Northeastern University

NEWS CLASSIFICATION & SENTIMENT ANALYSIS

Project Report

CSYE 7245: Big-Data Systems & Intelligence Analytics

April 28th, 2018

Author		**Supervisor**
Eklavya Saxena	001850025	Prof. Nicholas W. Brown
Ankur Jain	001206900	
Amandeep Singh	001271649	

Table of Contents

Introduction

Around the globe, there are more than 6500 'daily' newspapers, selling close to 400 million copies every day. Additionally, there are blogs, micro blogs, periodicals, magazines, fanzines etc. How to make sense of all this information? How to classify it and aggregate it so that quantitative analysis can be performed? This project explores one possible answer to these questions: **classification of news articles by sentiment and topic**.

The vision is to create the capability to track how sentiment on a topic has evolved over time, how different news outlets cover the same topic and, in the limit, to be able to predict future behavior through sentiment trends.

Data Sources

Majority of the data is web-scrapped from the following:

1. CNN (https://www.cnn.com)
2. BBC (www.bbc.com)
3. The Guardian (https://www.theguardian.com)
4. Infowars (https://www.infowars.com)
5. Fox News (www.foxnews.com)
6. NBC News (https://www.nbcnews.com)
7. Washington Post (https://www.washingtonpost.com)
8. The Onion (https://www.theonion.com)
9. ESPN (www.espn.com)
10. NDTV (https://www.ndtv.com)
11. Alzazeera (https://www.aljazeera.com)
12. Zee News (http://zeenews.india.com)
13. ABC News (http://abcnews.go.com)
14. India Today (https://www.indiatoday.in)
15. Bloomberg (https://www.bloomberg.com)

Approach

Following is the process flow chart to approach this research project:

News Scrapping

For scrapping, a special python library '*Article*' is used, which extracts the author name, content, title, category, URL of an article. The following JSON file is passed and looped through to get the URL of the news website into consideration.

```
{
  "cnn": {
    "link": "http://edition.cnn.com/"
  },
  "bbc": {
    "link": "http://www.bbc.com/"
  },
  "theguardian": {
    "rss": "https://www.theguardian.com/uk/rss",
    "link": "https://www.theguardian.com/us"
  },
  "breitbart": {
    "link": "http://www.breitbart.com/"
  },
  "infowars": {
    "link": "https://www.infowars.com/"
  },
  "foxnews": {
    "link": "http://www.foxnews.com/"
```

Important Libraries

```
#!pip install feedparser
#!pip install newspaper3k

import feedparser as fp
import numpy as np
import json
import newspaper
from newspaper import Article
from time import mktime
from datetime import datetime
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
%matplotlib inline
import csv
```

Importing JSON File

Importing the mentioned JSON file and setting up the limit to download 10^9 articles:

```
# Set the limit for number of articles to download
LIMIT = 1000000000
articles_array = []

data = {}
data['newspapers'] = {}

# Loads the JSON files with news sites
with open('NewsPapers.json') as data_file:
    companies = json.load(data_file)
```

```
    article['title'] = content.title
    article['text'] = content.text
    article['authors'] = content.authors
    article['top_image'] =  content.top_image
    article['movies'] = content.movies
    newsPaper['articles'].append(article)
    articles_array.append(article)
    print(count, "articles downloaded from", company, ", url: ", entry.link)
    count = count + 1
```

Exporting to CSV Post-download

Once the articles are downloaded, exporting to a CSV file:

```
#Finally it saves the articles as a CSV-file.
try:
    f = csv.writer(open('Scraped_data_news_output.csv', 'w', encoding='utf-8'))
    f.writerow(['Title', 'Authors','Text','Image','Videos','Link','Published_Date'])
    #print(article)
    for artist_name in articles_array:
        title = artist_name['title']
        authors=artist_name['authors']
        text=artist_name['text']
        image=artist_name['top_image']
        video=artist_name['movies']
        link=artist_name['link']
        publish_date=artist_name['published']
        # Add each artist's name and associated link to a row
        f.writerow([title, authors, text, image, video, link, publish_date])
except Exception as e: print(e)
```

