Madhuri Dilliker

Ruid :191004207

Capstone Project Topic: News Article analysis using NLP and Sentimental Analysis

Table of Contents

Introduction

All around the world there can be events that are both good as well as bad, and we are aware of only those which we see or read on social media platform. Media has played a very important contribution in spreading these to majority people, its media's primary responsibly to make people aware of what is happening around the globe. As bigger responsibility this sounds, media houses is the way in which they express the content of events happening to the people. Media house content should be unbiased, original and free of exaggeration as the same story can be interpreted in many different ways by its readers. It is very well known to us that, we become who we are by what we read and what we say.

If a person reads a positive story which is filled with positive words makes that person feel positive and vise versa. It is clear that words used in the content plays an important role as that of original content. Humans are able to classify any content or article into positive or negative subconsciously. For example "That women is very generous", will give us a positive sentiment. We can surely say this as words used in this sentence are positive, this results in overall sentiment to be positive.

We can now use sentimental analysis to predict what would a person feel or think about particular news article. Term sentimental analysis calculates the sentiment score along with the combined power of natural language processing and text analysis to classify if the sentiment is 'positive', 'neutral' or 'negative'. Sentiment analysis will combine the human language semantics and symbolic representation, this will be input for the algorithm, algorithm will be trained with news text to predict if the input news text sentiment is positive or negative or neutral. This is supervised learning approach where we will train the model and predict its sentiment class.

For prediction and classification, we make use of machine learning process and Natural Language processing technique to understand how the text data is and characteristics of this text data and classify the sentiment. We need a computational model that can classify the test into sentiment category. With increasing in information and development of online news forum, we need a effective tool to accurately classify this information into category. In this way we could easily filter, search and store useful information. For news agencies this will be a accurate tool as they receive lot of articles in a day.

With emerging demand in the field of digital news, it is observed that there are vast developments in deep learning for Natural Language processing. In current times deep learning model can be used for NLP task which involves machine translation, speech recognition, sentiment analysis and classification problems. We need a system for online news which would associate with problems faced by journalism to classify if they are spreading a news which has a negative impact to the world.

The scope of this project is to use the new article data which is in the form of raw textual data format and interpret if the news represent a positive and negative sentiment.

Dataset

Dataset: News Articles

Data source: Kaggle

Description: This dataset has data collected from various media houses home page to see which News media shares/writes articles with less gory words. This dataset stands as sample to find out which media house conveys the NEWS in more optimistic way. Data is collected from multiple news websites from October 2017 to Nov 2017. The dataset consist of 155 null values and 1100 rows. The dataset has following attributes:

1. **Title** – Title of the news article

2. **Summary** – Gist of the article

3. **Text** – Full text inside an article

4. **Url** – url of the article

5. **Keywords** – important words used in the article

In this Kaggle data source we have dataset for all the popular house media. We will move ahead with cnn dataset for this project.

Index	ID	KEYWORDS	SUMMARY	TEXT	TITLE	URL
0	0	['energy', 'sugars', '…	Story highlights Do…	Story highlights Don't be fooled by the word "energy" Some energy bars contain as much saturated fat as a	Are energy bars healthy?	https://www.cnn.com/2017/08/25/health/energy-bars-healthy-food-drayer/index.html
1	1	['facebook', 'whats', 'w…	Chat with us in Facebook M…	Chat with us in Facebook Messenger. Find out what's happening in the world as it unfolds.	Tamagotchi is back	http://www.cnn.com/videos/cnnmoney/2017/10/10/tamagotchi-is-back.cnn-tech
2	2	['jedi', 'shots', 'r…	ESPN's "Monday Night Footbal…	ESPN's "Monday Night Football" had bears, vikings and Jedi on Monday night.	'Star Wars: The Last Jedi' trailer debuts on 'Monday Night Football'	http://money.cnn.com/2017/10/09/media/star-wars-the-last-jedi-trailer/index.html
3	3	['clients', 'art', 'sci…	Lyn Harris' independent s…	This feature is part of ' Details ,' a new series that captures the creation of some of the world's most int…	Art and science collide in this one-of-a-kind scent	https://www.cnn.com/style/article/details-perfumer-lyn-harris/index.html
4	4	['akufoaddo…	(CNN) A tanker exploded near…	(CNN) A tanker exploded near a gas station in Accra on Saturday killing seven people, officials in Ghana sai…	Seven killed, dozens injured in Ghana tanker explosion	https://www.cnn.com/2017/10/08/africa/ghana-tanker-explosion/index.html
5	5	['spanish', 'independen…	Carles Puigdemont, t…	(CNN) Pro-independence Catalans gathered on the streets of Barcelona Tuesday ahead of a hotly anticip…	Catalans' future on line as parliament meets	https://www.cnn.com/2017/10/10/europe/catalonia-puigdemont-parliament-address/index.html
6	6	['press', 'excuse', '…	"I think it's fake news, bu…	(CNN) In a Forbes magazine interview published Tuesday morning, President Donald Trump was asked about repor…	The Trump White House's 'joke' excuse	http://www.cnn.com/2017/10/10/politics/trump-joking/index.html
7	7	['pollution…	President Barak Obama s…	(CNN) The days when all three of her children were simultaneously gasping for air are interminably etche…	Health impact of Trump environmental repeal	https://www.cnn.com/2017/10/10/health/health-effects-clean-power-repeal/index.html
8	8	['look', 'response',…	(CNN) Finally lumbering int…	Michael D'Antonio is the author of the book " Never Enough: Donald Trump and the Pursuit of Success " (St…	Trump in Puerto Rico: A narcissist's tour de force	https://www.cnn.com/2017/10/03/opinions/trump-in-puerto-rico-narcissists-tour-de-force-opinion-dan…
9	9	['okunoin', 'tohoku', '…	(CNN) — Upon hearing I wou…		Yamadera Risshakuji in Tohoku: 1,015 steps to Japan temple	https://www.cnn.com/travel/article/yamadera-temple-tohoku-japan/index.html
10	10	['land', 'life', 'pu…	Doug Tompkins and Kristine …	Catch "The Wonder List" on CNN Saturdays at 9 p.m. ET/ PT	They purchased paradise ... then gave it all away	http://www.cnn.com/travel/article/wonder-list-bill-weir-patagonia/index.html

Figure 1:Sample Data

Problem Statement

In this digital world people tend to consume news from social media, online news paper, online media house, popular search engines. And it becomes difficult to identify if the content behind the news. To classify if news article has positive or negative sentiment. Sentimental scores are extracted for each row in the dataset and classified if it belong to “positive”, “negative” or “neutral” sentiment.

Exploratory Data Analysis

Word Count

Gaining some insights from the new articles data. For now we just have our new text data so let's find the word count for each news article and news length distribution for each article.
We can see that mean of the distribution for word count lies near 800 words per abstract. The minimum and maximum word count ranges from 2 to 6000. This gives us insight into word count range for the data set we are handling and how it is varied for each rows. To have number of word count for each row, we can create a new column "Wordcount" using lambda function which will describe word count for each row.

Index	ID	KEYWORDS	SUMMARY	TEXT	TITLE	URL	Wordcount
0	0	['energy', 'sugars', 'ba…	Story highlights Do…	Story highlights Do…	Are energy bars healthy?	https://www.cnn.com/2…	293
1	1	['facebook', 'whats', 'wor…	Chat with us in Facebook M…	Chat with us in Facebook M…	Tamagotchi is back	http://www.cnn.com/v…	16
2	2	['jedi', 'shots', 'rey…	ESPN's "Monday Night Footbal…	ESPN's "Monday Night Footbal…	'Star Wars: The Last Jedi…	http://money.cnn.com…	370
3	3	['clients', 'art', 'scien…	Lyn Harris' independent s…	This feature is part of ' …	Art and science colli…	https://www.cnn.com/s…	172
4	4	['akufoaddo', 'tanker', 'in…	(CNN) A tanker exploded near…	(CNN) A tanker exploded near…	Seven killed, dozens injure…	https://www.cnn.com/2…	140
5	5	['spanish', 'independence…	Carles Puigdemont, t…	(CNN) Pro-independence …	Catalans' future on lin…	https://www.cnn.com/2…	1019
6	6	['press', 'excuse', 'se…	"I think it's fake news, bu…	(CNN) In a Forbes magazi…	The Trump White House's…	http://www.cnn.com/2…	429
7	7	['pollution', 'repeal', 'ke…	President Barak Obama s…	(CNN) The days when all thre…	Health impact of Trump envi…	https://www.cnn.com/2…	1300
8	8	['look', 'response', '…	(CNN) Finally lumbering int…	Michael D'Antonio is …	Trump in Puerto Rico: …	https://www.cnn.com/2…	1237
9	9	['okunoin', 'tohoku', 'ja…	(CNN) — Upon hearing I wou…		Yamadera Risshakuji in…	https://www.cnn.com/t…	584
10	10	['land', 'life', 'purc…	Doug Tompkins and Kristine …	Catch "The Wonder List" …	They purchased paradise ... …	http://www.cnn.com/t…	943
11	11	['kids', 'parental', '…	I'm not sure I needed a stud…	Kelly Wallace is CNN's digi…	Parental burnout: It's…	https://www.cnn.com/2…	2363

