MiniProject 2
IMDB Sentiment Analysis

Han Wen Xie	Michel Abdel Nour	Le Nhat Hung
260641053	260725050	260793376

Abstract

In this study, we investigated the performance of different Natural Language Processing (NLP) models to perform sentiment analysis on Internet Movie Database movie reviews. We applied three different classifiers for binary classification : Bernoulli Naïve Bayes, Logistic Regression and Support vector Machines. The features extracted for binary classification include bag of words, tf-idf weighting and N-grams. We found that the best performing model was the Logistic Regression model, performing with a validation score of 0.906 and 0.904 on the Kaggle test set, using bigrams, tf-idf weighting, L2 regularization and a 5-fold cross validation.

Introduction

Sentiment analysis is an important aspect of NLP. This project introduces the different ways for feature extraction and how different binary prediction models compare to each other. In the following text, we will present the performance of the Bernoulli Naïve Bayes model, Logistic Regression and Support Vector Machines. We are given a 50,000 movie review dataset from Internet Movie Database fetched from Kaggle. Our goal is to predict if a movie review is positive or negative. Varying various parameters, hyper-parameters and features can greatly impact the performance of each model. Using the features: tf-idf and different N-grams, we perform an in depth analysis of the best performing model: logistic regression. We improved our model using K-cross validation and L2 regularization. The logistic regression model achieved a mean score of 0.906 on the validation set compared to 0.870 from Bernoulli Naïve Bayes and 0.886 from Support Vector Machines.

Related work

The sentiment analysis problem takes its roots in the studies on public opinion analysis at the start of the 20^{th} century (Mantyla et al., 2014). The Sentiment classification problem in Natural language processing is a recent and fast growing research area in computer science, as most papers are published after 2004 (Mantyla et al., 2014. Liu, 2012). The problem of sentiment analysis became widely researched because for the first time in history, a large quantity of opinionated data is available on the web (Mantyla et al., 2014. Liu, 2012). From a relatively old paper from 2002, investigating the same issue on hand as us regarding movie review classification, they learn that standard machine learning models outperform human-produced baselines (Pang et al., 2002). Using the basic models of Naïve Bayes, Maximum Entropy and SVM, they achieved results of up to 0.827 accuracy compared to a human inspection reaching a maximum of 0.69 accuracy (Pang et al., 2002).

Nonetheless, the sentiment analysis problem is not restricted to online product reviews only. It can be applied to a broad range of industries including social media posts, stock markets, medicine and elections to name a few (Mantyla et al., 2014). The sentiment analysis problem drives every industry and social domain (Liu, 2012). We, as humans, are susceptible to public opinion. Our decisions and how we perceive reality are affected by how everyone around us sees the world (Liu, 2012). This is why we seek the opinion of others when making choices (Liu, 2012). In this text, we will research on a domain of sentiment analysis by classifying movie reviews using the written content only.

Dataset and setup

In this experiment, 50,000 raw text Internet Movie Database (IMDB) movie reviews were extracted from Kaggle. It is separated into a training and test set, each consisting of 25,000 data points. The training set consists of 12,500 positive and 12,500 negative reviews. The test set has its review scores hidden. Each data point in the test set is simply a random review associated with an ID. To evaluate how well our models perform on the test set, a csv file is generated where each row contains the ID number and the binary classification from our model.

All of the data was pre-processed to clean up the data. We use tokenization on each word after lower casing them. As for removing stop words, we investigated the usefulness of this extra step, and through multiple research papers, it is concluded that often removing stop words can negatively impact classification performance when doing natural language processing (Saif, 2015. Yuang, 2012).

Proposed approach

Following the pre-processing steps, we computed different numerical features using the built-in libraries in scikit learn:

Bag of words (BOW):
The normal bag of words is the most widely used first extraction step for text processing as it is the simplest feature to implement. We first find all, n, unique words in the given training set by tokenizing. Each data point will have a vector of size n where each digit will represent how many times that specific word occurs. This will usually create a rather large matrix depending on the size of the training set and the number of words we limit to.

Binary bag of words:
A binary bag of words is simply a modification of the simple BOW feature. The major difference is that each vector will only have a binary value to show if the word is present or absent in this specific data point instead of a count vector.

