# Tell me a good joke!

Evelina Dumitrescu
Marian Bou

March 26, 2015

## 1 Introduction

In this project, we want to determine what makes a good joke and as relevant data for telling the quality of the joke we considered the users' comments. This turns out to be a problem of sentiment analysis and classification. Based on the emotions extracted from the comments, we can classify them as positive, negative or neutral and tell how funny a joke is. For the moment, we chose to limit to the comments from the joke pages available on Facebook (ie: 9GAG, Joke of the Day). In the second section we present current techniques used for sentiment analysis and classification and in the third section we present our approach.

## 2 Related work

In the last decade, the subject of sentiment analysis and classification for social media networks has received increased attention. We have identified three major techniques relevant for our project: lexicon based, noisy labels and machine learning methods.

The first approach consists in using a dictionary of words or concepts annotated with their semantic orientation or polarity. In order to achieve a better precision and performance, domain-specific lexicons should be used, providing better word polarities for a given domain. However, building such lexicons is extremely expensive, so many systems use a general purpose lexicon. It is common to extract Part of Speech (POS) information for overcoming word-sense ambiguity and to consider some parts to be more relevant than others (ie: adjective and adverbs can provide more intense emotions than verbs or nouns). Examples of domain-independent dictionaries are SenticNet [1] and SentiWordNet [2]. SenticNet uses the bag of concepts approach, inferring the polarity of common sense concepts from natural language text at a semantic level, rather than at the syntactic level. It integrates approximative 15000 entries. On the other side, the SentiWordNet lexicon uses the "bag of words" technique and is based on an English dictionary called WordNet. It groups grammatical categories into synonym sets called synsets. SentiWordNet associates three scores for a synset to indicate the sentiment from the text: positive, negative

[1] http://sentic.net/
[2] http://sentiwordnet.isti.cnr.it/

and neutral. It has over 100000 entries. Lexicon based techniques are proposed in [1], [2], [3].

Due to the fact that in social media platforms the majority of messages are short sized (ie: Facebook status messages, Twitter reviews, some comments), the noisy label technique can lead also to good results. [4] and [5] use emoticons as noisy labels to train classifiers on a dataset collected from the Usenet newsgroup or from Twitter reviews. One can assume that an emoticon represents an emotion and all the words from the message are related to this emotion.

Alternatively, supervised machine learning techniques are used to build models or discriminators for different classes. For instance, [6] uses the Naive Bayes algorithm to separate positive reviews from the negative ones. Other research articles [7] realise a comparative approach between several machine learning algorithms, such as Naive Bayes, Support Vector Machines and Maximum Entropy.

# 3 Proposed solution

In this section we describe our approach for classifying jokes and for evaluation their perceived level of amusement. Our approach can be structured in five stages: corpus creation, composing the training data set, building the classfier, evaluate new jokes and determine the accuracy of our system. For creating the corpus, we extract comments from the joke pages available on Facbook using the Facebook Graphs API [3]. Using the Natural Language Toolkit(NLTK) [4], we tokenize the sentences and apply stemmatization algorithms on the resulting words. For the begining, we will work with "bag of words". We consider emoticons as standalone words. Increased attention should be given for repteated letters (ie: "happy"), negations, contractions and stopwords elimination.

In order to form the training data set, we split the processed comments into positive and negative/neutral categories. For every word from a comment, compute the relevance for the category that the specific comment belongs. In order to do this, we use the term frequency-inverse document frequency(TF-IDF) score. This is a numerical statistic metric that increases proportionally to the number of times a word appears in the document, but is offset by the frequency of the word in the corpus; this helps to adjust to the fact that some words appear more frequent in general. For instance, "laugh", "funny", ":D" are relevant for a positive comment and "cry", "horrible", "disgusting" and ";(" are relevant for a negative comment.

For classifying new comments, we intend to use the Transformed Weight-normalized Complement Naive Bayes [8]. We need to take into account also the TF-IDF score, not just the probabilities, because it is important to know its relevance, besides how frequent the term occurs in a specific class. The Scikit-learn [5] framework already has this implemented.

To evaluate the quality of the joke, we need to count the number of positive/negative comments. In order to tell how funny is the joke considered by a specific user, we can use a score weighted by the positive/negative polarity, but also by the relevance for the

[3] https://developers.facebook.com/docs/graph-api

[4] http://www.nltk.org/

[5] scikit-learn.org/

positive category $\sum_{word \in Words} Polarity_{word} * TFIDF_{word}$. For the polarity of words, we can use the SenticNet database [6] and include additional polarities for emoticons. For instance, "think" and "funny" have positive polarities of 0.061 and of 0.619; however, "think" won't be as relevant as "funny". On the other hand, "ugly" has a negative polarity of -0.581.

It is essential to know how accurate our classifier is. To achieve this, we should use testing samples for comments and see if classifier's prediction corresponds to our judgement. We need to identify false positives (FP), false negatives (FN), true positives (TP) and true negative (TN) cases. Furthermore, the F1, precision (positive predictive value), recall (sensitivity, true positive rate) scores have to be computed for measuring the efficiency.

$Precision = \frac{TP}{TP+FP}$

$Recall = \frac{TP}{TP+FN}$

$F1 = 2 * \frac{Precision*Recall}{Precision+Recall}$

Crossvalidation aproaches might help improve the evaluation.

# References


[1] P. Gonçalves, M. Araújo, F. Benevenuto, and M. Cha, "Comparing and combining sentiment analysis methods," in *Proceedings of the first ACM conference on online social networks*, pp. 27–38, ACM, 2013.

[2] M. Taboada, J. Brooke, M. Tofiloski, K. Voll, and M. Stede, "Lexicon-based methods for sentiment analysis," *Computational linguistics*, vol. 37, no. 2, pp. 267–307, 2011.

[3] G. Gezici, R. Dehkharghani, B. Yanikoglu, D. Tapucu, and Y. Saygin, "Su-sentilab: A classification system for sentiment analysis in twitter," in *Proceedings of the International Workshop on Semantic Evaluation*, pp. 471–477, 2013.

[4] J. Read, "Using emoticons to reduce dependency in machine learning techniques for sentiment classification," in *Proceedings of the ACL Student Research Workshop*, pp. 43–48, Association for Computational Linguistics, 2005.

[5] A. Pak and P. Paroubek, "Twitter as a corpus for sentiment analysis and opinion mining.," in *LREC*, vol. 10, pp. 1320–1326, 2010.

[6] B. Pang, L. Lee, and S. Vaithyanathan, "Thumbs up?: sentiment classification using machine learning techniques," in *Proceedings of the ACL-02 conference on Empirical methods in natural language processing-Volume 10*, pp. 79–86, Association for Computational Linguistics, 2002.

[7] A. Go, R. Bhayani, and L. Huang, "Twitter sentiment classification using distant supervision," *CS224N Project Report, Stanford*, pp. 1–12, 2009.

[8] J. D. Rennie, L. Shih, J. Teevan, D. R. Karger, *et al.*, "Tackling the poor assumptions of naive bayes text classifiers," in *ICML*, vol. 3, pp. 616–623, Washington DC), 2003.


[6] view-source:http://sentic.net/api/en/concept/