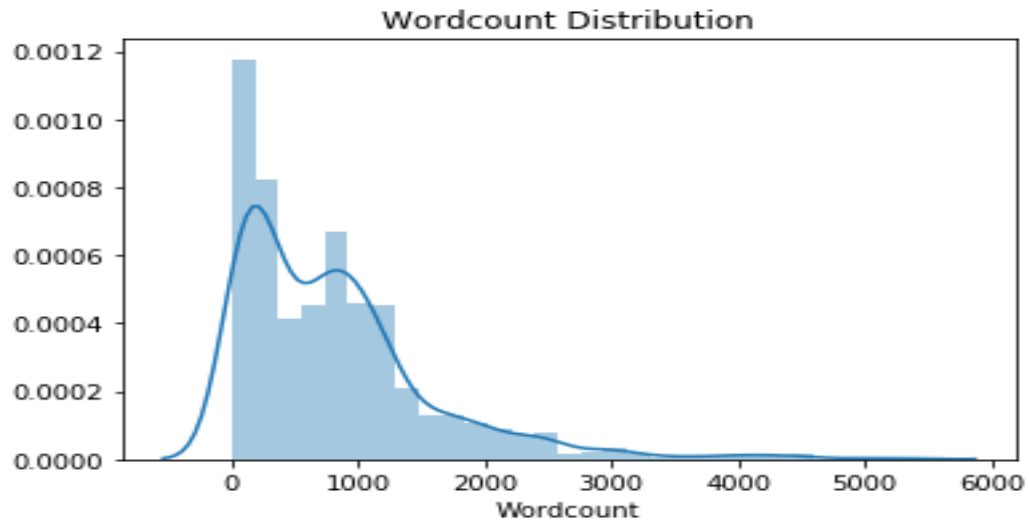

Figure 2: Word count distribution

News Length

For news length distribution the mean length lies around 5000 characters for this CNN dataset. The minimum and maximum text character ranges from 6 to 33000. This gives us insight into text character count range for the data set we are handling and how it is varied for each rows. To have text length for each row, we can create a new column “News Length” using lambda function which will describe text length for each row.

Index	ID	KEYWORDS	SUMMARY	TEXT	TITLE	URL	Wordcount	News Text Length
0	0	['energy', 'sugars', 'ba…	Story highlights Do…	Story highlights Do…	Are energy bars healthy?	https://www.cnn.com/2…	293	1713
1	1	['facebook', 'whats', 'wor…	Chat with us in Facebook M…	Chat with us in Facebook M…	Tamagotchi is back	http://www.cnn.com/v…	16	89
2	2	['jedi', 'shots', 'rey…	ESPN's "Monday Night Footbal…	ESPN's "Monday Night Footbal…	'Star Wars: The Last Jedi…	http://money.cnn.com…	370	2191
3	3	['clients', 'art', 'scien…	Lyn Harris' independent s…	This feature is part of ' …	Art and science colli…	https://www.cnn.com/s…	172	1069
4	4	['akufoaddo', 'tanker', 'in…	(CNN) A tanker exploded near…	(CNN) A tanker exploded near…	Seven killed, dozens injure…	https://www.cnn.com/2…	140	863
5	5	['spanish', 'independence…	Carles Puigdemont, t…	(CNN) Pro-independence …	Catalans' future on lin…	https://www.cnn.com/2…	1019	6694
6	6	['press', 'excuse', 'se…	"I think it's fake news, bu…	(CNN) In a Forbes magazi…	The Trump White House's…	http://www.cnn.com/2…	429	2493
7	7	['pollution', 'repeal', 'ke…	President Barak Obama s…	(CNN) The days when all thre…	Health impact of Trump envi…	https://www.cnn.com/2…	1300	8408
8	8	['look', 'response', '…	(CNN) Finally lumbering int…	Michael D'Antonio is …	Trump in Puerto Rico: …	https://www.cnn.com/2…	1237	7285
9	9	['okunoin', 'tohoku', 'ja…	(CNN) — Upon hearing I wou…		Yamadera Risshakuji in…	https://www.cnn.com/t…	584	3750
10	10	['land', 'life', 'purc…	Doug Tompkins and Kristine …	Catch "The Wonder List" …	They purchased paradise ... …	http://www.cnn.com/t…	943	5557

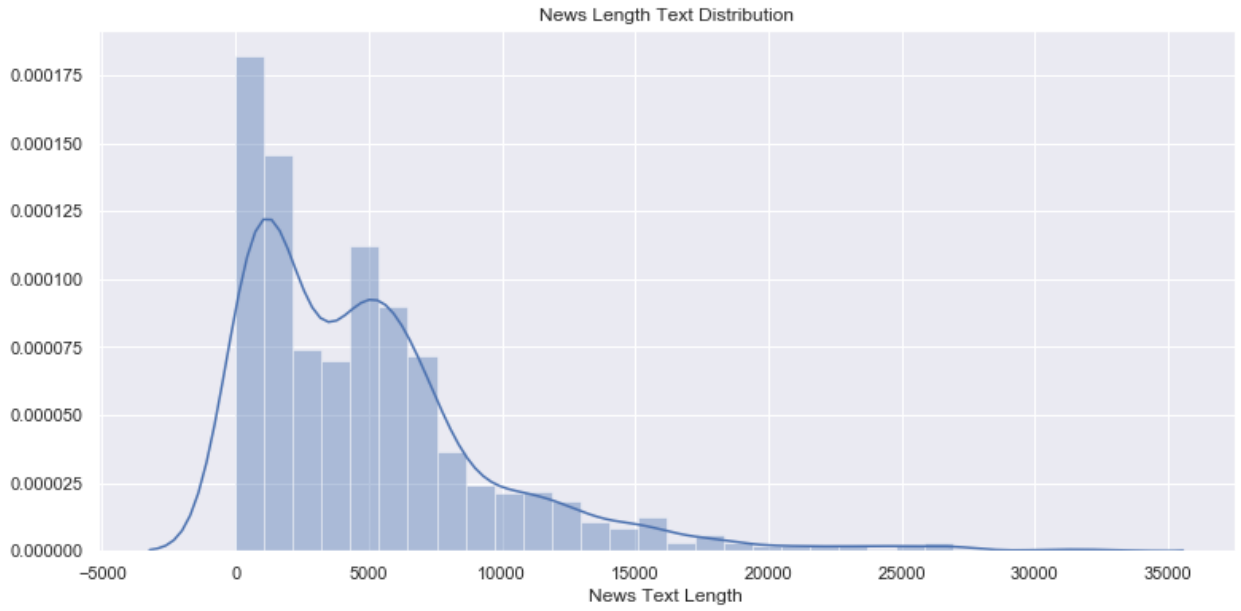

Figure 3. News Length Text Distribution

Most common and uncommon words

Getting the list of common and uncommon words will give us the words which are used frequently which can be later added to custom list of stop words. Following is the list of common

words with their value count, we can use this list later to add these frequently used words in custom stop word list

```
In [15]: Common_Words = pd.Series(' '.join(df['TEXT']).split()).value_counts()[:20]

In [16]: Common_Words
Out[16]:
the      37143
of       21587
to       21051
and      19428
a        17914
in       15220
that      7868
is        7338
for       6757
on        6439
with      5219
was       4955
The       4437
are       4351
as        4240
at        3991
have      3560
he        3475
I         3403
it        3325
dtype: int64
```

Methodology

Phase 1: Cleaning the data set
Phase 2: Use rules of NLP and NTLK for text vectorization for preparing text data for classification and text pre processing
Phase 3: Extracting Sentiment Scores
Phase 4: Use the effective classification technique such as Naïve Bayes, SVM and LSTM model.
Phase 5: Check the classification accuracy and represent the result graphically

Feature Engineering

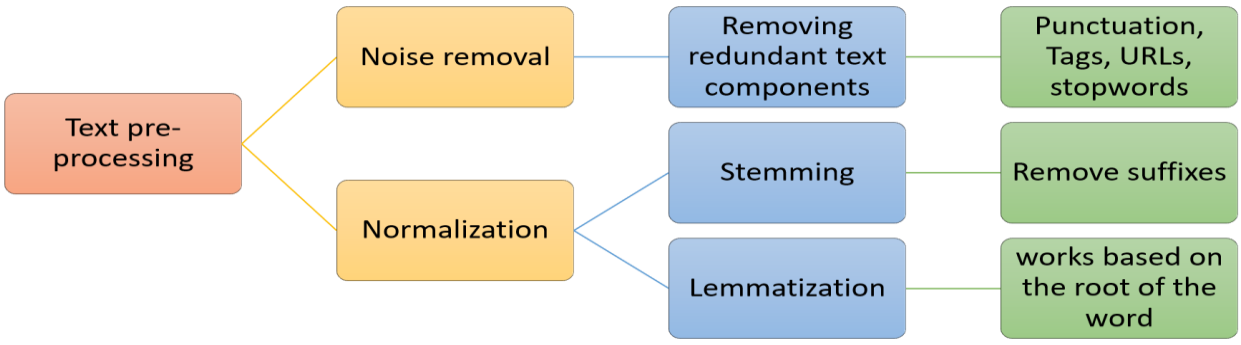

Figure 4: Text pre-processing

Here we will transform our raw news data into feature which can be used as input for classification models.

