CSE 587 - Lab 3 - Spring 2019

Analyze News Headlines and Predict Article Success

An implementation on Apache Spark using MLLib and PySpark

Shwetasree Chowdhury 50296995

Sowmith Nallu 50286932

Table of Contents:

Abstract:

We observe that having sensational headlines for an articles intrigues our interest to read. But do the headlines really matter for the article to become popular? We aim to answer this question in our project.

In this project we use word2vec model to create word and title embedding and calculate the sentiment analysis scores of the article titles, visualize the relationship between title sentiment and article popularity, and then attempt to predict the article popularity from the embeddings and other available features.

Apache Spark is advantageous for text analysis because it provides a platform for scalable, distributed computing and is faster when dealing with JSON objects and provides fast computing using intermediate caching. We collect the data from News API and NY Times API.

Why Spark?
Spark is a general-purpose distributed data processing engine that is suitable for use in a wide range of circumstances.There are libraries for SQL, machine learning, graph computation, and stream processing, which can be used together in an application, reason why we chose to utilise Spark for our project.

Spark also provides immense scalability and speed when it comes to analysing the petabytes of data that we would need to process in order to come to a fair prediction

What are we trying to achieve?
We aim to correlate and understand how a certain news article and its corresponding headline add to the article's popularity. In other words, what poignant key words can be used in a headlines articles so as to ensure, it gets maximum views and subsequent shares.
For the purpose of this project, we aim to use Sentiment analysis on the article headline, which can also be solved using Spark Streaming (for real time data), apart from pyspark and utilising word2vec package of MLLib on the article contents to achieve our desired result.

When do we use Spark?
Social media is a mammoth platform because of how easily it is accessible to anyone at any end of the globe.
Our project is of vital significance and comes at a time when competition among media houses are on the rise. One needs to be of public importance and relevance in order to stay ahead.

The successful implementation of the project would mean that media houses can hit the right chord and connect to as many people as they can based on just 10 words of a headline.
This implementation cannot however be done on a small amount of data and thus needs petabytes of the already existing data that are available on the internet.
This is where Spark comes in, because of its scalability and the ease with which it manages to run the model with such huge chunks of data at such a low time

How do we harness Spark to achieve our purpose?
Spark comes packed with a lot of features and libraries that support ML models.
For the purpose of our project, we are using pyspark for sentiment analysis
SparkML and SparkMLLib are libraries that support packages like pyspark and word2vec.

Problem Statement:

Importance and necessity
The headline is the most important part of any piece of writing - whether it is an article, newsletter, sales copy, blog entry, web page, email or business report. Without a good headline the rest will not be read.
Given, the advent of social media and the increasing competition among media houses, we need to formulate articles that are more relevant and have maximum number of shares. With so much content on the internet, relevant articles of importance often are lost among the sea of content.

How do we go about doing it?
Our project is an implementation to predict the popularity of an article by training an ML model. For the same, we have to look back and analyse the millions of articles that are already on the internet and compare the relevance between the headline and it's content.
We would widely be using two algorithms : Sentiment analysis and word2vec.
Word2vec helps us achieve a vector representation of a corpus of words while Sentiment analysis relies on a model training method to associate a particular input (i.e. a text) to the corresponding output (tag) based on the test samples used for training. The feature extractor transfers the text input into a feature vector. Pairs of feature vectors and tags (e.g. *positive*, *negative*, or *neutral*) are fed into the machine learning algorithm to generate a model.

Impact:
The impacts of being able to predict popularity of an article based on its headline has a lot of commercial as well as social impacts.
Commercially, media houses can augment the newsletters and their articles to something more suited for the reader's pallette- something that they are most likely to view and share, thereby increasing the popularity of an article and edging over other competitors.

Socially, it is possible to use this implementation to channelise more socially relevant and important topics to be read more and not get lost among the billions of articles released everyday.

