Amazon Reviews Sentiment Analysis

Prepared By:

Yazan AL-Hroob

The National ICT Up skilling program

Data Science Class

Al-Hussain Technical University

Prepared for:

Dr. Ibrahim Abu Al-Haol

Dr. Mohammad Abu Al-Houl

Dr. Rawan Al kurd

Dr. Omar Mesmar

Mr. Ghassan Gammoh

JAN 28, 2021

Abstract

Natural language processing is one of the most researched and worked sciences today. One of its most important parts is sentiment analysis, which studies the positive or negative extent of a text consisting of several words or sentences.

Due to the tremendous technological development that we are witnessing these days, there are huge amounts of textual data that we need to study and analyze. Like people's posts on social media, their comments, emails that customers send to companies, product reviews on online shopping sites, and much more.

The aim of this project is to study customer reviews of products on online shopping site Amazon and rank them for positive or negative reviews based on the text and context it contains using machine learning techniques. Building a model with a high accuracy rate enables us to rely on it to classify these reviews.

Table of Contents

Table of Figures

Introduction

Analyzing the vast amount of text data manually is impossible. Therefore, we must use artificial intelligence techniques and machine learning to help us analyze such a volume of data. A site like Amazon has millions of products offered and millions of reviews come on it. They need to employ these technologies to analyze the data they have to help them understand how satisfied customers are with the offered products.

This project is a study of customer reviews on Amazon products to analyze and classify them, whether they are positive or negative. A large number of reviews have been used regarding several product categories and the use of natural language processing, machine learning, and deep learning techniques to study them. And develop a classifier that can be used in determining whether this review is positive or negative with high accuracy.

Statement of the Problem

The problem here that Amazon have a huge number of reviews on products sold on their website and they want to know how the customers feel about these products. Are they satisfied or not. So, here they want to use this data that they have to study this case and based on the results they can improve their sales or can detect if this product satisfying the customers or not.

To solve this problem I want to use this data set to build a classifier that can classify the review to Positive or Negative review.

Significance of the project

The biggest benefit of this project is to help the online shops like Amazon to easily study the satisfaction of their customers about the products they sell and try to improve the customer shopping experience based on this.

Background

Natural Language Processing, which is usually abbreviated as NLP, is the ability of a computer program to understand human language as it is spoken and process it. NLP is one of the most useful and powerful branches in Artificial Intelligence (AI). So, it deals with the interactions happened between the human and computer using human natural language [1] [2].

Tokenization is the process of dividing a series of strings into parts like words, keywords, phrases, symbols, and other elements called Tokens [3]. Stemming reduces words from their roots (root) by removing suffixes. Suffixes are word endings that modify the meaning of a word. These include, but are not limited to : est, ed, ing, able, ful, etc. [4].

Sentiment analysis is a well-researched problem in linguistics and machine learning, with different language classifiers and models used in previous work. It is common to express this as a classification problem where a given text needs a rating of P ositive, Negative, or Neutral [5].

Vectorization is essential for textual data processing in natural language processing applications. Vectorization enables machines to understand textual contents by converting them into meaningful digital numerical representations [6].

Methodology

1. Data Set

The dataset used is a 4 Million reviews collected from amazon website and uploaded as a public dataset on Kaggle, and divided into two files:

- Training File: 3.6 Million reviews
- Testing File: 400 Thousands reviews

These two files have the same format **__Label__ Title: Review** in each row, Sample:
"__label__2 Great CD: My lovely Pat has one of the GREAT voices of her generation. I have listened to this CD for YEARS and I still LOVE IT. When I'm in a good mood it makes me feel better. A bad mood just evaporates like sugar in the rain. This CD just oozes LIFE. Vocals are jusat STUUNNING and lyrics just kill. One of life's hidden gems. This is a desert isle CD in my book. Why she never made it big is just beyond me. Everytime I play this, no matter black, white, young, old, male, female EVERYBODY says one thing "Who was that singing ?""
The dataset labeled based on negativity or positivity of the review (There is no neutrality in the dataset).
It built using reviews with 1 or 2 stars and 4 or 5 stars:

- __Label__1 : 1 or 2 stars ==> Negative review
- __Label__2 : 4 or 5 stars ==> Positive review

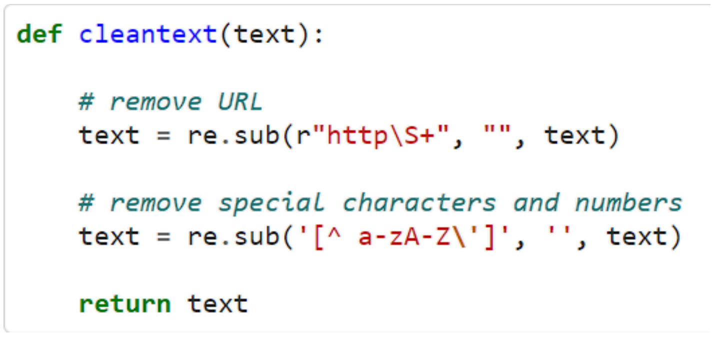
Also, this dataset is a product category independent. (Books, Electronics, Food, DVDs, etc.). In this project I have used a 1 Million reviews for train the model and 200 Thousands reviews for the testing process. These reviews selected randomly from the whole data to give some randomness in the process of choosing the reviews, not take the first 1 M reviews for training or first 200K for the test.

2. Data Cleaning and NLP

This part of report is about the processes of text cleaning and NLP processes, Stop words removal and stemming:

2.1 Data Cleaning

For data cleaning process, I have used a method that cleans the text from any url and any numbers or special characters, to keep only the alphabets words using Regular Expressions. Then, transforming the labels __Label__ to 0 and 1 labels.



```
def cleantext(text):

    # remove URL
    text = re.sub(r"http\S+", "", text)

    # remove special characters and numbers
    text = re.sub('[^ a-zA-Z\']', '', text)

    return text
```

Figure 1 Text Cleaning Function

2.2 Stop Words removing and stemming.

Stop words are the words that filtered out in the processes of NLP, they are the most commonly used words in the language. Examples of stop words are: "a," "and," "but," "how," "or," and "what."