1.Text cleaning

Following steps are used to perform cleaning of data

- Removing special character and tags: Tags like '&lt' and special characters such as "\\d" which do not add value to text data are needed to be removed from the text.
- Change text to lower case: Example the word price with P in caps and one with all lower case "price" and "Price" all will be treated as different entity, so as to treat these words as equal converting the text to lower case would solve this problem.
- Punctuation signs: Characters in the text which includes '?', '!' or ','should be removed
- Possessive Pronouns: The words with apostrophe s i.e 'John' or 'John's' should be treated as same so we need to remove this from the given text

2.Text processing

NTKL: NTKL is Natural Language Toolkit is text processing library which is used in building python programs which involves work with human language data for Natural language processing.

Stopwords: A stop word is list of frequently used words which can be ignored to derive better results. NTKL package gives downloaded list of common words. We can modify this list and add some new words which are common in the news text and delete it from the text data. Stop words is mostly referred as noise data which should be eliminated to generate better results

```
   ...: print(Stopwords)
{'were', 'when', "doesn't", 'which', 'its', 'they', 'these', 'we', 'way', 'at', 'each',
've', 'i', 'into', 'wasn', 'this', 'more', 'our', "won't", 'be', 'her', 'having', 'y',
'doesn', 'same', "mustn't", 'whom', 'mustn', 'am', 'any', 'both', 'about', 'there',
'two', 'has', 'them', 'again', 'until', 'itself', 'once', 'other', 'than', 'have',
'above', 'that', 'him', 'under', 'm', 'really', 'not', 'couldn', "you've", 'say',
'mightn', 'yourself', 'down', 'are', 'below', 'the', "didn't", 're', 'o', 'isn', 'told',
'will', 'hasn', 'take', 'my', 'because', 'shan', 'also', 'up', 'during', 'your', 'where',
'yourselves', 'what', 'go', 'or', "you'd", 'she', 'their', 'was', 'most', 'myself',
'hers', 'must', 'his', 'hadn', "weren't", 'he', 'with', 'just', 'said', 'after',
"shan't", 'may', 'out', 'does', 'ain', 'on', 'much', "you'll", 'only', 'ma', 'to',
'themselves', 'here', 'by', "you're", 'herself', 'needn', 'had', 'in', "should've",
'didn', 't', 'would', "wouldn't", 'very', 'such', "hasn't", 'caption', 'between', 'too',
's', 'why', 'a', 'those', 'being', 'make', "isn't", 'aren', 'shouldn', 'like', 'ours',
'do', "it's", 'it', 'well', 'through', 'so', "needn't", 'from', "don't", 'own', 'haven',
'nor', 'won', 'but', 'weren', 'ourselves', "that'll", "aren't", 'one', "mightn't", 'new',
"hadn't", 'did', 'll', 'for', 'over', 'you', 'yours', 'me', 'as', 'all', 'wouldn',
'could', 'don', 'many', 'while', 'even', 'an', 'before', 'off', 'know', 'doing', 'd',
'get', 'then', 'can', 'some', 'no', 'ago', 'come', 'against', 'if', 'of', 'should',
"she's", 'been', 'how', "shouldn't", 'and', "couldn't", 'further', 'is', 'few',
"haven't", 'theirs', "wasn't", 'who', 'now', 'himself'}
```

Word Cloud

Word cloud is another way to which represents frequency of a word in given text. If the size of word is larger this means the word is commonly used in the given text or represents importance of each word. Below is the word cloud for our text column in dataset.

Figure 5: Word Cloud

3.Text Vectorization

When words are converted to numbers this process is called vectorization. Word vectorization is a method which maps words from vocabulary to corresponding vectors of real numbers which can later be used for predictions or word semantics. We convert group of sentences into tokens, this is also called process of segmentation. We split the data into smallest possible chunk. Each word is converted to its numerical vector format. In simple words tokenization process will break the stream of text in meaningful entities called tokens. Tokenization can be performed on paragraph to break it into meaningful sentences and sentence can be further broken down to words. We are using countVectorizer function to map each word to its feature, which can later be transformed into a sparse matrix

4.Normalization

Normalization is the process of converting the list of words to uniform sequence. This will aid in processing the text for training. This is achieved using stemming and lemmatization . Stemming and lemmatization is normalization of words in the text data, i.e. reducing word to it original root form.

Stemming: Stemming is the process of reducing derived words to their word stem, root or root form. Words which are spelled differently because of their tense can be deduced to one central word, if they have the same meaning. For example, cook, cooking and cooked will be deduced to one word ‘cook’. Stemming is applied to single word .**Lemmatization** is the process of reducing

a word to its lemma ie resolving words to their dictionary form and it requires to know the part of speech for the word. Lemmatization needs to know the structure of language

5.N Gram Language model

N gram is sequence of words or N tokens. N gram language model can predict probability of input N-gram with any sequence of words. We use this language model to predict the probability of seeing word x, if we know which all words are used previously ie y where previous words contain n-1 list of words. Before moving forward to use Term frequency and Inverse document frequency for our text column. Lets visualize unigram, bigram and trigram for our text dataset Using count vectorization.

Unigram

Sequence of that contains one word at a time. Following is the visualization for top 20 words in our text data set.

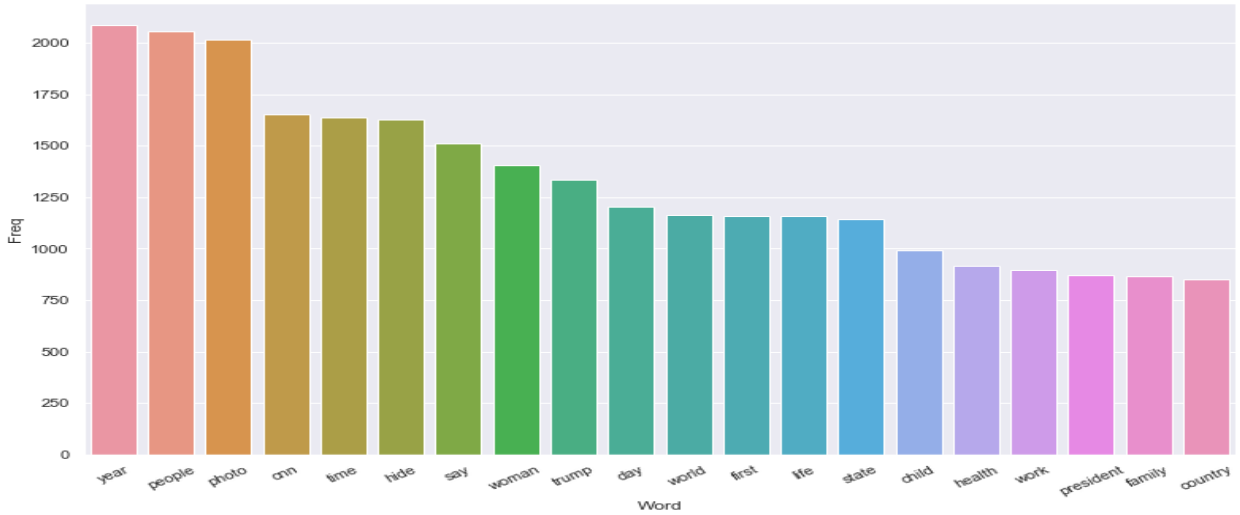

Figure 6: Unigram Visualization

Bigram

Bigram is when sequence contains 2 words at a time, following is bar plot visualization of bigram for our data set

