CS 6350.002 - Big Data Management and Analytics
Spring 2018
Project Report

Kruthika Vishwanath (kxv150930)
Radhika Kalaiselvan (rxk162030)
Eswar Chowdary Ganta (exg151430)
Poonam Purushottam Pathak (ppp160130)

Introduction

Analysis of news has become very crucial in recent years. It can help in many ways from voting the right candidate in an election to getting records on GDP of a country. In most cases, it gives insights to people to understand quickly the analytical view on what is going in the surroundings in no time.

Keeping the above in mind, we choose "All news" dataset from Kaggle. When we first read the description, there was no existing problem being stated to solve in the challenge. Hence, we went on to define our own challenge to learn as to what happened in the year 2016 & 2017 that it made world headlines. We made different analysis to understand different perspectives of the news in the aforementioned years.

We used Topic modeling techniques on the data to get the top topics that were trending in the years considered. We did analyze the sentiments and actors in the news. Also, we tried clustering the news around the topics to rank them based on cosine similarity measure.

Overview of the Dataset

"All News" (unlabeled) Kaggle dataset has the following features:

- id
- title: Title of the news
- publication: name of news publisher. Eg: CNN
- date: date of publication. Eg: 06/11/2017
- year: year of publication. Eg: 2016
- month: month of publication. Eg: 12 (December)
- content: news content

Having news title, date and news content, we were able to come up with some interesting analysis. Before that, we went ahead with data preprocessing. The data contains the news records related to the years 2016 and 2017.

Preprocessing

We used PySpark for the preprocessing of the data. The following are the various steps involved in the process.

- Deleted index column
- Deleted url column which had Nan (Removal of NAs)
- Deleted author column which had Nan
- Converting year and month to integer
- Cleaning the month column to have values between 1 & 12
- Filtered out publication column

Once we had the clean data, we went ahead with brainstorming as to how to make good use of the news content. As a team of 4, we all had different ideas and interests to play with the dataset. Hence, we divided our ideas into three different parts.

Part 1: Getting top talked topics for the year 2016 & 2017 (Topic Modeling)

Part 2: Lets evaluate the top talked topics from Part 1 using Cosine similarity

Part 3: Understanding the trends of sentiment score of any actor for the year 2016

Top talked topics for the year 2016 & 2017

We aim to get the most talked about topics from the news content for the years 2016 and 2017.

The **algorithm** is as follows:

1. Load and pre-process the data:
    a. Take the news content column from the dataset and token it using regex tokenizer
    b. Remove stop words from the tokenized news content
    c. Vectorise words to feature vectors
2. Combine the above stages with LDA model in a pipeline to get the most top topics.
3. Examine results by sending the results from step 2 to a user defined function (explained in the later section as Evaluation).

Challenges faced

- Some of our most talked about topics were words such as “like”, “new” etc. To resolve this, we included such words in regex tokenizer.
- Displaying results using user defined function but, later on with some help from online resources we were able to understand user defined function well.

Overall, due to some practice from assignment 3, we didn't face much of difficulty in doing this part of the project. But after this, our question was how do we evaluate this topic modelling. On reading some of the blogs and some research papers online, we came up with some dummy way of creating labels for the dataset since our dataset was not labelled to apply classification methods.

The results of the topic modelling are plotted as follows:

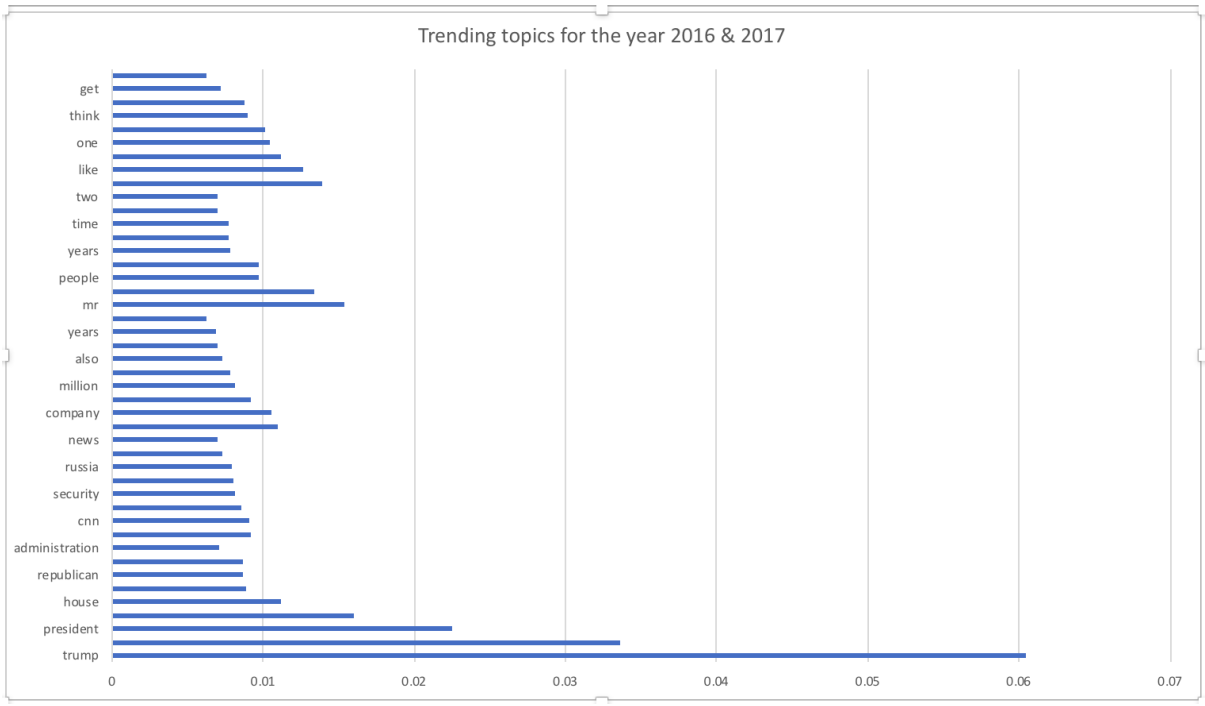


Fig.: since the news content was for the year 2016 & 2017, it was valid to see words like trump at the top

Evaluation

We took the new title and wrote a Scala code to check whether the news title contains the top talked topics from above section. If it does, create a label as famous and assign value 1, else, 0. After performing the same, our data looked like below:

	A	B
1	title	famous
2	Would a Trump Presidency Mean Economic Disaster? Let,Äôs Take a Look - The New York Times	1
3	U.S. Senate, Orlando, Donald Trump: Your Tuesday Evening Briefing - The New York Times	1
4	In ‚ÄòBrexit‚Äô Vote, David Cameron Faces Problem of His Own Making - The New York Times	0
5	How to Survive Being an Airbnb Host - The New York Times	0
6	Falluja Restaurant Is Reborn in Baghdad, Offering Nostalgia With Its Kebab - The New York Times	0
7	What J. Dennis Hastert, Ex-House Speaker, Will Face in Prison - The New York Times	0
8	Rift Between Officers and Residents as Killings Persist in South Bronx - The New York Times	0
9	Tyrus Wong, ‚ÄòBambi‚Äô Artist Thwarted by Racial Bias, Dies at 106 - The New York Times	0
10	Among Deaths in 2016, a Heavy Toll in Pop Music - The New York Times	0
11	Kim Jong-un Says North Korea Is Preparing to Test Long-Range Missile - The New York Times	0
12	Sick With a Cold, Queen Elizabeth Misses New Year‚Äôs Service - The New York Times	0
13	Taiwan‚Äôs President Accuses China of Renewed Intimidation - The New York Times	1
14	After ‚ÄòThe Biggest Loser,‚Äô Their Bodies Fought to Regain Weight - The New York Times	0

To achieve this, we used the .contains available in Scala. Once we created the above as train data, for the test, we manually labelled it for some 500 news content. Having the train data and test data (manually labelled), we applied some of the classification models like logistic regression & random forest.

Steps to evaluating topic modelling

1. Load and preprocess the data
    a. Tokenize the news title (we took news title instead of news content was to reduce preprocessing times as the news content was really large.)
    b. Remove stop words from tokenized news title
    c. Apply hashing TF model
    d. Create the logistic regression model & Random forest model with parameters
2. Separately pipeline the above models
3. Evaluate the pipeline using multiclass evaluator to fit the model on train data using cross validator.
4. Transform the test data using the above fit model and get metrics such as accuracy, precision, recall, f1score.

