Capstone Project-2 Milestone Report -2

Topic: Mining Amazon Product Reviews using NLP

1. Introduction:

Online shopping is a form of electronic commerce which allows consumers to directly buy goods or services from a seller over the Internet using a web browser. Consumers find a product of interest by visiting the website of the retailer directly or by searching among alternative vendors using a shopping search engine, which displays the same product's availability and pricing at different e-retailers. But online shopping comes with its own limitations. One of the biggest challenges is verifying the authenticity of a product. Is it as good as advertised on the e-commerce site? Will the product last more than a year? Are the reviews given by other customers true or are they false advertising? These are important questions customers need to ask before spending their money.

2. Problem Statement:

Ratings alone do not give a complete picture of the products we wish to purchase. So secondary option is looking at the reviews. Review's plays an important role in the decision-making process. If the number of reviews is less, it is easy to read and understand but what if there are thousands of reviews. So, the problem is How we can analyze great number of online reviews using Natural Language Processing (NLP).

Client: Amazon customers

This project will serve three purposes

- Helps consumers to understand the sentiment of the review.

- Help the consumers to get consumer feedback in the form of topics covered by the reviews without having to go through all of them.
- Enable consumers to quickly extract the summary of the reviews without reading the entirely.

3. Data Description:

Dataset: https://www.kaggle.com/snap/amazon-fine-food-reviews

Context: This dataset consists of reviews of fine foods from amazon. The data span a period of more than 10 years, including all ~500,000 reviews up to October 2012. Reviews include product and user information, ratings, and a plain text review. It also includes reviews from all other Amazon categories.

```
RangeIndex: 568454 entries, 0 to 568453
Data columns (total 10 columns):
#   Column                  Non-Null Count   Dtype
---  ------                  --------------   -----
0   Id                      568454 non-null int64
1   ProductId               568454 non-null object
2   UserId                  568454 non-null object
3   ProfileName             568438 non-null object
4   HelpfulnessNumerator    568454 non-null int64
5   HelpfulnessDenominator   568454 non-null int64
6   Score                   568454 non-null int64
7   Time                    568454 non-null int64
8   Summary                 568427 non-null object
9   Text                    568454 non-null object
```

For the analysis I have used only three features (Score, Summary, Text)

4. Text Wrangling and Pre-processing:

There are usually multiple steps involved in cleaning and pre-processing textual data. I have done some of the most important steps which are used heavily in Natural Language Processing (NLP) pipelines.

Step1: Removing HTML Tags

Unstructured text contains a lot of noise, HTML tags are typically one of these components which do not add much value towards understanding and analyzing text. So, in the first step removed unnecessary HTML tags and retain the useful textual information from the review.

Step:2 Expanding Contractions

Contractions are the shortened versions of words like don't for do not and how'll for how will. These are used to reduce the speaking and writing time of words. We need to expand these contractions for a better analysis of the reviews. I have created a dictionary of common English contractions that I will use for mapping the contractions to their expanded forms. Using a function called *'remove_contractions'* I have removed all the contractions in the review.

Step:3 Removing Special Characters

Special characters and symbols are usually non-alphanumeric characters or even occasionally numeric characters (depending on the problem), which add to the extra noise in unstructured text. Usually, simple regular expressions (regexes) can be used to remove them.
I have kept removing digits as optional, because often we might need to keep them in the pre-processed text.

Step4: Lemmatization

Lemmatisation is the algorithmic process of determining the lemma of a word based on its intended meaning. Remove word affixes to get to the base form of a word. However, the base form in this case is known as the root word, also known as the ***lemma***, will always be present in the dictionary. WordNetLemmatizer() is used for lemmatization process.