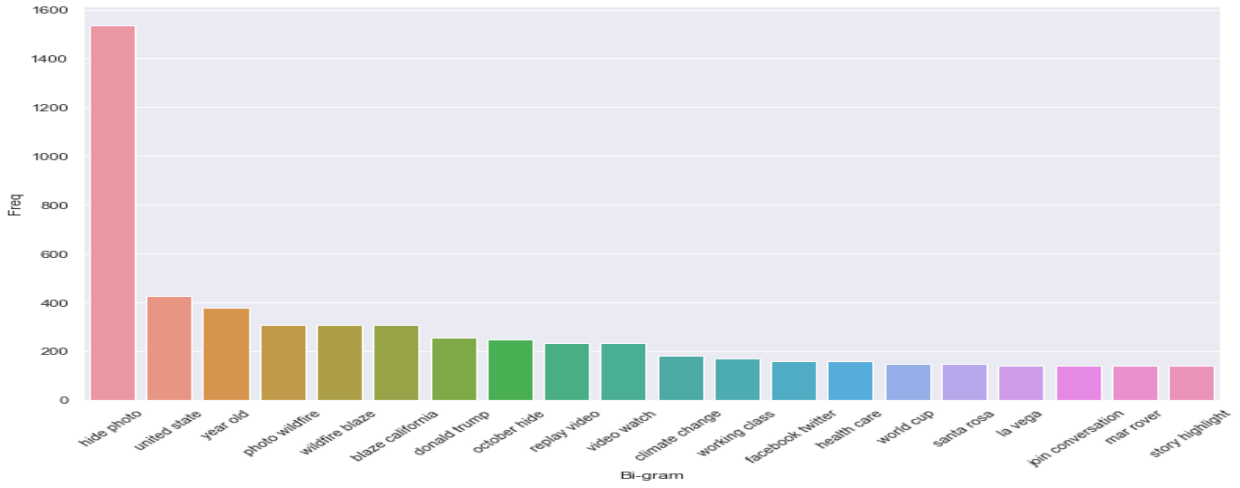

Figure 7- Bi-gram Visualization

Trigram: Tri-gram is when sequence contains 3 words at a time, following is bar plot visualization of bigram for our data set

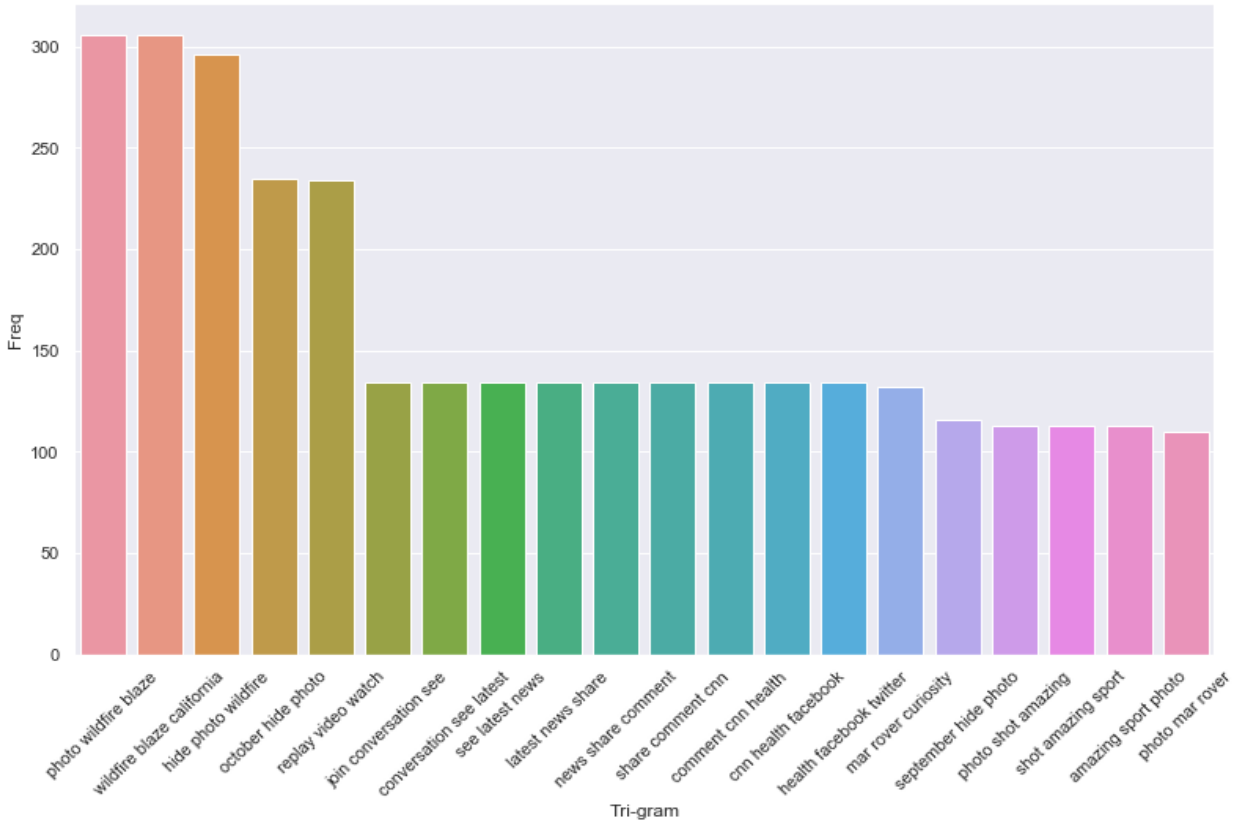

Figure 8: Trigram Visualization

6.TF-IDF

Term- Frequency:

Term Frequency can be defined as how frequently a particular word appears in the document. In a document each sentence will be of different length and it is a least possibility that a sentence would be of same length, so there is a slight possibility a word will appear in long sentence more than one time as compared to sorter sentences. Formula to calculate term frequency is as follows:

$$TF = \frac{No\ of\ time\ word\ appear\ in\ the\ document}{Total\ no\ of\ word\ in\ the\ document}$$

Inverse Document Frequency:

This is another way of finding the importance of a word. It is based on the fact that less frequent used word are more important and informative than common words in the document. IDS formula is given below:

$$IDF = log_{10} \frac{Number\ of\ Document}{Number\ of\ document\ in\ which\ word\ appear}$$

TF-IDF

When we multiply the values of Term Frequency (TF) and Inverse Document Frequency (IDF) we get TF-IDF score.TF-IDF score will represent relative importance of a term/word in the entire document. TF-IDF will reduce value of the common word that are used in text document. TF- is term frequency which will represent frequency of a term in given document and IDF – is Inverse Document Frequent. The formula for TF-IDF and algorithm is given below

$$TFIDF(t,d) = TF(t,d) \times \log\left(\frac{N}{DF(t)}\right)$$

Being:

- t: term (i.e. a word in a document)
- d: document
- $TF(t)$: term frequency (i.e. how many times the term t appears in the document d)
- N: number of documents in the corpus
- $DF(t)$: number of documents in the corpus containing the term t

TF-IDF value will increase proportionally with number of time a particular word appears in the text document and is offset by the number of document in the corpus that contain the word, which would help in adjusting to the fact that some words appear more frequently generally. TF-IDF feature creation process is fast and we can avoided overfitting by fine tuning the features extracted using this process.

Top Feature score extracted from our text column

```
 Top Featured Word Score:
insurance 0.6
survivor 0.267
insurance company 0.261
pre existing condition 0.169
pre existing 0.169
existing condition 0.169
company 0.155
turner 0.147
obamacare 0.134
health 0.13
pre 0.13
health insurance 0.13
existing 0.128
law 0.128
domestic 0.127
abuse 0.119
policy 0.102
condition 0.1
rape 0.094
post 0.094
state 0.085
domestic violence 0.082
medical 0.069
violence 0.066
counseling 0.066
```

Sentimental Analysis

Text Blob

Text blob in python is a library for processing textual data. It provides a simple interface for dividing into the natural language processing task like classification, sentiment analysis. **Textblob,sentiments** function will calculate polarity and sentiment for given textual data.We can also classify sentiment into positive and negative sentiment using **NaiveBayesAnalyzer** which will classify whole text data into positive and negative score.

Polarity

When the floating point sentiment value lies within the range of -1.0 to 1.0 where 0 represents neutral sentiment, +1 represents more positive sentiment and -1 represents more negative sentiment.

Subjectivity

When a floating point sentiment value lies within range of 0.0 to 1.0 where 0.0 tends to more objective and 1.0 tends to more subjective. Objective sentence represent sentence are factual and subjective sentence refer to emotion, feelings, believe and views.

We have calculated subjectivity and polarity for each row in our dataset.