TF-IDF weighting:
Term Frequency Times Inverse Document Frequency (tf-idf) is a very useful feature in NLP. This feature allows us to give an importance measurement to each word as they are not equally important (Hamilton, 2019). Words that appear more often or a rare word has more meaning (Hamilton, 2019). Given a dataset/corpus, The TF is simply the word count (Hamilton, 2019). IDF is given by *Eq.1*:

$$IDF(termT, Corpus) = \log \frac{TotalDocs}{(DocsWithTermT) + 1} \tag{1}$$

N-grams:
N-grams are continuous sequences of n words, they can be unigrams, bigrams, trigrams or even bigger sequences (Hamilton, 2019). N-grams are an useful additional feature as words can have different meanings depending on the context. We tested using different ranges of N-grams from (1,1) to (1,4).

Three different binary classification models were implemented and compared using the features described:

Bernoulli Naïve Bayes model:

The Naïve Bayes method is based on applying Bayes' theorem, where we assume that x_j are conditionally independent given y. Even though this assumption does not usually hold in real-world examples, the Naïve Bayes model still perform well (Lewis, 1998). Moreover, it is shown that the model approaches the asymptotic error much faster compared to other discriminative models such as logistic regression (Ng et al., 2002). The Bernoulli version implements the Naïve Bayes model where each feature is a binary value. The decision formula shown in *Eq.2* allows us to predict: if $\delta(x) \geq 0$

classify as 1, if $\delta(x) < 0$ then classify as 0 (Hamilton, 2019). We used the binary bag of words feature to test the model's implementation correctness. Following, that we tested by varying features and parameters.

$$\delta(x) = \log \frac{\mathrm{P}(y=1)}{\mathrm{P}(y=0)} = \log \frac{\theta_1}{1-\theta_1} + \sum_{j=1}^{m} \left(x_j \log \frac{\theta_{j,1}}{\theta_{j,0}} + (1-x_j) \log \frac{1-\theta_{j,1}}{1-\theta_{j,0}} \right) \tag{2}$$

with $\theta_1 = \frac{\text{examples where } y=1}{\text{all examples}}$, $\theta_{j,1} = \frac{\text{examples with } x_j=1 \text{ and } y=1}{\text{examples where } y=1}$ and $\theta_{j,0} = \frac{\text{examples with } x_j=1 \text{ and } y=0}{\text{examples where } y=0}$.

The Bernoulli Naïve Bayes model was implemented from scratch as a class with:

- 2 functions:
  - $fit(X_{train}, y_{train})$
  - $predict(X)$
- 4 fields:
  - k: constant for Laplacian smoothing
  - $binarize$: threshold. Feature values becomes 1 if $> binarize$, else 0
  - w_0: part of the decision boundary computation
  - w: part of the decision boundary computation

The log-odds ratio $\delta(x)$ *(Eq.2)* can be rewritten as

$$\begin{aligned} \delta(x) &= \log \frac{\theta_1}{1-\theta_1} + \sum_j \log \frac{1-\theta_{j,1}}{1-\theta_{j,0}} + \sum_j \left(\log \frac{\theta_{j,1}}{\theta_{j,0}} - \log \frac{1-\theta_{j,1}}{1-\theta_{j,0}} \right) x_j \\ &= \left(\log \frac{\theta_1}{1-\theta_1} + \sum_j w_{j,0} \right) + \sum_j (w_{j,1} - w_{j,0}) x_j \\ &= w_0 + x^T w \end{aligned} \tag{3}$$

In $fit(X_{train}, y_{train})$, the model learns w_0 and w. In $predict(X)$, for each example x, we compute $\delta(x)$ and classify the sentiment as positive if $\delta(x) \geq 0$, and negative if not.