Step5: Removing Stop words

Words which have little or no significance, especially when constructing meaningful features from text, are known as stopwords or stop words. These are usually words that end up having the maximum frequency if you do a simple term or word frequency in a corpus. Typically, these can be articles, conjunctions, prepositions and so on. Some examples of stopwords are *a*, *an*, *the*, *and* the like. used a standard English language stopwords list from **nltk.**

5. Exploratory Data Analysis (EDA)

We have already cleaned our data and have our corpus ready, yes – it is finally time for Exploratory Data Analysis! It is a crucial part of any data science project because that is where you get to know more about the data. In this phase, we can reveal hidden patterns in the data and generate insights from it.

Analysis of 'Score' has been done using "sns.countplot()" method, to understand the count of each score.

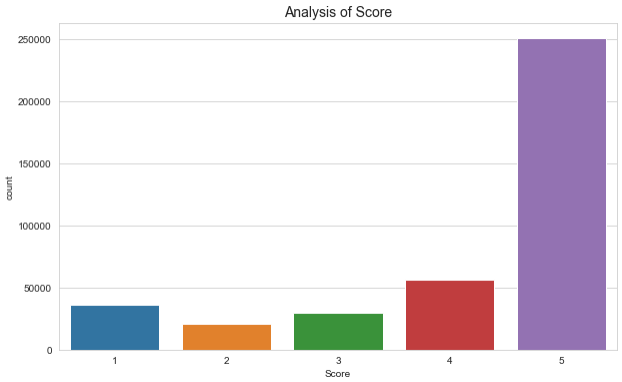


Fig:1

From the graph we can conclude that most of the reviews have score 5 and very a smaller number of reviews got score 2.

Word count distribution plot.

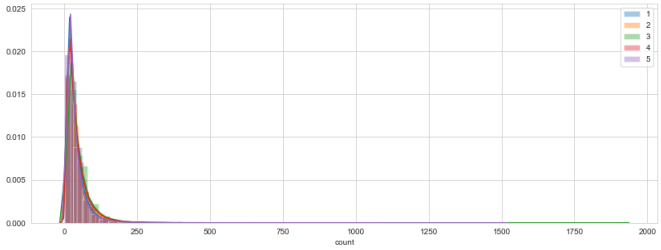


Fig:2

The word count distribution plot tells us that the length of all most all reviews are within the range of 0-250, but there are few outliers.

6. Sentiment Analysis

Sentiment analysis is basically the process of determining the attitude or the emotion of the writer, i.e., whether the review is positive or negative or neutral. Sentiment Analysis is done using TextBlob (*TextBlob* is a Python (2 and 3) library for processing textual data. It provides a simple API for diving into common natural language processing (NLP) tasks such as part-of-speech tagging, noun phrase extraction, sentiment analysis, classification, translation, and more.)

The sentiment function of textblob returns two properties, polarity, and subjectivity. Polarity is float which lies in the range of [-1.0,1.0] where 1 means positive statement and -1 means a negative statement. Subjective sentences generally refer to opinion, emotion, or judgment whereas objective refers to information. Subjectivity is also a float which lies in the range of [0,1].

I have classified the reviews into three, Positive, Negative and Neutral based on the polarity.

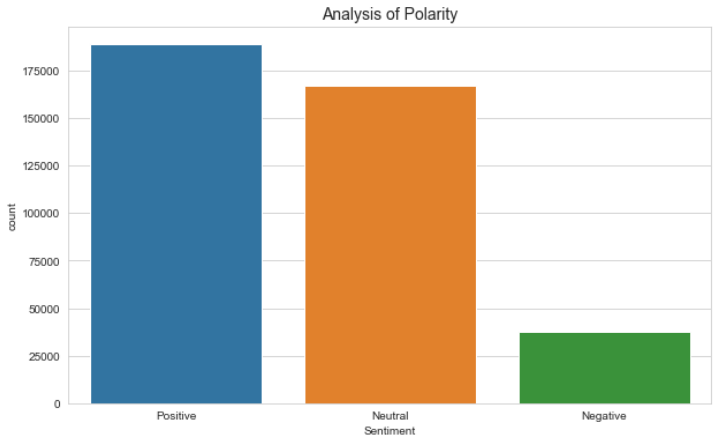


Fig:3

The above figure shows the distribution of polarity among reviews. Most of the reviews are classified as positive and a very less reviews classified as Negative.