Dataset Information and Description

Libraries Involved

Name of the Library	Description
import re	For regular expression
import nltk	A natural language toolkit
import string	For common string operations
import pandas	For data manipulation & analysis
import numpy	For scientific computing
import seaborn	A visualization library
import matplotlib.pyplot	2D plotting
from textblob import TextBlob	For simplified text processing
from newspaper import Article	For extracting & curating newspaper articles
from collections import Counter	Implements dict subclass for counting hashable objects (a specialized container datatypes)
from nltk.corpus import stopwords	This nltk module contains a list of stop words like 'the', 'is', 'are'
from stop_words import get_stop_words	Get list of common stop words in various languages
from nltk.tokenize import RegexpTokenizer	A tokenizer helps to divide a string into substrings by splitting on the specified string (defined in subclasses). A regexptokenizer splits a string into substrings using a regular expression
from nltk import sent_tokenize, word_tokenize	sent_tokenize - Return a sentence-tokenized copy of text, using NLTK's recommended sentence tokenizer. word_tokenize - Return a tokenized copy of text, using NLTK's recommended word tokenizer
from sklearn.naive_bayes import MultinomialNB	Naive Bayes classifier for multinomial models
from wordcloud import WordCloud, STOPWORDS	WorldCloud - A little word cloud generator. STOPWORDS - The build-in list of words to be eliminated
from sklearn.neural_network import MLPClassifier	A Multi-layer Perceptron classifier
from sklearn.model_selection import train_test_split	Split arrays or matrices into random train and test subsets
from sklearn.feature_extraction.text import CountVectorizer	Convert a collection of text documents to a matrix of token counts

from sklearn.metrics import confusion_matrix, classification_report	confusion_matrix - Compute confusion matrix to evaluate the accuracy of a classification. classification_report - Build a text report showing the main classification metrics

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 6243 entries, 0 to 6242
Data columns (total 6 columns):
index        6243 non-null int64
TITLE        6243 non-null object
PUBLISHER    6243 non-null object
Content      6243 non-null object
Author       6243 non-null object
Category     6243 non-null object
dtypes: int64(1), object(5)
memory usage: 292.7+ KB
```

The downloaded dataset has 6243 rows and 6 columns. It has no null values and columns are textual. The below picture describes the first 5 rows of the dataset:

```
df.head()
```

	index	TITLE	PUBLISHER	Content	Author	Category
0	1	Fed's Charles Plosser sees high bar for change...	Livemint	Paris/London/Atlanta: Federal Reserve Bank of ...	['Mark Deen']	Business
1	13	ECB FOCUS-Stronger euro drowns out ECB's messa...	Reuters	FRANKFURT, March 10 (Reuters) - The European C...	['Reuters Editorial']	Business
2	28	Forex Market: EUR/USD retreats from 2-1/2-year...	Binary Tribune	The euro retreated from highs unseen since Oct...	['Vladimir Manev']	Business
3	38	ECB's Noyer: Low inflation may hamper adjustment	MarketWatch	PARIS--The slow pace of price increases and no...	['Gabriele Parussini']	Business
4	45	ECB Unlocks Door for Further Euro Strength - W...	DailyFX	Fundamental Forecast for Euro: Neutral\r\n\r\n...	['Christopher Vecchio']	Business

Stemming and Lemmatization of Title and Content Column

Stemming

It is the process of reducing inflected (or sometimes derived) words to their word stem, base or root form—generally a written word form. The stem need not be identical to the morphological root of the word.

Lemmatization

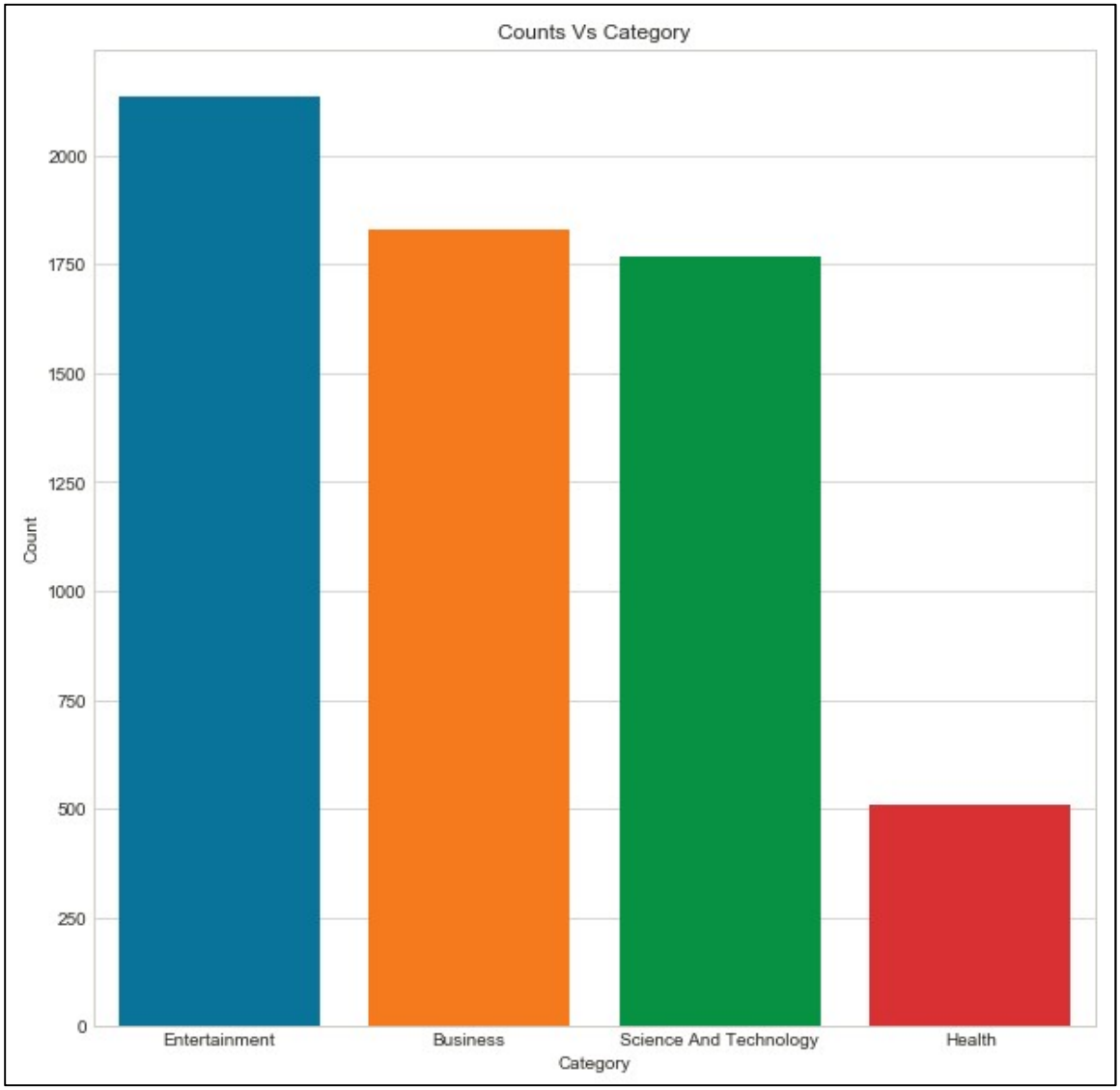
In linguistics, it is the process of grouping together the inflected forms of a word so they can be analyzed as a single item, identified by the word's lemma, or dictionary form.

```
w_tokenizer = nltk.tokenize.WhitespaceTokenizer()
lemmatizer = nltk.stem.WordNetLemmatizer()
ps = PorterStemmer()


def lemmatize_text(text):
    return [lemmatizer.lemmatize(w) for w in w_tokenizer.tokenize(text)]

df['Title_lemmatized'] = df.TITLE.apply(lemmatize_text)
df['Content_lemmatized'] = df.Content.apply(lemmatize_text)

lemmatization_title=df['Title_lemmatized']
lemmatization_content=df['Content_lemmatized']

