OPINION MINING ON MOVIE REVIEWS

GROUP MEMBERS:

1.Rajesh Nemani
Mail Id – rajeshnemani@my.unt.edu
Role- WebApp Design, Deployment, Data Gathering

2.Tejeeswar Kommineni
Mail Id – tejeeswarkommineni@my.unt.edu
Role-Feature detection, Data preparation, Training Model

3.Anuhyasiri Kondepati
Mail Id – anuhyasirikondepati@my.unt.edu
Role-Training Model, Word Cloud, Documentation

4.Balaji Narravula
Mail Id – balajinarravula@my.unt.edu
Role-Documentation, WebApp Design

5.Jayakishan Presingu
Mail Id – jayapresingu@my.unt.edu
Role-WebApp Integration

ABSTARCT

The computational examination of people's opinions, sentiments, attitudes, and emotions conveyed in written language is known as opinion mining or sentiment analysis. In recent years, it has become one of the most active study fields in natural language processing and text mining. Because of its potential applicability to a wide range of sectors, opinion mining has become increasingly important in both commercial and scientific applications. As a result, a great number of businesses have made consumer sentiment and opinion analysis a component of their purpose. The rise of social media platforms such as reviews, forum discussions, blogs, microblogs, Twitter, and social networks has increased the importance of sentiment analysis. Sentiment analysis and sentiment classification are both text classification tasks in which you are given a phrase or a series of phrases and asked to determine whether the sentiment underlying them is positive, negative, or neutral. To maintain it a binary classification task, the third property is sometimes ignored.

INTRODUCTION

PROBLEM

Classifying the polarity of a given text at the document, sentence, or feature/aspect level — whether the conveyed viewpoint in a document, a sentence, or an entity feature/aspect is positive, negative, or neutral — is a basic task in sentiment analysis. When analyzed, reviews are typically effective in generating a large amount of sentiment data. These statistics are valuable in determining public opinion on a variety of topics. In order to compute the customer perspective, we must create an Automated Machine Learning Sentiment Analysis Model.

MOTIVATION

Sentiment Analysis is motivated by two factors. Consumers and producers alike place a high value on "customer feedback" on products and services. It is critical for us to know the opinions of those around us before making a decision. Previously, this circle was small, consisting of only a few close friends and family members. People are now sharing their thoughts on blogs and forums, thanks to the Internet. People who are looking for information about a specific entity are now actively reading these reviews(product, movie etc.). Vendors demand a system that can detect patterns in customer reviews and use them to improve their product or service while also identifying future needs.

Past Related Works

Many studies incorporating review data, such as movie review analysis and customer review analysis, have been completed as part of AI and machine learning initiatives. Many of these projects' implementations have proven to be beneficial to our project.

Learning Goals

The objective of sentiment analysis, regardless of the label, is the same: to determine a user's or audience's opinion on a target object by evaluating a large volume of text from numerous sources. We plan to learn by focusing on text classification in our scenario.

Apporach

We will start with a dataset that is publicly available. We started it with Kaggle dataset that has over 50 k reviews segmented. We will clean the dataset as the dataset contains so many extra spaces, characters. Then we would follow Hybrid based approach, later we split the dataset into training data and testing data in the form of 80:20, where 80% is training data and 20% is testing data.

Architecture

The main modules and models that were used in project are :

NLTK: The major library for creating Python programs that interact with human language data is NLTK. It provides easy-to-use interfaces to over 50 corpora and lexical assets, including WordNet, as well as a suite of text preparation libraries for tagging, parsing, classification, stemming, tokenization, and semantic reasoning wrappers for NLP libraries and an active chat debate. The NLTK library is available for Windows, Mac OS, and Linux. The nicest aspect is that NLTK is a free, open-source, community-driven endeavor. It has certain drawbacks as well. Matching the needs of production usage is sluggish and complex