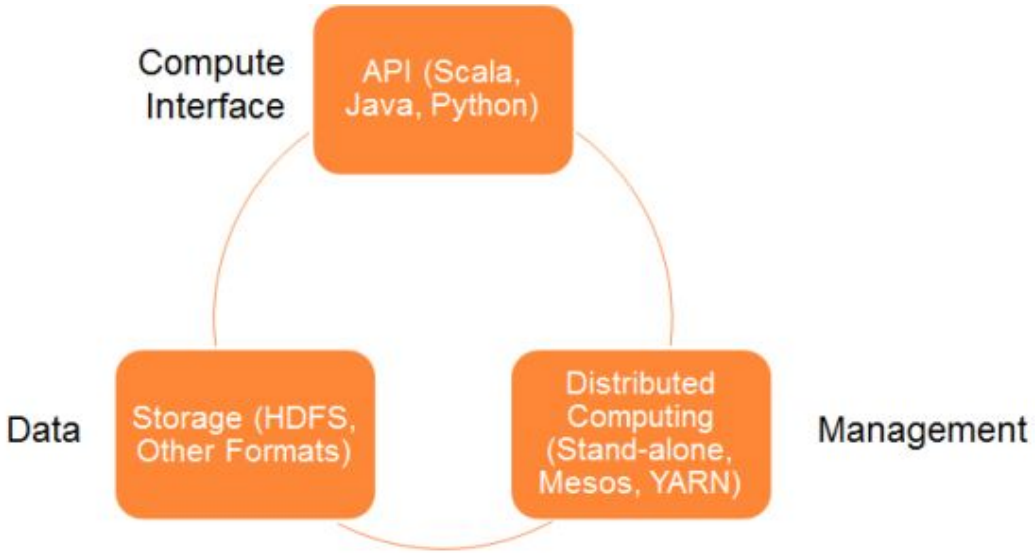
Implementation:

Methodology and Pipeline Architecture

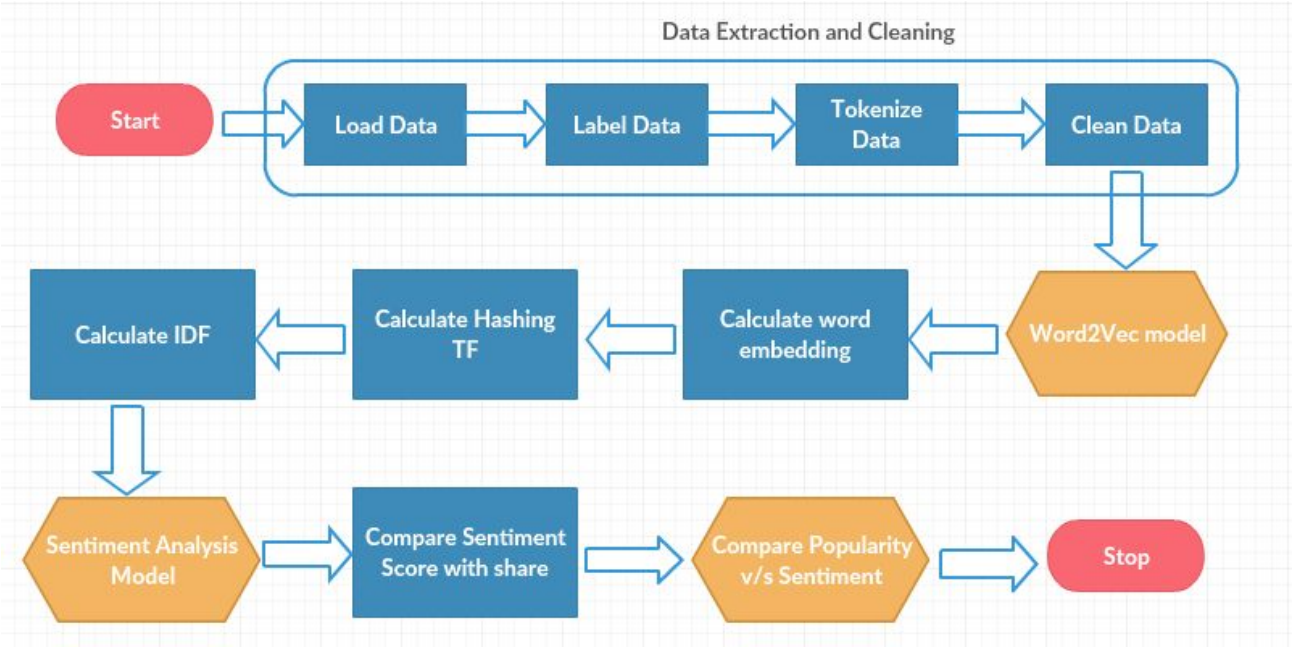
Methodology: In this project we take the data, calculate the word2vec embeddings score and observe the relevant content in the article and then perform sentiment analysis on the article title or headlines. Which gives us the sentiment score and then we compare the popularity of the article with its sentiment score

