CS 412 Final Report

Elijah Charbel

1 Sentiment Analysis

This project seeks to investigate and compare the performances of three machine learning models.

1.1 Problem Overview

Given a collection of text documents, the goal of sentiment analysis is to classify the polarity that each document carries. Most commonly, this task involves ascribing a positive, negative, or neutral label to each document. There exist specialized models capable of multiclass classification, which assign specific labels (ex. excitement, anger, sadness) to documents. This project directs its attention towards binary positive/negative sentiment classification.

The ability to accurately classify the polarity of written information holds value across a multitude of domains. Researchers may be interested in analysing business reputation, customer opinions of products, and the effectiveness of marketing campaigns. Social media posts, product reviews, and customer interactions such as customer support chats can all be analyzed to extract and understand public opinion. These opinions can be used downstream to monitor brands, predict preferences, and update marketing strategies.

As an example, we can use sentiment analysis to classify the following fictitious YouTube comments:

> "Thanks for uploading this amazing presentation. This video really helped clarify the topic"
>
> "There goes 10 minutes of my life that I'll never get back. This has to be the worst explanation I've ever suffered through. Truly terrible."

It is fairly easy for humans to identify that these are highly polarized comments; the first carries a positive sentiment, and the second carries a negative sentiment. Next we will explain how it is possible for machines to do the same.

2 Approaches Used

2.1 Dataset

This project uses the Large Movie Review Dataset from the Stanford AI lab, which contains 50000 highly polarized movie reviews written by users of the Internet Movie Database (IMDb).

Table 1: Dataset details

SENTIMENT	**COUNT**
Positive	25000
Negative	25000

From the table above, we can see that the dataset is extremely class balanced. Due to memory limitations, this project uses the first 10000 entries of the dataset. Class balance is preserved after this truncation.

2.2 Models Used

The following models were implemented using open source Python library, Scikit-learn.

2.2.1 Logistic Regression

Logistic Regression transforms linear combinations of input features to a sigmoid curve that represents probability of class assignment. The parameters are optimized using maximum likelihood estimation, estimating parameters based on an assumed probability distribution given our data.

For our binary classification task, this technique predicts the probability of the positive class as

$$P(y_i = 1 \mid X_i) = \hat{p}(X_i) = \frac{1}{1 + e^{-X_i w - w_0}}$$

and minimizes the cost function across n training examples:

$$\min_w \frac{1}{S} \sum_{i=1}^{n} s_i(-y_i \log(\hat{p}(X_i)) - (1 - y_i) \log(1 - \hat{p}(X_i)) + \frac{r(w)}{SC}$$

where s_i are training example specific weights with $S = \sum_{i=1}^{n} s_i$, and regularization terms $r(w)$ and C.

2.2.2 Naive Bayes

Naive Bayes model is a probabilistic classifier operating under the assumption that input data features are independent conditional on class assignment. We calculate prior probability of each class, $P(Y_i)$, and likelihoods, $P(X_i \mid Y_i)$, to determine posterior probability, $P(Y_i \mid X_i)$, using Bayes' Rule:

$$P(Y \mid X) = \frac{P(X \mid Y)P(Y)}{P(X)}$$

The class with the greatest posterior probability becomes our model's prediction. In this project, we use a Multinomial Naive Bayes classifier which assumes a Multinomial distribution of the data.

2.2.3 K-Nearest Neighbors

For a given test example, the K-Nearest Neighbors model measures the distances between our test example and each of the training examples. The majority class among the K closest training examples is chosen as the model prediction.

To optimize performance, we must carefully choose the best K value. This is accomplished through 5-fold cross validation. Choosing a K value from a collection of potential candidates, we partition the training data into 5 class balanced sets. The model is trained using 4 of these sets and performance is assessed on the left out set. This is repeated until each of the 5 sets has been used for validation. We then use the mean of these 5 scores to evaluate the performance for each k value we are considering.

