IMDb Sentiment Analysis

COMP 551 - Group 17

Beatrice Lopez
260654565

Minh Anh Nguyen
260671180

Xavier Sumba
260900337

February 2019

Abstract

IMDb is one of the most popular online databases for movies and personalities, a platform where millions of users read and write movie reviews. This provides a large and diverse dataset for sentiment analysis. In this project, we were tasked to implement different classification models to predict the sentiment of IMDb reviews, either as positive or negative, using only text each review contains. The goal is to find the model with the highest F1 score and best generalization. We trained different models using multiple combinations of text features and hyper-parameter settings for both the classifiers and the features, which we found could potentially impact the performance significantly. Every model was evaluated by k-fold cross validation to ensure the consistency of their performance. We found that our best performing model was the Naïve Bayes - Support Vector Machines classifier with bag of words, which reported an accuracy score of 91.880 on the final test set.

1 Introduction

IMDb contains approximately 83 million registered users, 5.6 million movie titles, and 9.5 million personalities[1]. Many of the users are actively posting movie reviews, which provides us with a rich and diverse dataset of people's sentiments and opinions.

The task of classifying the overall sentiment polarity of movie reviews is highly desirable. From a researcher's perspective, the sentiment of a movie review is usually associated with a rating (e.g. the number of stars), which is helpful for the classification process. From a user's perspective, it serves as a recommendation tool for movie selection. From a producer's perspective, it can be effectively used for movie marketing and advertising purposes.

For this task, we trained many models using a training set of 25,000 movie reviews and targets to decide on one with highest F1 score. The classifiers used were logistic regression, Multinomial Naïve Bayes, support vector machines, and decision trees from the SciKit Learn package[2], our own implementation of Bernoulli Naïve Bayes with Laplace smoothing, as well as replications of a few state-of-the-art algorithms. We used different combinations of text features, including TF-IDF, stemming, binary occurrences, bag of words (BoW), n-grams, maximum and minimum number of words, and mutual information. Each model was trained with multiple hyper-parameter settings to determine the best parameters, then evaluated using k-fold cross validation. Among the traditional classifiers, we found that logistic regression performed best, while multinomial Naïve Bayes and SVM performed noticeably better than decision trees and Bernoulli Naïve Bayes. We also explored and replicated state-of-the-art models to further improve our accuracy. Our best performing model was Naïve Bayes - Support Vector Machines (NB-SVM) with bag of words, which reported a F1 score of 91.880 on the test set on Kaggle.

[1]`https://www.imdb.com/pressroom/stats/`

[2]`https://scikit-learn.org`

2 Related Work

Sentiment classification is a relatively well-studied task in machine learning. We will reference some relevant papers to compare the results and discuss what we learned and applied from them.

[Wang and Manning, 2012] suggested the use of Naïve Bayes - Support Vector Machines (NB-SVM), an SVM variant using NB log-count ratios as feature values. The authors tested the performance of NB, SVM, and NB-SVM on 7 different datasets of movie reviews, customer reviews, opinion polarity and subjectivity. NB-SVM consistently performed well across tasks and datasets. This NB-SVM algorithm become the basis for some of our best performing models.

Given sufficient computer power, time, and training data, a single layer multiplicative LSTM with 4096 units [Radford et al., 2017] would outperform NB-SVM in all classification accuracies of small datasets. However, the training took one month to complete, and we did not have the time or compute power to train this model, even though we implemented it.

An ensemble of generative and discriminative techniques is a simple yet powerful method for sentiment analysis [Mesnil et al., 2015]. By combining Recurrent neural network (RNN), NB-SVM with trigrams, and sentence vectors as an ensemble, the authors were able to surpass the state-of-the-art baselines. Motivated by this result, we created ensembles of our own, using boosting and stacking methods.

[Martineau and Finin, 2009] presented Delta TF-IDF, a technique to efficiently weigh word scores before classification. While regular TF-IDF gives more weight to frequently occurring words that are more rare in the corpus, delta TF-IDF takes a step further by giving more weight to words that occur more often in that text, and are comparatively rare in oppositely labeled documents. It achieves this by weighting a feature's word count by the log of the ratio of positive and negative training documents using this word. Delta TF-IDF produced significantly better results than regular TF-IDF.

3 Dataset and Setup

The training set contains the raw text of 25,000 IMDb movie reviews, each labeled as positive or negative, with no other features. The test set contains the text of 25,000 unlabeled movie reviews. For evaluation and model selection, we used k-fold cross validation to take different portions of the training set to be used as the validation set.