Since we were running it on our local machine and could not take training data as 50,000 news as our systems ran low on memory, we reduced the training data to 5,000 news content after taking your permission. Test as 500 news content. With that, we obtained the following results:

Metrics for Random Forest

- Accuracy for Random Forest: 0.9563492063492064
- Precision for Random Forest: 0.9580686451123542
- Recall for Random Forest: 0.9563492063492064
- F1 Score for Random Forest: 0.9550302168503377

Metrics for Logistic Regression

- Accuracy for Logistic Regression: 0.9623015873015873
- Precision for Logistic Regression: 0.9622110817941952
- Recall for Logistic Regression: 0.9623015873015872
- F1 Score for Logistic Regression: 0.962251390738236

Conclusion from code: Logistic regression gives better result than Random Forest.

While we were choosing the dataset, we were under the assumption that, most of the news was on election since the news was from 2016 (US elections) & 2017. But, our results prove that the dataset also contained other news like Iraq, China's GDP, Women harassed etc. This was all coming from CNN as CNN

reports world's news apart from US news. But other four publications which we were considered in this part was New York times, Atlantic, Breitbart, Business Insider which mainly reported US news.

We do agree that, we went ahead with dummy evaluation since we had to create labels all by ourselves based on top talked topics. But to better support this, one of our team member suggested using cosine similarity to evaluate the top talked topics.

News Content Clustering as per Top Most Headlines

Once we get a refined set of 20 most talked about topics in the year (2016-2017) from LDA clustering. We then focus on clustering the news around those topics to rank those topics as a measure of their popularity.

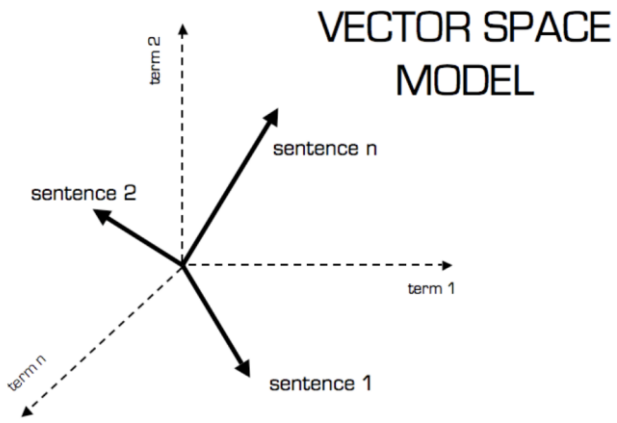
We utilize the concept of cosine similarity in a vector space model where each news headline and a topic are represented as document and query vectors respectively.

Libraries used: Apache spark ml library

Cosine Similarity Concept

Cosine similarity is one of the popular approaches used to determine how similar are the documents. This approach is based on the fact that the most similar documents will have the most relevant words common between them.

To execute this school of thought, we represent words in the query and document as vectors in a 2D model where value of each vector is the weighted frequency of the word appearing in the document. Once we get the vector representation, the angle between each pair of them can be measure of how close these vectors are. Therefore, cosine values of that angle are a good measure of similarity between those vectors and therefore the documents representing them.



Vector Space Model

Implementation

To implement this approach on the news data, we addressed the following issues and features:

1) Pre-processing and tokenization: Since the model relies heavily on vectors, it is important that the vectors representing the news and topics are relevant. So, the first process involves removing the stop words and special characters from the news content and tokenize them using apache.spark.ml tokenizer and stop words library.

2) Calculating TF and IDF weight: Then, the following formulae were used to calculate the tf-idf weight of each word to be represented as a vector in cosine similarity:

$$TFIDF(t, d, D) = TF(t, d) \cdot IDF(t, D).$$

where TF(t,d) is given by :

TF(t,d) = 1 + log(tft,d)

tft,d - the count of word t appearing in news headline d and

DF(t,D) = number of news headline word t appears in out of total headlines (D).

Calculating TF(t,D) and DF(t,D)

Since the Tf weight for a word appearing in topic and a news content should be with respect to its own respective vector space, we cannot use apache.spark.HashingTF library as it outputs the hashed index based on features. Instead we calculate the term frequency as per the above formula for TF(t,d) by iterating over each row for each topic.

Similarly IDF(t,D) is calculated over all the news content in data frame for each topic.

3) Normalization:

To get accurate cosine similarity, we used apache.spark.ml.Normalizer library to normalize the tf-idf weight of each vector.

