = 1, 2, ..., 34$$

In the Banker joke, $s_{Bi} = 2$. There are three groups of words that have two words in them: $B_1 = 19$, Results Of Reasoning: *be*, *lose*; $B_2 = 24$, Volition In General: *use*, *interest*; $B_3 = 31$, Affections In General: *banker*, *interest*. Ideally, there should be a group of about three words, and collocations, describing a person's inner state (*used to be*, *lose*, *interest*), and two words (*lose*, *interest*) in W_A are a target phrase. However, due to the shortage of data about collocations in dictionaries and weak points of collocation extraction, W_B splits into several smaller groups. Consequently, to find the target word, we have to appeal to other word features. Testing the system on homographic puns, we relied on the polysemantic character of words. If in a joke, there is more than one value of B, W_B candidates merge into one, with duplicates removed, and every word in W_B becomes the target word candidate: $c \in W_B$. In the Banker joke, W_B is a list of *be*, *lose*, *use*, *interest*, *banker*; $B = \{19, 24, 31\}$. Based on the definition of the target word in a homographic pun, words from W_B that are also found in W_A should have a privilege. Therefore, the first value v_α assigned to each word is the output of the Boolean function:

$$v_\alpha(c) = \begin{cases} 2 & \text{if}(c \in W_A) \wedge (c \in W_B) \\ 1 & \text{if}(c \notin W_A) \wedge (c \in W_B) \end{cases}$$

The second value v_β is the absolute frequency of a word in the union of B_1, B_2 etc., including duplicates: $v_\beta(c) = f_c(W_{B_1} \cup W_{B_2} \cup W_{B_3})$.

[7] s_{ki} is always an integer; W_A and W_B are always lists of words; A is always an integer, B is a list of one or more integers.

Word form	v_α	v_β	v_γ	v_δ	$v_{W_{Bk}}$
be	1	1	4	0.338	1.352
lose	2	1	9	0.338	6.084
use	2	1	2	0.338	1.352
interest	2	2	10	0.502	**20.08**
banker	2	1	6	0.502	6.024

Table 4: Values of the Banker joke.

The third value v_γ is a word's position in the sentence: the closer the word is to the end, the bigger this value is. If the word occurs several times, the algorithm counts the average of the sums of position numbers.

The fourth value is part-of-speech probability v_δ. Depending on the part of speech, the word belongs to, it gets the following rate:

$$v_\delta(c) = \begin{cases} 0.502 & if \quad c \quad - \quad Noun \\ 0.338 & if \quad c \quad - \quad Verb \\ 0.131 & if \quad c \quad - \quad Adjective \\ 0.016 & if \quad c \quad - \quad Adverb \\ 0.013 & \qquad\qquad\quad otherwise \end{cases}$$

The final step is to count rates using multiplicative convolution and choose the word with the maximum rate:

$$z_1(W_B) = \left\{c | \max_c (v_\alpha \times v_\beta \times v_\gamma \times v_\delta)\right\}$$

Values of the Banker joke are illustrated in Table 4.

In the solution for heterographic puns, we built a different model of B-group. Unlike homographic puns, here the target word is missing in W_B (the reader has to guess the word or phrase homonymous to the target word). Accordingly, we rely on the completeness of the union of W_A and W_B: among the candidates for W_B (the second largest groups), such groups are relevant that form the longest list with W_A (duplicates removed). In Ex. 2 (the Church joke), $W_A = go, gas, annual, barbecue, propane$, and two groups form the largest union with it: $W_B = buy, proceeds + sacred, church$. Every word in W_A and W_B can be the target word. The privilege passes to words used only in one of the groups. Ergo, the first value is:

$$v_\alpha(c) = \begin{cases} 2 & \text{if}(c \in W_A) \oplus (c \in W_B) \\ 1 & \text{otherwise} \end{cases}$$

Frequencies are not calculated; values of position in the sentence and part-of-speech distribution remain the same. The output of the function is:

$$z_1(W_B) = \left\{c | \max_c (v_\alpha \times v_\gamma \times v_\delta)\right\}$$

Word form	v_α	v_γ	v_δ	$v_{W_{Ak}}, v_{W_{Bk}}$
propane	2	18	0.502	**18.072**
annual	2	8	0.131	2.096
gas	2	5	0.502	5.02
sacred	2	15	0.338	10.14
church	2	3	0.502	3.012
barbecue	2	9	0.502	9.036
go	2	12	0.338	8.112
proceeds	2	11	0.502	11.044
buy	2	4	0.338	2.704

Table 5: Values of the Church joke.

Values of the Church joke are illustrated in Table 5.

4 Subtask 3: Mapping Roget's Thesaurus to Wordnet

In the last phase, we implemented an algorithm that maps Roget's Sections to synsets in Wordnet. In homographic puns, definitions of a word in Wordnet are analyzed similarly to words in a pun when searching for semantic fields the words belong to. For example, words from the definitions of the synset *interest* belong to the following Roget's Sections: Synset(interest.n.01)=a sense of concern with and curiosity about someone or something: (21, 19, 31, 24, 1, 30, 6, 16, 3, 31, 19, 12, 2, 0); Synset(sake.n.01)=a reason for wanting something done: 15, 24, 18, 7, 19, 11, 2, 31, 24, 30, 12, 2, 0, 26, 24, etc. When A-Section is discovered (for example, in the Banker joke, A=30 (Possessive Relations)), the synset with the maximum number of words in its definition that belong to A-Section becomes the A-synset. The B-synset is found likewise for the B-group, with the exception that it should not coincide with A-synset. In heterographic puns, the B-group is also a marker of the second target word. Every word in the index of Roget's Thesaurus is compared to the known target word using Damerau-Levenshtein distance. The list is sorted in the increasing order, and the algorithm begins to check what Roget's Sections every word belongs to, until it finds the word that belongs to a Section (or the Section if there is only one) in the B-group. This word becomes the second target word.