For pre-processing, we cleaned the data by removing HTML tags, non-word characters, accents and stop words. Then we turned each word to lowercase before tokenizing. For certain methods, we observed that limiting the number of words helped the classification task. We believed this is because some of the top words, such as *zzzz (73286 occurrences)*, did not provide semantic content for the analysis of opinions. We found that limiting the words within a certain range of frequency helped to remove noise and improve our scores. Hence, we removed the 1 to 5 least frequent words and only allowed the 4000 to 6000 most frequent words[3].

In addition, the number of occurrences of a word does not necessarily imply its importance throughout the documents. Thus, we used TF-IDF to quantify this value. Mutual information was also used to select the features with more information with regards to the target variable. Finally, stemming and n-grams (up to trigrams) were used in some models.

[3]Find the complete analysis in the Jupyter notebook provided.

4 Proposed Approach

The following two lists contain the foundations of our approach. The *algorithms* column lists the algorithms used for the classification task and the *methods* column lists the basic mechanisms for feature engineering. We obtained acceptable results with common algorithms such as logistic regression (LR), Bernoulli Naïve Bayes (BNB), Multinomial Naïve Bayes (MNB), decision trees (DT), and support vector machines (SVM). However, after analyzing the state-of-the-art models on this dataset, we decided to go further and attempt to replicate those results, using only the models with classical machine learning algorithms[4]. We found that NB-SVM[5] [Wang and Manning, 2012] performed well on this dataset. NB-SVM introduces the hyper-parameter β to weight the predictions of SVM and MNB ($\beta = 1$ is normal SVM).

Algorithms

- Logistic Regression (LR)
- Bernoulli Naïve Bayes (BNB)[6]
- Multinomial Naïve Bayes (MNB)
- Support Vector Machine (SVM)
- Naïve Baye - Support Vector Machines (NB-SVM)
- Decision Trees (DT)

Pre-processing methods

- TF-IDF
- Bag of Words (BoW)
- Binarization
- Stop words removal
- N-grams (up to trigrams)
- Fixed vocabulary
- Normalization
- Stemming

First, we made a random search, varying the parameters of the methods and algorithms; we executed each model with 10-fold cross validation (CV). This reduced our search space and gave us an intuition of the set of features that works well based on the F1 score. Here, we noticed that stop words, words within a certain frequency range (mostly minimum occurrence 2 and maximum occurrence 4400), TF-IDF, and bigrams were important to the models. We found stemming and normalization to not be helpful for this dataset. We also realized that the combination of best parameters depends on the nature of each algorithm.

In addition, we calculated the mutual information between the words and the sentiment and built a vocabulary of words with the most information. The top words in the vocabulary are commonly used words for describing emotions. For example, the top 10 words are *bad, worst, waste, great, awful, excellent, terrible, worse, stupid, no.* We select the top 40%, 50%, and 80% words to be used while building the feature vector.

After selecting the main methods for feature engineering, we performed a random search focusing more on hyper-parameters using 10-fold CV. We repeated this process, while reducing the search space (based on the results of the F1 score) for random search and increasing the number of folds for CV up to 200. At the end, we executed an exhaustive grid search over the best set of parameters with an increased number of folds to test the variability of the models.

[4]We attempted to execute a model with a LSTM RNN. However, the GPU in Google Colab was a limitation. The code that accompanies this project has the implementation for it. We aimed to obtain 0.94 with LSTM.

[5]We found an implementation in Python that we slightly modified for our own pipeline, see `https://github.com/Joshua-Chin/nbsvm`

[6]Our implementation of Bernoulli Naïve Bayes did not scale well when dealing with a lot of features. However, this problem could be alleviated using sparse matrices.

5 Results

Using the general settings described above, we reported the top models with each algorithm in Table 1, in which the listed scores used 10-fold CV. There was not a major difference in runtime[7]. In general, DT and BNB did not perform well on this dataset. MNB and SVM performed significantly better and gave similar results to each other. We found that LR and NB-SVM performed best on this dataset, providing a F1 score of 0.922 and 0.914, respectively.

Method	**F1 score**
LR - BoW	0.90843
LR - TF-IDF	0.91369
LR - MI	0.90370
DT - BoW	0.70256
SVM - TF-IDF	0.89586
NB-SVM - BoW	**0.91640**
NB-SVM - MI	**0.92289**
BNB - Binary	0.85285
MNB - BoW	0.88754

Table 1: Top result for each algorithm

In addition, we selected our top-3 models to perform stacking and our best LR model to perform boosting. Even though the results improved by $1e-3$ to $3e-3$, they didn't seem to generalize equally on the Kaggle leaderboard.

