Sentiment Analysis on Movie Reviews

Mihaela Sorostinean, Katia Sana, MOHAMED Mohamed, Amal Targhi

February 12, 2017

1 Introduction

Expressing opinions and posting reviews about places visited or movies seen has become really popular nowadays. This has led to the need to automatically make sense of this huge amount of data. The human language is complex therefore teaching a machine to analyze the various grammatical nuances, cultural variations, slang and misspellings that occur in reviews provided by users is a difficult process. Teaching a machine to understand how context can affect tone is even more difficult. Advancements in machine learning and natural language processing techniques made it possible to analyze user reviews and identify the user's opinions towards them. This methods of sentiment analysis are useful in a wide range of domains, such as business or politics. The purpose of the challenge proposed by Kaggle is to experiment different machine learning models for the task of sentiment analysis, using the Rotten tomatoes movie review corpus. More specifically, phrases from the dataset have to be classified in 5 categories, according to the intended tone of the user: negative, somewhat negative, neutral, somewhat positive, positive.

This paper represents a preliminary report on our project. We start with a study of the current state-of-the-art for sentiment analysis in order to gain a deeper understanding of the problem as well as the methods used to approach it. In the third part we present the database provided on this challenge, followed by the preprocessing techniques which we applied on this database. Thereafter, we present the methods we have investigated and implemented so far. Finally we present our preliminary conclusions on this task as well as the directions which we will develop next.

2 State of the art

This Kaggle competition has been inspired by the work of Socher *et al.* [9], in which the authors propose the use of a Recursive Neural Network that builds on top of grammatical structures. Moreover, they introduce a dataset called Sentiment Treebank, which contains sentiment labels for 215,154 phrases in the parse trees of 11,855 sentences. By applying their technique on this dataset, the authors reported a 5% improvement in the accuracy of sentiment prediction compared with the current state-of-the-art.

While the method presented above showed best results for the task of sentiment analysis, we also investigated other works published on this subject. The Rotten tomatoes dataset, which is employed in this challenge was originally collected by Pang *et al.* [5]. In this work, the authors propose a meta-algorithm based on a metric labeling of the phrases and used an SVM for the classification task. A general approach observed for sentiment analysis is the use of bag-of-words for the data representation [6]. In another work that meant to analyze social media data and extract user's sentiments, Pak *et al.* [4] used N-grams and POS-tags features to train a Multinomial Naive Bayes Classifier.

3 The database

The data provided for this challenge is comprised of phrases extracted from the Rotten tomatoes dataset. The training data contains more than 156000 phrases, which were parsed from about 8500 sentences by the Stanford parser. Each phrase has an associated sentiment label. There are 5 sentiment labels considered:

- 0 - negative

- 1 - somewhat negative
- 2 - neutral
- 3 - somewhat positive
- 4 - positive

Figure 1 shows the partition of classes in the dataset. Phrases which belong to the same sentence share a common id. Additionally, 66291 phrases, extracted from 3310 sentences, have been provided as test set.

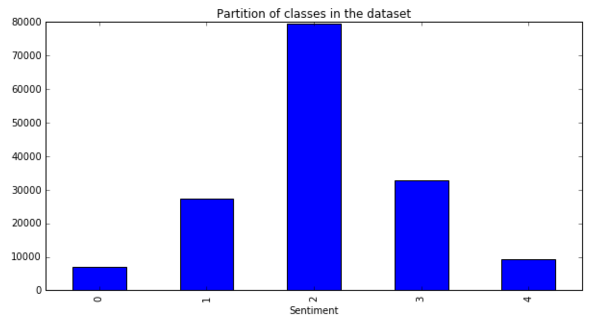
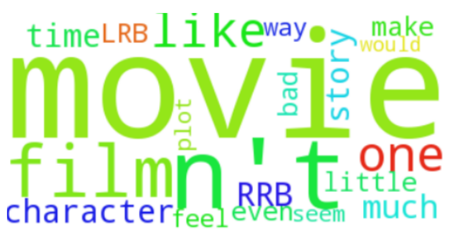
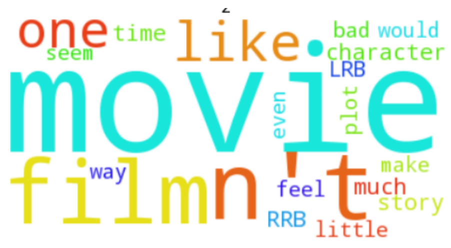