def title_stemming(text):
    l = []
    for i in text:
        for j in i:
            #converts list item to lower case
            p=j.lower()
            # removes punctuation,numbers and returns list of words
            q=re.sub('[^A-Za-z]+', ' ', p)
            l.append(ps.stem(q))
    return l
```

News Categories

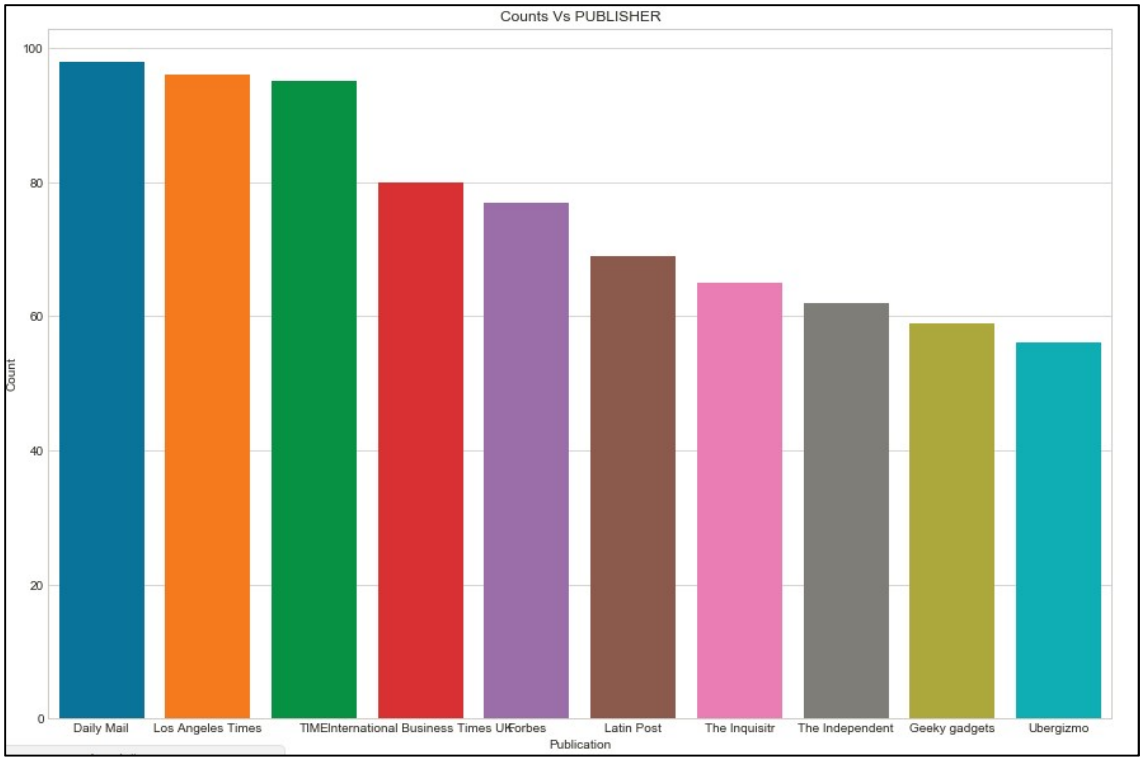
- Entertainment
- Science and Technology
- Business
- Health

The dataset has more news in ‘Entertainment’ category:



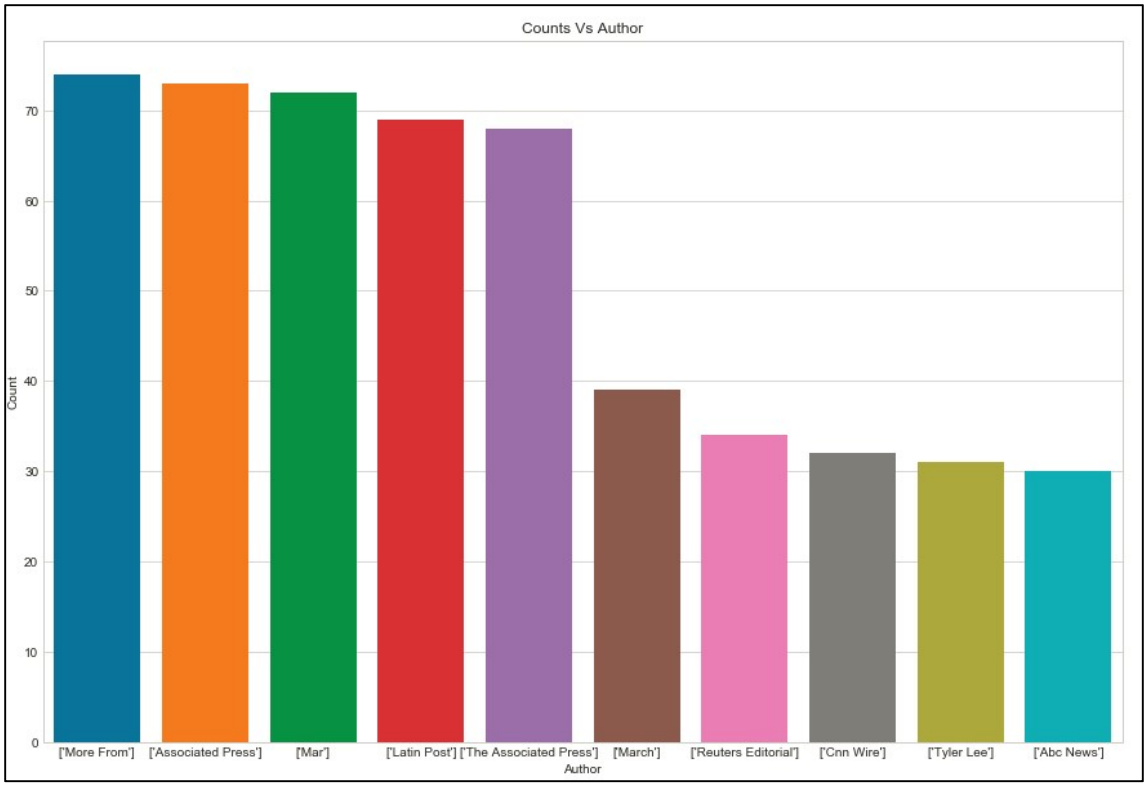
Publishers

The bar plot shows the number of articles released by the publishers, displayed in descending order:



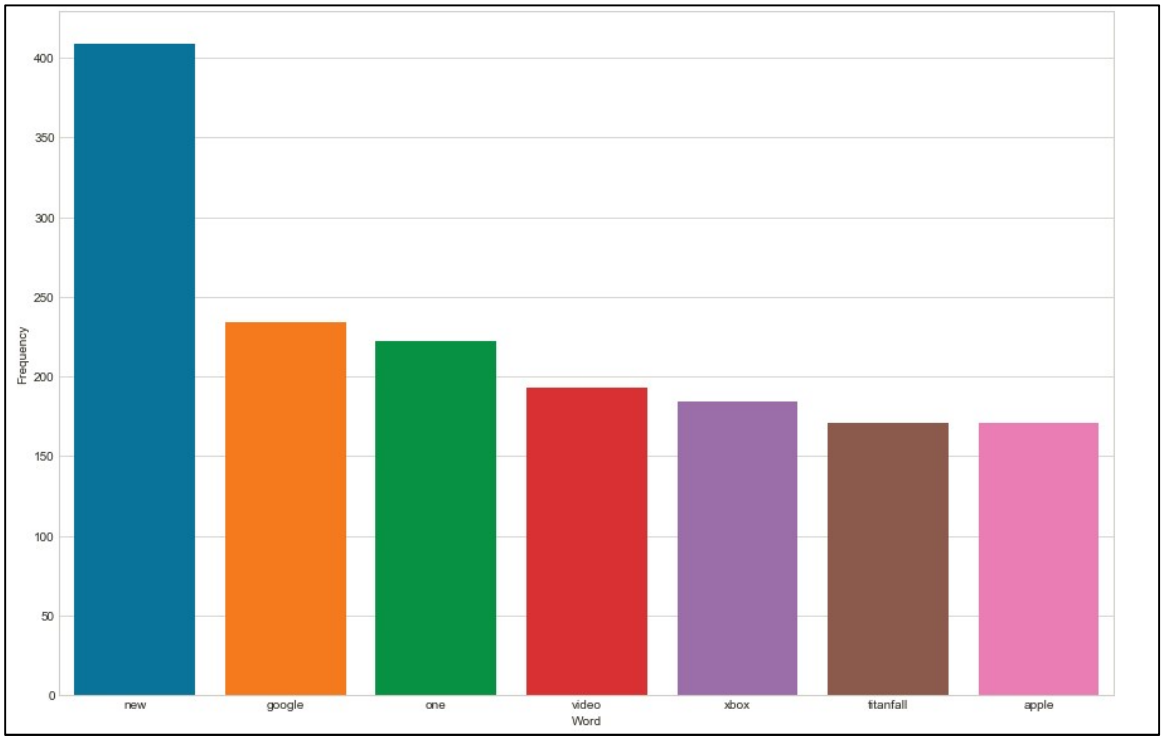
Authors

On visualization, following are the number of the articles published by an author:



Frequency of the Word in Title/Content Column

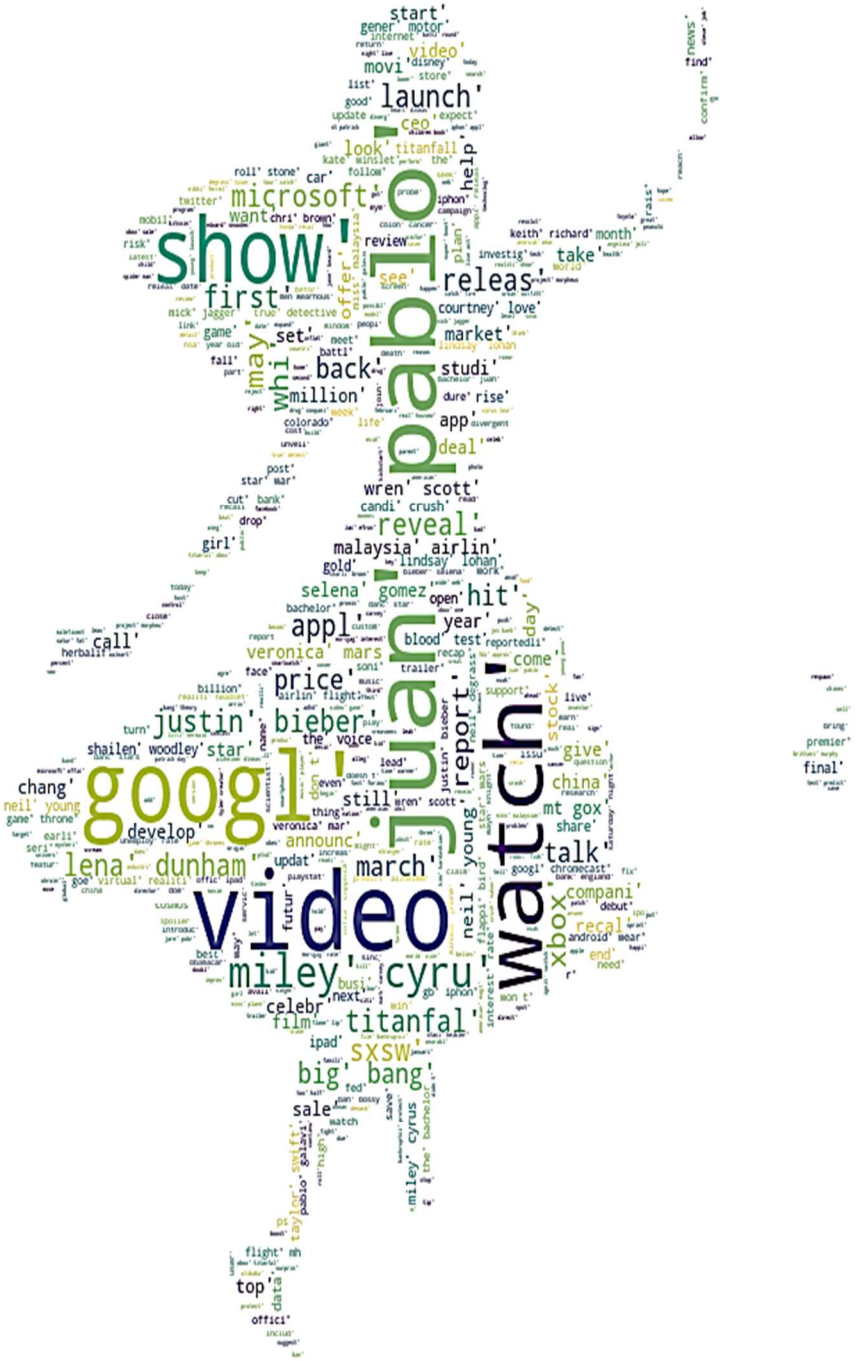
After eliminating STOPWORDS from the 'Title' column, the below figure shows the frequency of the remaining words:



Word Clouds

Word Cloud of the 'Title' Column

The below figure shows the most frequent occurring words in the 'Title' column:



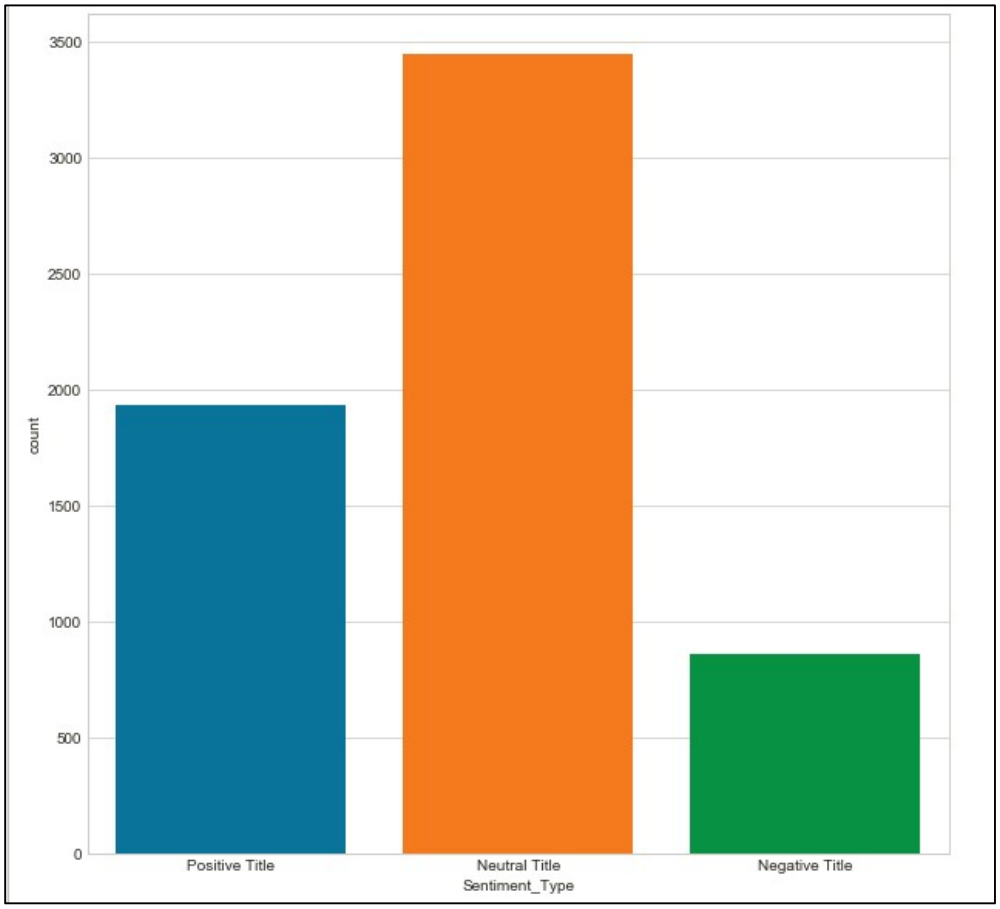
Word Cloud of the 'Content' Column

The below figure shows the most frequent occurring words in the 'Content' column:

Categorization

Categorization of the 'Title' column into sentiments based on its sentiment value. If sentiment value is greater than 0 then it is a positive sentiment.

	TITLE	sentiment	polarity	Sentiment_Type