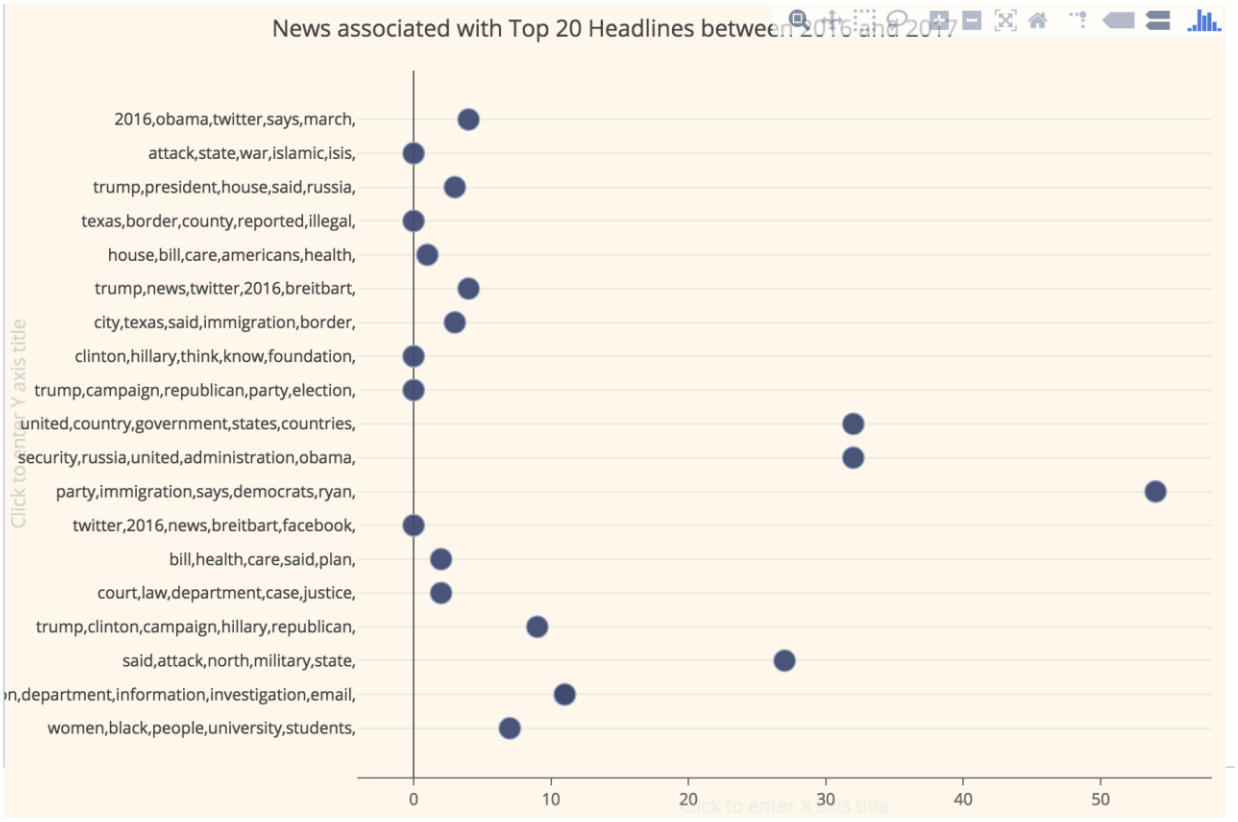
4) Cosine Value calculation and mapping:

Cosine value is calculated as part of dot product on normalized vectors, the topic with maximum cosine value for a news headline is considered to be the closest topic to that news.

Sample results and inference

We took dataset of 200 rows news content to cluster them around 20 top most talked topics:

Following are the graphical representation of the results:



The X-Axis represents the number of news headlines around the topic listed in Y-Axis.

Resultant Ranking of the top topics:

1) Democratic Party Immigration
2) Country United States government
3) Russia Security Obama Administration
4) North Military Attack
5) Clinton Investigation Department email information
6) Trump Campaign republican Hillary Clinton
7) University Students Black people women
8) Twitter news 2016 Trump Breitbart
9) Twitter 2016 Obama says march
10) Immigration border Texas city
11) President Trump house Russia
12) Court law case justice department

