ISSN (Online) 2394-2320

International Journal of Engineering Research in Computer Science and Engineering (IJERCSE)

Vol 7, Issue 9, September 2020

Deep Learning for Movie Review Sentiment Analysis

[1] Rajeev Dixit

[1] Indian Institute of Information Technology - Allahabad

Abstract: Sentiment analysis [6] has long been an important problem of study in NLP and machine learning, for finding public sentiments on products, brands or services. Previous approaches to sentiment analysis have included unsupervised learning, Naive Bayes classifiers and SVM. In this paper, we focus on sentiment analysis of movies using text reviews. Sentiment analysis can be a challenging problem to solve because our language is rather complex and a single word can have either positive or negative connotations based on the context. We will be using the Large Movie Review Dataset [4] given by Stanford AI lab, which is a binary sentiment classification dataset based on IMDB reviews of movies, and contains 50,000 reviews with a 50:50 train: test split. The objective is to classify a movie as good or bad, based on its text review. We will approach this problem using the Tf-Idf vector of the corpus and applying a deep learning model on top of it. This model achieved an accuracy of 90.7%, which is a significant improvement over the current approaches. Future extensions to this approach could include more powerful deep learning models like LSTM or GRU, which can extract even more contextual information.

Keywords— Sentiment analysis, Tf-Idf, Neural networks, Deep learning, NLP, n-grams, Bag of Words.

1. INTRODUCTION

Sentiment analysis is an extensive field of study in NLP and text mining, which is used to contextually identify, extract and quantify subjective information from textual data. It can be used to identify social sentiment or perception of a product, brand or service. Hence, it has long been an important problem to tackle for business, marketing and management functions. Sentiment analysis can be used to derive general public opinions such as "happy", "angry", "sad", "satisfied", etc. towards the product being reviewed.

In case of movies, reviews are critical for their evaluation, as they provide insights into public opinion of the movie. Opinions on movies are subjective, and often vary vastly from person to person. While star rating can quantify the success or failure of a movie, textual reviews give a deep qualitative insight into different smaller aspects to the movies such as acting, storyline and direction. Earlier, very few movie reviews were available, since they were published by established critics and journalists in newspapers and magazines. However, since the last decade, due to ubiquity of the internet, thousands of reviews are published by the general public for a single movie on online aggregators such as Imdb, Rotten Tomatoes and Metacritic. Thus, deriving Sentimental insights from large scale data is an interesting problem to solve.

In this paper, we aim to perform Sentiment Analysis on a dataset of Imdb reviews of different movies, which have a textual review as well as star rating on a scale of 1 to 10. Our objective is to perform binary classification of movies as good or bad based on the text review. The dataset "Large Movie Review Dataset" is published by Andrew Mass [4], and was originally used in the paper "Learning Word Vectors for Sentiment Analysis" by researchers from Stanford University. Our approach is to apply a multilayer neural network to the Tf-Idf matrix and compare performance improvements over other approaches. We aim to provide solutions to challenges faced due to the scale of the dataset, which has 50,000 reviews and possible human spelling errors caused in reviews. The tools and frameworks that we have used include NLTK [9], Scikit-Learn [10] and Keras [11] with Tensorflow backend.

2. Related work

Sentiment Analysis has been subject to a lot of research work as a Natural Language Processing and Machine Learning task. In this section, we will briefly summarize all relevant previous work. Peter Turney's [1] work in classifying reviews as thumbs up or thumbs down was probably the first paper of this type. He used a simple unsupervised learning algorithm to estimate the semantic orientation of different extracted phases from a review and predicted the output class as the average of all phrases in a review. This algorithm achieved an average accuracy of 74% on different kinds of reviews and 66% accuracy on movie reviews.

The next important work was by Pang, Lee [2] where they used three machine learning methods: Naive Bayes, maximum entropy classification, and SVM on unigrams and bigrams on an Imdb movie review data. Their research yielded a best accuracy of about 83% on using SVM with Unigrams and Bigrams. The work of Hu & Liu [3] also brought about a new approach in which opinion words are extracted using Parts of Speech Tagging and aggregated to determine user sentiment.

Lastly, the work of Andrew Mass, et al [4] uses a mixture of supervised and unsupervised learning to learn word vectors, which capture semantic term–document information as well as rich sentiment content. This research group at Stanford AI lab has also published the Large Movie Review Dataset, consisting of 50,000 movie reviews which is being used in this paper.

