Recognizing Humor Without Recognizing Meaning

Jonas Sjöbergh and Kenji Araki

Graduate School of Information Science and Technology
Hokkaido University
Sapporo, Japan
{js, araki}@media.eng.hokudai.ac.jp

Abstract. We present a machine learning approach for classifying sentences as one-liner jokes or normal sentences. We use no deep analysis of the meaning to try to see if it is humorous, instead we rely on a combination of simple features to see if these are enough to detect humor. Features such as word overlap with other jokes, presence of words common in jokes, ambiguity and word overlap with common idioms turn out to be useful. When training and testing on equal amounts of jokes and sentences from the British National Corpus, a classification accuracy of 85% is achieved.

1 Introduction

Humor is an interesting part of human language use, usually used many times every day. Research on humor has been done in many different fields, such as psychology, philosophy, sociology and linguistics. When it comes to computational linguistics, two main approaches have been explored. One is humor generation, where systems for generating usually quite simple forms of jokes, e.g. word play jokes, have been constructed [1–4]. The second is humor recognition, where systems to recognize whether a text is a joke or not have been constructed [5, 6], which is what this paper is about.

Theories of humor often state that humor appears when a setup suggests one interpretation but is followed by an ending that does not agree with this interpretation, but is consistent with some other interpretation of the setup. Thus, the listener must shift from the assumed reference frame to a new interpretation, which can be humorous.

Doing this kind of analysis automatically with current language processing techniques seems prohibitively difficult. In our experiments we instead use quite low level information sources and see if enough hints can be gathered to determine if something is a joke or not without actually having any understanding of the meaning. Our work closely resembles the work in [6], where humor detection was treated as a text classification problem. Machine learning using content based information and some stylistic features present in many jokes gave results far above baseline performance.

We also treat humor detection as text classification and use machine learning. Our method differs in what features are made available to the machine learning system. Examples of useful features are presence of words common in jokes, word overlap with known jokes and word overlap with idiomatic expressions.

2 Method for Detecting Humor

There are many machine learning algorithms available, and in many cases the performance is very similar. We put very little effort into selecting a good machine learning algorithm, instead focusing on the information features given to the machine learner. The interesting part is thus what language features can be used for classifying text as jokes or not.

The machine learning method we use is very simple. For each feature a threshold value is calculated separating the training examples into two groups. The threshold is selected so as to make the mean entropy of these as low as possible. To classify a new example, which group it belongs to is checked for each feature, and the proportion between positive and negative examples in this group used. The proportion of positive examples for each group the example belongs to is multiplied together and compared to the product in the same way for the negative examples. If the product for the positive examples is larger the example is classified as a joke, otherwise as a non-joke. This method is not very powerful, but fast.

Since we have many features that represent almost the same information, performing some feature reduction to remove redundant or useless features improves performance. It is also interesting in the respect that it shows what kind of information is useful in detecting humor.

We perform a very simplistic feature reduction. Using disjoint sets of test data (i.e. training and classifying several times), we remove one feature at a time. The feature that gives the best result when not present is then permanently removed, and the process repeated. When all features have been removed, the best result achieved in the process is found, and all features that were removed when reaching this result are discarded, the rest kept.

The rest of this section presents the features given to the machine learner. Feature names are given in *italics*. Most features are developed in a very shallow way. This means that they might be too simplistic to capture the sought after information, but we wanted to see how well the classification could be done using only readily available and simple to use information. Many much more sophisticated features than ours could be designed and implemented, at least for narrow domains.

2.1 Text Similarity

Since we are doing text classification, it is of course useful to use common text classification features such as the closeness in some word space model between a new sentence and the labeled sentences in the training data. The first group of features contain information of this type:

Closest joke, the word overlap between the sentence to classify and the joke in the training corpus with the highest word overlap. Overlap is divided by the square root of the number of words in the sentences in the training corpus, since otherwise the overlap will tend to be higher for long sentences (since they contain more words to overlap). *Closest non-joke*, is the same but for non-joke sentences, and *closest difference*, is the difference between the two previous features.