Figure 1: Partition of classes in the dataset

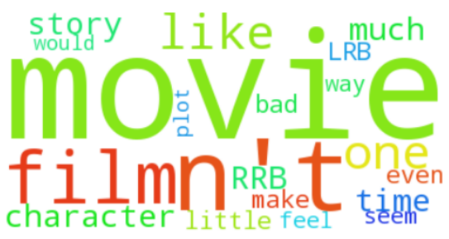
We can interpret the class of a review (sentiment) by exploring it through the words that compose it, and their probability. For this, we create a world cloud that is simply a graphical representation of the word frequencies. In order to do this we used the library word cloud of python.
Note: we did not include the stop words in the word cloud below.



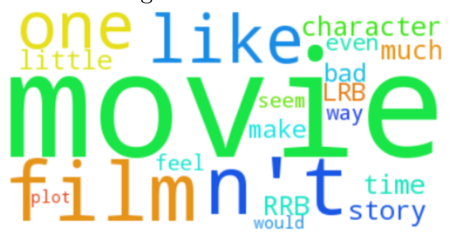
(a) frequency of word in positive class



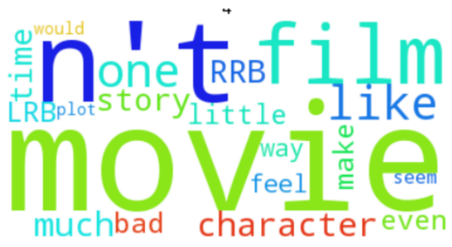
(b) frequency of word in neutral class in somewhat negative class



(c) frequency of words in neutral class



(d) frequency of words in somewhat negative class



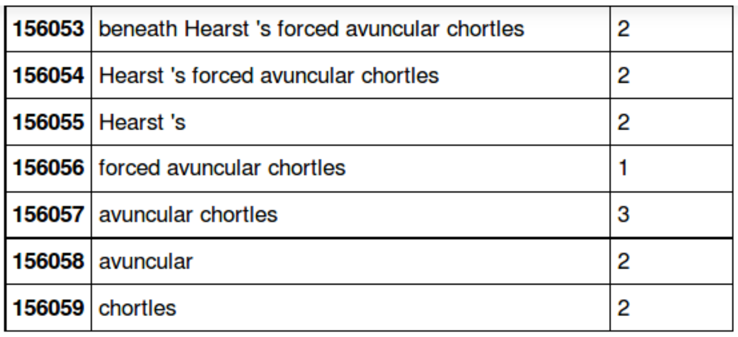
(e) frequence of word in somewhat postive class

Figure 2: frequency of words across the 5 classes

Based on the figure above 2 we can notice that there are some words like; movie, film,story etc... which are really common since they are related to the cinema context so we should not take them in consideration in our classification task (they should not be part of our vocabulary) so we have to delete them in the pre-processing phase, especially that they have a high frequency so they will bias our result.

4 Preprocessing

After observing the data we have noticed that there are stop words, punctuation, Unicode and uppercase characters (figure 3). Therefore we proceeded to delete stop words because they not provide any information for the overall meaning of the text and we have also added some words on the list of stop words, such as: will,now,today etc.. we have also add the words which are related to the cinema context for example: movie,serie,story, film etc... Moreover we have deleted the punctuation and convert Unicode characters to ASCII. And finally we stemmed every word to treat each flexion of a word equally, for example: wait(infinitive, imperative present), waits (present, 3rd person, singular), waited (simple past or past participle), waiting (progressive), etc. are all described by wait. **Notice:** All methods require a tokenization process except recurent neural networks (LSTM)



156053	beneath Hearst 's forced avuncular chortles	2
156054	Hearst 's forced avuncular chortles	2
156055	Hearst 's	2
156056	forced avuncular chortles	1
156057	avuncular chortles	3
156058	avuncular	2
156059	chortles	2

Figure 3: example of data before pre_processing

4.1 Tfidf

TF-IDF [8] [3] (Term Frequency-Inverse Document Frequency) is a text mining technique used to categorize documents. It stands for "term frequency * inverse document frequency" and is a method for emphasizing words that occur frequently in a given document, while at the same time de-emphasising words that occur frequently in many documents.