3. Methodology

Data Preparation:
The dataset comes in text files of each separate review. The first step would be tokenization and stemming of the text. For tokenization [7], Treebank word tokenizer was selected, after comparison with other options such as Word tokenizer and Punctuation tokenizer. Stemming [8] is used to get the root word from its different variation by removing the suffixes. This is useful since different words with a common stem will have the same meaning, and hence we may replace the word only by the stem. For that purpose, Porter stemmer was a natural choice.
Feature generation
After preparing our data with tokenization and stemming, we need to convert out text into a vector of learnable features. For that, following methods were used.

1) *BOW*: Bag of words represents a document just as a frequency of all the words present in it, without giving any importance to the order of occurrence. It is the most basic model for text representation.
2) *N-grams*: N-grams extends the concept of BOW, where we represent a document as counts of contiguous sequences of n words. This gives a bit more context to the features in our model. For example, in the sentence, "The direction is not good", the one-gram "good" gives positive sentiment, but if we consider bigrams, we get the phrase "not good" which in fact reverses the meaning.
3) *TF-IDF*: While n-grams do a fairly good job of representing our text, we still have some issues which can be improved. For example, the word "movie" will occur in almost all movie reviews frequently, so it will have a large value in the n-gram model, but since it is supposed to be common in positive and negative reviews, it doesn't add any value to our classifier. The Tf-Idf [8] metric can help us here to quantify the importance of a word in a document with respect to the entire corpus. Tf-Idf is the product of two terms:

- Term Frequency (Tf): Tf is the relative frequency of a term in a document. It represents the importance of a particular term in context of a single document.

$$tf(i,j) = \frac{\text{Frequency of term i in document j}}{\text{Total terms in document j}}$$

- Inverse Document Frequency (Idf): Idf is the inverse frequency of a term with respect to all documents in the corpus. A high Idf signifies that a word his highly discriminatory as it appears in very few documents.

$$idf(i) = log\left(\frac{Total\ number\ of\ documents}{documents\ with\ term\ i}\right)$$

$$tfidf(i,j) = (tf(i,j) \times idf(i))$$

Scikit-Learn provides a good implementation of Tf-Idf with its TfIdfVectorizer. We calculated Tf-Idf with a combination of unigrams, bigrams and trigrams as this would add contextual phrases to our corpus which can

IFERP
connecting engineers... developing research

ISSN (Online) 2394-6849

International Journal of Engineering Research in Computer Science and Engineering (IJERCSE)
Vol 7, Issue 9, September 2020

provide better discriminative power. The Tf-Idf was additional smoothed by removing highly frequent words which maybe pronouns or prepositions like “a”, “the”, “he”, which don’t hold any importance (f > 25%) and very infrequent words (ct < 3) as they maybe spelling mistakes. After this a word corpus of around 500,000 words was used for training our machine learning model.
.

Machine Learning Model
The following two machine-learning models were used to fit the feature vector obtained in step B:

1) *Logistic regression (Baseline)*: A binary Logistic regression model with l2 regularization is used to train on this feature vector to predict whether a movie is good or bad. This model is a good baseline model, as it performs better than naive Bayes. On the test data, this model achieved an accuracy of 89.3%.

2) *Multilayer neural network:* Logistic regression gives a fairly impressive result, but it is only a linear model, we can do better than that with deep learning. Deep learning seems to perform better with larger data, so we propose a multi-layer neural network [12] for this. Deep learning is a very powerful model that can help capture non-linearity and complicated sub characteristics in the data that logistic
regression cannot. I experimented with different architectures ranging from 1 to 3 hidden layers, with different number of parameters. Dropout was also added in the initial dense layer to reduce overfitting, and overdependence on certain parameters.

4. Result

Logistic regression as a base model on top of Tf-Idf provides pretty good results.It achieved a maximum accuracy of 89.3% on for 1-gram to 3-grams with smoothing, on the test dataset of 25,000 reviews. The final weights of the word parameters can be used to analyze positive sentiment words (very high coefficient) and negative sentiment words (very negative coefficient).

TABLE 1
Top Opinion Word

Top Positive Words	Top Negative words
great	bad
love	worst
excel	waste
enjoy	bore
best	awful
beautiful	nothing
perfect	poor
favourite	terrible
amazing	stupid

Table I shows the top 10 positive and negative opinion words based on the learned coefficients of the logistic regression. We can easily observe that these are exactly the words we would expect in the reviews of a positive movie and negative movie respectively.
The multilayer artificial neural network beats this model across all categories by a sizable margin of around 1%.
The comparison between accuracy achieved by the machine-learning algorithms on the validation data is shown in Table II and Fig 1. The best neural network achieved an accuracy of 90.7% on Tf-Idf for 1-gram to 3-grams with smoothing. This boost in performance can be attributed to neural networks ability to learn complex and non-linear relations from the features. Neural networks give a larger improvement for larger feature size which is the case for unigrams to trigrams. Given a more powerful GPU setup and finer hyper-parameter tuning, an even better accuracy can be obtained using this model.