6.1 Understanding the common words used in reviews

A wordcloud is a visualization wherein the most frequent words appear in large size and the less frequent words appear in smaller sizes.

Fig:4 wordcloud for Positive and Negative reviews

The above figure shows that the words ‘highly recommend’, ’well’,’ taste great’, ‘delicious’ are the common words used to represent positive review and the words ‘think’, ‘product’, ’use’ are used to represent negative review.

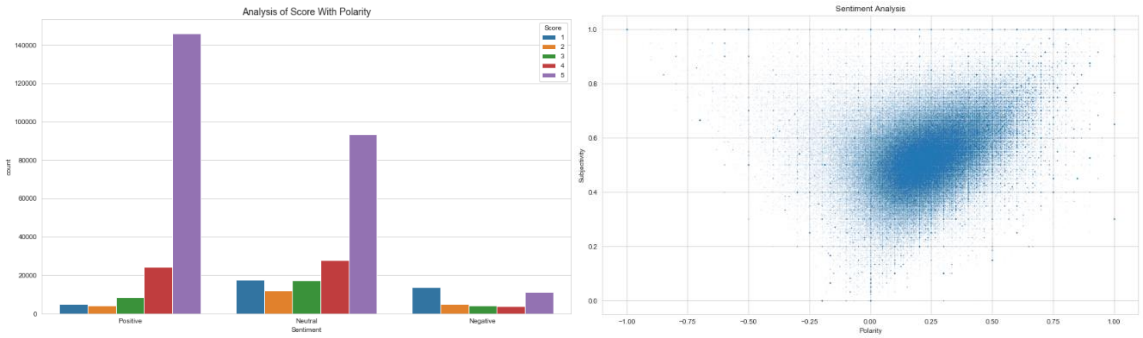


Fig:5

The Sentiment VS Score bar chart shows that, most of the positive reviews got score 5, but there are few positive reviews, which got very less Score. On the other side some of the negative reviews got high rating.
The Polarity VS Subjectivity scatter plot tells us that, most of the people are happy with Amazon food services.

The most common words give insights not only on the frequently used words but also words that could also be potential data specific stop words.

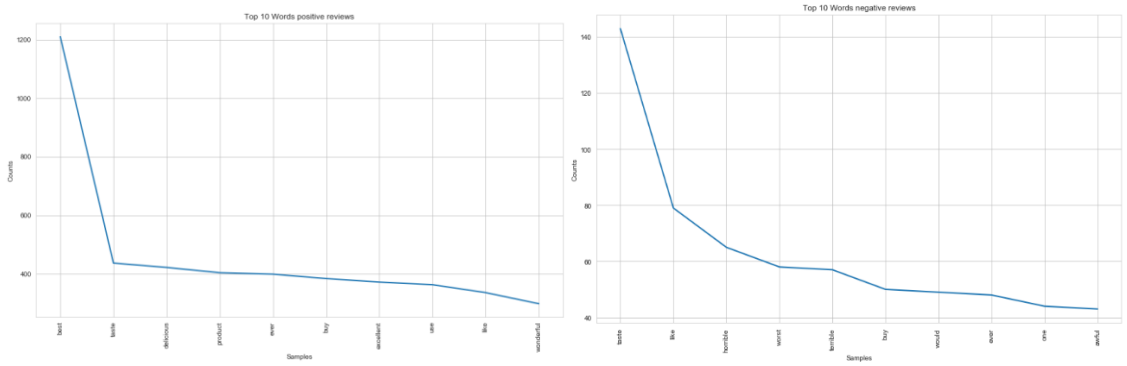


Fig:6

Top 10 Words for positive reviews are [('best', 1211),('taste', 437),('delicious', 422),('product', 404),('ever', 399),('buy', 384),('excellent', 372),('use'363),('like', 336),('wonderful', 298)] and the Top 10 Words for negative reviews are [('taste', 143),('like', 79),('horrible', 65),('worst', 58),('terrible', 57),('buy', 50),('would', 49),('ever', 48),('one', 44),('awful', 43)].

