Assignment 2: Predicting the sentiment of IMDB review comments using machine learning

Jean-Sébastien Grondin
McGill Id:260345519
jean-sebastien.grondin@mail.mcgill.ca

Jonathan El Baze
McGill Id:260893222
jonathan.elbaze@mail.mcgill.ca

David Gilbert
McGill Id:260746680
david.gilbert@mail.mcgill.ca

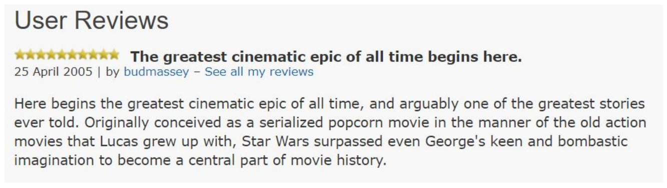
Abstract

In this project, machine learning and natural language processing techniques (NLP) were used for a classification task consisting in predicting whether the sentiment of IMDB review comments is positive or negative. IMDB is a popular website where users can find movie ratings and reviews for movies and TV series. Users can also post their own reviews for movies they have seen. To predict the sentiment of such reviews, various classifiers were tested: Bernoulli and Multinomial Naïve Bayes, Support Vector Machines, as well as logistic regression. The popular tool for sentiment analysis VADER was also used and compared against machine learning methods [1]. For training and validation purposes, a total of 25 000 review comments were available, and of which half were positive reviews and the other half were negative. NLP techniques such as count vectorization (binary and frequency based), TF-IDF weighing, stemming and lemmatization as well as n-grams were also investigated. For each classifier, the impact of several hyperparameters on the classification performance was also assessed via the SKLearn function *RandomizedSearchCV()*. The accuracy score (ratio of correct classifications) was used for all comparisons. It was found that the best performing model was a linear SVM classifier. This method obtained an accuracy score of **0.90266** on the test set.

1 Introduction

The objective of this project was to implement and evaluate different classifiers and natural language processing (NLP) techniques for a classification task consisting in predicting whether the sentiment of IMDB review comments is positive or negative. An additional goal was also to implement a Bernoulli Naïve Bayes training and prediction algorithm without using the SKLearn libairy and use it as a simple baseline when evaluating other models built using the SKLearn libairy. A total of 25000 labeled instances of text comments were available (50% positive and 50% negative) for training and validation purposes. Another 25000 unlabeld instances were also available and used as a test set. Various classifiers were tested: Bernoulli and Multinomial Naïve Bayes, Support Vector Machines, as well as logistic regression. The popular tool for sentiment analysis VADER was also used and compared against machine learning methods [1]. NLP techniques such as count vectorization (binary and frequency based), TF-IDF weighing, stemming and lemmatization were also investigated. For each classifier, the impact of several hyperparameters on the classification performance was also assessed via the SKLearn function *RandomizedSearchCV*. The accuracy score (ratio of correct classifications) was used for all comparisons. The best model was found to be a linear SVM model which used mono-grams and bi-grams in addition to TF-IDF. This method obtained an accuracy score of **0.90266** on the test set.

IMDB is a popular website where users can find movie ratings and reviews for movies and TV series. Users can also post their own reviews for movies they have seen. Predicting the sentiment in text was attempted in recent years [1][2][3] and has been found to be amenable to machine learning.



User Reviews

★★★★★★★★★★ **The greatest cinematic epic of all time begins here.**
25 April 2005 | by budmassey – See all my reviews

Here begins the greatest cinematic epic of all time, and arguably one of the greatest stories ever told. Originally conceived as a serialized popcorn movie in the manner of the old action movies that Lucas grew up with, Star Wars surpassed even George's keen and bombastic imagination to become a central part of movie history.

2 Related Work

This section briefly surveys previous work on sentiment text categorization. Text sentiment classification is an interesting domain in machine learning, and several researchers already studied a similar subject. Pang, Lee and Vaithyanathan (2002) [2] in their study compared

Naive Bayes to maximum entropy classification, and support vector machines. They concluded that these methods performed way better than human-produced baselines, but didn't perform as well as they expected. Wang & Manning (2012) [3] tried another approach, closer to the optimal solution described in this study. They implemented bigrams as text preprocessing and obtained way better results than previous studies on similar subject and datasets. Another approach of sentiment analysis was done by Hutto and Gilbert (2014) [1]. They tried a simple rule-based model for sentiment analysis : VADER. It outperformed all the machine learning algorithms used as baseline and even others. Elements from these interesting approaches shall be studied and/or combined further in this paper.