As mentioned in the background section, the stemming process is used to get the root of word. So, a lot of reviews words refer to the same root, then we provide more generality to the words. Example: the stem of the words “eating”, “eats”, “eaten” is eat.

3. Analysis and plotting

First step that I have used to analyze this data is to draw the word cloud that can give a full view of what are the most frequently used words.

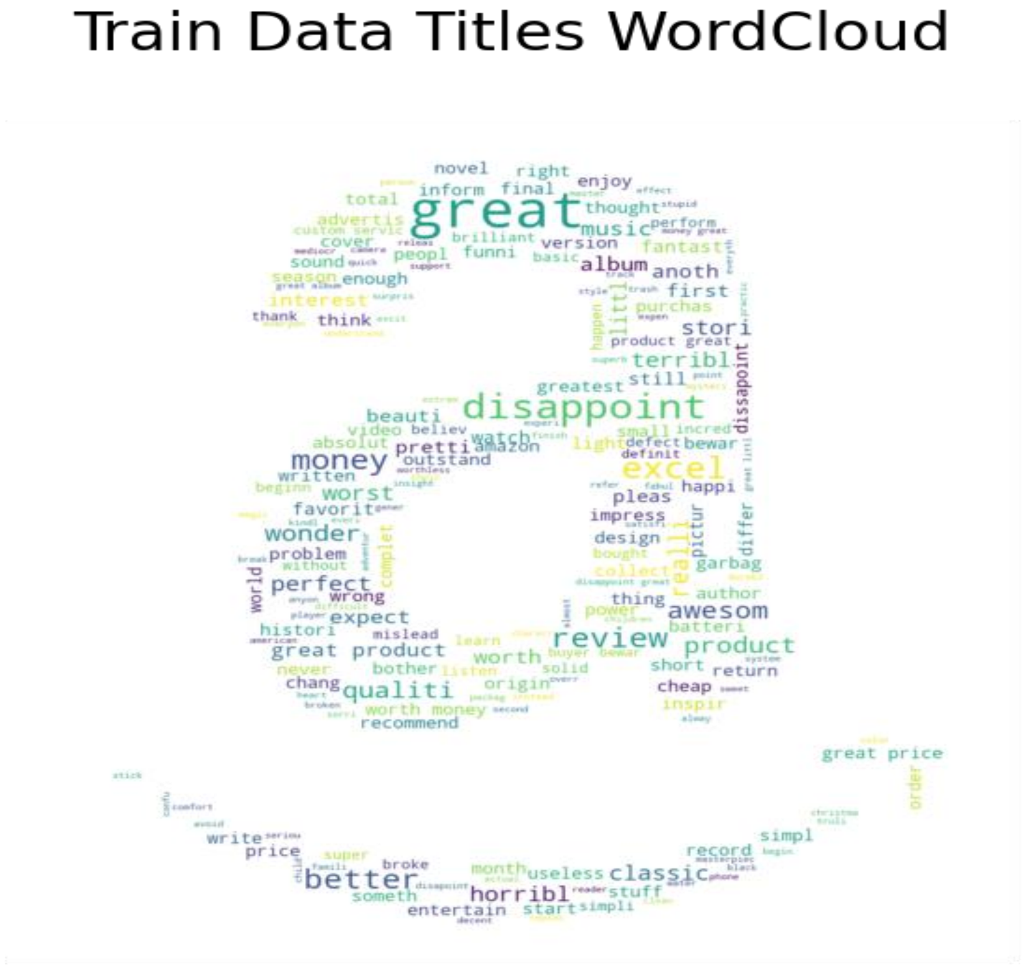


Figure 2 Training reviews titles word cloud

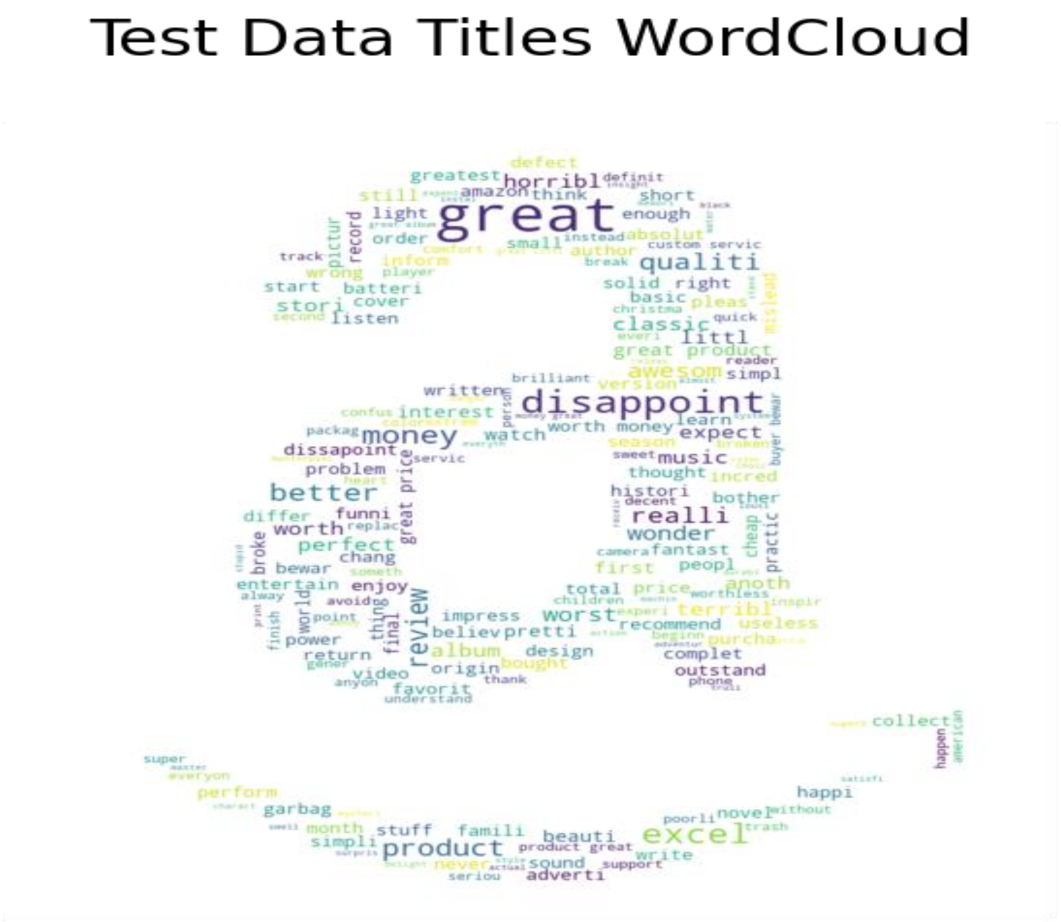


Figure 3 Testing reviews titles word cloud

Second analysis step was the checking of the balance of the data. Is the reviews count for each label are almost the same or not.