13) Plan Health Care bill

14) Health Care bill House

15) Trump Campaign republican party election

16) Hillary Clinton Foundation

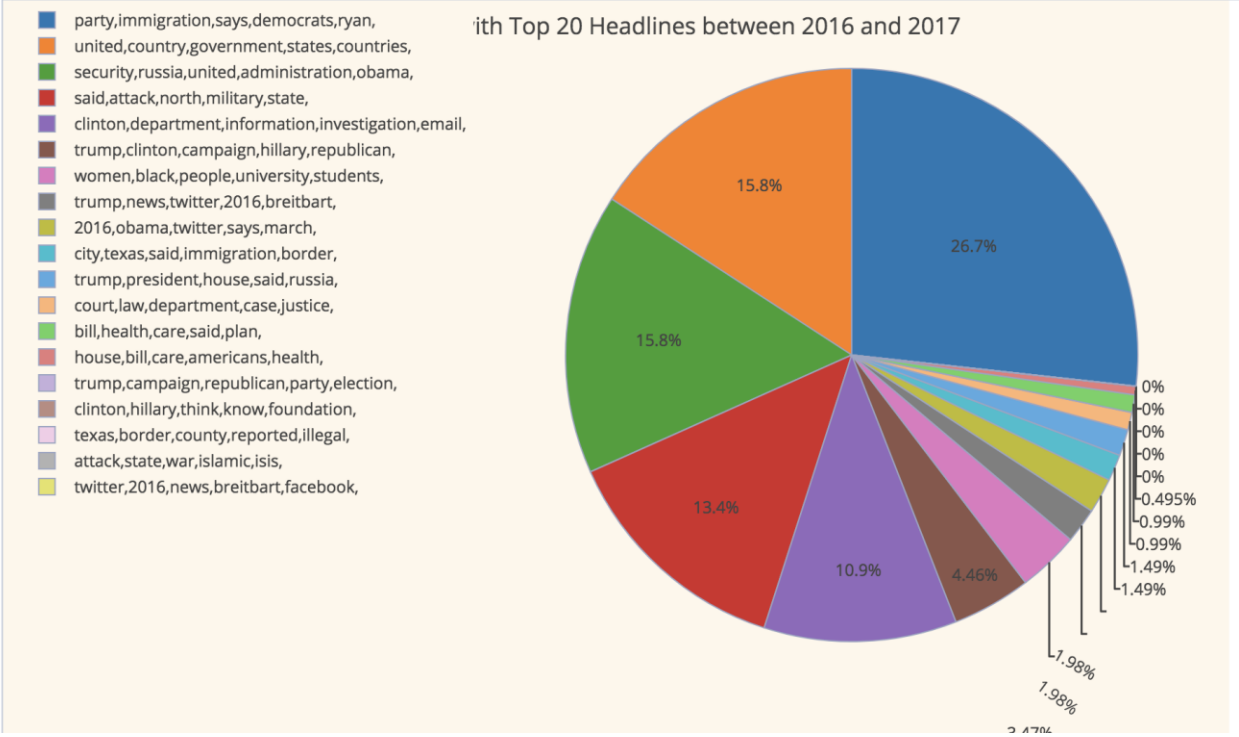
17) Reported illegal Texas border county

18) Islamic State ISIS attack war

19) Twitter 2016 news Breitbart Facebook

20) North Trump Attack says twitter

The pie-chart representation:



Inferences

1) Most of topics in year 2016-2017 revolved around topics “Democratic Party Immigration”, “Country United States government “, “Russia Security Obama Administration “, “north, military, attack” and “Hillary Clinton Investigation email”.