Index	ID	KEYWORDS	SUMMARY	TEXT	TITLE	URL	Wordcount	News Text Length	New Text	Polarity	Subj
0	0	['energy', 'sugars', 'ba…	Story highlights Do…	Story highlights Do…	Are energy bars healthy?	https://www.cnn.com/2…	293	1713	story highlight foo…	0.208333	0.485802
1	1	['facebook', 'whats', 'wor…	Chat with us in Facebook M…	Chat with us in Facebook M…	Tamagotchi is back	http://www.cnn.com/v…	16	89	chat u facebook mess…	0	0
2	2	['jedi', 'shots', 'rey…	ESPN's "Monday Night Footbal…	ESPN "Monday Night Footbal…	'Star Wars: The Last Jedi…	http://money.cnn.com…	370	2191	espn monday night footbal…	0.0715995	0.383361
3	3	['clients', 'art', 'scien…	Lyn Harris' independent s…	This feature is part of ' …	Art and science colli…	https://www.cnn.com/s…	172	1069	feature part detail series…	0.303835	0.616356
4	4	['akufoaddo', 'tanker', 'in…	(CNN) A tanker exploded near…	(CNN) A tanker exploded near…	Seven killed, dozens injure…	https://www.cnn.com/2…	140	863	cnn tanker exploded near…	0.0925926	0.475926
5	5	['spanish', 'independence…	Carles Puigdemont, t…	(CNN) Pro-independence …	Catalans' future on lin…	https://www.cnn.com/2…	1019	6694	cnn pro independence …	0.0361107	0.34582
6	6	['press', 'excuse', 'se…	"I think it's fake news, bu…	(CNN) In a Forbes magazi…	The Trump White House's…	http://www.cnn.com/2…	429	2493	cnn forbes magazine inte…	0.0318919	0.494653
7	7	['pollution', 'repeal', 'ke…	President Barak Obama s…	(CNN) The days when all thre…	Health impact of Trump envi…	https://www.cnn.com/2…	1300	8408	cnn day three child simulta…	0.0552183	0.441313
8	8	['look', 'response', '…	(CNN) Finally lumbering int…	Michael D'Antonio is …	Trump in Puerto Rico: …	https://www.cnn.com/2…	1237	7285	michael antonio autho…	0.0385732	0.53878

Below is the sentiment score for our overall text data set

```
blob.sentiment
Sentiment(classification='pos', p_pos=1.0, p_neg=3.633586025330017e-19)
```

Below is the subjectivity and polarity line graph for our overall text data

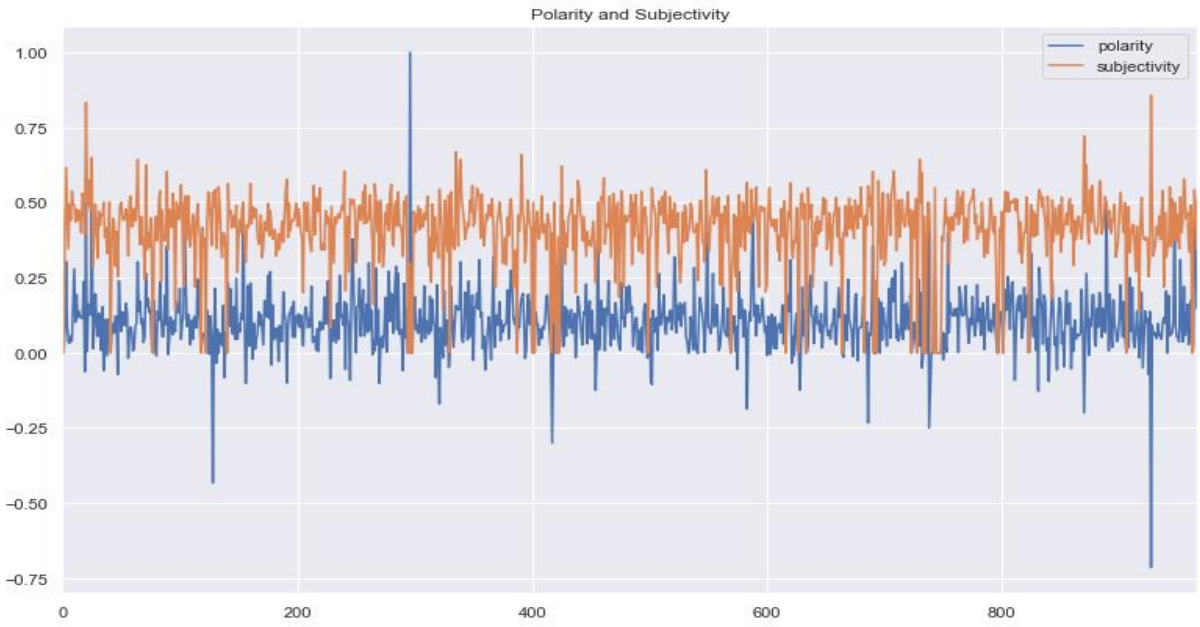

Figure 9: Sentiment line plot

Extracting Sentiment Score

In order to perform sentiment analysis we do not have sentiment score in our dataset. We need a score so that we can apply classification model to our text dataset, to classify if the news text represent positive or negative news.

Vader Sentiment Analyzer

Vander stands for valence aware dictionary and sentiment reasoner. It is a sentiment analysis library is built in NLTK which is lexicon and rule based which is specifically designed for sentiments expressed in online or social media. It uses a combination of sentiment lexicon, which is a special list of lexical feature which are labelled as positive, neutral or negative score. **polarity _scores** method is used to calculate the sentiment score. All these score should be add up to 1. Vander generates positive and negative scores and tell us how positive and negative a sentiment is. This library is fast enough and doesn't suffer from speed performance. It not only gives the scores for positive and negative but also proves a compound score, **compound** score is calculates sum of all lexicon ratings and is normalized between -1(extremely negative) and 1(extremely positive).

- When compound score is greater than **0.05** then sentiment is **positive.**
- When compound score lies between the range which is **greater than -0.05 and less than 0.05** then sentiment is **neutral.**
- When compound score is less than **-0.05** then sentiment is **negative**

Below is the snippet of our data set after applying **polarity _scores** method in vander sentiment analyzer library

Index	ID	KEYWORDS	SUMMARY	TEXT	TITLE	URL	Wordcount	News Text Length	New Text	Polarity	Subj	Sentiment Analyser
0	0	['energy', 'sugars', 'ba…	Story highligh…	Story highlig…	Are energy b…	https:// www.cnn.com/2…	293	1713	story highlight foo…	0.208333	0.485802	{'neg': 0.024, 'neu': 0.864, 'pos': 0.112, 'compound': 0.9778}
1	1	['facebook', 'whats', 'wor…	Chat with us in Fa…	Chat with us…	Tamagotc…	http:// www.cnn.com/v…	16	89	chat u facebook mess…	0	0	{'neg': 0.0, 'neu': 1.0, 'pos': 0.0, 'compound': 0.0}
2	2	['jedi', 'shots', 'rey…	ESPN's "Monday …	ESPN "Monday…	'Star Wars: Th…	http:// money.cnn.com…	370	2191	espn monday night footbal…	0.0715995	0.383361	{'neg': 0.084, 'neu': 0.848, 'pos': 0.069, 'compound': -0.8878}
3	3	['clients', 'art', 'scien…	Lyn Harris' …	This feature…	Art and science …	https:// www.cnn.com/s…	172	1069	feature part detail series…	0.303835	0.616356	{'neg': 0.008, 'neu': 0.865, 'pos': 0.127, 'compound': 0.9717}
4	4	['akufoaddo', 'tanker', 'in…	(CNN) A tanker e…	(CNN) A tanker …	Seven killed, …	https:// www.cnn.com/2…	140	863	cnn tanker exploded near…	0.0925926	0.475926	{'neg': 0.1, 'neu': 0.834, 'pos': 0.066, 'compound': -0.8225}
5	5	['spanish', 'independence…	Carles Puigdemo…	(CNN) Pro-ind…	Catalans' future o…	https:// www.cnn.com/2…	1019	6694	cnn pro independence …	0.0361107	0.34582	{'neg': 0.094, 'neu': 0.817, 'pos': 0.089, 'compound': -0.9419}
6	6	['press', 'excuse', 'se…	"I think it's fak…	(CNN) In a Forbe…	The Trump White Ho…	http:// www.cnn.com/2…	429	2493	cnn forbes magazine inte…	0.0318919	0.494653	{'neg': 0.099, 'neu': 0.778, 'pos': 0.123, 'compound': 0.9506}
7	7	['pollution', 'repeal', 'ke…	President Barak Ob…	(CNN) The day…	Health impact o…	https:// www.cnn.com/2…	1300	8408	cnn day three child simulta…	0.0552183	0.441313	{'neg': 0.068, 'neu': 0.801, 'pos': 0.131, 'compound': 0.9984}
8	8	['look', 'response', '…	(CNN) Finally …	Michael D'Anton…	Trump in Puerto R…	https:// www.cnn.com/2…	1237	7285	michael antonio autho…	0.0385732	0.53878	{'neg': 0.17, 'neu': 0.719, 'pos': 0.11, 'compound': -0.9987}

Sentiment Category

Now we have compound score we can follow following steps to extract the sentiment category from this score

1. Extract only **compound score** into a new column in pandas dataframe
2. Classifying the score as positive negative and neutral in new column called 'Sentiment category'
3. Converting column type to category type
4. Adding a column to have numerical columns to category
5. Treating neutral type as positive category

Following is the snippet of dataset were we have a column which indicate sentiment category

We now have following new columns in our dataframe.