6.2 Sentiment Prediction using Reviews

Model Building: Logistic Regression

To analyze a preprocessed data, it needs to be converted into features. Depending upon the usage, text features can be constructed using assorted techniques – Bag-of-Words, TF-IDF, and Word Embeddings. Here I have used only Bag-of-Words.
Bag of Words Model is a classical approach in expressing text as a set of features is getting the token frequency. Each entry to the dataframe is a document while each column corresponds to every unique token in the entire corpora. The row will identify how many times a word appears in the document. I used logistic regression to build the models. It predicts the probability of occurrence of an event by fitting data to a logit function.

Making Predictions and Evaluating the Model

Once the model has been trained, the last step is to make predictions on the model. Finally, to evaluate the performance of the machine learning models, I used classification metrics such as a confusion Metrix, F1 measure, accuracy, etc. To find the values for these metrics, we can use classification_report, confusion_matrix, and accuracy_score utilities from the sklearn.metrics library.

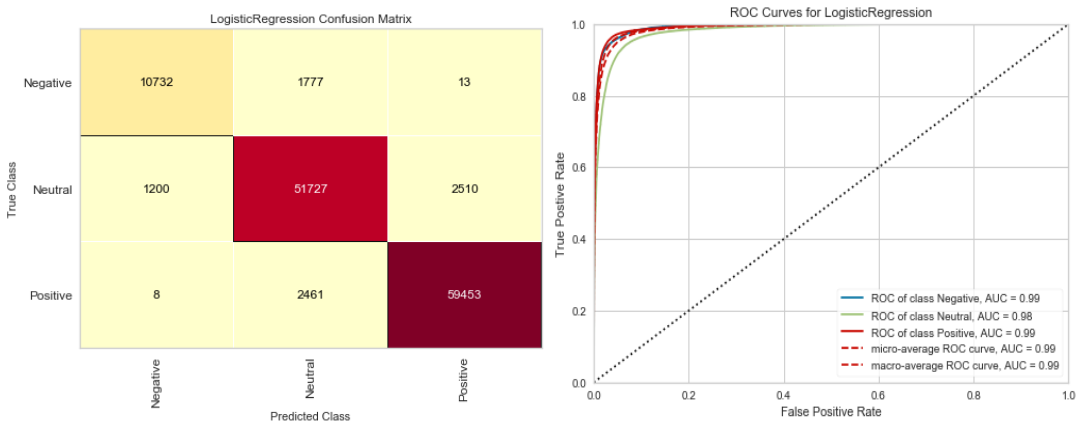


Fig:7

The logistic regression model on the Bag-of-Words features gave an accuracy 93.8% for the test set. Fig.7 shows the Confusion Matrix and AUC.

Model Building Using ANN

Finally, I trained the model using Artificial Neural Networks using keras API and TensorFlow and managed to get a better result of 95.3% accuracy on validation data.

