SENTIMENT ANALYSIS KAGGLE COMPETITION

A PREPRINT

Thierry Bleau
260 743 240
team 57

Botond Maros
260 874 605
team 57

Anasthasisa Statakos
260 874 605
team 57

February 23, 2019

ABSTRACT

This paper seeks to explore different approaches to the sentiment analysis of IMDb movie reviews. Sentiment analysis is the analysis of extracting emotions and opinions of the reviewers towards a particular topic from differently structured textual data. The purpose of this sentiment analysis is to categorize the reviews and label them as positive or negative. The challenge is to predict the sentiment of a new review. In the following write-up we explain several of our approaches. For this task we used various text processing methods, feature extraction techniques and classification models to attain the best prediction. Pipelines were compared to each other and evaluated based on their accuracy. Our approach using classification techniques has the best accuracy of 0.91.

1 Introduction

The objective of this project is to build a consistent classifier with reliable prediction accuracy on a database of IMDb movie reviews. We will point out the differences and understand the varying costs and benefits that come with all the different natural language processing techniques. For this analysis we'll be using a dataset of 25000 movie reviews taken from IMDb. The dataset is divided equally between positive and negative reviews. Since IMDb lets users rate movies on a scale from 1 to 10. To label these reviews the curator of the data labeled anything with less than 4 stars as negative and anything with more than 7 stars as positive. Reviews with 5 or 6 stars were left out. To start our classification task we have to prepare and clean the data, select features and models. To prepare the data we used lemmatization, and stopwords removal. For feature selection we used binary representation of words and tf-idf and n-grams. The models we compared were Bernloulli Naive Bayes, Logistic regression and Decision tree classifiers.

2 Previous literature

A lot of methods were devised to extract meaningful features and find accurate models to predict the sentiment of raw text sources. [7] The most commonly used techniques developed over the years include:

- lemmatization : reducing the most of the words to their general form.
- tf-idf: evaluating the frequecies of words employed in the text compared to whole corpuses.
- logistic regression: combining weights of features to see if they pass a certain threshold.
- support vector machines: finding a separation between vectorized data point where the margins are maximized.
- deep learning: More advanced methods like Convolutional Neural Networks and LSTM's are also commonly used.

3 Data and Setup

Before vectorizing the text we cleaned our data by lemmatization, which is getting to the root of the words by for example removing verb ending 'ing' and past tense 'ed', and stopwords removal such as 'the', 'I', ... to reduce noise in

our data. Stopword removal has varying effectiveness for different corpuses. In our case, this slightly improved our predictions. After this, we converted our corpus into a matrix where every entry represents a word. In each comment we labeled the words with 0's or 1's representing which words occur in the review from the corpus. We are now ready to move to the next step in our workflow which is feature extraction and selection.

4 Approaches

Feature selection in machine learning has just as big impact on the learning process as finding the right model. Nevertheless, it is very difficult to represent textual data numerically since if done incorrectly, it is very easy to lose a lot of valuable information.

After lemmatization and stop-word removal we represent the occuring words binarily in a review. This only represents occurrence but for positive or negative emotion we know some terms are more important than others. Term frequency–inverse document frequency can be used for this purpose, which weights the words reflecting on how important they are in a corpus. Using this statistics yields better results in the prediction but we are still losing valuable grammatical information for example the sequence of the words, which we believe is important. Therefore, we tried to solve this issue by using n-grams. This slightly improved our prediction.

After the feature extraction we can build our classifier. We used linear models because they tend to perform well on sparse data-sets like this one, and they learn very fast compared to other algorithms like neural networks. We compared support vector machines, decision trees, logistic regression and implemented a Bernoulli Naive Bayes classifier from scratch.

Decision trees require relatively little effort for data preparation but they don't perform well on text analysis. They produce wrong results if complex, hardly tangible factors are present such as emotions. They are better suited to model a decision process, but text data is too difficult for that.

Logistic regression on the other hand is well suited for binary classification. Therefore it is always a baseline model for this purpose. It needs more attention for feature engineering than decision trees but they are very effective.

The Bernoulli naive Bayes is a generative learning algorithm that computes a conditional probability uses Bayes rule. It is easy to interpret and uses elementary probability theory. Features can only be binary data because representing term occurrence. Therefore we can't engineer features which is a huge drawback compared to other models.