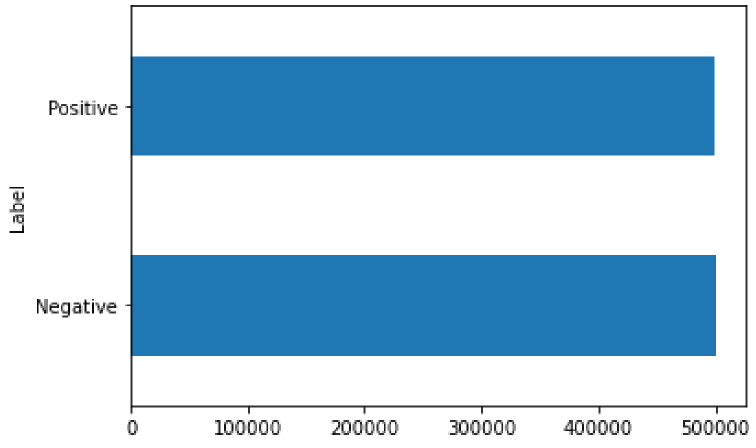


Figure 4 Training Data reviews count per label

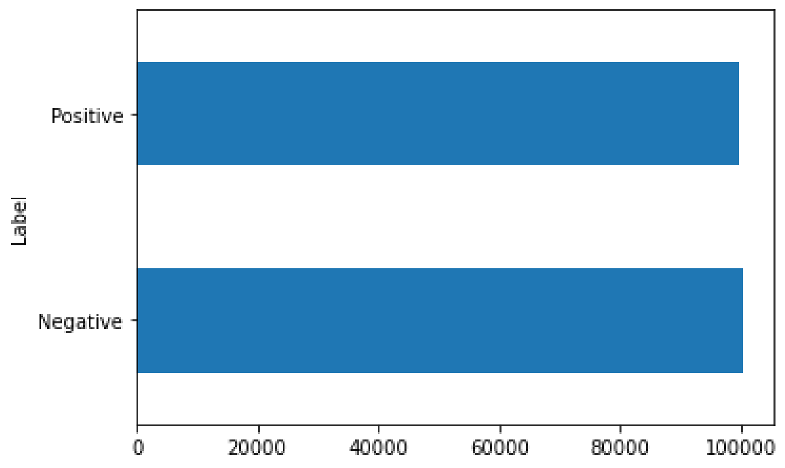


Figure 5 Testing Data reviews count per label

Then I plotted the distribution of review length by words, and calculated the average number of characters and words for each label in the training and the testing datasets. After that, I have plotted count plots to show the most ten frequented words in negative, positive training reviews titles and in negative, positive testing reviews titles.