In order to compute the tfidf more efficiently, we performed the following pre-processing methods on the dataset:

- we removed most frequent words from documents. In fact, we discarded words appearing in more than 80% of the documents because they don't give information and can bias the result.
- we have also removed rare words: discard words appearing in less than 5 documents. It can cause some kind of noise.
- we also considered n-grams [1], combinations of n sequential words, more precisely bi_grams. Why do n-grams work? Consider this phrase: "movie not fantastic". it's clear that it have a negative sentiments, but if we take each word separately we won't detect this. On the opposite, the model will probably learn that "good" is a positive sentiment word, which doesn't help at all here. On the other hand, bi grams will do the trick: the model will probably learn that "not good" has a negative sentiment.

4.2 Improvement of Negative Forms

Das et Chen [2] implemented a method in their work for extracting feelings from stock market forums messages. They have determined that the negation in a sentence reverses the meaning of the sentence. They discussed how words such as not, never or no are used to reverse the meaning

of a sentence. To emphasize the negation of the sentence, they begin by detecting the words of the sentences and mark them with a negative tag. Pang and Lee [7] use the same technique by adding the word of negation to each word up to the first mark of punctuation following the word of negation found.
Here is an example illustrating their method:
« I do not NOT like NOT this NOT movie , but I like this one . »
We will try this method later to see if it can improve the result.

4.3 Feature Selection

In the text classification, the feature selection is a method for selecting a specific subset of terms from the training set and only using that subset for the classification task. The main advantages of this method are the reduction in the size of the data, select relevant features(words) and, therefore, allows for faster workout, but it can also improve the accuracy of the system by removing features that can be seen as noise.
in our case after tfidf we have a matrix of 11402 columns and 156060 rows columns. then we have to apply fetaure Seelection.

We use the SelectKBest implemented in Scikit-learn with the Chi Square method (chi2) for a first feature selection. This method is mainly used in statistics to test the independence of two events. In the feature selection, chi2 calculates whether the occurrence of a specific term and the occurrence of a specific class are independent. Thus each term is evaluated and all terms end up ordered by their score. A high score indicates that the null hypothesis of independence must be rejected and that the occurrence of the term and the class are dependent. If the class and term are dependent on each other, the feature is selected for classification (the first k features).

5 Methods

In this section we describe the methods we have implemented so far for the classification of phrases into categories based on the sentiment communicated in each phrase. In order to evaluate the performance of various methods and tune their parameters, we split the train dataset into 3 parts: 60% for training, 20% for validation and 20% for testing.

5.1 Multinomial Naive Bayes

The first approach which we considered as a baseline for this problem is the use of Multinomial Naive Bayes classifier trained on the bag of words representation of the data. The Sklearn implementation of the Multinomial Naive Bayes Classifier was used, with the default parameters. However, even if this method is not supposed to give the best classification accuracy, the first classifier which we trained obtained a really low accuracy, therefore we need to investigate it in detail.

5.2 Support Vector Machines

Suport Vector Machines is a powerful classification algorithm, which depends on creating decision boundaries between the classes samples. These boundaries have to maximize the margins between the classes. This algorithm was proposed for Binary classification but adapted for the multi classification using an approach called "one to many". This approach proposes to put the samples of one class in a category alone and the remaining classes samples in another category, and this step is repeated for all categories. Sklearn library was used to implement SVM for multi-classification. Linear and Redial Bias were used as kernel functions. Similarly to Naive Bayes, this classifiers showed low accuracies.