Also, our best model (NB-SVM - MI) did not generalize well with the test set on Kaggle. Our intuition for this was because even though the fixed vocabulary contains words with the most information help to get a high F1 score during training, the test set might not contain words that are important to this classification task. For future improvement, we could incorporate a lexicon to make our vocabulary richer, so that it covers the same amount of words in the training set and test set.

Our second best model generalized better. We used NB-SVM with BoW and bigrams, with the pre-processing procedure described in section 3, achieving a score of 0.911 on Kaggle. However, after some more parameter tuning, using trigrams, and using a tokenizer from the NLTK library[8], we found a model that generalized better in the test set. We used the model *NB-SVM - BoW* with $\beta = 0.3193$, $\alpha = 1$, and $C = 0.405$, and obtained n score of 0.9188 in the Kaggle leaderboard, slightly beating the benchmark.

Our approach improved by 0.0066 from the one in [Wang and Manning, 2012]. This can be due the limitation of words, minimum 2 and maximum 6000. In addition, we found trigrams to work better than bigrams.

6 Discussion and Conclusion

In this project, we classified movie reviews from the IMDb dataset as positive or negative. We used common pre-processing steps such as HTML and non-word removal and tokenization. We found a small improvement of 0.0066 compared to the state-of-the-art results, using traditional machine learning techniques on this dataset. In addition, we found that logistic regression and

[7]Except for BNB because we are using dense matrices.

[8]`https://www.nltk.org/api/nltk.tokenize.html`

NB-SVM, a variation of SVM, worked best on this dataset. We achieved a F1 score of 0.9188 in the Kaggle leaderboard.

We learned that not only is it important to know the mechanisms of the algorithms, but that feature construction requires creativity and understanding of the problem. We also learned that maintaining good practices while building models is important because it allows reproducibility, and sound model comparison and model selection.

For future work, we would like to train the obtained models with the addition of a lexicon. We want to experiment with the data augmentation techniques in [Wei and Zou, 2019]. Their work is applied in the context of deep learning to recurrent neural networks, but we would like to test those operations with classical machine learning algorithms. We noticed that, in general, deep learning models scale well with large datasets. However, the models built in this project that employs mutual information seemed to work well in the training set. We think the reason why they did not perform equally well in the test set is not because of overfitting, but because we did not shown the model enough words to be generalizable.

Finally, having a larger training set with the rating score of each review (e.g. from 1 to 10) could potentially correlate to better performance of our models. We would also want to implement the concept of intensity (e.g. low, medium, high, extreme), by which we would train our models using not only positively or negatively labeled data, but low positive, medium positive, highly positive, extremely positive, and similarly for intensities of negative reviews.

7 Statement of Contributions

All three members contributed to the project quite equally. The main framework of Bernoulli Naïve Bayes, and the SciKit Learn classifiers and pipelines were implemented by Sumba, with Nguyen coming alongside him to implement and train some of the models. Lopez and Sumba were mainly in charge of reading and summarizing relevant research papers. We all contributed to the pre-processing task, execution of experiments, and writing this report.

References

[Martineau and Finin, 2009] Martineau, J. and Finin, T. (2009). Delta tfidf: An improved feature space for sentiment analysis. In *Proceedings of the Third International ICWSM Conference*, pages 258–261. Association for the Advancement of Artificial Intelligence.

[Mesnil et al., 2015] Mesnil, G., Mikolov, T., Ranzato, M., and Bengio, Y. (2015). Ensemble of generative and discriminative techniques for sentiment analysis of movie reviews. *arXiv preprint arXiv:1412.5335.*

[Radford et al., 2017] Radford, A., Jozefowicz, R., and Sutskever, I. (2017). Learning to generate reviews and discovering sentiment. *arXiv preprint arXiv:1704.01444.*

[Wang and Manning, 2012] Wang, S. and Manning, C. D. (2012). Baselines and bigrams: Simple, good sentiment and topic classification. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Short Papers-Volume 2*, pages 90–94. Association for Computational Linguistics.

[Wei and Zou, 2019] Wei, J. W. and Zou, K. (2019). Eda: Easy data augmentation techniques for boosting performance on text classification tasks. *arXiv preprint arXiv:1901.11196.*