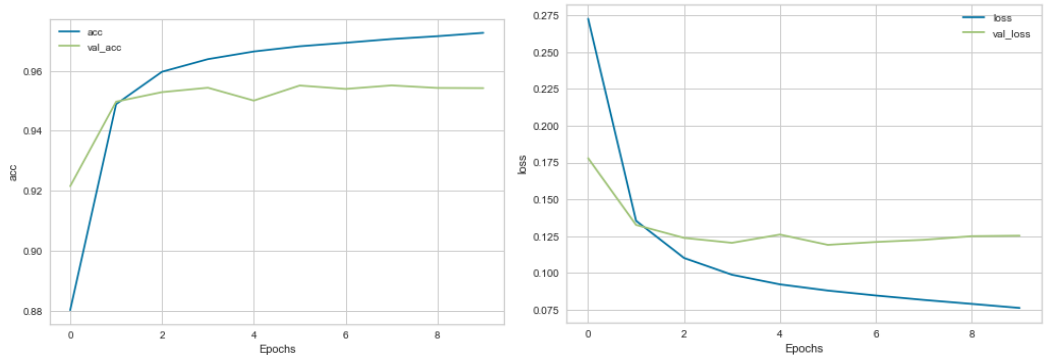


Fig:8

7. Topic Modeling

Topic Modeling is a text mining approach. A Topic Model can be defined as an unsupervised technique to discover topics across various text documents. Topic modeling helps in exploring large amounts of text data, finding clusters of words, similarity between documents, and discovering abstract topics. This part of project will serve two purposes

- Enable consumers to quickly extract the key topics covered by the reviews without having to go through all of them.
- Help the sellers/retailers get consumer feedback in the form of topics (extracted from the consumer reviews).

Topic Modeling is a technique to extract the hidden topics from large volumes of text. There are several algorithms for doing topic modeling. The most popular ones include LSA or LSI – Latent Semantic Analysis or Latent Semantic Indexing – Uses Singular Value Decomposition (SVD) on the Document-Term Matrix- Based on Linear Algebra, NMF – Non-Negative Matrix Factorization – Based on Linear Algebra, LDA – Latent Dirichlet Allocation – The one we'll be focusing here. Its foundations are Probabilistic Graphical Models.

Latent Dirichlet Allocation (LDA) is a popular algorithm for topic modeling with excellent implementations in the Python's Gensim package. The LDA is based upon two general assumptions:

- Documents that have similar words usually have the same topic
- Documents that have groups of words frequently occurring together usually have the same topic.

7.1. Tokenize words and clean-up text

I tokenized each sentence into a list of words, removed punctuations and unnecessary characters altogether. Gensim's simple_preprocess() is a great tool for this. Additionally, I have set deacc=True to remove the punctuations.

7.2. Creating Bigram and Trigram Models

Bigrams are two words frequently occurring together in the document. Trigrams are 3 words frequently occurring. Gensim's Phrases model can build and implement the bigrams, trigrams, quadgrams and more. The two important arguments to Phrases are min_count and threshold. The higher the values of these param, the harder it is for words to be combined to bigrams.

7.3. Feature Engineering

Bag of word Model

The classical approach in expressing text as a set of features is getting the token frequency. Each entry to the dataframe is a document while each column corresponds to every unique token in the entire corpora. The row will identify how many times a word appears in the document.

TF-IDF Model

Tf-idf stands for term frequency-inverse document frequency, and the tf-idf weight is a weight often used in information retrieval and text mining. This weight is a statistical measure used to evaluate how important a word is to a document in a collection or corpus. The importance increases proportionally to the number of times a word appears in the document but is offset by the frequency of the word in the corpus.

Shortcomings of count-based feature engineering model

Traditional count-based feature engineering models are bag of words and TF-IDF. While they are effective methods for extracting features from text, due to the inherent nature of the model being just a bag of

unstructured words, we lose additional information like the semantics, structure, sequence and context around nearby words in each text document.

The need for word embeddings

In case of raw text data, especially count based models like Bag of Words, we are dealing with individual words which may have their own identifiers and do not capture the semantic relationship amongst words. This leads to huge sparse word vectors for textual data and thus if we do not have enough data, we may end up getting poor models or even overfitting the data due to the curse of dimensionality. Predictive methods like Neural Network based language models try to predict words from its neighboring words looking at word sequences in the corpus and in the process, it learns distributed representations giving us dense word embeddings.