Step 1: Preprocess and clean the data, using NLTK
Step 2: Use word2vec to create word and title embeddings and then visualize them as cluster using t-SNE
Step 3: Perform the Sentiment Analysis on the headlines.
Step 4: Attempt to predict article popularity from the embeddings and other available features.
Step 5: Visualize the relationship between title sentiment and article popularity
Step 6: Drawing Inferences

Spark Architecture



Project Pipeline



Data Generation

1. For the Data Generation, we use NY Times most popular api https://developer.nytimes.com/docs/most-popular-product/1/overview/
2. Using this api, using this api we generate the most popular articles.
3. We have 3 field in the most popular article along with the other fields which presents us with the news
    a. number of views it received
    b. number of email and number of shares on Facebook.
    c. number of times it has been emailed

NY Times Most Popular API:
NY Times api provides a plethora of articles with a number of fields, which are very useful in data analysis.

The NY Times Provides services for getting the most popular articles on NYTimes.com based on emails, shares, or views.

There are number of options for Data Generation:

1. Get the most emailed articles

   https://api.nytimes.com/svc/mostpopular/v2/emailed/7.json?api-key=3XNf2AuXStd7KRvhqMFedwTw8N3oIwPA

2. Get the most number of Views

   https://api.nytimes.com/svc/mostpopular/v2/viewed/7.json?api-key=3XNf2AuXStd7KRvhqMFedwTw8N3oIwPA

3. Get the number of shares on facebook
   https://api.nytimes.com/svc/mostpopular/v2/shared/7/facebook.json?api-key=3XNf2AuXStd7KRvhqMFedwTw8N3oIwPA

Code snippet for Data Generation:

In [16]:

```
import requests
from bs4 import BeautifulSoup
import urllib.request,json
import sys
import math

#NY Times search article Api
#https://api.nytimes.com/svc/mostpopular/v2/viewed/1.json?api-key=3XNf2AuXStd7KRvhqMFedwTw8N3oIwPA

url="https://api.nytimes.com/svc/mostpopular/v2/viewed/1.json?api-key=3XNf2AuXStd7KRvhqMFedwTw8N3oIwPA"

data = urllib.request.urlopen(url)
response = json.load(data)
docs = response['results']
#print(docs)
data=[]
row={}
for i in docs:
    row['views']=i['views']
    row['abstract']=i['abstract']
    data.append(row)

with open('file.json', 'w') as outfile:
    json.dump(data, outfile)
```

Snippets of Data collected:

Data for number of Views


```
"results": [
  {
    "url": "https://www.nytimes.com/2019/04/28/arts/television/game-of-thrones-battle-of-winterfell-review.html",
    "adx_keywords": "Game of Thrones (TV Program);Television",
    "column": "Game of Thrones",
    "section": "Arts",
    "byline": "By JEREMY EGNER",
    "type": "Article",
    "title": "'Game of Thrones' Season 8, Episode 3 Recap: An Epic Battle Presented at a Human Scale",
    "abstract": "Sunday's Winterfell clash delivered tension and release, goose bumps and heartbreak, grandiosity and intimacy.",
    "published_date": "2019-04-28",
    "source": "The New York Times",
    "id": 100000006483036,
    "asset_id": 100000006483036,
    "views": 1,
    "des_facet": [
      "TELEVISION"
```

```
"url": "https://www.nytimes.com/interactive/2019/04/30/opinion/privacy-targeted-advertising.html",
"adx_keywords": "Advertising and Marketing;Political Advertising;Online Advertising;Acxiom Corp;Data-Mining and Database Marketing;Privacy",
"column": "",
"section": "Opinion",
"byline": "By STUART A. THOMPSON",
"type": "Interactive",
"title": "These Ads Think They Know You",
"abstract": "We bought some ad space and targeted readers using the invisible technology of the internet.",
"published_date": "2019-04-30",
"source": "The New York Times",
"id": 100000006470273,
"asset_id": 100000006470273,
"views": 20,
"des_facet": [
    "ADVERTISING AND MARKETING",
    "POLITICAL ADVERTISING",
    "ONLINE ADVERTISING",
    "DATA-MINING AND DATABASE MARKETING"
],
```