2.3 Text Representation Methods

Our dataset consists of many text reviews, and we would like to represent our data's features numerically (instead of as words) so our machine learning models can understand and process them. We accomplish this using two text representation methods.

2.3.1 Bag of Words

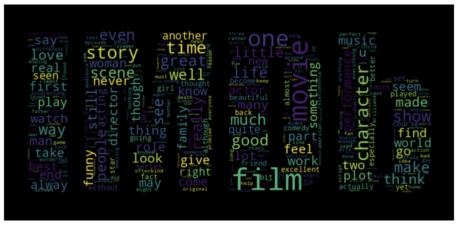
The Bag of Words model represents text as an unordered collection of word counts. Despite this method's disregard of word order and grammar, this approach can work surprisingly well.

2.3.2 Term Frequency - Inverse Document Frequency

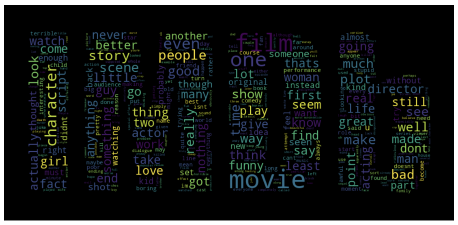
TF-IDF considers two statistics: term frequency, which captures the relative frequency of a word within a document, and inverse document frequency, which captures term frequency across all documents (i.e. how many documents in the corpus contain the term or word).

2.4 Dataset Preprocessing

Before converting raw text to BoW and TF-IDF representations, we have to do some common NLP data preprocessing. This includes the following tasks: Converting sentiment labels from string (negative/positive) to numbers (0/1). Converting text to lowercase. Remove HTML tags. Remove punctuation. Perform word tokenization. Remove stopwords (words that carry little semantic meaning). Perform lemmatization (convert words to their root words ex. 'better' → 'good'). We generate the following WordClouds to inspect the most common words among positive and negative reviews respectively.



Positive Words



Negative Words

Figure 1: Common Words in Positive and Negative Reviews

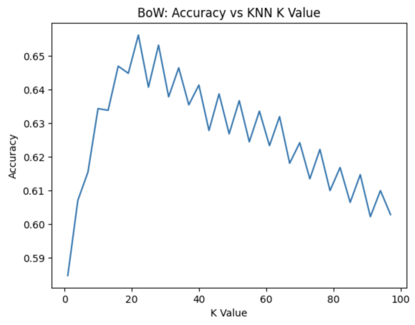
3 Evaluation Measures

For each model, we construct confusion matrices detailing the numbers of true positives, true negatives, false positives, and false negatives from our predictions. We use these values to calculate various performance measures including accuracy (the ratio of true predictions over tall predictions), precision (the ratio of true positives over true and false positives), recall (the ratio of true positives over true positives and false negatives) and F1 score (the harmonic mean of precision and recall). Since our data is class balanced, we can expect accuracy and F1 scores to be similar. Regardless, we will look at these two evaluation measures.

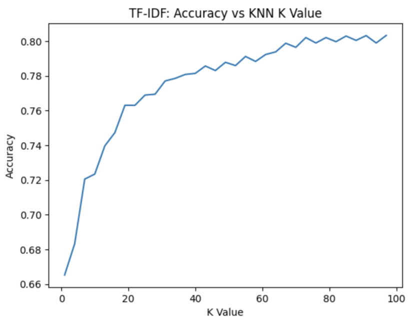
4 Results

4.1 K Value Cross Validation Results

After performing an 80:20 split on our data for training and testing, we conduct cross validation to determine the best K values for the K-NN models. Figure 2 reveals the optimal K Values for BoW and TF-IDF text representations.