0	Fed's Charles Plosser sees high bar for change...	0.160000	0.540000	Positive Title
1	ECB FOCUS-Stronger euro drowns out ECB's messa...	0.000000	0.300000	Neutral Title
2	Forex Market: EUR/USD retreats from 2-1/2-year...	0.000000	0.000000	Neutral Title
3	ECB's Noyer: Low inflation may hamper adjustment	0.000000	0.300000	Neutral Title
4	ECB Unlocks Door for Further Euro Strength - W...	0.000000	0.500000	Neutral Title
5	Both ways	0.000000	0.000000	Neutral Title
6	Carl Icahn Rift Hurts eBay (EBAY)	0.000000	0.000000	Neutral Title
7	EBay rejects Icahn board nominees, asks invest...	0.000000	0.125000	Neutral Title
8	Icahn Targets Ebay Chief Donahoe After Company...	0.000000	0.000000	Neutral Title
9	EBay Asks Shareholders To Vote Against PayPal ...	0.000000	0.000000	Neutral Title



In the dataset, there are mostly Neutral Titles:

No. of Neutral Titles > No. of Positive Titles > No. of Negative Titles

Algorithms and Error Metrics

To predict the sentiments - Multinomial Naïve Bayes, Multi-Layer Perceptron and XG Boost algorithms are used. The model's trained and sentiments (Positive/Negative) of the 'Title' column has been predicted.

The below figure shows the error metrics to predict the sentiments of the title using the above models:

	Naive Bayes_Model	MLP_Model	XGB_Model
Metrics_Train	0.97316	1.00000	0.86847
Metrics_Test	0.85101	0.85752	0.82984

Since, the Naïve Bayes and MLP model overfits, the best model for this dataset is XGB despite less accuracy as compared to other models.

Validation of the Sentiments of the 'Title' column