- Compound score
- Sentiment category
- Sentiment Category code
- Sentiment Category code 1 (neutral is treated as positive)

Sentiment Analyser	Compound Score	Sentiment_Category	Sentiment_Category_Code	ntiment_Category_code
{'neg': 0.024, 'neu': 0.864, 'pos': 0.112, 'compound': 0.9778}	0.9778	Positive	2	1
{'neg': 0.0, 'neu': 1.0, 'pos': 0.0, 'compound': 0.0}	0	Neutral	1	1
{'neg': 0.084, 'neu': 0.848, 'pos': 0.069, 'compound': -0.8878}	-0.8878	Negative	0	0
{'neg': 0.008, 'neu': 0.865, 'pos': 0.127, 'compound': 0.9717}	0.9717	Positive	2	1
{'neg': 0.1, 'neu': 0.834, 'pos': 0.066, 'compound': -0.8225}	-0.8225	Negative	0	0
{'neg': 0.094, 'neu': 0.817, 'pos': 0.089, 'compound': -0.9419}	-0.9419	Negative	0	0
{'neg': 0.099, 'neu': 0.778, 'pos': 0.123, 'compound': 0.9506}	0.9506	Positive	2	1
{'neg': 0.068, 'neu': 0.801, 'pos': 0.131, 'compound': 0.9984}	0.9984	Positive	2	1
{'neg': 0.17, 'neu': 0.719, 'pos': 0.11, 'compound': -0.9987}	-0.9987	Negative	0	0
{'neg': 0.029, 'neu': 0.911, 'pos': 0.06, 'compound': 0.9678}	0.9678	Positive	2	1
{'neg': 0.06, 'neu': 0.792, 'pos': 0.148, 'compound': 0.9987}	0.9987	Positive	2	1

Model

Multinomial Naïve Bayes

Naïve Bayes belongs to family of algorithms based on application of bayes theorem with a strong assumption, that every feature is independent of others, and to predict the category of a given input sample. This algorithm is probability based classifier which will calculate the Bayes theorem and category which represents the highest will be the output of classification. Naïve Bayes are commonly used in NLP problems. This algorithm is considered to be fast, reliable and accurate in NLP applications

SVM:

It works by finding a hyperplane that separates the two classes with maximum margin. SVM algorithm for text classification problem can obtain into good results as its very effective in high dimension space

LSTM Modeling

- Vectorizing text column by converting each text into a vector or integer sequence.
- Limiting and setting max features to 30000 words
- Truncating and padding input sequence so the length of all the sequence is same before they are processed

- The activation function layer used is soft max and model loss function is compiled using categorical crossentropy as it is multi class classification problem
- The output layer will create 2 output values one for each class.

Result

Model	Accuracy
Multinomial Naïve Bayes	82
Random Forest	75
Svm	66
LSTM	97

We can see from above evaluation Multinomial Naïve Bayes and LSTM could classify news article text in with better accuracy as compared to SVM and Random Forest.

With these result, we can construct execution measurements that are valuable for a speedy evaluation on how well a classifier functions:

- **Accuracy**: gives level of writings that were anticipated with the right tag.
- **Precision**: gives level of models the classifier got directly out of the complete number of precedents that it anticipated for a given tag.
- **Recall**: gives level of precedents the classifier anticipated for a given tag out of the absolute number of models it ought to have anticipated for that given tag.
- **F1 Score**: gives the consonant mean of exactness and review

For Multinomial Naïve Bayes we have following classification report which indicates precision, recall and F1 score for positive (1) and negative sentiment. If we calculate the average of all these values it turns out to be around 0.82

```
In [116]: print(classification_report(y_test,predicted_result))
              precision    recall  f1-score   support

           0       0.80      0.70      0.75       109
           1       0.83      0.90      0.86       182

    accuracy                           0.82       291
   macro avg       0.82      0.80      0.80       291
weighted avg       0.82      0.82      0.82       291
```

We have divided our data into test and training split into 70/30 ratio. Below is the model summary for LSTM model.

```
In [177]: print(model.summary())
Model: "sequential_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
embedding_1 (Embedding)      (None, 2778, 150)         4500000
_________________________________________________________________
lstm_1 (LSTM)                (None, 200)               280800
_________________________________________________________________
dense_1 (Dense)              (None, 2)                 402
=================================================================
Total params: 4,781,202
Trainable params: 4,781,202
Non-trainable params: 0
_________________________________________________________________
```

Loss function is used in machine learning algorithm, to optimize it and loss is calculated on training and testing data. And this graph will provide the insight on how well model is performing in training and testing dataset. It reflects sum of errors made for each training and testing set. Loss value also indicates how well or bad model behaves after each iteration. The graph shows overfitting model during training as there is sudden rise in the start of training model and then value is decreased.

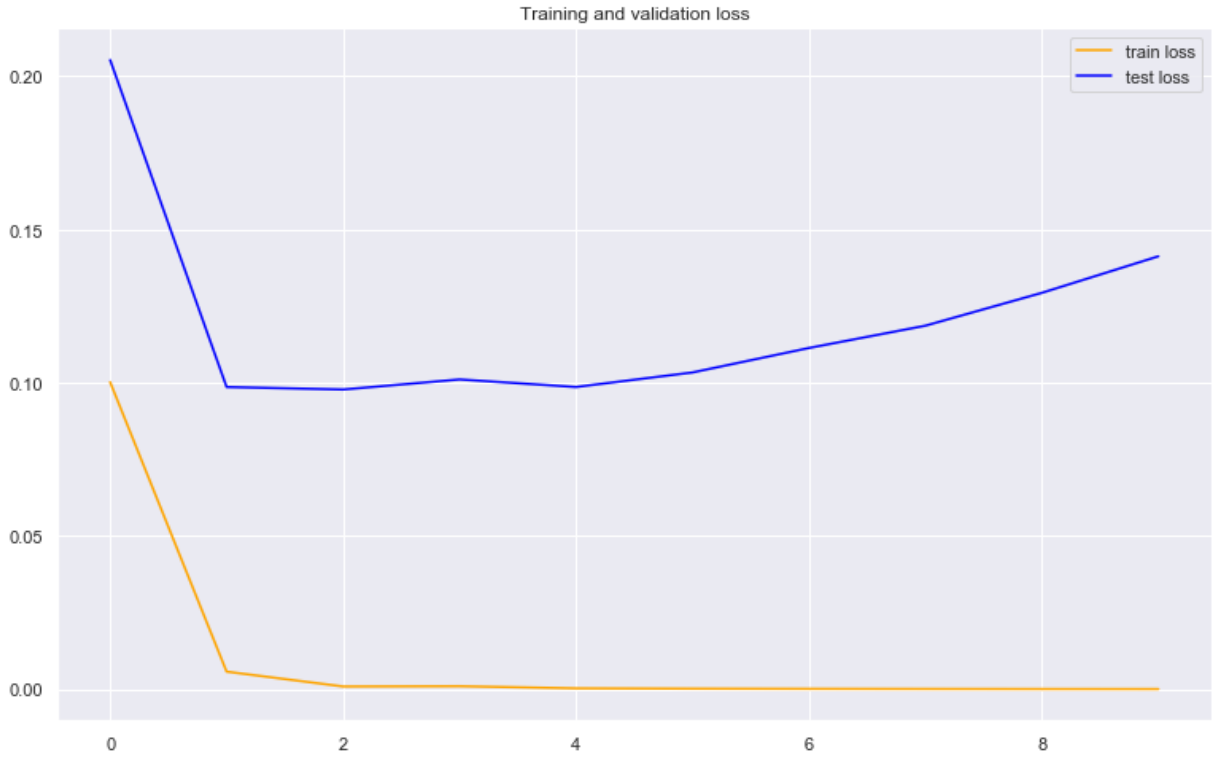

Figure 10: Training and Validation loss

An accuracy graph will provide the insight into how algorithm performs. The model accuracy is determined after model is trained it is measure of model prediction against its test or true data and calculated in percentage form. As we can see from the accuracy graph our accuracy is close to 97 percent