Nevertheless, as we did not have many trial data, but for the four examples released before the competition, the first trials of the program on a large collection returned many errors, so we changed the algorithm for the B-group as follows.

Homographic puns, first run. B-synset is calculated on the basis of sense frequencies (the output is the most frequent sense). If it coincides with A-synset, the program returns the second frequent synset.

Homographic puns, second run. B-synset is calculated on the basis of Lesk distance using built-in NLTK Lesk function (Bird et al., 2009). If it coincides with A-synset, the program returns another synset based on sense frequencies, as in the first run.

Heterographic puns, first run. The second target word is calculated based on Thesaurus and Damerau-Levenshtein distance; words missing in Thesaurus are analyzed as their WordNet hypernyms. In both runs for heterographic puns, synsets are calculated using the Lesk distance.

Heterographic puns, second run. The second target word is calculated on the basis of Brown corpus (NLTK (Bird et al., 2009)): if the word stands in the same context in Brown as it is in the pun, it becomes the target word. The size of the context window is (0; +3) for verbs, (0;+2) for adjectives; (-2;+2) for nouns, adverbs, and other parts of speech within the sentence where a word is used.

Type of pun	Precision	Recall	F1
Ho.	0.7993	0.7337	0.7651
Change	-0.0026	-0,0448	-0,0249
He.	0.7580	0.5940	0.6661
Change	-0.0005	-0,0386	-0,0237

Table 7: Task 1, overlap removed.

5 Results

Table 6 illustrates SemEval results (Miller et al., 2017) of our system PunFields[8] (Ho. - homographic, He. - heterographic).

In one of the reviews, we were prompted to check if the training and test set overlap. The overlap was 742 puns (30% of the pun training set). When we removed them and an equal number of random sentences from the training set and recalcualted the result using Gold set, the result fell within the scope of 4.5% which puts PunFields at the same place in the scoring table. We tend to think that the results went down not only because of the overlap removal, but also due to the reduction of the training set by 30%. This encourages us to state that the sematic fields hypothesis on which we build the classification model was tested successfully.

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

Task	Precision	Recall	Accuracy	F1
1, Ho.	0.7993	0.7337	0.6782	0.7651
1, He.	0.7580	0.5940	0.5747	0.6661
Task	**Coverage**	**Precision**	**Recall**	**F1**
2, Ho., run 1	1.0000	0.3279	0.3279	0.3279
2, Ho., run 2	1.0000	0.3167	0.3167	0.3167
2, He., run 1	1.0000	0.3029	0.3029	0.3029
2, He., run 2	1.0000	0.3501	0.3501	0.3501
3, Ho., run 1	0.8760	0.0484	0.0424	0.0452
3, Ho., run 2	1.0000	0.0331	0.0331	0.0331
3, He., run 1	0.9709	0.0169	0.0164	0.0166
3, He., run 2	1.0000	0.0118	0.0118	0.0118

Table 6: Competition results.

In comparison with the results of other systems, PunFields showed its best performance in classification which is probably due to the following factors. First, supervised learning algorithms like SVM have been historically very efficient in classification tasks. Although they fail on short texts more often than on long ones. Second, the training set was rather large (twice larger than during experimentation). However, the results for heterographic puns are lower than even in the preliminary tests. Probably, our training set contains too few heterographic puns, or their vectors are more alike with random sentences (or, rather, more scattered across the vector space).

The rule-based system of finding the target word and its WordNet meaning turned out to be less successful. Although later after fixing some programming errors, we managed to improve the result for Subtask 2[9]. Furthermore, multiplying values turned out to be a wrong decision, and the reasons for it will be reflected in a further publication. As for Subtask 3, we tried to combine two very different dictionaries: Roget's Thesaurus and Wordnet. When used in Subtask 1, Thesaurus provided reliable information on meanings of puns and was more or less successful in Subtask 2 (considering the mentioned improvements). But in Subtask 3 it was very much below the baseline results suggested by the Task Organizers. At the same time, we did not have any experimental data to test different variations of the algorithm before the competition. Especially, it concerns combinations of the own system with existing methods of WordNet (Lesk and sense frequencies). Initially, employing Thesaurus instead of WordNet was a solution made for convenience of parsing and experimenting with data. Further research will show whether these dictionaries can combine and solve issues together, or one of them should become a more preferrable source of data.

[9]The current result is 0.5145 for homographic puns, 0.3879 for heterographic puns.

References

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

Adam Kilgarriff and Christiane Fellbaum. 2000. Wordnet: An electronic lexical database.

Tristan Miller and Iryna Gurevych. 2015. Automatic disambiguation of English puns. In *ACL (1)*. pages 719–729.

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)*. Association for Computational Linguistics, Vancouver, Canada, pages 60–70. http://www.aclweb.org/anthology/S17-2005.

Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, et al. 2011. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research* 12(Oct):2825–2830.

Peter Mark Roget. 2004. Roget's thesaurus of English words and phrases. Project Gutenberg.