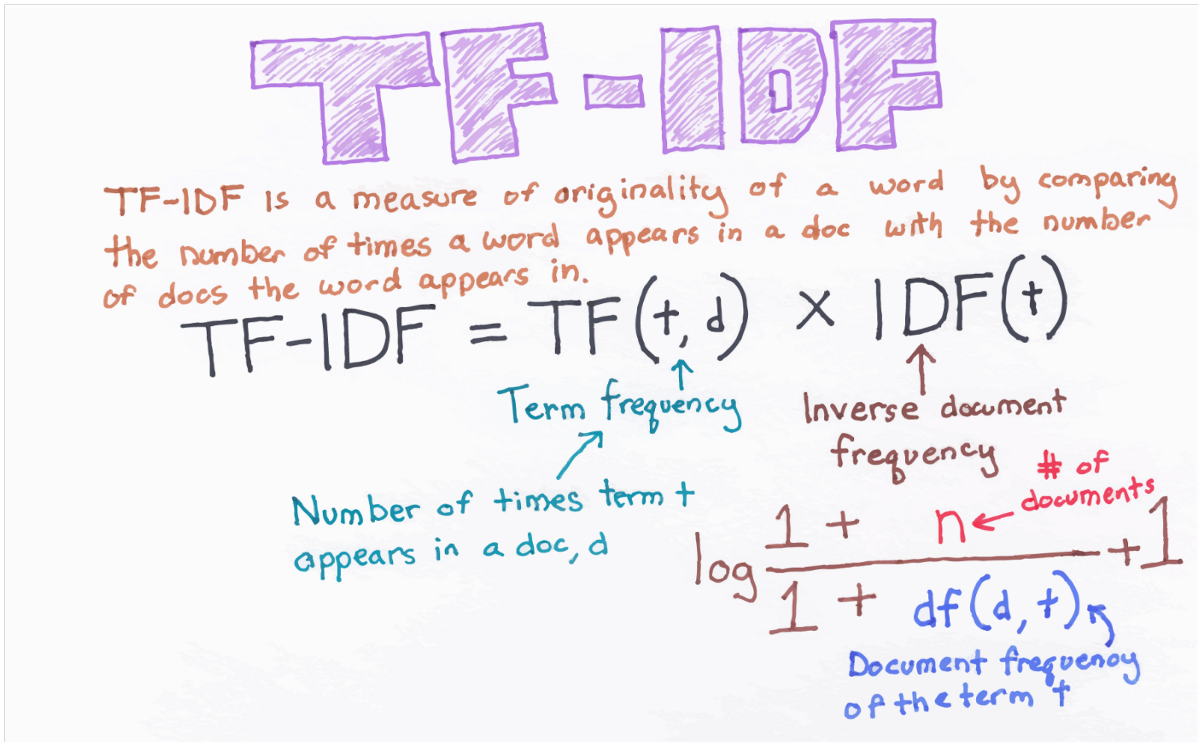
Stopword: Stop words are a group of words that are widely used in a language. Stop words in English include "a," "the," "is," "are," and others. Stop words are frequently used in Text Mining and Natural Language Processing (NLP) to exclude terms that are so widely used that usually contain very little meaningful information. We are using NLTK module for removing the stop words.

WordNet Lemmatizer: The process of reducing a word to its basic form is known as lemmatization. The difference between stemming and lemmatization is that lemmatization takes the context into account and transforms the word to its meaningful base form, whereas stemming just eliminates the final few letters, which sometimes results in inaccurate meanings and spelling mistakes.

Wordnet is a huge, freely and accessible to the public English lexical database that aims to develop organized semantic associations between words. It also has lemmatization capabilities and was one of the first and most widely used lemmatizers. It has an interface provided by NLTK, but you must first download it in order to use it.

Word Tokenize : Tokenization is the process of dividing a big amount of text into smaller pieces known as tokens. These tokens are extremely valuable for detecting patterns and are regarded as the first stage in stemming and lemmatization. Tokenization also aids in the replacement of sensitive data components with non-sensitive data elements. To separate a statement into words, we utilize the word tokenize() function. For improved text interpretation in machine learning applications, the result of word tokenization can be translated to Data Frame. It can also be used as an input for additional text cleaning procedures including punctuation removal, numeric character removal, and stemming. To be taught and produce predictions, machine learning models require numerical data. Tokenization of words becomes an important aspect of text to integer data conversion.