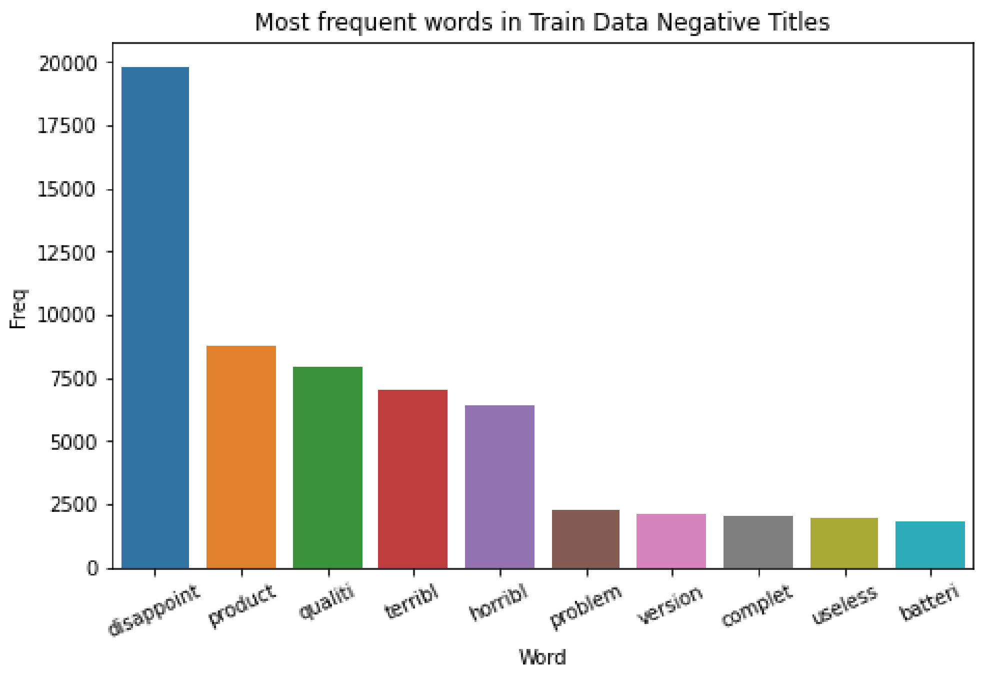
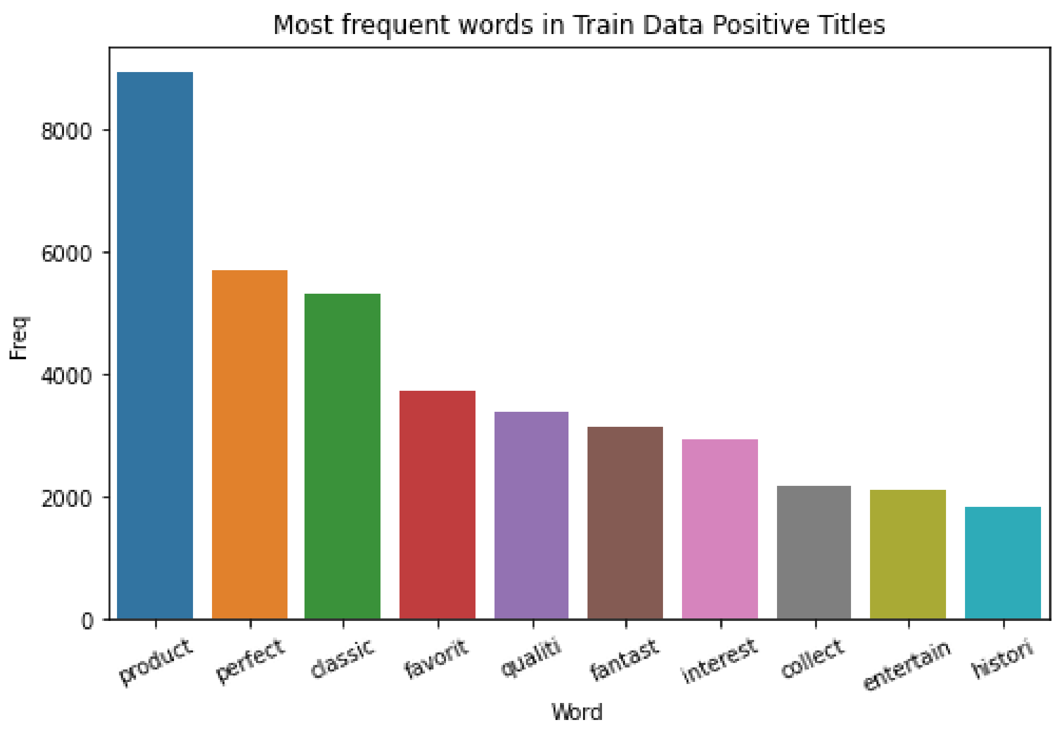
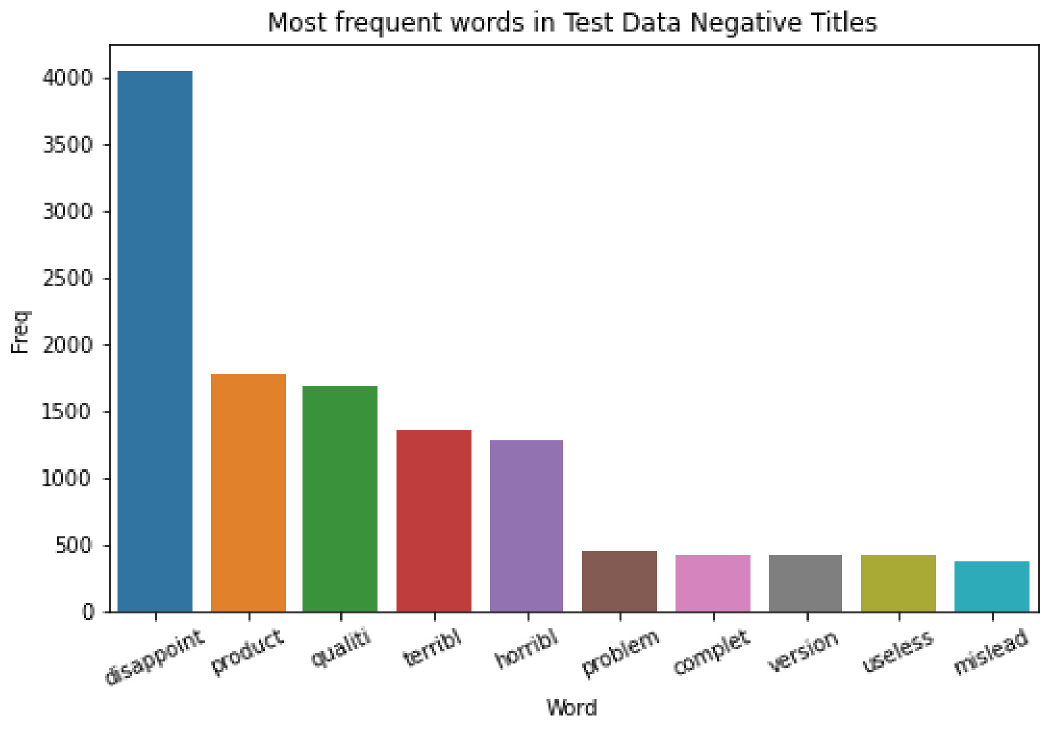
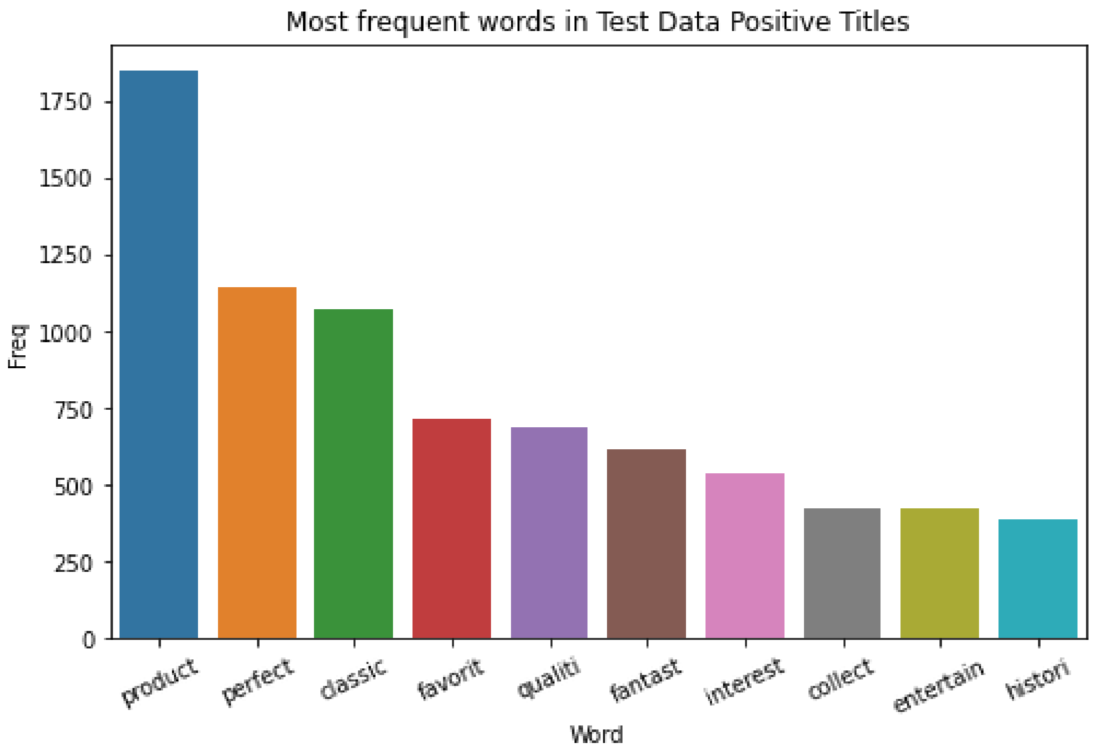