Logistic regression:

Logistic regression is a type of discriminative learning where we can directly estimate the probability of $P(y|x)$ (Hamilton, 2019). Logistic regression outputs a probability between 0 and 1 using the real values and the logistic sigmoid function shown in *Eq.4* which can be mapped to discrete classes, in our case to a binary classification. We set the decision boundary to where x=0 (Hamilton, 2019).

$$\sigma = \frac{1}{1+\exp^{-x}} \tag{4}$$

To learn the weights, we can maximize the log-likelihood function which is also equivalent to minimizing the negative of the log-likelihood known as the cross-entropy loss shown in *Eq.5* (Hamilton, 2019). The cross-entropy loss measures how many bits of information is needed to correct the errors from our model (Hamilton, 2019).

$$cross-entropy(D) = -\sum_{i=1}^{n} y_i ln(\sigma(w^T x_i)) + (1-y_i) ln(1-\sigma(w^T x_i)) \tag{5}$$

The main feature used in logistic regression is tf-idf weighting. However, since the word count is so high using the whole corpus, we use L2 regularization to penalize the complexity and reduce all the

feature weights. This in turn will avoid over-fitting issues as it decreases variance. We also use k-fold cross validation to reduce over-fitting. The logistic regression model implementation is directly taken from sklearn.

Support Vector Machine:
Support vector machines (SVMs) are very effective in text categorization and usually outperform Naïve Bayes (Joachims, 1998). They are considered as margin classifiers in contrast to the probabilistic Naïve Bayes and Logistic Regression models (Hamilton, 2019). It is a constrained optimization problem. The goal of this model is to find a vector $\vec{w}$ that represent a hyper-plane that can separate the data points from one class from another, but also maximize this margin (Hamilton, 2019). We can simply use the perceptron learning rule where we will loop through all misclassified examples x_i and perform the update: $w = w + ay_i x_i$ until all training examples are correctly classified (Hamilton, 2019). The solution is simply:

$$\vec{w} = \sum_{i}^{n} a_i c_i \vec{d_i} \tag{6}$$

Where the a_i are obtained from solving a dual optimization problem. The c_i are either -1 or 1 for binary classification. The d_i, documents, that have a a_i greater than 0 are the support vectors used to build this margin identified by the vector w (Pang, 2002). We used the built-in library for SVM in sklearn.

Results

Classifier results:

We used all three classifiers introduced in our text to perform binary classification. Using cross validation, as shown in *Table 1*, the Bernoulli Naïve Bayes model performed at best with a validation score of 0.870. We tested the BNB model using a binary TF-IDF, regularization and cross validation. SVM performed with a maximum score of 0.886 and the best logistic regression reached the highest validation score of 0.906. We tested all three models in a similar fashion by varying the parameters and features. In the following section, we will discuss on how we achieved these results and show the details of the implementation for the logistic regression model as an example.

Table 1: Classifier performance

Models	Performance
Bernoulli Naïve Bayes	0.870
Support Vector Machine	0.886
Logisitic Regression	0.906

Logistic regression pipelines:

We used many different feature extraction pipelines for processing the text data for the logistic regression model. As seen in *Table 2*, we used SciKit Learn's GridSearch to test using a range of parameters on the different features. As for N-grams, we tested over multiple ranges from 1 to 4 word sequences. The tf-idf weighting was tested by including the idf part in our test models or not. As for L2 regularization, we used the hyperparameter C to control. The default regularization hyperparameter C is set at 1.0. C is the inverse of the regularization strength. In other words, the lower C is, the stronger the regularization. We tested by varying the range from as little as 0.001 to 100. Lastly, we used a 5-fold cross validation to eliminate over fitting issues in all our runs.

Table 2: GridSearch Parameters

Parameters	Parameter range
vect ngram range	[(1,1),(1,2),(2,2),(1,3),(1,4)]
tfidf use idf	[True, False]
clf C	[0.001, 0.01, 0.1, 1, 10, 100]

Gathered in *Table 3* are the results from some of our best models while varying parameters and features in the pipeline for logistic regression. As seen, the mean fit time increases by a significant amount when testing with larger N-gram ranges. Additionally, larger N-grams do not provide better results compared to only adding bigrams. By testing, we see that the hyperparameter, C, performs the best at a value of 100. We can also deduce that as the model gets more complex, a higher validation score can be achieved, but K-fold cross validation and regularization is necessary for large text feature models to remove the overfitting issue. Our best performing model, resulting in a validation score of 0.906, has the following parameters: TF-IDF, bigrams, L2 regularization under a hyperparameter C of 100. We tested our best model from our validation results and ended with a 0.90413 test score on Kaggle.