Finally, we have the *knn* feature, a weighted vote between the five closest sentences in the training data, each with a weight equal to their word overlap. Jokes have positive sign and non-jokes have negative sign.

2.2 Joke Words

Some words are common in jokes, while some are rarely used to be funny. For instance, according to the Laughlab study [7], animals are often mentioned in jokes, especially ducks. To capture this, words that occur at least five times in the jokes in the training data and are at least five times more common in the jokes than in the non-jokes are collected in a list. Each word is given a weight that is the relative frequency of the word among the jokes divided by the relative frequency among the non-jokes.

To capture short phrases that are common in jokes, such as "change a light bulb", similar lists are constructed for word bigrams and trigrams too. To detect signs of unfunniness, the same is also done for words that are underrepresented in jokes. Using this information the following features are calculated:

Joke words, the sum of the weights of all words common in jokes. *Joke bigrams*, similarly for bigrams and *joke trigrams* for trigrams. *Joke word pairs*, the same for any pair of words occurring in the sentence, and *joke word triples* similarly for three words from the same sentence. Finally, *joke words and pairs*, the sum of *joke words* and *joke word pairs* is also used.

Non-joke words, the same as for *joke words*, except using words rare in jokes but common in non-jokes. *Non-joke bigrams*, *non-joke trigrams*, *non-joke word pairs*, *non-joke word triples*, and *non-joke words and pairs* in the same way as above.

These features are, like the previous section, fairly typical text classification features, though instead of using typical weighting schemes such as Inverse Document Frequency, we use weighting related to how common the words are in jokes compared to non-jokes. Since these lists are based on the sentences in the training data, these features of course give a much stronger separation between jokes and non-jokes in the training data than can be expected in sentences to classify later.

2.3 Ambiguity

Most theories on humor agree that ambiguity is useful in humor. A simple feature to capture this is the ambiguity of the words in the sentence. The ambiguity of a word is measured by looking up the word in the online dictionary at dictionary.com and counting the number of senses listed. Two features are based on this, *average ambiguity*, the average number of word senses of the words in the sentence, and *maximum ambiguity*, the highest ambiguity of the words in the sentence.

Of course, ambiguity is not only caused by word senses. Another measure of ambiguity is calculated by running the link parser [8] on each sentence. The number of possible parses found is then used as the feature *parse ambiguity.* To account for the fact that longer sentences generally have more possible parses, the value is divided by the sentence length in words.

2.4 Style

In a very similar study to ours [6] an analysis of the jokes in their corpus showed some possibly useful characteristics of jokes. For instance, they often contain human related words such as "you" or "I"; they often use negations, dirty words, antonymy etc. Inspired by their results, we use the following stylistic features:

Dirty, the number of dirty words present in the sentence. A list of 2,500 dirty words downloaded from the Internet was used to decide if a word is dirty or not.

Human, the number of words present from the list: "you", "your", "I", "me", "my", "man", "woman", "he", "she", "his", "her", "guy", and "girl"; and *negation*, the number of occurrences of "not" or "n't".

Using the CMU Pronunciation Dictionary[1] for finding the pronunciation of words, three features were calculated: *rimes*, the number of word pairs that have at least four letters, at least one of which is a vowel, pronounced the same way at the end of the words; *similar*, the same as *rimes* but using the beginning of the words instead of the end; and *alliteration*, the number of words that start with the same sound and have a maximum of two words in between.

The *new words* feature also uses the CMU Pronunciation Dictionary, simply counting the number of words that are not in the dictionary.

Repeated words features: *repeated words*, the number of words of at least five letters that occur more than once, and *repeated substring*, the longest substring (in letters) that occurs more than once in the sentence. *Average repeated words* and *average repeated substring* are the same but divided by sentence length.

