Donia Zaheri (DoniaZ2)
Abishek Samuel (asamuel4)

PROJECT 3
Movie Review Sentiment Analysis

Introduction And Goal

The goal of this project is to perform a sentiment analysis on movie reviews. A subset of the IMDB movie review dataset is divided into five different splits, each containing a training set and a test set with 25000 data points respectively. The goal is to achieve an AUC level of 0.96 or better using a vocabulary size of 1000 words or fewer.

Preprocessing the Data

Each split contains 25000 data rows and four data columns as described below:

- "Id": The identification number
- "Sentiment": 0 = negative and 1 = positive
- "Score": The 10-point score assigned by the reviewer. Scores 1-4 correspond to negative sentiment; Scores 7-10 corrspond to positive sentiment. This dataset contains no reviews with score 5 or 6 (training data doesn't have the score column)
- "Review": the review of the reviewer

The data pre-processing includes cleaning the data to separate out useful words, and converting the remaining words to numerical tokens, which can be used as inputs to fit to the selected model.

Pre-processing 1: Removing HTML Tags And URLs:

For the first step, all html urls and tags were removed as they are irrelevant to the data analysis. The python package 're', which refers to the regular expression function, was used for this process.

Pre-processing 2: Removing Special Characters, Upper-Case Letters, Stop Words and Performing Tokenization

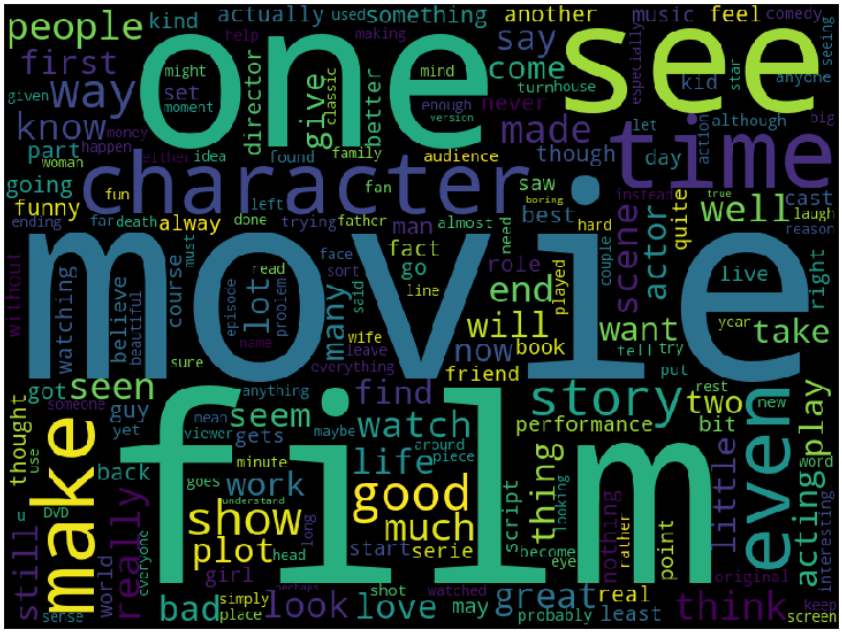
Every language syntax contains common words and connectors such as 'they', 'is', 'and', etc. These words are frequently used in sentences, but they do not accurately convey the meaning or context of a document. These words are referred to as 'stop words' in sentiment analysis. Removing 'stop words' from a document improves the performance of the model, particularly a binary classification model. The vectorizer function ('CountVectorizer' function from 'sklearn' package, in this case) is useful for word removal, ngram filtering and document frequency filtering. The function also automatically removes special characters not relevant to word context. For this project, the list of common stop words were provided by the instructors for data cleaning.

Words were also converted to lowercase format, and the final filtered words were tokenized to a numerical format for use in the classification models.

After Tokenization, a LinearSVC model was used to create the final vocabulary list containing 994 words. Refer to the HTML markup file for this vocabulary generation process.

Data Analysis:

An initial Exploratory data analysis was performed to better understand the data. These techniques included plots of the distributions of positive and negative sentiments which illustrates that the data distribution is balanced and is safe to use for training classification models. Another data analysis step was to get distributions of the number of words in reviews for training and test sets. Finally, a word cloud of all the words in the training data was plotted, to illustrate word frequency. (figure below).



Classification Model Implementation:

Three different models were implemented to achieve the goal of an AUC score of 0.96 or better. All model implementations were from the 'sklearn' library. SInce this is a classification problem, we started with logistic regression. The best result that was obtained for logistic regression was 0.9024. Then, the SVM model was implemented to see if the AUC score can be improved. However, the best score obtained from SVM was still very similar to logistic regression. At the end, Ridge Classification model was the only model that provided an AUC score close to benchmark results.