TABLE II
Test Data Accuracy

	Logistic Regression	Multilayer Neural Network
Bag of Words	85.6	86.1
Tf-Idf with unigrams	88.0	88.7
Tf-Idf with smoothing	87.6	88.3
Tf-Idf with 1-gram to 3-gram with smoothing	89.6	90.7

ISSN (Online) 2394-6849

International Journal of Engineering Research in Computer Science and Engineering (IJERCSE)
Vol 7, Issue 9, September 2020

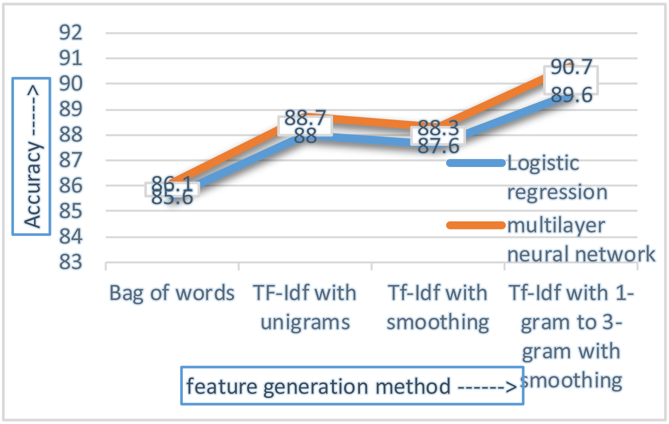

Fig .1 Test data accuracy

5. Conclusion

From the results, we can conclude that deep learning on Tf-Idf vectors is a promising approach for Sentiment Analysis, especially on large-scale data. Our neural network model achieved a decent improvement over the baseline logistic regression model as seen in Fig .1. Using bigrams and trigrams boosts the performance of our classifier due to the capture of contextual data in phrases which can't be captured in unigrams. Using an even more powerful deep learning model like LSTM or GRU may improve the accuracy even further.

Abbreviations

BOW:	Bag of Words
GRU:	Gated Recurrent Unit
LSTM:	long short term memory
NLTK:	Natural Language Toolkit
POS:	Parts of Speech
SVM:	Support Vector Machine
TF-Idf:	Term frequency inverse document frequency

REFERENCES

1. Peter D Turney "Thumbs up or thumbs down?: semantic orientation applied to unsupervised classification of reviews" July 2002 Proceedings of the 40th Annual Meeting on Association for Computational Linguistics (ACL). Philadelphia, pp. 417-424
2. Bo Pang, Lillian Lee, Shivakumar Vaithyanathan, "Thumbs up? Sentiment Classification using Machine Learning Techniques". Proceedings of the 2002 Conference on Empirical Methods in Natural Language Processing (EMNLP 2002), Association for Computational Linguistics, July 2002, pp. 79–86. doi:10.3115/1118693.1118704
3. Minqing Hu and Bing Liu. 2004. Mining and Summarizing Customer Reviews. In Proceedings of the tenth acm sigkdd international conference on knowledge discovery and data mining. https://www.cs.uic.edu/~liub/publications/kdd04-revSummary.pdf
4. Andrew L. Maas, Raymond E. Daly, Peter T. Pham, Dan Huang, Andrew Y. Ng, and Christopher Potts. 2011. Learning Word Vectors for Sentiment Analysis. The 49th Annual Meeting of the Association for Computational Linguistics (ACL 2011). http://ai.stanford.edu/~amaas/papers/wvSent_acl2011.pdf
5. Shahzad Qaiser and Ramsha Ali. Text Mining: Use of TF-IDF to Examine the Relevance of Words to Documents. International Journal of Computer Applications 181(1):25-29, July 2018.
6. Mika V. Mäntylä, Daniel Graziotin, Miikka Kuutila, The evolution of sentiment analysis—A review of research topics, venues, and top cited papers, Computer Science Review, Volume 27, February 2018, Pages 16-32, ISSN 1574-0137
7. Dr. S.Vijayarani and Ms. R.Janani. TEXT MINING: OPEN SOURCE TOKENIZATION TOOLS – AN ANALYSIS, Advanced Computational Intelligence: An International Journal (ACII), Vol.3, No.1, January 2016
8. Jivani, Anjali. (2011). A Comparative Study of Stemming Algorithms. Int. J. Comp. Tech. Appl.. 2. 1930-1938.
9. NLTK documentation: https://www.nltk.org/
10. Scikit Learn documentation: https://scikit-learn.org/stable/

11. Keras documentation: https://keras.io/
12. Skapura, David M. Building Neural Networks. Menlo Park, CA: Addison-Wesley Publishing Company, 1996