Data for number of shares on facebook


```
"results": [
  {
    "url": "https://www.nytimes.com/2019/05/03/world/asia/cyclone-fani-india-evacuations.html",
    "adx_keywords": "Evacuations and Evacuees;Odisha (India);Cyclones;Cyclone Fani",
    "subsection": "asia pacific",
    "share_count": 1,
    "count_type": "SHARED-FACEBOOK",
    "column": null,
    "eta_id": 0,
    "section": "World",
    "id": 100000006492463,
    "asset_id": 100000006492463,
    "nytdsection": "world",
    "byline": "By HARI KUMAR, JEFFREY GETTLEMAN and SAMEER YASIR",
    "type": "Article",
    "title": "How Do You Save a Million People From a Cyclone? Ask a Poor State in India",
    "abstract": "Officials showed seasoned skill in evacuating a poor crowded area in Cyclone Fani's path. "If you don't learn from past experiences, you will drown," one said.",
    "published_date": "2019-05-03",
    "source": "The New York Times",
    "updated": "2019-05-04 14:44:10",
    "des_facet": [
      "CYCLONES"
    ]
```

```
    "url": "https://www.nytimes.com/2019/04/30/business/media/obama-netflix-shows.html",
    "adx_keywords": "Television;Movies;Netflix Inc;Obama, Barack;Obama, Michelle;Higher Ground Productions;Documentary Films and Programs",
    "subsection": "media",
    "share_count": 19,
    "count_type": "SHARED-FACEBOOK",
    "column": null,
    "eta_id": 0,
    "section": "Business",
    "id": 100000006485822,
    "asset_id": 100000006485822,
    "nytdsection": "business",
    "byline": "By JOHN KOBLIN",
    "type": "Article",
    "title": "The Obamas and Netflix Just Revealed the Shows and Films They're Working On",
    "abstract": "Former President Barack Obama has a multiyear deal with Netflix in which he and the former first lady, Michelle Obama, will produce shows and films for the streaming service.",
    "published_date": "2019-04-30",
    "source": "The New York Times",
    "updated": "2019-05-01 17:55:12",
    "des_facet": [
      "TELEVISION",
      "MOVIES"
```

Data for number of emails:


```
"url": "https://www.nytimes.com/2019/04/29/smarter-living/the-case-for-doing-nothing.html",
"adx_keywords": "",
"subsection": "",
"email_count": 2,
"count_type": "EMAILED",
"column": null,
"eta_id": 0,
"section": "Smarter Living",
"id": 100000006442321,
"asset_id": 100000006442321,
"nytdsection": "smarter living",
"byline": "By OLGA MECKING",
"type": "Article",
"title": "The Case for Doing Nothing",
"abstract": "Stop being so busy, and just do nothing. Trust us.",
"published_date": "2019-04-29",
"source": "The New York Times",
"updated": "2019-05-02 21:45:30",
```

```
"url": "https://www.nytimes.com/2019/04/30/smarter-living/best-advice-youve-ever-received.html",
"adx_keywords": "Families and Family Life;Parenting;Computers and the Internet;Labor and Jobs",
"subsection": "",
"email_count": 11,
"count_type": "EMAILED",
"column": "Crowdwise",
"eta_id": 0,
"section": "Smarter Living",
"id": 100000006434237,
"asset_id": 100000006434237,
"nytdsection": "smarter living",
"byline": "By DAVID POGUE",
"type": "Article",
"title": "The Best Advice You've Ever Received (and Are Willing to Pass On)",
"abstract": "We asked you for the best advice anyone's ever given you, and how it made an impact on your life. Here's what you said.",
"published_date": "2019-04-30",
"source": "The New York Times",
"updated": "2019-05-03 01:19:02",
"des_facet": [
    "FAMILIES AND FAMILY LIFE",
    "PARENTING"
],
"org_facet": [
    "COMPUTERS AND THE INTERNET",
    "LABOR AND JOBS"
],
```


Solution:

In this project we propose a solution based on the results of word2vec and Sentiment analysis.

word2vec model:
Word2vec is a two layer neural net that process text. Its input is text corpus and output is a set of vector
It converts the text to a deep neural form.