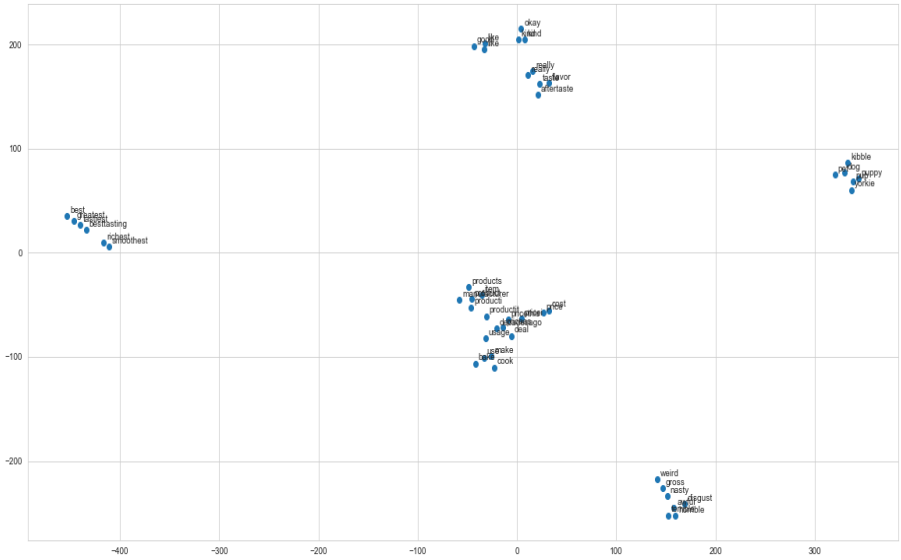
The Word2Vec Model

This is a predictive deep learning-based model to compute and generate high quality, distributed and continuous dense vector representations of words, which capture contextual and semantic similarity. Essentially these are unsupervised models which can take in massive textual corpora, create a vocabulary of possible words, and generate dense word embeddings for each word in the vector space representing that vocabulary.

Exploratory Data Analysis

t-SNE

t-Distributed Stochastic Neighbor Embedding (t-SNE) is an unsupervised, non-linear technique primarily used for data exploration and visualizing high-dimensional data. t-SNE gives a feel or intuition of how the data is arranged in a high-dimensional space.



Word Algebra

Since Word2Vec characterizes words into quantified tokens, we can consequently add or subtract word vectors together. To add is to combine the meaning of the components and to subtract is to take out the context of one token from another. The following are examples of this vector algebra and their similarity scores: ***Dog + Food=Kibble, Coffee + tea=Brew***

Building the Topic Model

LDA's approach to topic modeling is it considers each document as a collection of topics in a certain proportion. And each topic as a collection of keywords, again, in a certain proportion. Once you provide the algorithm with the number of topics, all it does it to rearrange the topics distribution within the documents and keywords distribution within the topics to obtain a good composition of topic-keywords distribution. A topic is nothing but a collection of dominant keywords that are typical representatives. Just by looking at the keywords, we can identify what the topic is all about. I have given number of topics as 4.

View the topics in LDA model

```
Topic #1:
['like', 'taste', 'good', 'flavor', 'try', 'make', 'tea', 'really', 'great',
'one']

Topic #2:
['food', 'dog', 'treat', 'cat', 'hair', 'eat', 'salmon', 'grain', 'chew', 'dr
y']

Topic #3:
['coffee', 'chocolate', 'cup', 'bean', 'milk', 'roast', 'blend', 'dark', 'cak
e', 'coconut']

Topic #4:
['buy', 'get', 'product', 'find', 'order', 'love', 'price', 'time', 'bag', 'b
ox']
```

Visualize the topics-keywords

Evaluating topic model quality

We can use perplexity and coherence scores as measures to evaluate the topic model. Typically, lower the perplexity, the better the model. Similarly, the lower the UMass score and the higher the Cv score in coherence, the better the model.

8. Text Summarization

Text summarization is a subdomain of Natural Language Processing (NLP) that deals with extracting summaries from huge chunks of texts. Text summarization can broadly be divided into two categories — Extractive Summarization and Abstractive Summarization.