Antonymy features look up the antonyms of a word at dictionary.com. If any of the listed antonyms are present in the sentence, a score of one divided by the number of possible antonyms is given. Two features are based on this, *maximum antonymy score*, the highest value (i.e. the antonym pair present with the fewest other possible antonyms), and *antonymy*, the sum of all antonymy scores.

2.5 Idiomatic Expressions

One-liner jokes are often amusing reformulations based on common idioms or proverbs. An Internet search for "English proverbs" and taking the first results

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

that contained actual collections of proverbs and idioms gave a list of 3,000 proverbs and idioms. The following features are used based on this:

Short idioms, the number of idioms of two or three words present in the sentence, and *long idioms*, similarly for those of length at least four.

Idiom overlap, the number of proverbs or idioms of at least four words that overlap with the sentence for at least four words in a row. *Vague overlap* is the same, though one non-matching word can be skipped in the middle of the match.

Longest overlap is the longest matching word sequence of any idiom, and *longest common substring* is the longest common substring (in words) between the sentence and any idiom. Both of these scores are divided by the square root of the length of the idiom, since longer idioms often overlap other sentences on common words such as prepositions.

3 Evaluation

In our evaluations we use a corpus of 6,100 jokes collected from the Internet. All jokes were automatically downloaded from collections of jokes were the classification was "oneliner". They are thus generally of the oneliner type, though some jokes that are perhaps not really oneliners (such as "What do you call 5000 dead lawyers at the bottom of the ocean? A good start!") were also downloaded if they were classified as oneliners by the original collector. All jokes written in English.

After downloading, all jokes were manually checked to see if they were indeed jokes, removing for instance descriptions such as "Dirty one-liners section 2". Jokes with a high word overlap with other jokes in the corpus were also checked and only one version of the same joke was kept. The shortest joke is only one word long ("Contentsmaysettleduringshipping"), the longest is 80 words. On average, the jokes contain 12 words.

As non-jokes we use sentences from the British National Corpus, BNC [9], which was the hardest to distinguish from one-liners in other studies [6], achieving an accuracy of 79%. We use a different set of sentences from the BNC, and a different set of jokes, so results are not necessarily comparable though.

Data for training and testing was created by taking an equal number of jokes and BNC sentences to each data set. Any sentences from the BNC that were shorter than the shortest joke or longer than the longest joke were ignored. The data was divided into 95% training data and 5% test data. More training data normally gives better results with machine learning, but repeating the training and testing for each of the (in our case) 20 test sets takes quite some time. A one in twenty split was deemed a good compromise between the time required to test on a large number of examples and the amount of training data made available to the machine learner.

Five sets of test data were used for feature reduction. The system was then evaluated using the remaining 15 test data sets, which gives slightly more than 9,000 sentences to classify in the tests.

Which features were removed during the feature reduction is shown in Table 1. Since there are redundant features of several of the feature types, many are removed.

Table 1. Features that are discarded during the feature reduction.

Maximum ambiguity	*Alliteration*
Maximum antonymy score	*Antonymy*
Joke word trigrams	*Joke word pairs*
Non-joke word pairs	*Human*
Average repeated words	*Negation*
Average repeated substring	*Short idiom*
Vague overlap	*Long idiom*

The general usefulness of the different feature types is shown in Table 2, where the classification accuracy based only on one group of features or when a whole group is removed is shown.

Table 2. Classification accuracy (%). Accuracy when removing or using only a single feature group is also shown.

	Only	Without
All features	85.4	50.0
Similarity	75.7	83.8
Joke words	84.1	76.8
Ambiguity	62.5	84.8
Style	59.1	85.4
Idioms	63.5	85.0

Presence of words common or rare in jokes seems to be the most useful information, giving almost as high accuracy as when using all features. This is followed by word overlap with training examples. These are fairly standard text classification features, so that they are useful in joke related text classification too is not surprising. Similarity to idioms or proverbs helps a little, as does ambiguity. The style features are sadly of little use together with the other features, though using only these is a little better than guessing, so they do provide some information.