Table 3: Logistic Regression model performance with 5-fold cross validation

Logistic Regression	N-gram range	C(Regularization)	Mean Fit Time (s)	Mean Validation Score
Bag of Words	N/A	N/A		0.867
TF-IDF	N/A	N/A		0.871
TF-IDF, L2	N/A	10		0.881
TF-IDF, L1, N-grams	(1-2)	10		0.888
TF-IDF, L2, N-grams	(1-4)	100	103.71	0.897
TF-IDF, L2, N-grams	(1-3)	100	67.24	0.902
TF-IDF, L2, N-grams	(1-2)	10	25.83	0.903
TF-IDF, L2, N-grams	(1-2)	100	28.74	0.906

Discussion and Conclusion

To summarize, the project compared the performance of Bernoulli Naïve Bayes model, logistic regression model and Support Vector Machine for binary classification of IMDB reviews. The primary takeaways are that k-cross validation and regularization are necessary on large dataset where large amounts of text features are present to avoid overfitting issues. Additionally, bigrams are the most useful feature compared to higher N-grams by accuracy and fitting time. Most importantly, varying various parameters, hyperparameters and features can greatly impact the performance of each model. Logistic regression performed the best achieving a validation score of 0.906 with very specific parameters: TF-IDF, bigrams, L2 regularization under a hyperparameter C of 100 and a 5-fold cross validation.

To conclude, we think that our models are performing poorly on data points where the author uses many words against the general sentiment of the entire review or reviews where the author states both positive and negative points from the movie. As others note for online reviews, (Turney, 2002), "the whole is not necessarily the sum of the parts". Better investigation on the data can be additional features deciding the overall meaning of a sentence instead of looking only at words. With enough testing, a more desirable outcome of Logistic Regression can be achieved when testing on the validation set. We also believe that features involving the time and date of posting can increase the models prediction accuracy. This could be further explored using classes such as TimeseriesGenerator, part of the keras library, that allows to validate with data acquired at equal time intervals.

Statement of Contributions

Han Wen Xie: Write-up, Pre-processing, testing tf-idf and logistic regression

Le Nhat Hung: Bernoulli Naïve Bayes, Logistic Regression, Feature Testing, L2 Regularization, Feature Extraction Pipeline, Cross-Validation

Michel Abdel Nour: Write-up, L2 Regularization, Feature Testing, Cross-Validation

References

Hassan Saif, Miriam Fernandez, Yulan He, Harith Alani. 2015. On Stopwords, Filtering and Data Sparcity for Sentiment Analysis of Twitter. *In LREC 2014, Ninth International Conference on Language Resources and Evaluation. Proceedings.*

Chong Tze Yuang, Rafel E. Banchs and Chng Eng Sion. 2012. An Empirical Evaluation of Stop Word Removal in Statistical Machine Translation. *In Proceedings of the 13th Conference of the European Chapter of the Association for Computational Linguistics.*

Bo Pang, Lillian Lee, Shivakumar Vaithyanathan. 2002. Thumbs up? Sentiment Classification using Machine Learning Techniques. *In Proc. 2002 Conf. on Empirical Methods in Natural Language Processing (EMNLP).*

Mika V. Mäntylä, Daniel Graziotin, Miikka Kuutila. The Evolution of Sentiment Analysis - A Review of Research Topics, Venues, and Top Cited Papers. 2018. *In Computer Science Review.*

Gang Wang, Jianshan Sun, Jian Ma, Kaiquan Xu, Jibao Gu. 2014. Sentiment classification: The contribution of ensemble learning. *In Decision Support Systems.*

Thorsten Joachims. 1998. Text categorization with support vector machines: Learning with many relevant features. *In Proc. of the European Conference on Machine Learning (ECML).*

Andrew Y. Ng, Michael I. Jordan. On Discriminative vs Generative classifiers: A comparison of logisitic regression and Naïve Bayes. 2002. David D. Lewis. 1998. Naïve (Bayes) at forty: The independence assumption in information retrieval. *In Proc. of the European Conference on Machine Learning (ECML). Invited talk.*

Bing Liu. 2012. Sentiment Analysis and Opinion Mining. *In Synthesis Lectures on Human Language Technologies.*

William L. Hamilton. 2019. COMP 551 - Applied Machine Learning Lectures.

Peter D. Turney and Michael L. Littman. 2002. Unsupervised learning of semantic orientation from a hundred-billionword corpus. *In Technical Report EGB-1094, National Research Council Canada.*