We also attempted to implement a convolutional neural network from [1] to see if better scores could be achieved by means of pure computational power.

We finally, we tried an ensemble method of both stochastic gradient descent classifier and polarizing scores from a sentiment lexicon from the Vader package [6] and later added logistic regression to the mix.

5 Results

We implemented and tested different models on our training set which was split into 5 five folds for the KFold cross validation.

As we can see the bernoulli naive bayes we implemented only predicted with a score of 0.52. This accuracy was caused by using binary occurrences and only using the top 5000 words. This was necessary to speed up the computation. The huge loss in precision is due to the fact that it was impossible to engineer features to better fit the classes.

The decision tree model performed better than the naive bayes. It achieved 0.71 with the text being vectorized and using 1- and 2-grams. Other features did not improve the model significantly.

Logistic regression with 1- and 2-grams achieved 0.87. With vectorized text and tf-idf this accuracy went up to 0.88. After lemmatization and stop-word removal with tf-idf this accuracy was 0.893. This was a very effective model given the very short runtime.

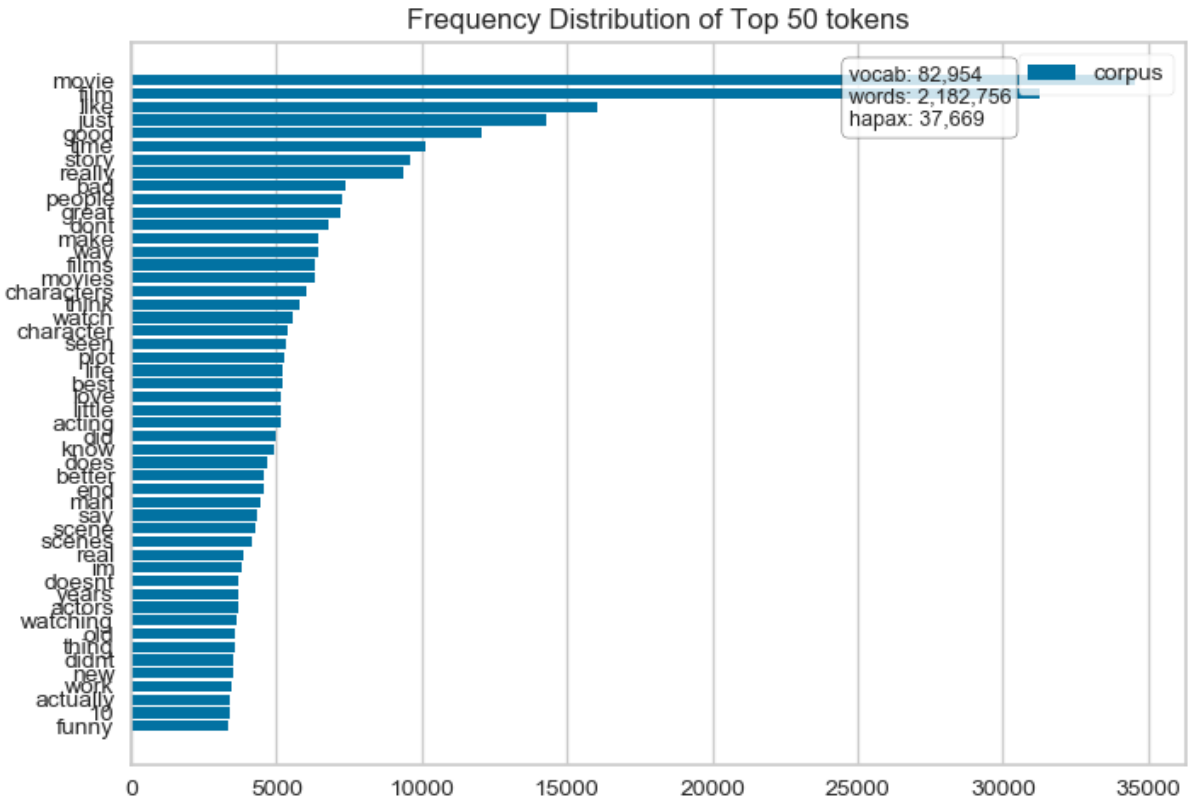

Figure 1: Unsurprisingly, the most frequently used words are "movie" and "film". However, we can see that the distribution exhibits what would be expected by Zipf's law.