Bag of Words: Best K value = 22

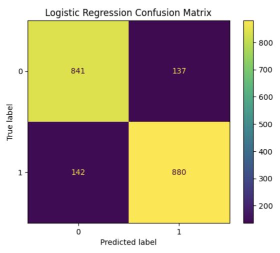


TF-IDF: Best K value = 97

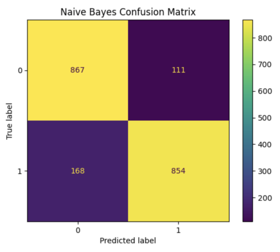
Figure 2: K Value Cross Validation Results

4.2 Bag of Words Results

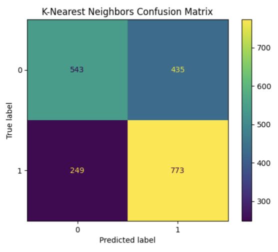
Using the BoW text representation for our models, we find that Logistic Regression model classifies reviews with 85.9% accuracy and 86.1% F1 score. The Naive Bayes model classifies reviews with nearly identical results: 86% accuracy and 85.9% F1 score. The K-NN model has 67.6% accuracy and 71.9% F1 Score. Figure 3 displays the confusion matricies for each of these models.



Logistic Regression



Naive Bayes

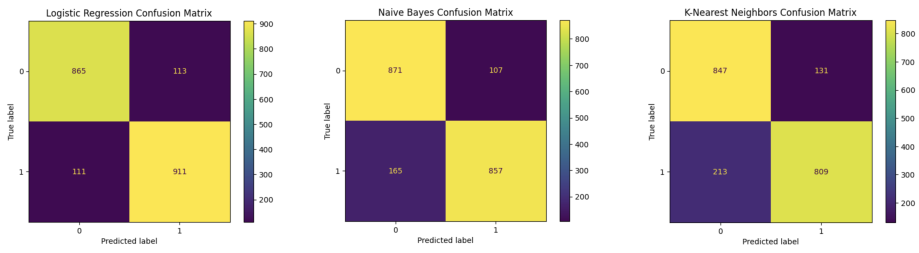


KNN

Figure 3: Bag of Words Model Confusion Matricies

4.3 Term Frequency-Inverse Document Frequency Results

Using the TF-IDF text representation for our models, we find that Logistic Regression model classifies reviews with 88.8% accuracy and 89.1% F1 score. The Naive Bayes model classifies reviews with 86.4% accuracy and 86.3% F1 score. The K-NN model has 82.8% accuracy and 82.5% F1 Score. Figure 4 displays the confusion matricies for each of these models.



Logistic Regression Naive Bayes KNN

Figure 4: TF-IDF Model Confusion Matricies

5 Lessons Learned

The results of our experiments demonstrate the importance of cross validation for tuning the hyperparameter K in for use in our K-Nearest Neighbors models. By default, the Scikit-learn library uses K=5, which would provide far lower performance.

Additionally, we have learned that our three models using Term Frequency - Inverse Document Frequency text representation outperform the same three models using Bag of Words text representation. We conclude that document frequency is an important statistic to consider when converting text to numerical representation, and this reveals the importance of word frequency within individual examples and the corpus as a whole in classifying text as positive or negative.

Experiment results also indicate that Logistic Regression outperforms Naive Bayes and K-Nearest Neighbors models in binary sentiment classification.

References

[1] Maas, Andrew L. and Daly, Raymond E. and Pham, Peter T. and Huang, Dan and Ng, Andrew Y. and Potts, Christopher (2011) Learning Word Vectors for Sentiment Analysis, *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pp. 142-150. Portland, Oregon: Association for Computational Linguistics.

[2] Pedregosa, F. and Varoquaux, G. and Gramfort, A. and Michel, V. and Thirion, B. and Grisel, O. and Blondel, M. and Prettenhofer, P. and Weiss, R. and Dubourg, V. and Vanderplas, J. and Passos, A. Cournapeau, D. and Brucher, M. and Perrot, M. and Duchesnay, E. (2011) Scikit-learn: Machine Learning in Python, *Journal of Machine Learning Research*, pp. 2825-2830.