Figure 6 Most frequent words in Negative training reviews


Most frequent words in Train Data Positive Titles
Freq
8000
6000
4000
2000
0
product
perfect
classic
favorit
qualiti
fantast
interest
collect
entertain
histori
Word


Most frequent words in Test Data Negative Titles
Freq
4000
3500
3000
2500
2000
1500
1000
500
0
disappoint
product
qualiti
terribl
horribl
problem
complet
version
useless
mislead
Word

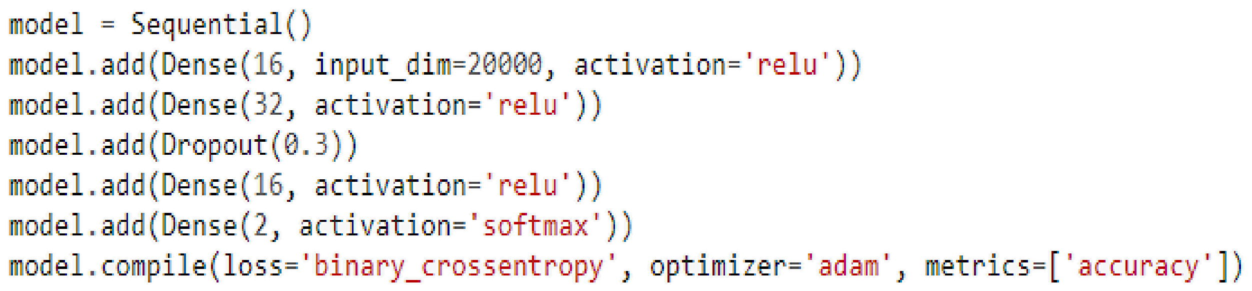


4. Fitting the Vectorizers and Building the Classifiers

I have used two vectorizers, Hashing Vectorizer with 3 cases of n-grams {(no n-gram), (1,2), (1,3)} and TF-IDF Vectorizer n-grams (1,2) with three supervised machine learning models for each one. Also Deep Feed Forward Neural Network Classifier with ach vectorizer. Finally, Embedding Neural Network with pad sequencer implementation on the whole dataset 3,600,000 reviews for training and 400,000 reviews for testing. ML models used are:

- Stochastic Gradient Descent Classifier with alpha parameter equals to 0.000001.
- Logistic Regression Classifier with parameters C=1000, solver ='saga'.
  “Saga” Solver used for large datasets with high dimensions (Large numbers of features (Columns)).
- Linear Support Vector Classifier with parameter C =1.

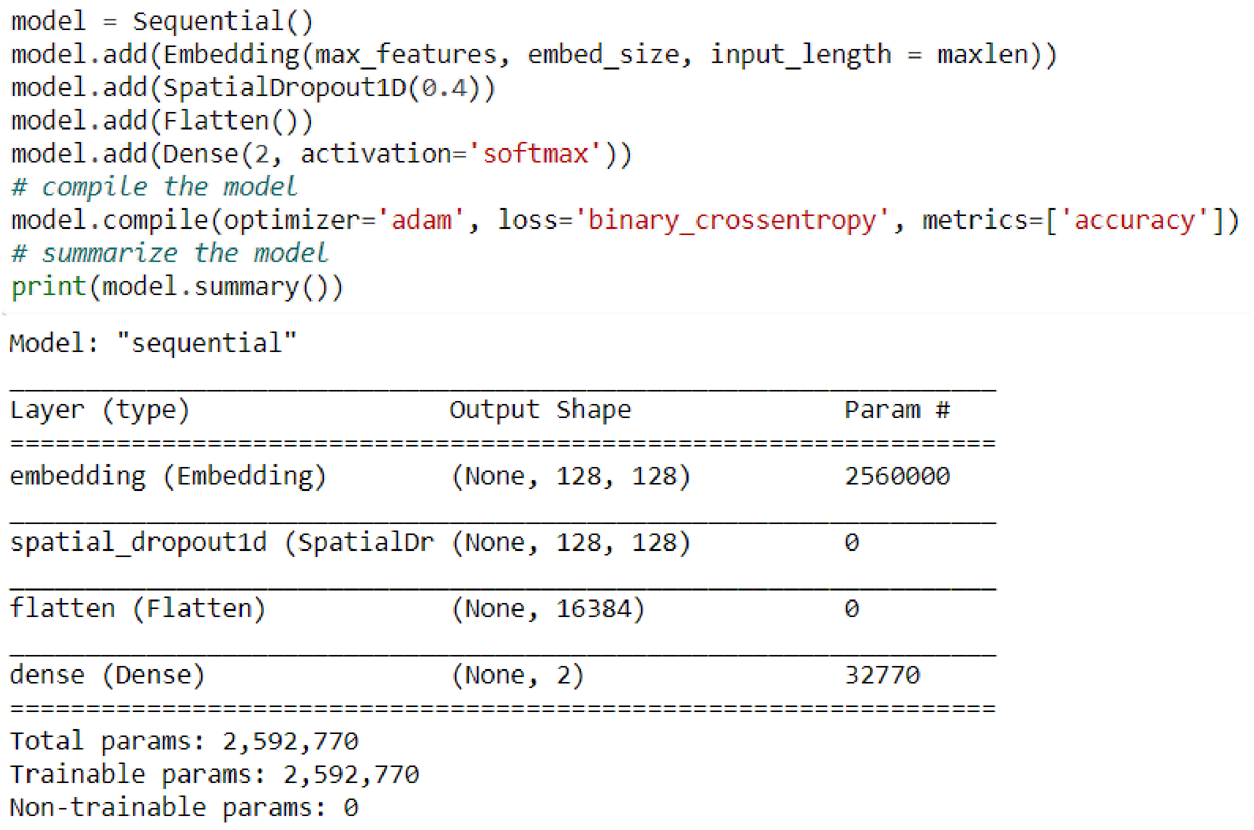
DNN structure:



```
model = Sequential()
model.add(Dense(16, input_dim=20000, activation='relu'))
model.add(Dense(32, activation='relu'))
model.add(Dropout(0.3))
model.add(Dense(16, activation='relu'))
model.add(Dense(2, activation='softmax'))
model.compile(loss='binary_crossentropy', optimizer='adam', metrics=['accuracy'])
```