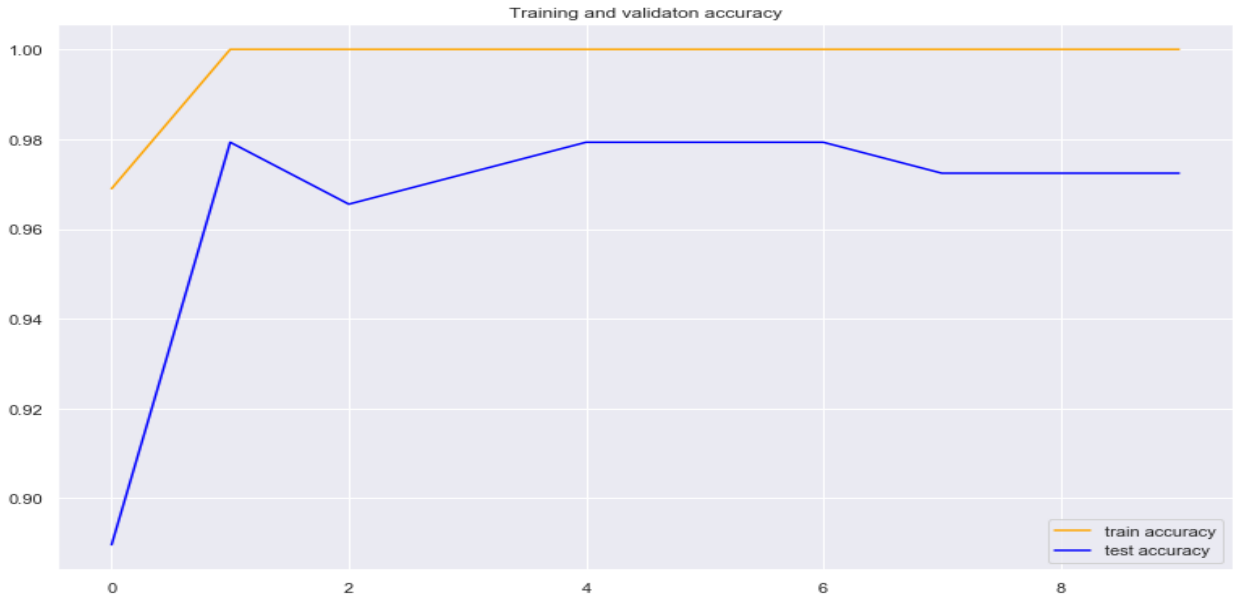

Figure 11: Training and validation Accuracy

Conclusion

In this project we have used NLP text processing and vectorization models to pre process our text data so that we can use this text for classifying it into positive and negative sentiment. The model proposed is being tested for various performance measures which consist of accuracy, Precision, recall and F1 score. The accuracy ranges from 66 to 97 for all the training models. LSTM model with convoluted learning has outperformed all the other models

Future Works

As we have seen the resultant curve for training loss and accuracy showed overfitting of data we can overcome this using cross validating algorithm. We can also implement hashing vectorization which would improve results drastically. We would perform further testing on datasets with larger values to see if we can further improve the results from this project.

References

http://www.uvm.edu/pdodds/files/papers/others/2007/godbole2007a.pdf
https://www.kaggle.com/harishcscode/all-news-articles-from-home-page-media-house