Tf-idf Vectorizer: Count Vectorizer returns the number of frequencies with relation to the index of vocabulary, whereas tf-idf considers the entire weight of words in the texts.



TF-IDF

TF-IDF is a measure of originality of a word by comparing the number of times a word appears in a doc with the number of docs the word appears in.

$$\text{TF-IDF} = \text{TF}(t, d) \times \text{IDF}(t)$$

Term frequency

Number of times term t appears in a doc, d

Inverse document frequency

$$\log \frac{1 + n}{1 + df(d, t)} + 1$$

# of documents

Document frequency of the term t

Modelling

Data Properties

We are drawing the data from Kaggle. It has tabular format which has 2 features namely revies and sentiment. The sentiment is further divided into 2 labels which are positive and negative. The dataset has equal number of positive and negative reviews.

Data Preprocessing

First we load our data from the .csv file into the dataframe. Then we check if the data is loaded properly or not using the head(), later we check the description of the dataframe , which is followed by check for null values or missing values. Then we start the preprocessing of data.In preprocessing we remove punctuations, extra spaces, html breaks, and any unnecessary text in the review which can be considered as noise. Then we go with Lemmatization.

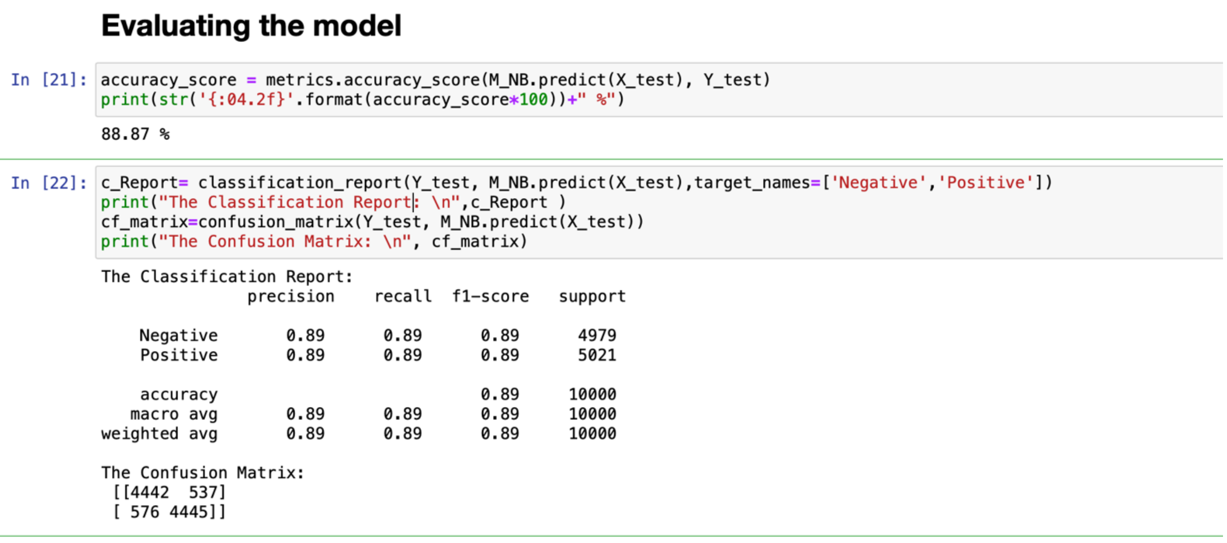
Model

The data which is processed would be fed to the model. Before we feed it into the model we would split the data.In this case we are using MultinomialNB and LinearSVC to check which has given more accuracy.

Performance Metrics

The below are the results of the model performance on the test data.