```
title_positive=df['TITLE'][0]
title_positive
```

```
"Fed's Charles Plosser sees high bar for change in pace of tapering"
```

First, I want to test with the positive title. I have chosen the above title and its sentiment is Positive. After evaluating it should predict the sentiment as Positive.

```
title_positive_transformed = bow_transformer.transform([title_positive])
nb.predict(title_positive_transformed)[0]
```

```
'Positive Title'
```

Second, I want to test with the negative title. I have chosen the below title and its sentiment is negative. After evaluating it should predict the sentiment as Negative.

```
title_negative=df['TITLE'][14]
title_negative
```

```
"$5 20-piece chicken nuggets didn't help McDonald's reverse US sales decline"
```

```
title_negative_transformed = bow_transformer.transform([title_negative])
nb.predict(title_negative_transformed)[0]
```

```
'Negative Title'
```

To predict the category, same models as above i.e. Multinomial Naïve Bayes, Multi-Layer Perceptron and XG Boost are used.

The below figure shows the error metrics to predict the category of the news using the above models:

	Naive Bayes_Model	XGB_Model	MLP_Model
Metrics_Train	1.00000	0.65365	0.99848
Metrics_Test	0.85752	0.58993	0.85637

In this prediction too, the best model is XGB because other models overfit.

Validation of the News Category

The below News Title comes under Business News Category. If the output gives 1 then it is predicted successfully.

```
Category_business=df_news_business_entertainment['TITLE'][0]
Category_business
```

```
"Fed's Charles Plosser sees high bar for change in pace of tapering"
```

```
Category_business_transformed = bow_transformer_category.transform([Category_business])
nb.predict(Category_business_transformed)[0]
```

```
1
```

The below News Title comes under Entertainment News Category. If the output gives 0 then it is predicted successfully.

```
category_entertainment=df_news_business_entertainment['TITLE'][6242]
category_entertainment
```

```
"REPORT: Older women far more likely to get Alzheimer's than breast cancer"
```

```
category_entertainment_transformed = bow_transformer_category.transform([category_entertainment])
nb.predict(category_entertainment_transformed)[0]
```

```
0
```

Model Deployment

Pickle

Pickle is the standard way of serializing objects in Python. Pickle operation is used to serialize machine learning algorithms and save the serialized format to a file. Later, this file can be loaded to de-serialize the model to make new predictions.

```
# save the model to disk
filename = 'finalized_model_big_data.pkl'
pickle.dump(pickle_models, open(filename, 'wb'))
```

Upload to AWS S3

Once, the pickle script is written, pickle file is uploaded on Amazon S3 using below script:

```
#function to upload the saved file on the cloud
def uploadToS3(destinationPath, filePath, arg_AWSuser, arg_AWSpass):

    AWS_ACCESS_KEY_ID = arg_AWSuser
    AWS_SECRET_ACCESS_KEY = arg_AWSpass

    bucket_name = 'bigdatamodeldevelopmentdeployment'
    conn = boto.connect_s3(AWS_ACCESS_KEY_ID,AWS_SECRET_ACCESS_KEY)

    bucket = conn.create_bucket(bucket_name,location=boto.s3.connection.Location.DEFAULT)

    testfile = filePath
    print ('Uploading '+testfile+' to Amazon S3 bucket '+bucket_name)
    def percent_cb(complete, total):
        sys.stdout.write('.')
        sys.stdout.flush()

    k = Key(bucket)
    k.key = destinationPath+"/"+testfile
    k.set_contents_from_filename(testfile,cb=percent_cb, num_cb=10)
```

Docker and GIT

Docker

It is a computer program that performs operating-system-level virtualization. Docker image automates the task and makes it OS independent. The below snippet is of the docker file. The link to the docker image created can be accessed via:

https://hub.docker.com/r/ankkur13/bigdatadockerimage/

```
# Use the basic Python 3 image as launching point
FROM python:3.6.3

# Add the script or text to the Dockerfile
ADD model_deployment_script.py /home
# ADD requirements.txt /home
ADD Scrapped_content.csv /home
ADD argv_input_syntax.txt /home

# Install required Libraries
# RUN pip install -r ./home/requirements.txt
RUN pip install numpy
RUN pip install pandas
RUN pip install seaborn
RUN pip install sklearn
RUN pip install scipy
RUN pip install sklearn
RUN pip install nltk
#RUN pip install string
#RUN pip install re
RUN pip install stop_words
#RUN pip install collections
RUN pip install wordcloud
RUN pip install textblob
#RUN pip install xgboost
#RUN pip install PIL
#RUN pip install pickle
RUN pip install boto
```

GIT

Implementation and documentation of the research project can be accessed via:

https://github.com/ankkur13/Big-Data-Systems-and-Intelligence-Analytics

References

- http://newspaper.readthedocs.io/en/latest/
- https://www.geeksforgeeks.org/newspaper-article-scraping-curation-python/
- https://www.smallsurething.com/web-scraping-article-extraction-and-sentiment-analysis-with-scrapy-goose-and-textblob/
- https://github.com/nikbearbrown