Appendices

```
# -*- coding: utf-8 -*-
"""
Created on Sat Apr  1 13:09:37 2020

@author: madhuri
"""
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn import model_selection, naive_bayes, svm
#loading the dataset
df = pd.read_csv('CNN_news.csv')
#checking and cleaning the mising values
df.isnull().sum()
df= df.dropna()
#viewing the dataset
df.head(10)
df.columns

#Calculating word count for each row for text column
df['Wordcount'] = df['TEXT'].apply(lambda x: len(str(x).split(" ")))
df[['TEXT','Wordcount']].head()

#descriptive statistics for the word count and its visualization
df.Wordcount.describe()
plt.figure(figsize=(12.8,6))
sns.distplot(df['Wordcount']).set_title('Wordcount Distribution');
#calculate news text length column and plot news text length
df['News Text Length'] = df['TEXT'].str.len()
plt.figure(figsize=(12.8,6))
sns.distplot(df['News Text Length']).set_title('News Length Text Distribution');
df['News Text Length'].describe()
#find common words
Common_Words = pd.Series(' '.join(df['TEXT']).split()).value_counts()[:20]
Common_Words

#find uncommom words
Uncommon_words =  pd.Series(' '.join(df ['TEXT']).split()).value_counts()[-20:]
Uncommon_words



# Libraries for text preprocessing
```

```
import re
import nltk
#nltk.download('stopwords')
from nltk.corpus import stopwords
#nltk.download('wordnet')
from nltk.stem.porter import PorterStemmer
from nltk.tokenize import RegexpTokenizer
from nltk.stem.wordnet import WordNetLemmatizer
#stop words
Stopwords = set(stopwords.words("english"))
#adding more words to stop words
##Creating a list of custom stopwords
addNewStopWords =
['said','say','also','could','even','get','would','like','one','caption','may','much','go','make','come','take
'
                ,'know','well','really','much','two','must','ago','new','many','say','way','told']
Stopwords = Stopwords.union(addNewStopWords)
print(Stopwords)
#remove possive pronoun
df['TEXT'] = df['TEXT'].str.replace("'s", "")
corpus = []
for i in range(0, 3847):
    #Remove punctuations
    text = re.sub('[^a-zA-Z]', ' ', df['TEXT'][i])
    #Convert to lowercase
    text = text.lower()
    #remove tags
    text=re.sub("&lt;/?.*?&gt;"," &lt;&gt; ",text)

    # remove special characters and digits
    text=re.sub("(\\d|\\W)+"," ",text)

    ##Convert to list from string
    text = text.split()

    ##Stemming
    ps=PorterStemmer()
    #Lemmatisation
    lemmen = WordNetLemmatizer()
    text = [lemmen.lemmatize(word) for word in text if not word in
          Stopwords]
    text = " ".join(text)
    corpus.append(text)

#Generating Word cloud
#!pip install wordcloud
```

```
from wordcloud import WordCloud, STOPWORDS, ImageColorGenerator
import matplotlib.pyplot as plt
wordcloud = WordCloud(
                      background_color='white',
                      stopwords=Stopwords,
                      max_words=100,
                      max_font_size=50,
                      random_state=42
                     ).generate(str(corpus))
print(wordcloud)
fig = plt.figure(1)
plt.imshow(wordcloud)
plt.axis('off')
plt.show()
fig.savefig("word1.png", dpi=900)
#adding changes in text to original data
df['New Text'] = corpus


#feature extraction
#count vectorization and transformation
from sklearn.feature_extraction.text import CountVectorizer
import re
cv=CountVectorizer(max_df=0.8,stop_words=Stopwords, max_features=10000,
ngram_range=(1,3))
X=cv.fit_transform(corpus)
X.shape
list(cv.vocabulary_.keys())[:10]

#visualiztion of words with top unigram, bigram and trigram
#frequently occuring words that is common in the text column
def Unigram_n_words(corpus, n=None):
    vec1 = CountVectorizer().fit(corpus)
    bag_of_words = vec1.transform(corpus)
    sum_words1 = bag_of_words.sum(axis=0)
    words_freqs = [(word, sum_words1[0, idx]) for word, idx in
                vec1.vocabulary_.items()]
    words_freqs =sorted(words_freqs, key = lambda x: x[1],
                    reverse=True)
    return words_freqs[:n]
#put common words in pandas data frame
words = Unigram_n_words(corpus, n=20)
df_uni = pd.DataFrame(words)
df_uni.columns=["Word", "Freq"]

#visulalize the bar plot using seaborn library - unigram grams
```

```
sns.set(rc={'figure.figsize':(13,8)})
g = sns.barplot(x="Word", y="Freq", data=df_uni)
g.set_xticklabels(g.get_xticklabels(), rotation=30)

#visulalize the bar plot using seaborn library - bi grams
def bigram_words(corpus, n=None):
    vec2 = CountVectorizer(ngram_range=(2,2),
           max_features=2000).fit(corpus)
    bag_of_words = vec2.transform(corpus)
    sum_words1 = bag_of_words.sum(axis=0)
    words_freqs = [(word, sum_words1[0, idx]) for word, idx in
                vec2.vocabulary_.items()]
    words_freqs =sorted(words_freqs, key = lambda x: x[1],
               reverse=True)
    return words_freqs[:n]
top_bigram_words = bigram_words(corpus, n=20)
df_bigram = pd.DataFrame(top_bigram_words)
df_bigram.columns=["Bi-gram", "Freq"]
print(df_bigram)
#Barplot of most freq Bi-grams
sns.set(rc={'figure.figsize':(13,8)})
h=sns.barplot(x="Bi-gram", y="Freq", data=df_bigram)
h.set_xticklabels(h.get_xticklabels(), rotation=45)


#visulalize the bar plot using seaborn library - tri grams
def trigram_words(corpus, n=None):
    vec1 = CountVectorizer(ngram_range=(3,3),
           max_features=2000).fit(corpus)
    bag_of_words = vec1.transform(corpus)
    sum_words = bag_of_words.sum(axis=0)
    words_freqs = [(word, sum_words[0, idx]) for word, idx in
                vec1.vocabulary_.items()]
    words_freqs =sorted(words_freqs, key = lambda x: x[1],
               reverse=True)
    return words_freqs[:n]
top3_words = trigram_words(corpus, n=20)
df_trigram = pd.DataFrame(top3_words)
df_trigram.columns=["Tri-gram", "Freq"]
print(df_trigram)
#Barplot of most freq Tri-grams
sns.set(rc={'figure.figsize':(13,8)})
j=sns.barplot(x="Tri-gram", y="Freq", data=df_trigram)
j.set_xticklabels(j.get_xticklabels(), rotation=45)
```

```
#convert word matrix to integer
from sklearn.feature_extraction.text import TfidfTransformer
cov_tfidf_transformer=TfidfTransformer(smooth_idf=True,use_idf=True)
cov_tfidf_transformer.fit(X)
# get feature names
feature_names=cv.get_feature_names()
# fetch document for which keywords needs to be extracted
Text_words=corpus[532]
#generate tf-idf for the given document
tf_idf_vector=cov_tfidf_transformer.transform(cv.transform([Text_words]))
tf_idf_vector.shape
print(tf_idf_vector)
#based on higest td-idf score we can extract higest scores
#Function for sorting tf_idf in descending order
from scipy.sparse import coo_matrix
def sort_coo_matrix(coo_matrix):
    values = zip(coo_matrix.col, coo_matrix.data)
    return sorted(values, key=lambda x: (x[1], x[0]), reverse=True)

def feature_from_vector(feature_names, sorted_items, topn=10):


    #use only topn items from vector
    sorted_items = sorted_items[:topn]

    score_vals = []
    feature_vals = []

    # word index and corresponding tf-idf score
    for idx, score in sorted_items:

        #keep track of feature name and its corresponding score
        score_vals.append(round(score, 3))
        feature_vals.append(feature_names[idx])

    #create a tuples of feature,score
    #results = zip(feature_vals,score_vals)
    results= {}
    for idx in range(len(feature_vals)):
        results[feature_vals[idx]]=score_vals[idx]

    return results

#sort the tf-idf vectors by descending order of scores
sorted_items=sort_coo_matrix(tf_idf_vector.tocoo())
```

```
#extract only the top n; n here is 10
Feautured_Words=feature_from_vector(feature_names,sorted_items,25)

# now print the features with their probalility
print("\nText:")
print(doc)
print("\n Featured Word Score:")
for f in Feautured_Words:
   print(f,Feautured_Words[f])


   #find higest weight of particular word
from sklearn.feature_extraction.text import TfidfVectorizer
tfidf_vector = TfidfVectorizer()
tfidf_vector.fit(df['New Text'])

X_transformation = tfidf_vector.transform(df['New Text'])
#text at loc[1]
df['New Text'][1]
#find importance of words using vectorization method
print([X_transformation[1,tfidf_vector.vocabulary_['find']]])

#check subjectivity and polarity of news text
from textblob import TextBlob
from textblob.sentiments import NaiveBayesAnalyzer
polarity=[]
subjectivity=[]
df['Polarity']=df.TEXT.apply(lambda x: TextBlob(x).sentiment.polarity)
df['Subj']=df.TEXT.apply(lambda x: TextBlob(x).sentiment.subjectivity)
df.head(10)
#Create 2 arrays
polarity=[]
subj=[]

#Get polarity and sentiment for each row and put it in either polarity or sentiment
for t in df.TEXT:
   tx=TextBlob(t)
   polarity.append(tx.sentiment.polarity)
   subj.append(tx.sentiment.subjectivity)
   blob = TextBlob(t, analyzer=NaiveBayesAnalyzer())
   #sentiment classification for the text
   blob.sentiment #Sentiment(classification='pos', p_pos=1.0, p_neg=3.633586025330017e-19)
#Put in dataframe polsubj which has a column of polarity values and a column of subjectivity
values
polsubj = pd.DataFrame({'polarity': polarity,'subjectivity': subj})
#Plot the line graph
```

```
polsubj.plot(title='Polarity and Subjectivity')

#sentiment analysis
#!pip install vaderSentiment
from vaderSentiment.vaderSentiment import SentimentIntensityAnalyzer
analyser = SentimentIntensityAnalyzer()

#sentiment analyser
df['Sentiment Analyser'] = df.TEXT.apply(lambda x:analyser.polarity_scores(x))
#create a different column of all scores which are present as json array
 k=df['Sentiment Analyser'].apply(pd.DataFrame,index=[0]).tolist()
final_df = pd.concat(k)
final_df.index = pd.Series(final_df.index).shift(-1).fillna(0).cumsum()
#add only compound score to our original data set

df = pd.DataFrame(df)
df1=pd.DataFrame(final_df)
#add score to original data frame
df['Compound Score'] = df1['compound'].tolist()

#function to determine category of sentiment
def Category_function(x):
    if x> 0.05 :
        return "Positive"
    elif x > -0.05 and x <0.05:
        return "Neutral"
    elif x <= -0.05:
        return "Negative"

 df['Sentiment_Category']= df['Compound Score'].apply(lambda x: Category_function(x) )
#Sentiment_Category is converted to category type
 df['Sentiment_Category'] = df['Sentiment_Category'].astype('category')
df.dtypes
#add a column to have numerical columns to category
 df['Sentiment_Category_Code'] = df['Sentiment_Category'].cat.codes
Sentiment_Category_Code= df['Sentiment_Category_Code']

 df['Sentiment_Category_code_1']= df['Sentiment_Category_Code'].apply(lambda x: 1 if x>=1
else 0)
Sentiment_Category_code_1=df['Sentiment_Category_code_1']
#naive bayes
from sklearn.model_selection import train_test_split
X_train1, X_test1, y_train1, y_test1 = train_test_split(X, Sentiment_Category_code_1,
test_size=0.3)
from sklearn.naive_bayes import MultinomialNB
clf=MultinomialNB()
```

```
clf.fit(X_train1, y_train1)
print (clf.score(X_train1, y_train1))
print (clf.score(X_test1, y_test1))
predicted_result=clf.predict(X_test1)
from sklearn.metrics import classification_report, confusion_matrix, accuracy_score
print(classification_report(y_test1,predicted_result))
Naive_bayes_score = accuracy_score(y_test1, predicted_result)
Naive_bayes_score
#accuracy score 0.82
#Using random forest
from sklearn.ensemble import RandomForestClassifier
rf = RandomForestClassifier()
rf.fit(X_train1, y_train1)

# calculating the random forest accuracy
y_pred_rf = rf.predict(X_test1)
random_forest_score = accuracy_score(y_test1, y_pred_rf)
#accuracy 0.749
random_forest_score
print(classification_report(y_test1,y_pred_rf))

#SVM
from sklearn import svm
svc = svm.SVC()
svc.fit(X_train1, y_train1)

#calculating the SVM accyracy
y_pred_svm = svc.predict(X_test1)
svc_score = accuracy_score(y_test1, y_pred_svm)
svc_score
#0.6632

#lstm
from sklearn.feature_extraction.text import CountVectorizer
from keras.preprocessing.text import Tokenizer
from keras.preprocessing.sequence import pad_sequences
from keras.models import Sequential
from keras.layers import Dense, Embedding, LSTM
from sklearn.model_selection import train_test_split
from keras.utils.np_utils import to_categorical
import re

max_fatures = 30000
tokenizer = Tokenizer(nb_words=max_fatures, split=' ')
tokenizer.fit_on_texts(df['New Text'].values)
X2 = tokenizer.texts_to_sequences(df['New Text'].values)
```

```
X2 = pad_sequences(X2)
Y2 = pd.get_dummies(df['Sentiment_Category_code_1']).values
X2_train, X2_test, Y2_train, Y2_test = train_test_split(X2,Y2, random_state = 42)
print(X2_train.shape,Y2_train.shape)
print(X2_test.shape,Y2_test.shape)


embed_dim = 150
lstm_out = 200
model = Sequential()
model.add(Embedding(max_fatures, embed_dim,input_length = X2.shape[1], dropout=0.2))
model.add(LSTM(lstm_out, dropout_U=0.2,dropout_W=0.2))
model.add(Dense(2,activation='softmax'))
model.compile(loss = 'categorical_crossentropy', optimizer='adam',metrics = ['accuracy'])
prssint(model.summary())

history=model.fit(X2_train, Y2_train, epochs=10, batch_size=250, validation_split=0.2)
#plot loss function
loss = history.history['loss']
val_loss = history.history['val_loss']
epochs = range(len(loss))
plt.plot(epochs, loss, 'orange', label='train loss')
plt.plot(epochs, val_loss, 'blue', label='test loss')
plt.title("Training and validation loss")
plt.legend()
#plot accuracy
acc = history.history['accuracy']
val_acc = history.history['val_accuracy']

plt.plot(epochs, acc, 'orange', label='train accuracy')
plt.plot(epochs, val_acc, 'blue', label='test accuracy')
plt.title("Training and validaton accuracy")
plt.legend()

# print the model summary
print (model.summary())

# test the model with pretrained weights
scores = model.evaluate(X2_test, Y2_test, verbose=1)

print("Accuracy: %.2f%%" % (scores[1]*100))

score,acc = model.evaluate(X2_test, Y2_test, verbose = 2, batch_size = 10)
print("score: %.2f" % (score))
print("acc: %.2f" % (acc))
```