MultinomialNB Results:



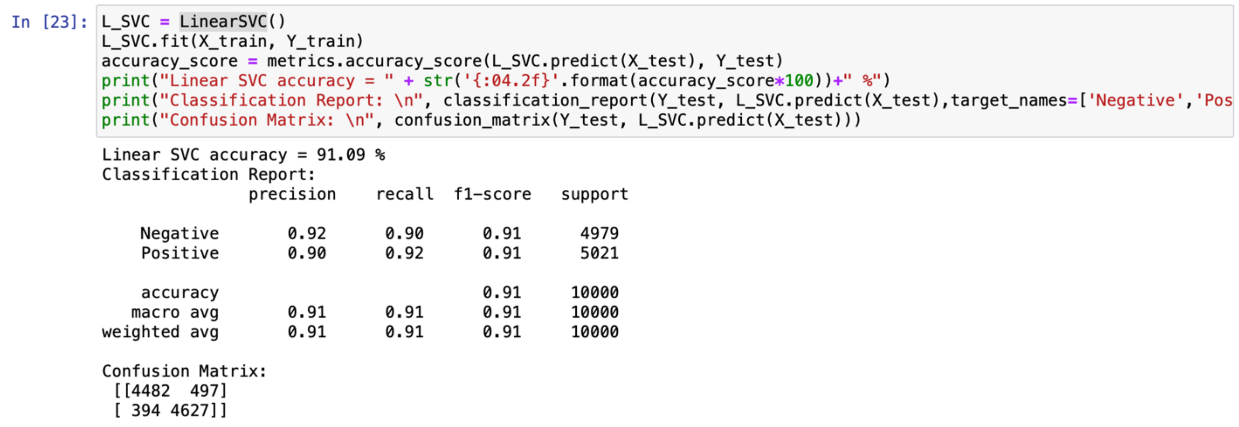
Evaluating the model

```
In [21]: accuracy_score = metrics.accuracy_score(M_NB.predict(X_test), Y_test)
         print(str('{:04.2f}'.format(accuracy_score*100))+" %")

88.87 %
```

```
In [22]: c_Report= classification_report(Y_test, M_NB.predict(X_test),target_names=['Negative','Positive'])
         print("The Classification Report: \n",c_Report )
         cf_matrix=confusion_matrix(Y_test, M_NB.predict(X_test))
         print("The Confusion Matrix: \n", cf_matrix)

The Classification Report:
               precision    recall  f1-score   support

    Negative       0.89      0.89      0.89      4979
    Positive       0.89      0.89      0.89      5021

    accuracy                           0.89     10000
   macro avg       0.89      0.89      0.89     10000
weighted avg       0.89      0.89      0.89     10000

The Confusion Matrix:
 [[4442  537]
 [ 576 4445]]
```

We can see that the accuracy is 88.87% .

LinearSVC Results:



```
In [23]: L_SVC = LinearSVC()
         L_SVC.fit(X_train, Y_train)
         accuracy_score = metrics.accuracy_score(L_SVC.predict(X_test), Y_test)
         print("Linear SVC accuracy = " + str('{:04.2f}'.format(accuracy_score*100))+" %")
         print("Classification Report: \n", classification_report(Y_test, L_SVC.predict(X_test),target_names=['Negative','Pos
         print("Confusion Matrix: \n", confusion_matrix(Y_test, L_SVC.predict(X_test)))
```

```
Linear SVC accuracy = 91.09 %
Classification Report:
               precision    recall  f1-score   support

    Negative       0.92      0.90      0.91      4979
    Positive       0.90      0.92      0.91      5021

    accuracy                           0.91     10000
   macro avg       0.91      0.91      0.91     10000
weighted avg       0.91      0.91      0.91     10000

Confusion Matrix:
 [[4482  497]
 [ 394 4627]]
```

We can notice that the accuracy of LinearSVC is 91.09%.

So, we are gonna create a pickle file for the LinearSVC model.