Pad Sequencer details and Embedding Layer NN structure:

- Number of words for the sequencer: "max_features = 20000"
- Sequencer pad Max length: "maxlen = 128"
- NN structure with Embedding size = 128:



```
model = Sequential()
model.add(Embedding(max_features, embed_size, input_length = maxlen))
model.add(SpatialDropout1D(0.4))
model.add(Flatten())
model.add(Dense(2, activation='softmax'))
# compile the model
model.compile(optimizer='adam', loss='binary_crossentropy', metrics=['accuracy'])
# summarize the model
print(model.summary())
```

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
embedding (Embedding)        (None, 128, 128)          2560000
_________________________________________________________________
spatial_dropout1d (SpatialDr (None, 128, 128)          0
_________________________________________________________________
flatten (Flatten)            (None, 16384)             0
_________________________________________________________________
dense (Dense)                (None, 2)                 32770
=================================================================
Total params: 2,592,770
Trainable params: 2,592,770
Non-trainable params: 0
```

Classification results discussed in the Results section of this report.

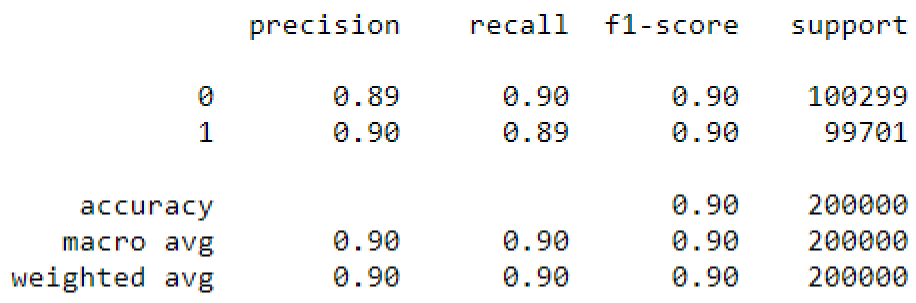
Results and discussion

In this section I am showing the Classification results of the all combinations of the vectorizers and ML, DL models. Classification report method results:

1. Hashing Vectorizer No n-gram

Used with parameter “alternate_sign=False” to generate only positive numerical values, because I used the “relu” Activation Function in the DNN models.

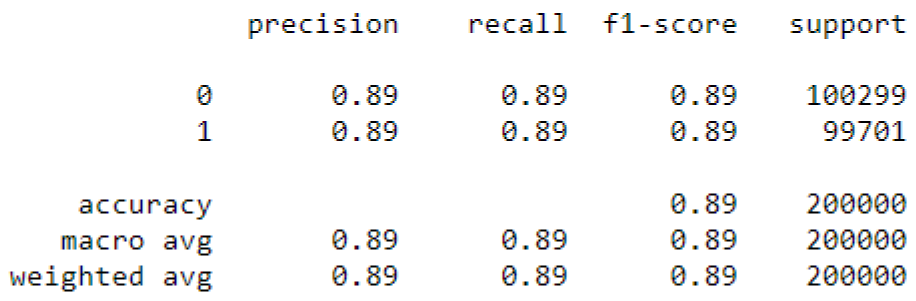
1.1 Stochastic Gradient Descent Classifier



```
              precision    recall  f1-score   support

           0       0.89      0.90      0.90    100299
           1       0.90      0.89      0.90     99701

    accuracy                           0.90    200000
   macro avg       0.90      0.90      0.90    200000
weighted avg       0.90      0.90      0.90    200000
```

Accuracy: 0.8961

1.2 Logistic Regression Classifier



```
              precision    recall  f1-score   support

           0       0.89      0.89      0.89    100299
           1       0.89      0.89      0.89     99701

    accuracy                           0.89    200000
   macro avg       0.89      0.89      0.89    200000
weighted avg       0.89      0.89      0.89    200000
```

Accuracy: 0.8902

1.3 Linear Support Vector Classifier

```
              precision    recall  f1-score   support

           0       0.90      0.90      0.90    100299
           1       0.90      0.90      0.90     99701

    accuracy                           0.90    200000
   macro avg       0.90      0.90      0.90    200000
weighted avg       0.90      0.90      0.90    200000
```

Accuracy: 0.8976

1.4 DNN

Training Process: Epochs = 2, Batch size = 1000, validation split = 0.1

```
Epoch 1/2
900/900 [==============================] - 81s 88ms/step - loss: 0.3997 - accuracy: 0.8490 - val_loss: 0.2516 - val_accuracy: 0.89
68
Epoch 2/2
900/900 [==============================] - 71s 77ms/step - loss: 0.2271 - accuracy: 0.9107 - val_loss: 0.2492 - val_accuracy: 0.89
82
```

Evaluation Results:

Loss: 0.2479, Accuracy: 0.8981

2. Hashing Vectorizer with n-grams (1,2)

Used with parameter “alternate_sign=False” to generate only positive numerical values.

2.1 Stochastic Gradient Descent Classifier