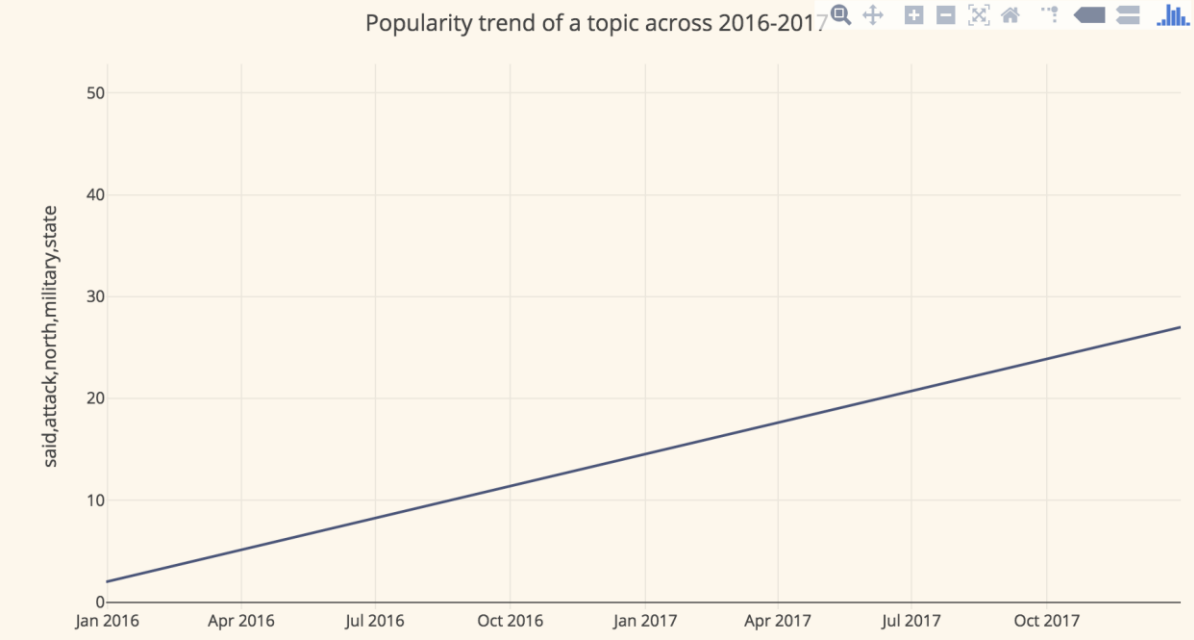
2) Topic “Democratic Party Immigration” is ranked the highest. On further investigation, we found that most of the news content which wasn’t related to the topics in relation with election 2016, got mapped to this topic as per the cosine vector concept modelling. Therefore, ranking of this topic as first might not give clear picture of news content clustering of 2016.

3) We can also see in pie chart representation, the equal percentage of document distribution for topics “Plan Health Care bill” and “Health Care bill House” as the both these topics are similar on the line on concept and there for rightfully the cosine value distribution is equivalent across both.

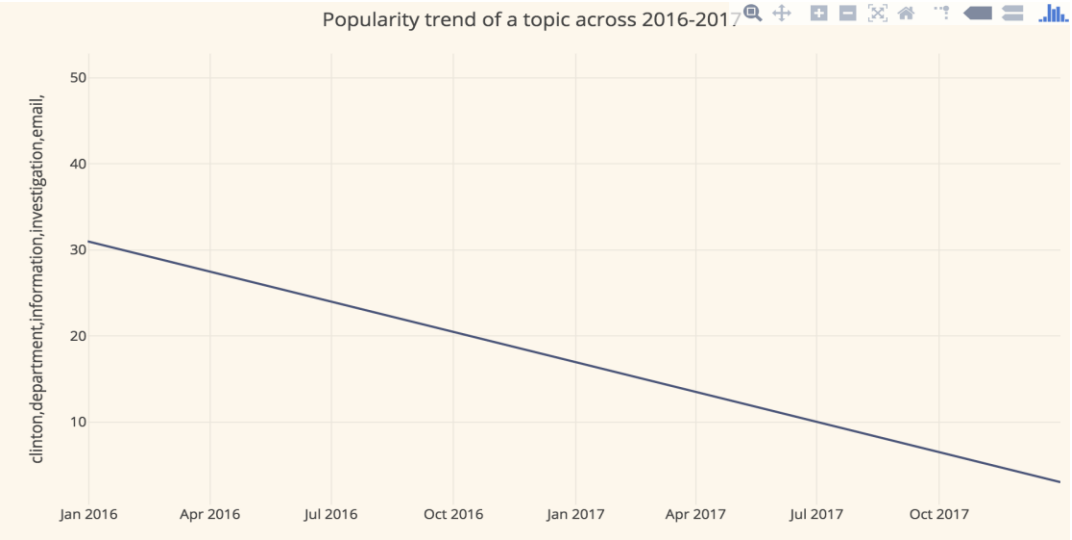
4) Few of the last ranked topics do not map with any of the news headlines in the sample. This is probably because the sample size taken (200) is very small compare to the topic modelling done on dataset of 50,000 and the model highly depends on how accurately the top topics for association are determined.

Following is the time series plot of popularity:

Popularity of topic “North Military Attack” across 2016 -2017



Popularity of topic “Clinton Investigation Department email information” across 2016 -2017



The represents the change in the distribution of news content around the given topics across 2016 to 2017. This distribution indicates the popularity trend of that topic.