Table 1: Scores and Runtimes for Different models

Model	Runtime (seconds per fold)	Score (μm)
Logistic Regression	2.7	0.88
Naive Bayes	642	0.52
SVM	3.04	0.91
Decision Tree	4.3	0.7
SGD	2.5	0.87
CNN (epochs = 5, batchsize = 16)	3174	0.82
SGD +Compound Lexicon	4.5	0.74
SGD +Negative Lexicon	4.4	0.8352
SGD + LogReg	2.14	0.88
SGD +Negative Lexicon + LogReg	266	0.82

After multiple tries, the convolutional neural network disappointingly only performed around 0.82 to 0.84 for computation times spanning over more than an hour.

On the other hand, a combination of stochastic gradient descent classification and the compound measure of a sentiment lexicon yielded 0.74. We quickly noticed that long reviews had a higher propensity to contain positive words at the same time as being a negative review overall. Using only the negative part of the lexicon slightly improved our score but simple logistic regressions were still better overall.

Finally, our best performing model was the support machine with 2-grams and l2-normalizers which yielded a score of 0.91 for a relatively short runtime. This is the model that we used to submit our prediction to Kaggle and it achieved a score of 0.907 on the public dataset.

6 Discussion and Conclusion

One of the key takeaways from the project is that deep learning techniques seldom perform better than regular statistical methods because of the lack of data and the necessity of huge amounts of computing power to achieve decent results [4]. In the case of regular statistical methods, a possible direction for future investigation would be a way to quantify the concept known as "Russell Conjugation". This concept describes the different emotional connotations that we give to words with the same meaning like "firm" and "stubborn". Quantifying Russell Conjugation in movie reviews would most likely give better sentiment lexicons to use for sentiment analysis.

7 Statement of Contributions

Botond did the logistic regression, the decision tree and naive-bayes implementations.

Thierry did the convolutional neural network, the support vector machine, and the Kfold cross-validation implementations.

Both worked on the write-up.

References

[1] *How to Develop an N-gram Multichannel Convolutional Neural Network for Sentiment Analysis*. 2018. URL: https://machinelearningmastery.com/develop-n-gram-multichannel-convolutional-neural-network-sentiment-analysis/.

[2] Alon Jacovi, Oren Sar Shalom, and Yoav Goldberg. "Understanding Convolutional Neural Networks for Text Classification". In: *CoRR* abs/1809.08037 (2018). arXiv: 1809.08037. URL: http://arxiv.org/abs/1809.08037.

[3] Vishal. A. Kharde and Sheetal. Sonawane. "Sentiment Analysis of Twitter Data : A Survey of Techniques". In: *CoRR* abs/1601.06971 (2016). arXiv: 1601.06971. URL: http://arxiv.org/abs/1601.06971.

[4] Evgeny Kim and Roman Klinger. "A Survey on Sentiment and Emotion Analysis for Computational Literary Studies". In: *CoRR* abs/1808.03137 (2018). arXiv: 1808.03137. URL: http://arxiv.org/abs/1808.03137.

[5] Olga Kolchyna et al. "Twitter Sentiment Analysis". In: *CoRR* abs/1507.00955 (2015). arXiv: 1507.00955. URL: http://arxiv.org/abs/1507.00955.

[6] Parul Pandey and Parul Pandey. *Simplifying Sentiment Analysis using VADER in Python (on Social Media Text)*. 2018. URL: https://medium.com/analytics-vidhya/simplifying-social-media-sentiment-analysis-using-vader-in-python-f9e6ec6fc52f.

[7] Rahul Tejwani. "Sentiment Analysis: A Survey". In: *CoRR* abs/1405.2584 (2014). arXiv: 1405.2584. URL: http://arxiv.org/abs/1405.2584.

[8] Yongping Xing et al. "A Convolutional Neural Network for Aspect Sentiment Classification". In: *CoRR* abs/1807.01704 (2018). arXiv: 1807.01704. URL: http://arxiv.org/abs/1807.01704.

references