```
              precision    recall  f1-score   support

           0       0.92      0.92      0.92    100299
           1       0.92      0.92      0.92     99701

    accuracy                           0.92    200000
   macro avg       0.92      0.92      0.92    200000
weighted avg       0.92      0.92      0.92    200000
```

Accuracy: 0.9227

2.2 Logistic Regression Classifier

```
              precision    recall  f1-score   support

           0       0.91      0.91      0.91    100299
           1       0.91      0.91      0.91     99701

    accuracy                           0.91    200000
   macro avg       0.91      0.91      0.91    200000
weighted avg       0.91      0.91      0.91    200000
```

Accuracy: 0.9139

2.3 Linear Support Vector Classifier

```
              precision    recall  f1-score   support

           0       0.92      0.92      0.92    100299
           1       0.92      0.92      0.92     99701

    accuracy                           0.92    200000
   macro avg       0.92      0.92      0.92    200000
weighted avg       0.92      0.92      0.92    200000
```

Accuracy: 0.9227

2.4 DNN

Training Process: Epochs = 2, Batch size = 1000, validation split = 0.1

```
Epoch 1/2
900/900 [==============================] - 91s 98ms/step - loss: 0.3829 - accuracy: 0.8460 - val_loss: 0.2012 -
val_accuracy: 0.9198
Epoch 2/2
900/900 [==============================] - 77s 84ms/step - loss: 0.0898 - accuracy: 0.9726 - val_loss: 0.2562 -
val_accuracy: 0.9074
```

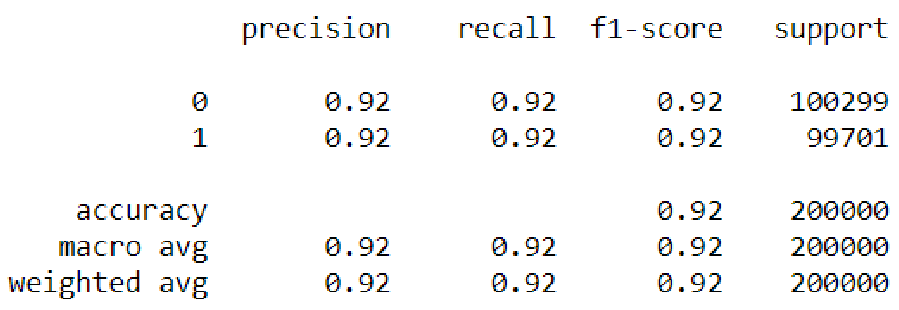
Evaluation Results:

Loss: 0.2557, Accuracy: 0.9089

3. Hashing Vectorizer with n-grams (1,3)

Used with parameter “alternate_sign=False” to generate only positive numerical values.

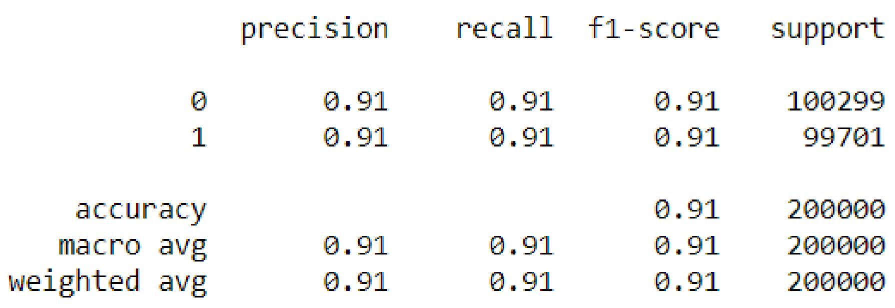
3.1 Stochastic Gradient Descent Classifier



```
              precision    recall  f1-score   support

           0       0.92      0.92      0.92    100299
           1       0.92      0.92      0.92     99701

    accuracy                           0.92    200000
   macro avg       0.92      0.92      0.92    200000
weighted avg       0.92      0.92      0.92    200000
```

Accuracy: 0.9201

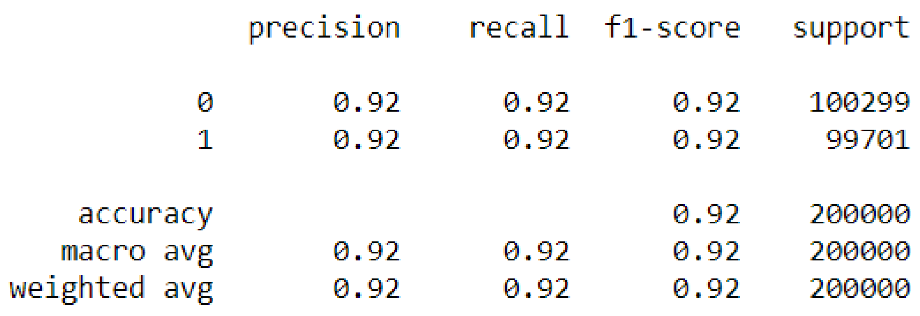
3.2 Logistic Regression Classifier



```
              precision    recall  f1-score   support

           0       0.91      0.91      0.91    100299
           1       0.91      0.91      0.91     99701

    accuracy                           0.91    200000
   macro avg       0.91      0.91      0.91    200000
weighted avg       0.91      0.91      0.91    200000
```

Accuracy: 0.9125

3.3 Linear Support Vector Classifier

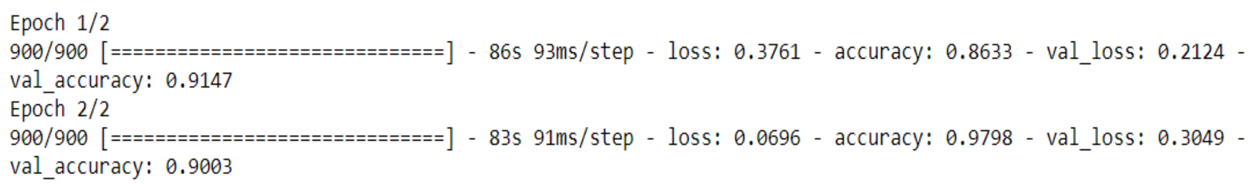