Gensim Model:
Word embedding is an approach to provide a dense vector representation of words. Word embeddings are improved version when compared to word counts and word frequencies.

T-SNE Model:
T-distributed Stochastic Neighbour Embedding(t-SNE) is a machine Learning technique well-suited for embedding high-dimensional data for visualization.

1. We grab the titles of all the articles.
2. Then we use nltk to tokenize the words and convert into words.
3. Then we use gensim model.
4. In the next step we will be using Dimensionality reduction with t-SNE.
5. Which provides the visualization in low dimensional space.

Sentiment Analysis:

1. Sentiment analysis is the most common classification tool that analyses an incoming message and tells whether the sentiment is positive, negative or neutral.
2. Next we find out the popularity of the article based on the sentiment analysis scores.
3. The sentiment analysis scores for the document headline is calculated.

Language:

We used python as the language for collecting the data from API's and we used pyspark as the implementation model to implement the project.

```
from pyspark import SparkContext
from pyspark.mllib.feature import Word2Vec


#Provide the file directory
USAGE = ("bin/spark-submit --driver-memory 4g "
         "examples/src/main/python/mllib/word2vec.py")

if __name__ == "__main__":
    if len(sys.argv) < 2:
        print(USAGE)
        sys.exit("Argument for file not provided")
    file_path = sys.argv[1]
#Initialize the spark context
    sc = SparkContext(appName='Word2Vec')
    inp = sc.textFile(file_path).map(lambda row: row.split(" "))


#Fit the INP machine learning model
    word2vec = Word2Vec()
    model = word2vec.fit(inp)

#Find the synonyms associated with the word "economy"
    synonyms = model.findSynonyms('economy', 40)

#Then we print the word distance, we then use t-SNE model to visualize the word embeddings score
    for word, cosine_distance in synonyms:
        print("{}: {}".format(word, cosine_distance))
    sc.stop()
```

Snippets of code:

word2vec model

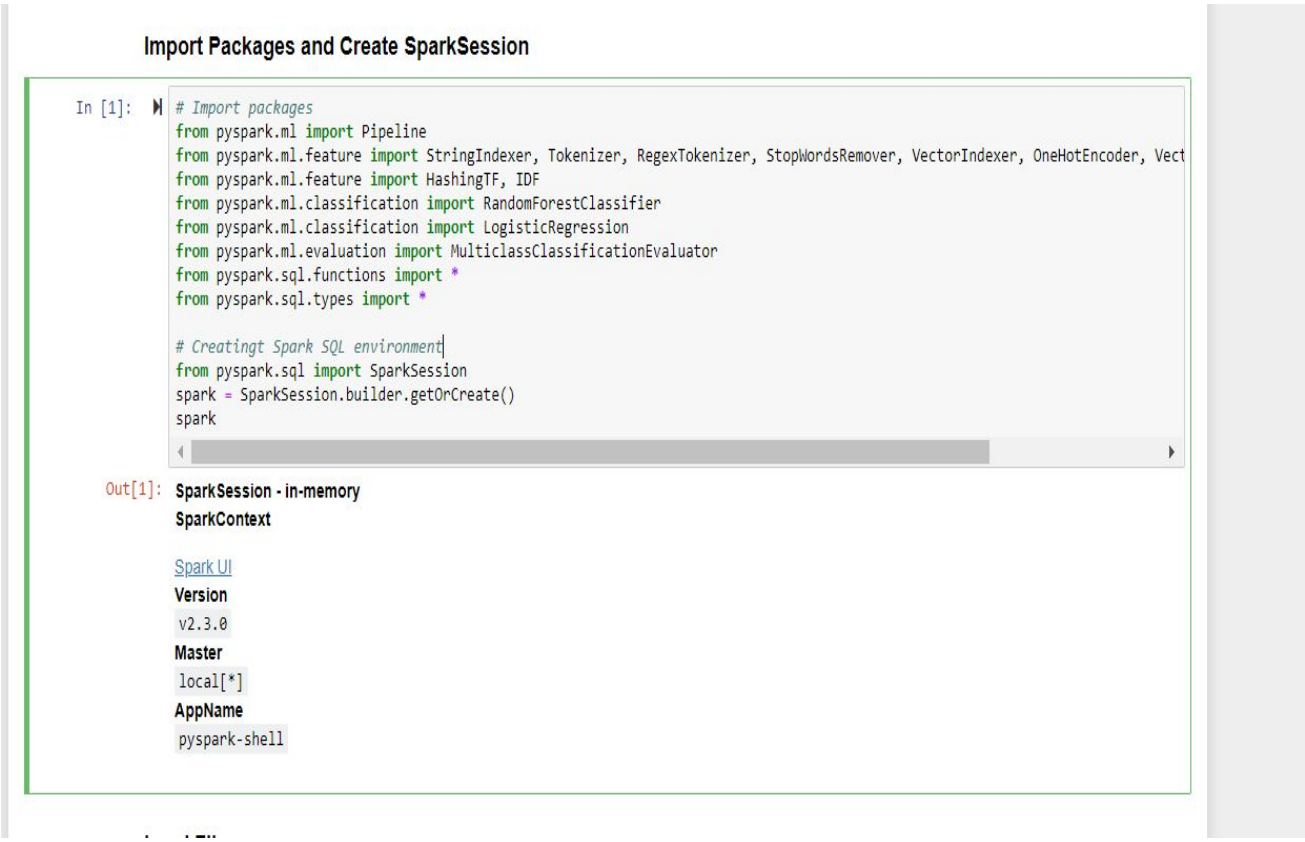
Steps performed in Word2vec Model:

1. In the first step spark context is initialized.
2. The stop words are removed and the data is cleaned
3. In the next step data is fed to the word2vec.
4. The word2vec output gives us how close is the context between the words in a document
5. Word2vec output gives how coherent the words are there in the document.
6. In the output we also get the distance between the points.

Sentiment Analysis

Steps carried out in Sentiment Analysis

1. In the first step spark context is initialized and spark session is initialized.



2. In the second step cleaning and formatting of the data is done

Clean and format data

In [3]:
```
# Create labels, tokenize and clean data

from pyspark.ml.feature import Tokenizer, RegexTokenizer, StopWordsRemover

# Create Labels for the record(containing text from different files of some category)
labeledData = StringIndexer(inputCol = "category", outputCol = "label").fit(articles_df).transform(articles_df)

# Tokenize the data
tokenizedData = RegexTokenizer(inputCol="text", outputCol="words", pattern="\\W").transform(labeledData)

# Clean data by removing stopwords
cleanData = StopWordsRemover(inputCol="words", outputCol="filtered").transform(tokenizedData)

cleanData.show(2)
```

```
+--------------------+--------------------+--------+-----+--------------------+--------------------+
|           file_name|                text|category|label|               words|            filtered|
+--------------------+--------------------+--------+-----+--------------------+--------------------+
|ny_business_artic...|Good Wednesday. H...|business|  1.0|[good, wednesday,...|[good, wednesday,...|
|ny_business_artic...|The German carmak...|business|  1.0|[the, german, car...|[german, carmaker...|
+--------------------+--------------------+--------+-----+--------------------+--------------------+
only showing top 2 rows
```

3. In the third step we calculate the sentiment analysis scores.

4. We then generate the sentiment analysis scores for the top headlines and compare the sentiment analysis scores with the popularity scores.

5. We can use this feature for fake article detection as well.

6. On the whole we observe that there is no clear marked difference between the number of times the article has been shared and the sentiment analysis scores it received.