We achieved a total classification accuracy of 85%, which is a lot better than random guessing (50%). It is also higher than previous results on similar data, though the results are not necessarily comparable. Without feature reduction, the accuracy is 81%. BNC seems to be relatively hard to distinguish from jokes, compared to other corpora. For example, a quick evaluation using sentences from the Open Mind Common Sense project [10] instead gave 93% accuracy.

4 Conclusions

We presented a text classification approach using machine learning to classify sentences as jokes or non-jokes. We made many different types of features available to the machine learner, and then did some simple feature reduction to find out what type of information is useful in detecting jokes. The classification accuracy, 85%, was higher than other reported results.

The most useful information was classical text classification features, such as word overlap with training examples or the presence of certain words. For the last type of feature, we use a novel weighting scheme, giving weight to words in proportion to how much more common or rare they are in jokes than in normal texts. Using this feature type alone gives almost as good accuracy as when using all feature types. We believe this means that jokes differ from other types of texts in what topics they treat, which agrees with other findings, such as the fact that dirty words or human related words are over represented in jokes.

We also believe that the low contribution from the other feature types is in part caused by our very shallow implementation of the features. While features such as dirty words and human related words will likely not contribute that much new information not already present in the content based features (since they too are basically content based features), other features could likely be improved. For instance, ambiguity in jokes is usually revealed by the punch line. Taking this into account more sophisticated ambiguity features than the average number of word senses in the sentence or the total number of parses possible could be created. One example could be how much of the parse ambiguity is caused by the end of the sentence being ambiguous, another is checking the number of word senses of the last word that "make sense" in the current context or how difficult it is to determine what sense is used this time.

Other features are probably relevant for only quite few jokes, thus being largely drowned out by other examples when using our fairly simplistic machine learning system. One example is the repeated words feature. Some jokes like "Kids in the back seat cause accidents; accidents in the back seat cause kids." use a lot of repetition. These are so rare, though, that the machine learner does not consider it important to distinguish sentences with very high repetition from other sentences. Using a different machine learning system might make these features more valuable.

Acknowledgments

This work has been funded by The Japanese Society for the Promotion of Science, (JSPS).

References

1. Binsted, K.: Machine Humour: An Implemented Model of Puns. PhD thesis, University of Edinburgh, Edinburgh, United Kingdom (1996)

2. Binsted, K., Takizawa, O.: BOKE: A Japanese punning riddle generator. Journal of the Japanese Society for Artificial Intelligence **13**(6) (1998) 920–927
3. Yokogawa, T.: Generation of Japanese puns based on similarity of articulation. In: Proceedings of IFSA/NAFIPS 2001, Vancouver, Canada (2001)
4. Stark, J., Binsted, K., Bergen, B.: Disjunctor selection for one-line jokes. In: Proceedings of INTETAIN 2005, Madonna di Campiglio, Italy (2005) 174–182
5. Taylor, J., Mazlack, L.: Toward computational recognition of humorous intent. In: Proceedings of Cognitive Science Conference 2005 (CogSci 2005), Stresa, Italy (2005) 2166–2171
6. Mihalcea, R., Strapparava, C.: Making computers laugh: Investigations in automatic humor recognition. In: Proceedings of HLT/EMNLP, Vancouver, Canada (2005)
7. Wiseman, R.: Laughlab: The Scientific Search For The World's Funniest Joke. Random House (2002)
8. Grinberg, D., Lafferty, J., Sleator, D.: A robust parsing algorithm for link grammars. In: Proceedings of the Fourth International Workshop on Parsing Technologies, Prague, Czech Republic (1995)
9. Burnard, L.: The Users Reference Guide for the British National Corpus (1995)
10. Singh, P.: The public acquisition of commonsense knowledge. In: Proceedings of AAAI Spring Symposium on Acquiring (and Using) Linguistic (and World) Knowledge for Information Access, Palo Alto, California (2002)