```
              precision    recall  f1-score   support

           0       0.92      0.92      0.92    100299
           1       0.92      0.92      0.92     99701

    accuracy                           0.92    200000
   macro avg       0.92      0.92      0.92    200000
weighted avg       0.92      0.92      0.92    200000
```

Accuracy: 0.9183

3.4 DNN

Training Process: Epochs = 2, Batch size = 1000, validation split = 0.1



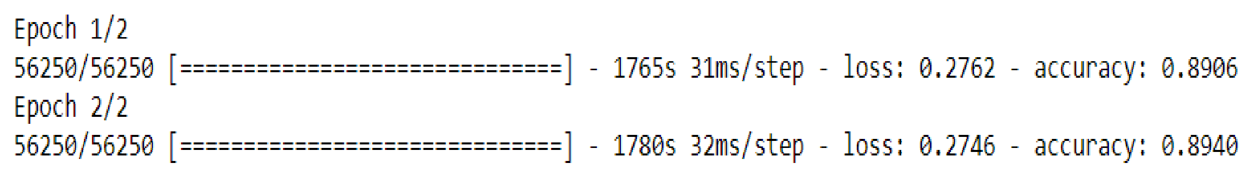
```
Epoch 1/2
900/900 [==============================] - 86s 93ms/step - loss: 0.3761 - accuracy: 0.8633 - val_loss: 0.2124 -
val_accuracy: 0.9147
Epoch 2/2
900/900 [==============================] - 83s 91ms/step - loss: 0.0696 - accuracy: 0.9798 - val_loss: 0.3049 -
val_accuracy: 0.9003
```

Evaluation Results:

Loss: 0.3052, Accuracy: 0.9012

4. Pad Sequencer with Embedding Layer NN

Training Process: Epochs = 2, Batch size = 64



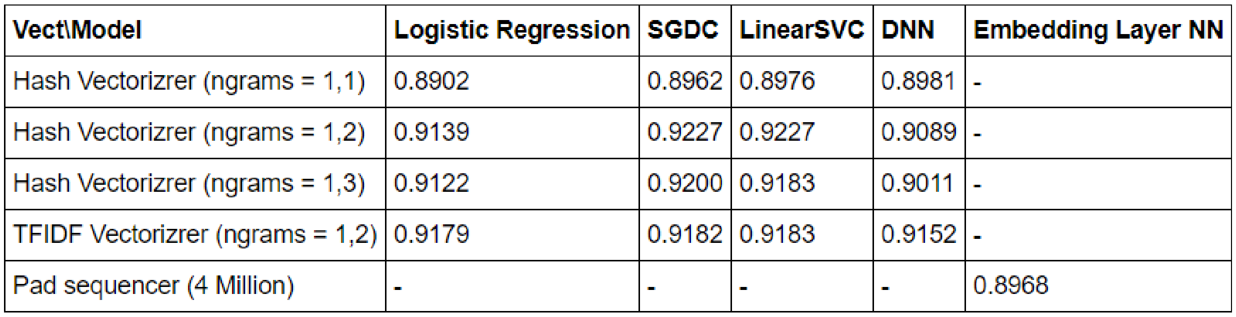
```
Epoch 1/2
56250/56250 [==============================] - 1765s 31ms/step - loss: 0.2762 - accuracy: 0.8906
Epoch 2/2
56250/56250 [==============================] - 1780s 32ms/step - loss: 0.2746 - accuracy: 0.8940
```

Evaluation Results:

Loss: 0.2674, Accuracy: 0.8968

Conclusions

As shown below in the table, model accuracy and classification repot results shows that the best methodology for this problem to gain the highest accuracy is to use Hashing Vectorizer with n-grams (1,2) and SGDC or LSVC Models.



Vect\Model	Logistic Regression	SGDC	LinearSVC	DNN	Embedding Layer NN
Hash Vectorizrer (ngrams = 1,1)	0.8902	0.8962	0.8976	0.8981	-
Hash Vectorizrer (ngrams = 1,2)	0.9139	0.9227	0.9227	0.9089	-
Hash Vectorizrer (ngrams = 1,3)	0.9122	0.9200	0.9183	0.9011	-
TFIDF Vectorizrer (ngrams = 1,2)	0.9179	0.9182	0.9183	0.9152	-
Pad sequencer (4 Million)	-	-	-	-	0.8968

For more implementation or analysis details you can refer to my Github repo where you can find the three notebooks that contains all of this work details. Use the link in this reference [7].

References

[1] A. Geitgey, "Natural Language Processing is Fun!," 18 July 2018. [Online]. Available: https://medium.com/@ageitgey/natural-language-processing-is-fun-9a0bff37854e. [Accessed January 2021].

[2] M. Rouse, "natural language processing (NLP)," May 2019. [Online]. Available: https://searchbusinessanalytics.techtarget.com/definition/natural-language-processing-NLP. [Accessed January 2021].

[3] R. Collobert, J. Weston, L. Bottou, M. Karlen, K. Kavukcuoglu and P. Kuksa,, "Natural Language Processing (Almost) from Scratch," *Journal of Machine Learning Research,* pp. 2493-2537, 2011.

[4] L. Chris, "A Beginners Guide To Stemming In Natural Language Processing," Medium.com, 26 Septemper 2020. [Online]. Available: https://towardsdatascience.com/a-beginners-guide-to-stemming-in-natural-language-processing-34ddee4acd37. [Accessed January 2021].

[5] S. Asur, B. A. Huberman, "Predicting the Future With Social Media," *2010 IEEE/WIC/ACM International Conference on Web Intelligence and Intelligent Agent Technology,* pp. 492-499, 2010.

[6] Kumari, Anita & Shashi, M., "Vectorization of Text Documents for Identifying Unifiable News Articles," *International Journal of Advanced Computer Science and Applications,* vol. 10, p. 305, 2019.

[7] Y. Al-Hroob, "Amazon Reviews Sentiment Analysis," January 2021. [Online]. Available: https://github.com/yazan9900/ds-projects/tree/main/Capstone%20Project. [Accessed January 2021].