5.3 LSTM Networks

In the next step of our project we plan to explore the efficiency of LSTMs on the task of sentiment analysis. Long short-term memory models are a type of recurrent neural network. In recurrent neural networks (RNN), predictions are made sequentially, and the hidden layer from one prediction is fed to the hidden layer of the next prediction. This gives the network "memory", in the sense that the results from previous predictions can inform future predictions. LSTMs add additional factors to a traditional RNN that give it more of a fine-grained control over memory. These factors control:

- how much the current input matters in creating the new memory.
- how much the previous memories matters in creating the new memory.
- what parts of the memory are important in generating the output.

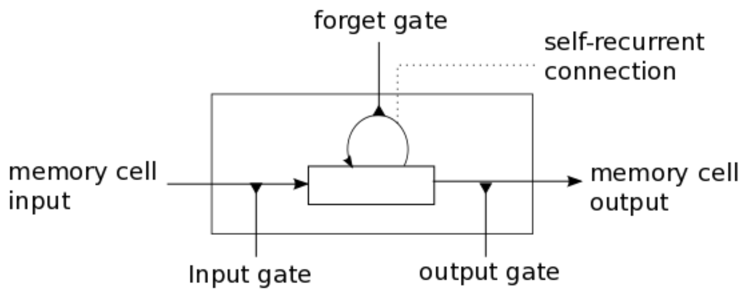


Figure 4: Illustration of an LSTM memory.

6 Conclusion and further steps

Analyzing tone and extracting sentiments form user text input is not an easy task. In the first part of our project we investigated the current state-of-the-art for the problem of sentiment analysis in order to discover which are the most efficient approaches.

We started by analyzing the dataset provided in this Kaggle challenge and we extracted some statistics which can help us in better understanding the data. Thereafter we proceeded with pre-processing the dataset in order to obtain a good representation of the text input. Finally we explored some classification algorithms which could be employed for this task. We considered Multinomial Naive Bayes in order to obtain a baseline result for the classification, as well as SVMs with various kernel functions. However, for the moment none of the classifiers had a good accuracy.

Further we plan to enhance the pre-processing phase by exploring some other techniques find in the literature. Moreover, we will try to explain why the baseline classifiers showed such a low accuracy, and improve their results by tuning their parameters. We also plan to use and analyze a tree based ensemble method such as Random Forest or Gradient Boosting. Finally, we will try a deep learning technique, such as RNN, since [9] reported the best results with this approach.

References

[1] Fotis Aisopos, George Papadakis, and Theodora Varvarigou. Sentiment analysis of social media content using n-gram graphs. In *Proceedings of the 3rd ACM SIGMM international workshop on Social media*, pages 9–14. ACM, 2011.

[2] Sanjiv Das and Mike Chen. Yahoo! for amazon: Extracting market sentiment from stock message boards. In *Proceedings of the Asia Pacific finance association annual conference (APFA)*, volume 35, page 43. Bangkok, Thailand, 2001.

[3] Justin Martineau and Tim Finin. Delta tfidf: An improved feature space for sentiment analysis. *Icwsm*, 9:106, 2009.

[4] Alexander Pak and Patrick Paroubek. Twitter as a corpus for sentiment analysis and opinion mining. In *LREc*, volume 10, 2010.

[5] Bo Pang and Lillian Lee. Seeing stars: Exploiting class relationships for sentiment categorization with respect to rating scales. In *Proceedings of the 43rd annual meeting on association for computational linguistics*, pages 115–124. Association for Computational Linguistics, 2005.

[6] Bo Pang, Lillian Lee, et al. Opinion mining and sentiment analysis. *Foundations and Trends® in Information Retrieval*, 2(1–2):1–135, 2008.

[7] Bo Pang, Lillian Lee, and Shivakumar 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*, pages 79–86. Association for Computational Linguistics, 2002.

[8] Juan Ramos et al. Using tf-idf to determine word relevance in document queries. In *Proceedings of the first instructional conference on machine learning*, 2003.

[9] Richard Socher, Alex Perelygin, Jean Y Wu, Jason Chuang, Christopher D Manning, Andrew Y Ng, and Christopher Potts. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the conference on empirical methods in natural language processing (EMNLP)*, volume 1631, page 1642. Citeseer, 2013.