3 Dataset and setup

Two datasets of 25000 instances were available for this study. The first was composed of 12500 positive comments and 12500 negative ones. These were labelled accordingly and used for training and validation. The latter was comprised of 25000 unlabelled comments which needed to be classified and which were used as a test set. The following text pre-processing steps were implemented as they were previously proven to be effective for text sentiment analysis and classification.

- **URLs:** were removed
- **email addresses:** were removed
- **capital letters:** were converted to lowercase characters
- **punctuation and special typography:** all punctuation and other special characters were removed (e.g. !, ?, ., ;, #, *, [,], {, }, etc). Only the following characters were kept : /, %, $, as they may carry information that would be relevant for this classification task (e.g. 10/10, 100%, 100,000,000$)
- **lemmatization or stemming:** the various inflected forms of a word are regrouped into a lemma (i.e. one base word) or reduced to their stems. These two techniques were used interchangeably but lemmatization was found to be the most effective.
- **stop words:** several strategies were investigated. Removing all stop words was first tested, then adapted to removing all stop words except *n't* and *not*. In the end, only a short list of neutral stop words that do not carry information (i.e. *in*, *of*, *at*, *a*, *the*) were removed and this was found to be most effective.
- **tokenization:** each comment was splitted into words and a 'bag of words' was created. Comment features were then vectorized as a binary output (i.e. present or not), or as a frequency output (i.e. count in each comment). These two different feature representations were tested in each model and results are discussed in subsequent sections.
- **TF-IDF:** the benefit of converting features into TF-IDF statistics was also evaluated. TF-IDF is short for term frequency-inverse document frequency, and is used to indicate how important each word is in a comment in relation to the corpus.
- **n-grams:** the benefit of introducing bi-grams and tri-grams was evaluated. N-grams are continuous sequence of n words and are often used in the fields of computational linguistics.

Once all text processing was completed, the feature matrix was ready to be used by the various models. The test data was left aside and 10% of the training data was randomly split and left aside for validation. The remaining instances (90%) was used to train the various classification models.

4 Proposed approach

To begin with, a Bernoulli Naïve Bayes classifier algorithm was developed without using the SKLearn librairy. One should note for a Bernoulli Naïve Bayes classifier to work, the features must be independent booleans, i.e. binary variables. The algorithm consists in first computing the marginal probabily of positive comments $P(y = 1)$ and negative comments $P(y = 0)$ in the training set. This is alreay known as this corresponds to the ratio of positive and negative comments in the training set (i.e. 0.5). Subsequently, the prior probabilities $\theta_{j,1} = P(x = 1|y = 1)$ and $\theta_{j,0} = P(x = 1|y = 0)$ can be computed for all features x_j by looping through all positive instances and calculating the ratio of instances where this feature $x_j = 1$, and repeating for all negative instances respectively. These two steps correspond to training the classifier. One can then use it for making predictions by calculating $\delta(x)$ and classifying the instance as positive if $\delta(x) > 0$, or as negative otherwise. This algorithm was used as the baseline model for this study, with only minimal text feature processing. The latter consisted in converting all capital letters in comments to lowercase characters, splitting comments in different tokens and creating binary feature variables indicating the presence (or absence) for the top 1000 most frequent words in the combination of positive and negative comments. This number was chosen semi-arbitrarily from the fact that the Naïve Bayes model is relatively quick to compute and can be trained with large numbers of features without incurring penalties. From Zipf's law, it is known that word frequencies is inversely proportional to their rank, which means that there would be less and less benefit in incrementally adding more words. The model accuracy on the validation set is **0.775**, which is the baseline for the models described below. The *BernoulliNB()* function form the SKLearn librairy gives the same answer.

$$\delta(x) = \log\left(\frac{\theta_1}{1-\theta_1}\right) + \sum_{j=1}^{m}\left(x_j \log\left(\frac{\theta_{j,1}}{\theta_j}\right) + (1-x_j)\log\left(\frac{1-\theta_{j,1}}{1-\theta_{j,0}}\right)\right)$$

Three different pipelines were created for processing the text data and each used a different classifier from SKLearn, namely a) Multinomial Naïve Bayes, b) Linear SVM and c) Logistic Regression. Each pipeline enabled using different text processing techniques such as using binary occurences, frequency based statistics and the TF-IDF weighing. These pipelines were also used to explore various hyperparameters with the *RandomizedSearchCV()* function and find the optimal settings for each.

The popular sentiment analyzer VADER and it's lexicon [1] were also implemented and tested on this dataset and compared against the ML pipelines as a non-ML rule based alternative. The algorithm was obtained from the author's Github account and used as is.