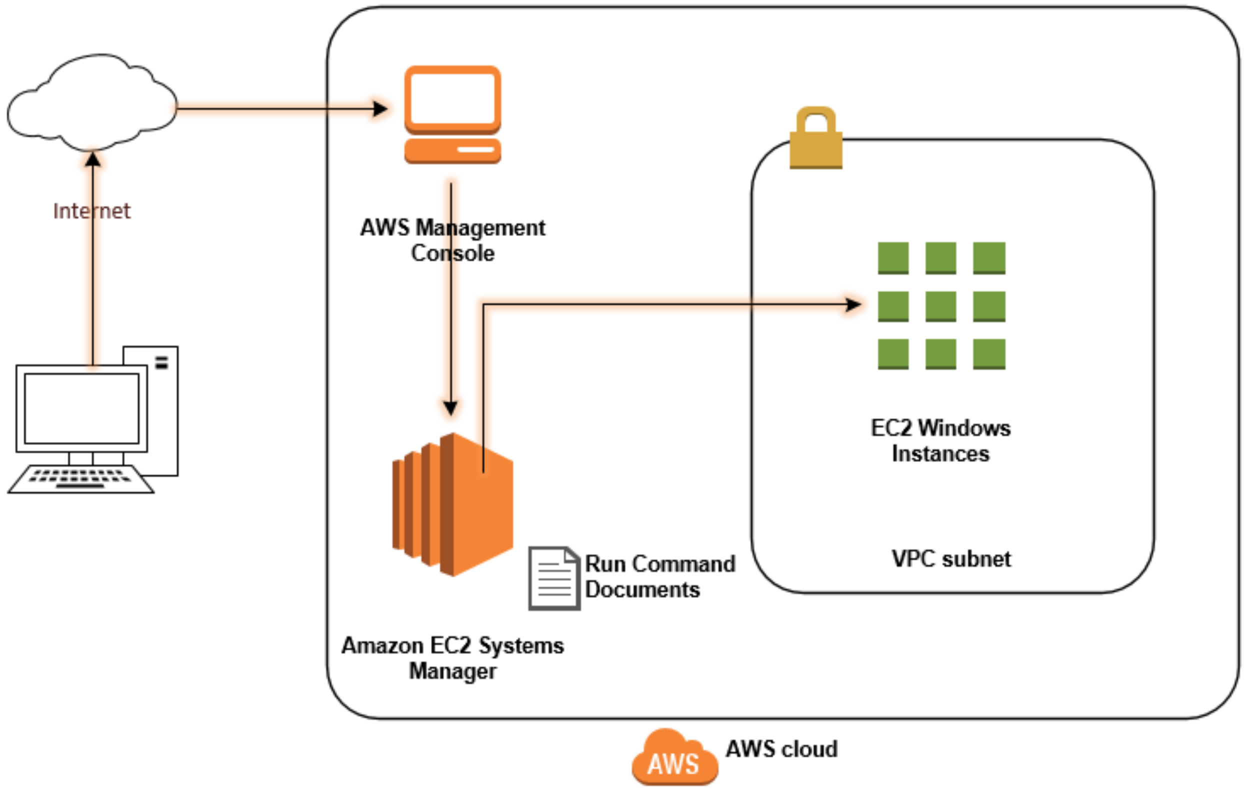
Implementation

Technologies Involved

1. **Cloud Tools:** AWS EC2 instance
2. **Jupyter Notebooks:** Google Colab, Kaggle Notebooks
3. **Mobile/client-side integration:** Web-App using Flask

Deployment :

We have created an web application to access the model and to evaluate the input sentence. We are using AWS EC2 instance as to deploy our model, as it is free. It does takes long time for the deployment, it would be done in couple of minutes and our model will be ready to use. To deploy and access the instance we need to have a private key.



Conclusion

Repository/Archives

GitHub Repo: https://github.com/rnemani96/CSCE_5214_SDAI.git

Exploratory and Extensible

1. The project is on analyzing the sentiment of the user. So, we can extend it into understanding the feeling of user about a product or a Facebook comments, posts, tweets.

References

1. AWS Documentation: AWS Documentation
2. Amazon EC2 Getting Started: https://aws.amazon.com/ec2/getting-started/
3. NLTK documentation: https://www.nltk.org/book/ch01.html
4. Deploying instance: https://docs.aws.amazon.com/codedeploy/latest/userguide/tutorials-windows.html
5. Understanding sentiment analysis concept: https://marutitech.com/introduction-to-sentiment-analysis/