- Extractive Summarization: These methods rely on extracting several parts, such as phrases and sentences, from a piece of text and stack them together to create a summary. Therefore, identifying the right sentences for summarization is of utmost importance in an extractive method.
- Abstractive Summarization: These methods use advanced NLP techniques to generate an entirely new summary. Some parts of this summary may not even appear in the original text.

Here I am considering only extractive summarization technique. I am not using any machine learning library in this project. Rather I will use Python's NLTK library for summarizing Amazon Reviews.

Text Summarization Steps

- **Convert Paragraphs to sentences**
  First need to convert the whole paragraph into sentences. The most common way of converting paragraphs to sentences is to split the paragraph whenever a period is encountered.
- **Text Preprocessing**
  After converting paragraph to sentences, we need to remove all the special characters, stop words and numbers from all the sentences.
- **Tokenizing the sentences**
  We need to tokenize all the sentences to get all the words that exist in the sentences.
- **Find Weighted Frequency of Occurrence**
  Next, we need to find the weighted frequency of occurrences of all the words. We can find the weighted frequency of each word by dividing its frequency by the frequency of the most occurring word.
- **Replace Words by Weighted Frequency in Original Sentences**
  The final step is to plug the weighted frequency in place of the corresponding words in original sentences and finding their sum. It is important to mention that weighted frequency for the words removed during preprocessing (stop words, punctuation, digits etc.) will be zero and therefore is not required to be added.
- **Sort Sentences in Descending Order of Sum**

The final step is to sort the sentences in inverse order of their sum. The sentences with highest frequencies summarize the text. For instance,

```
Text: 'I have bought several of the Vitality canned dog food product s
and have found them all to be of good quality. The product looks more
like a stew than a processed meat and it smells better. My Labrador is
finicky, and she appreciates this product better than most.'

Summary: The product looks more like a stew than a processed meat and
it smells better.
```

9. Score Prediction using reviews

The purpose of this analysis is to make up a prediction model where we will be able to predict the rating of the product. In this analysis, we will not focus on the sentiment.
Train/Test: Spited 80% of the dataset into a training set and 20% into a testing set, using a train_test_split.
Feature Engineering: To make statistical algorithms work with text, we first must convert text to numbers. To do so, three main approaches exist i.e. Bag of Words, TF-IDF and Word2Vec

Bag of words (BoW) model

Bag of Words model is used to preprocess the text by converting it into a bag of words, which keeps a count of the total occurrences of most frequently used words.
In the bag of words approach the first step is to create a vocabulary of all the unique words. The next step is to convert each document into a feature vector using the vocabulary. The length of each feature vector is equal to the length of the vocabulary. The frequency of the word in the document will replace the actual word in the vocabulary. If a word in the vocabulary is not found in the corresponding document, the document feature vector will have zero in that place. **CountVectorizer:** Counting the number of times each word appears in a document.

TFI-DF Vectorizer: Calculating the frequency that each word appears in a document out of all the words in the document. The idea behind the TF-IDF approach is that the words that occur less in all the documents and more in individual document contribute more towards classification.TF-IDF is a combination of two terms. Term frequency and Inverse Document frequency. They can be calculated as:

TF = (Frequency of a word in the document)/ (Total words in the document)

IDF = Log ((Total number of docs)/ (Number of docs containing the word))

The Word2Vec Model

This is a predictive deep learning-based model to compute and generate high quality, distributed and continuous dense vector representations of words, which capture contextual and semantic similarity. Essentially these are unsupervised models which can take in massive textual corpora, create a vocabulary of possible words, and generate dense word embeddings for each word in the vector space representing that vocabulary.
For the Evaluation of the model here I am using two metrics:
Accuracy: Accuracy is a valid choice of evaluation for classification problems. Accuracy = Correct Predictions / Total Predictions.
F1-Score: The F-Measure is a popular metric for imbalanced classification. F-Measure = (2 * Precision * Recall) / (Precision + Recall).