Logistic Regression

This was the first model that was implemented. After fitting and transforming the data with 'CountVectorizer', the training data was fitted to a logistic regression model with a regularization parameter and l2 (ridge) penalty. linear_model.LogisticRegression from 'sklearn' library was used for this purpose. The parameters selected for this model were below:

- C = 0.52: The inverse of a regularization strength applied to the model. The lower the parameter C, the stronger the regularization
- Max_iter = 1000000: maximum number of iterations. Increasing Max_iter too much causes overfitting of the data. It also increases the running time of the model.
- Penalty = 'l2': using l2 (ridge) penalty for regularization
- solver='Saga': The solver for the model

The training and test errors in terms of an AUC score, were very similar. Logistic regression did a decent job in predicting the sentiment with AUC of 0.9024. Given the simplicity of the model, this was still impressive - but did not make the benchmark.

SVC

In the second approach, a Support Vector Machine model was implemented. Fitting the transformed data from Countvectorizer again to the model, a very close result to Logistic Regression, was obtained. 'svm.SVC' model from sklearn library was used for this purpose. The parameters selected for this model were below:

- C = 0.001: The inverse of a regularization strength applied to the model. The lower the parameter C, the stronger the regularization.
- Gamma = 1e-1: Defines how much influence a single training example has. The larger gamma is, the closer other examples must be to be affected.
- kernel='rbf': Radial Basis Function was selected as the kernel function for SVM.

Ridge Classification

The final approach was implementing a Ridge Classifier. This provided the best performance so far, with a mean AUC score of .9538 across the five splits. The 'sklearn' implementation linear_model.Ridge was used for this implementation. The parameters selected for this model were below:

- alpha = 0.9: The inverse of a regularization strength applied to the model. The lower the parameter C, the stronger the regularization.
- tol = 0.001: Solution Precision

Performance

Split#	Ridge Classification	SVM	Logistic Regression
1	0.9546	0.8937	0.8948
2	0.9537	0.8942	0.8952
3	0.9529	0.8925	0.8935
4	0.9542	0.8966	0.8963
5	0.9538	0.8907	0.8906
Mean (AUC)	**0.9538**	**0.89354**	**0.89408**
Mean(Time) [sec]	**2.48131**	**2.95848**	**20.87164**

Interpretability, Limitation and Error Analysis of the model

Using ridge and logistic regression made the model more interpretable. These models give respective weights to the coefficients to their importance, hence we can understand the importance of each phrase in determining the meaning of the sentiment.

One reason for having errors in predicting review sentiment is because in some reviews there are both positive and negative phrases and their numbers are similar. Hence, it is harder for the model to make a correct prediction. Also the predictions for the models with moderate score had the most errors.

Another limitation with the model was where there were sarcastic comments or expressing the reviewers expectation versus what they thought of the movie after watching it, or quoting other reviewers in their review. Which makes the review and hence the model prediction more complicated. One suggested solution for this issue we found online was to use a high number of n-grams that would better convery the context in which the positive/negative phrases were used.

Conclusion

In this project sentiment analysis was performed with classification models for IMDB movie reviews. First the most significant words were selected with ridge regression and they were reduced to fewer than 1000 words with ridge regression with Countvectorizer. Then we used these words in our data to fit logistic, svm and ridge classification models. Logistic regression is limited due to its strict classification as it does not differentiate between 0.51 and 0.99 in assigning a label to a review. Models like SVM with more robustness towards outliers do better in these cases. For excellent prediction results on more complex text analysis deep learning models would be recommended.

System and Software Packages

System Specs: Operating System - Windows 10 (64 Bit), Processor - Intel i7 @ 2.60Hz, RAM - 32 GB

Software Packages: Python: 3.8.3; Scipy: 1.5.0; scikit-learn: 0.32.1; numpy: 1.18.5; pandas: 1.0.5;

Individual Contributions

Donia Zaheri (DoniaZ2) - Worked on data cleaning, data analysis, logistic regression model and svm model and writing the report.

Abishek Samuel (asamuel4) - Worked on the final vocabulary list selection and ridge classification model, along with other additional performance optimizations and writing the report.

Reference

[1]Original Project - https://www.kaggle.com/c/word2vec-nlp-tutorial

[2] Certain coding and model implementation practices were provided from the UIUC STAT 542 course by Prof. Feng Liang and the course's TA's.