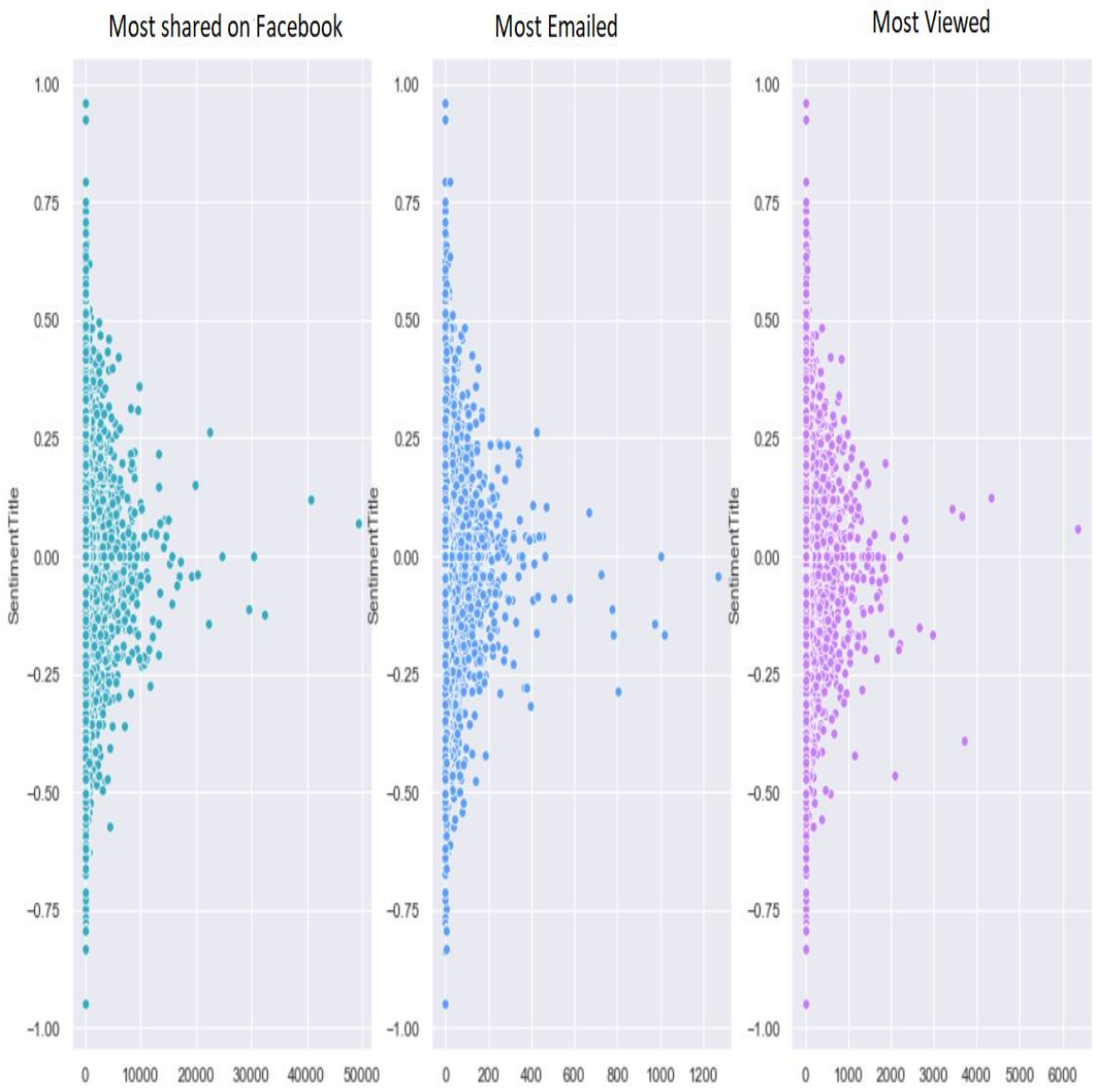
Outcomes and Visualizations

As a final step, we have the word embeddings score and the sentiment analysis score for the top headlines.