In the attempt to obtain more accuracy, various ensembling stacking meta-models were developed and tested. The best of each classifier models were selected and used to generate predictions and the VADER predictions was also used as an input, for a total of four different features. Different meta-model options were investigated: namely a voting method with and without the VADOR feature, and a logistic regression method with and without the VADOR feature, as well as with and without interaction features. All results are provided in the following section.

5 Results

Following are the incremental improvements made when adding in the text pre-processing techniques and pipeline improvements. It is interesting to notice that from the baseline accuracy of **0.775**, the multinomial NB model's accuracy is dropping significantly when including all words. Another key finding is that all of text preprocessing steps only essentially do not improve the accuracy score, even though they enable greater improvements with the subsequent steps. The main contributors to improving the accuracy are adding bi-grams, and removing terms that appear too frequently in the corpus (via max df in the count vectorizer).

Incremental change	accuracy score (validation set)
Multinomial NB all words	0.546
Remove URLs	0.546
Remove emails	0.546
Remove punctuation	0.552
Lemmatization & stop words	0.552
n-gram (1, 2)	0.712
n-gram (1, 3)	0.780
TF-IDF	0.780
$maxdf = 0.5$	0.893
$alpha = 0.0107$	**0.897**

The optimal multinomial NB machine learning pipeline thus yield an accuracy of **0.897**. The optimal linear SVM model performs the best with an accuracy of **0.903** on the validation set, which is obtained with including monograms and bigrams only, using a max df of 0.25, TF-IDF and an inverse regularization of 1. The logistic regression model also performs quite well with an accuracy of **0.890**, when using monograms and bigrams, a max df of 0.5, TF-IDF, a dual formulation and an inverse regularization strength of 2. The VADOR sentiment analysis tool performs less well and obtains an accuracy of **0.694**.

An ensemble stacking meta-model was also developed using the predictions from each model and that of the VADOR algorithm. Using a voting meta-model, the ensemble model yielded an accuracy of **0.902** on the validation set, which dropped to **0.900** whithout using the VADOR's prediction as an input. A logistic regression meta-model was also tested but the accuracy didn't improve beyond **0.900**. Interestingly, when adding interaction terms between model predictions in the form of the product of the classes, the metamodel's accuracy increased to **0.904** on the validation set, which is the top score reported in this study. However, this latter meta model was then used to generate predictions on the test set but didn't perform as well as the single SVM model and thus an ensemble model was not selected. In the end, the best model selected for this task was found to be the linear SVM model described above. The model was run on the test set and the solution was uploaded on **Kaggle** and the accuracy was **0.90266**.

6 Discussion and Conclusion

In this study, we have outlined several machine learning pipelines that can be used to predict whether the sentiment of IMDB comments is positive or negative. One key takeaway was that some pre-processing steps seem to have very little impact on the model performance by themselves, but may enable greater improvements in subsequent steps. The introduction of bi-grams and tri-grams was found to be very effective. Another takeaway was that it the models explored in this study can run relatively fast on all words but one measure that is quite effective for improving the accuracy consist in limiting the words considered to only the ones that do not appear too frequently in the corpus (via the max df hyperparameter). The selected model and hyperparameters had an accuracy score of **0.903** on the validation set and **0.90266** on the test set. It is believed that this score can be improved further. Possible directions for future improvements include:

- **A)** Try out other classifiers (e.g. Decision Trees, Random Forest) and ensembling methods (e.g. bagging and boosting) for better performance.
- **B)** Try out better lemmatization librairies (e.g. spacy)
- **C)** Try out word2vec neural networks

7 Statement of Contributions

- **J-S Grondin** Primary role was to implement the Bernoulli Naïve Bayes algorithm, building and experimenting with ML pipelines. Also was responsible for developing and evaluating an ensembling stacking meta-model. Finally, contributed to the writing of the report.
- **J El Baze** Primary role was to construct the dataset, extract features and build the text processing functions. Also responsible for building ML pipelines and investigate how the VADER sentiment lexicon can improve predictions. Also responsible for running experiments and writing the report.
- **D Gilbert** Managed to do absolutely nothing!

References

[1] C.J. Hutto & Eric Gilbert *VADER: A Parsimonious Rule-Based Model for Sentiment Analysis of Social Media*. Georgia Institute of Technology, Atlanta, 2014

[2] Bo Pang & al. *Thumbs up? Sentiment Classification using Machine Learning Techniques*. Cornell University, 2002

[3] Sida Wang & al. *Baselines and bigrams: simple, good sentiment and topic classification*. Jeju Island, Korea, July 2012