Logistic Regression as a classification model performed the best result in classifying the reviews as compared to other classification models. Logistic Regression can find the best hyperplane which could separate the reviews into different categories. I have also applied different models on all the featurization techniques and can check for the best result.

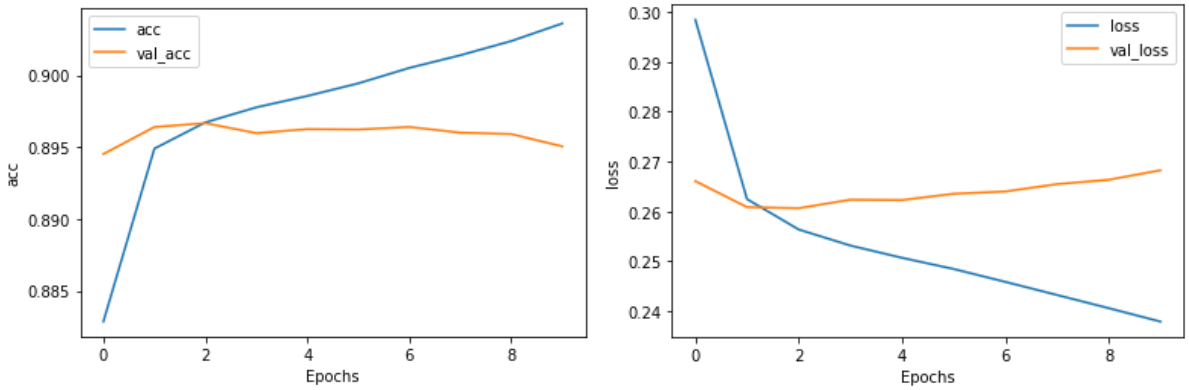
Algorithm	Evaluation metrics	BoW	TFI-DF	Word2Vec
Logistic Regression	Accuracy	.70	.67	.63
	F1-Score	.85	.81	.77
Random Forest	Accuracy	.63	.65	.63
	F1-Score	.77	.79	.77

From the above it is clear that Logistic Regression with Bag of Word model has given better result compared to Random Forest algorithm.

The Neural Network Model

The Keras library provides wrapper classes to allow us to use neural network models developed with Keras in scikit-learn. The hidden layer uses a rectifier activation function which is a good practice. Because we used a one-hot encoding for our dataset, the output layer must create 5 output values, one for each class. Used a "*softmax*" activation function in the output layer. This is to ensure the output values are in the range of 0 and 1 and may be used as predicted probabilities. Finally, the network uses the efficient Adam gradient descent optimization algorithm with a logarithmic loss function, which is called "*categorical_crossentropy*" in Keras.

```
Layer (type)                 Output Shape              Param #
=================================================================
embedding (Embedding)        (None, 100, 16)           160000
_________________________________________________________________
global_average_pooling1d (Gl (None, 16)                0
_________________________________________________________________
dense (Dense)                (None, 24)                408
_________________________________________________________________
dense_1 (Dense)              (None, 5)                 125
=================================================================
Total params: 160,533
Trainable params: 160,533
Non-trainable params: 0
```



The accuracy of the testing (89%) set has already surpassed our previous logistic Regression with BOW model, which is a great step further in terms of our progress.

10. Conclusion and Future Work

Ideally, the aim of the project was to help Amazon consumers to understand the sentiment of the review, to get consumer feedback in the form of topics covered by the reviews without having to go through all of them and Enable them to quickly extract the summary of the reviews without reading the entirely. As a future work I would like to implement the Text summarization using Artificial Neural Network.

11. References

- https://machinelearningmastery.com/multi-class-classification-tutorial-keras-deep-learning-library/
- https://realpython.com/python-keras-text-classification/
- https://github.com/dipanjanS/nlp_crash_course_plugin20/blob/master/notebooks/03_NLP%20Applications/07_NLP_Applications_Topic_Modeling.ipynb
- https://medium.com/analytics-vidhya/automated-keyword-extraction-from-articles-using-nlp-bfd864f41b34