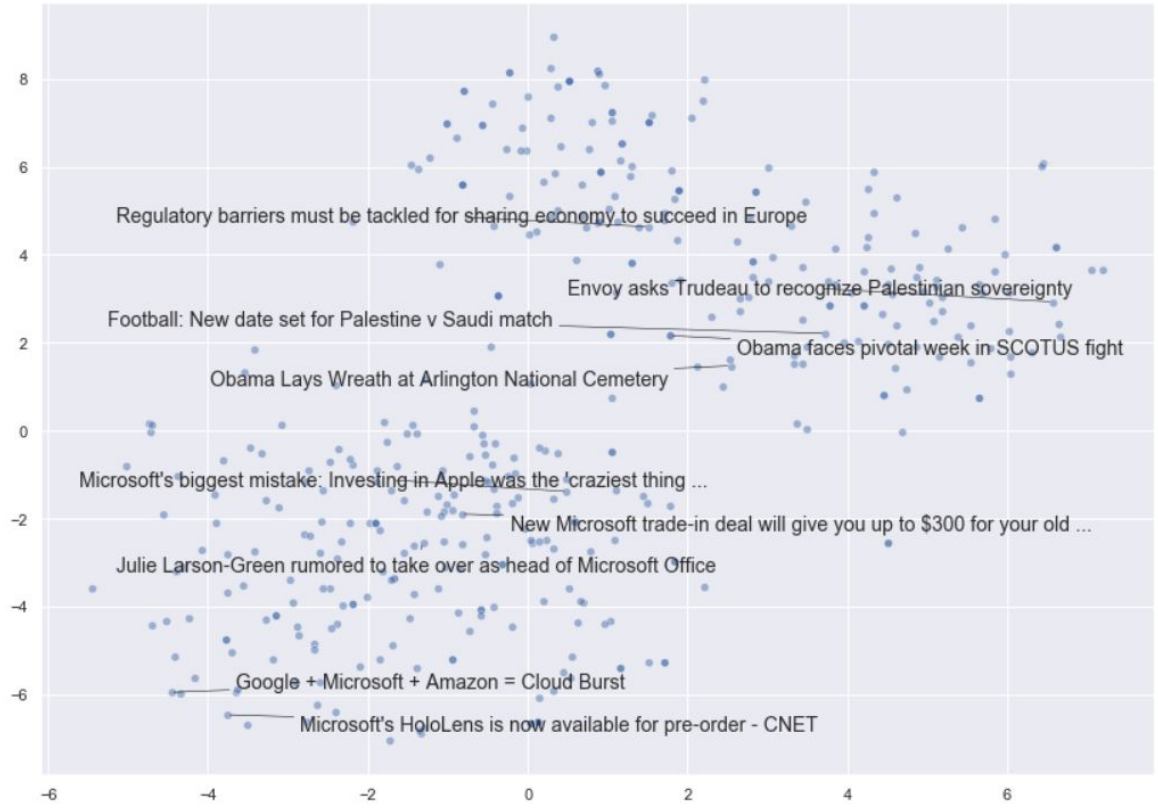
Now we have the sentiment analysis score for the headlines, the number of times the article has been shared on facebook, most emailed and most viewed.

Visualizations:

Sentiment analysis scores vs the shares of the article



Word2vec representation of the documents



Inferences:

1. We observe that there is no clear relationship between the sentiment analysis and the number of shares.
2. The embeddings score from the word2vec model show that there is a clear relationship between the articles which got the highest shares.
3. The sentiment analysis scores doesn't corroborate that having a good catchy headline will render more views.
4. The word2vec model helps us in finding the relationship between the overall article, it's context and the title.
5. We need to infer the relationship between the shares of the article vs the embedding score and the sentiment analysis score.

Future Implementation:

We can extend the word2vec model along with sentiment analysis scores for the title and the article and find whether the article is fake or not.
Another implementation of the above logic would be if we were to ever categorize all the contents available on the internet.

Summary:

In a nutshell, we used two machine learning models word2vec and sentiment analysis to deduce whether or not an article would be popular based on it's headline.

In conclusion, having a higher sentiment analysis score on the headline alone didn't have any positive effect on the popularity of the article. But the word2vec distance scores did help us in understanding the popularity of the article.

This implies that the users generally read through the article before sharing it and do not just go by the headline. The popularity of the article has more to do with its contents and not on it's clickbaity- headline.

References:

https://towardsdatascience.com/using-word2vec-to-analyze-news-headlines-and-predict-article-success-cdeda5f14751
https://nbviewer.jupyter.org/github/chambliss/Notebooks/blob/master/Word2Vec_News_Analysis.ipynb
https://github.com/apache/spark/blob/master/examples/src/main/python/mllib/word2vec.py
https://towardsdatascience.com/sentiment-analysis-with-pyspark-bc8e83f80c35
https://mapr.com/blog/spark-101-what-it-what-it-does-and-why-it-matters/
https://www.edureka.co/blog/spark-streaming/#Use_Case_Twitter_Sentiment_Analysis