For example, the first time series graph for topic “North Military Attack “increased from 2016 to 2017 after the election whereas discussion around the topic “Clinton Investigation Department email information” decreased drastically post 2016 election.

Challenges and Shortcomings:

- Executing Cosine similarity is a computationally intensive, to process it on dataset of 200 rows alone takes 10-15 mins to complete. This is typically because we need to iterate the whole dataset each time for TF calculation, IDF calculation, normalization and dot product. As a result, inference drawn on the results may not be completely correct.

- As discussed in the inference section, a topic like “Democratic Party Immigration” can get associated with news which does not strongly capture the essence of other major topics of concern. This is because a cosine value is not good in removing the dissimilar topics as the similarity.

- Evaluation of cosine similarity is difficult as labelling of the train data if done manually is very subjective.

Conclusion

As per the analysis we did across the news data and the inferences drawn on it, we can conclude that cosine similarity model is overall a good approach to cluster the data topics or labels provided, the dataset worked on is large enough and labels are accurate.

Future Scope of Improvement

- As stated in challenges and shortcomings section, there is definitely a scope to optimize the algorithm in calculating the tf, idf and cosine values to facilitate running over large dataset to improve the accuracy.

- The results of topic modelling can be further refined by integration with sentiment analysis and actor identification using Stanford NLP libraries to remove irrelevant topics. More accurate the topic labels, more accurate the clustering will be.

Actor identification and sentiment analysis

Identify the top 10 personalities based on the maximum occurrences of their names on the news dataset and identity the average sentiment score of the context of their occurrence. In this way we identify the trend of the sentiment score of any actor through a year.

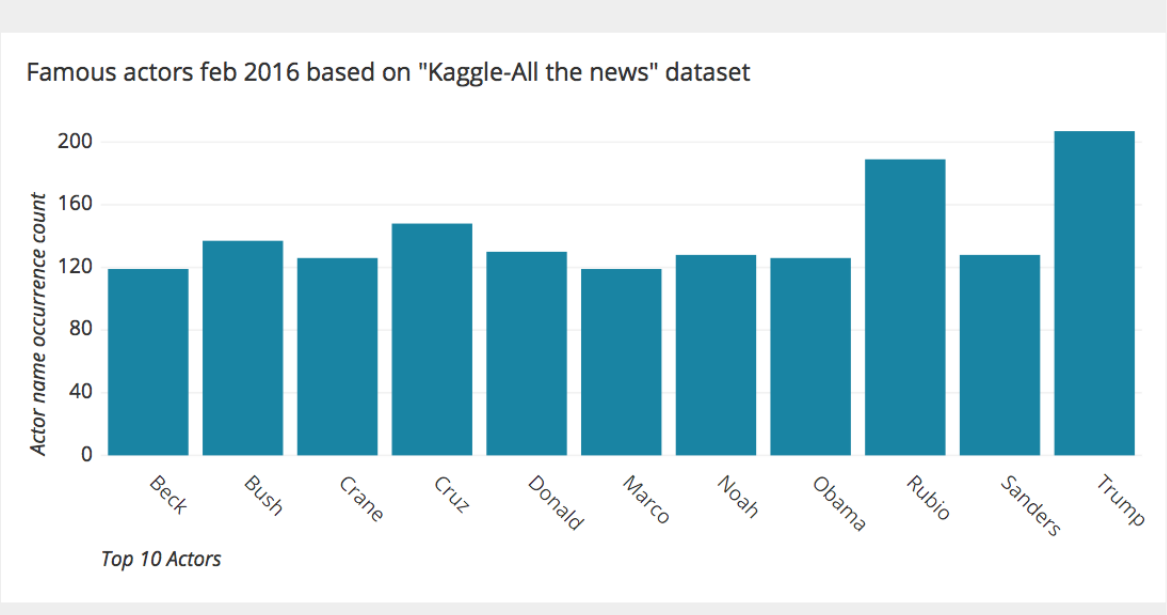
Libraries used: Stanford Core NLP, Spark MlLib, Databricks Core NLP

Stanford named entity recognizer (NER) identifies sentences in large text dataset and tags the words into one of the 3 classes (PERSON, ORGANIZATION, LOCATION). Used this library to identity the PERSON tags/actors and the sentiment score of the sentence containing this word and sorted them based on the occurrence count.

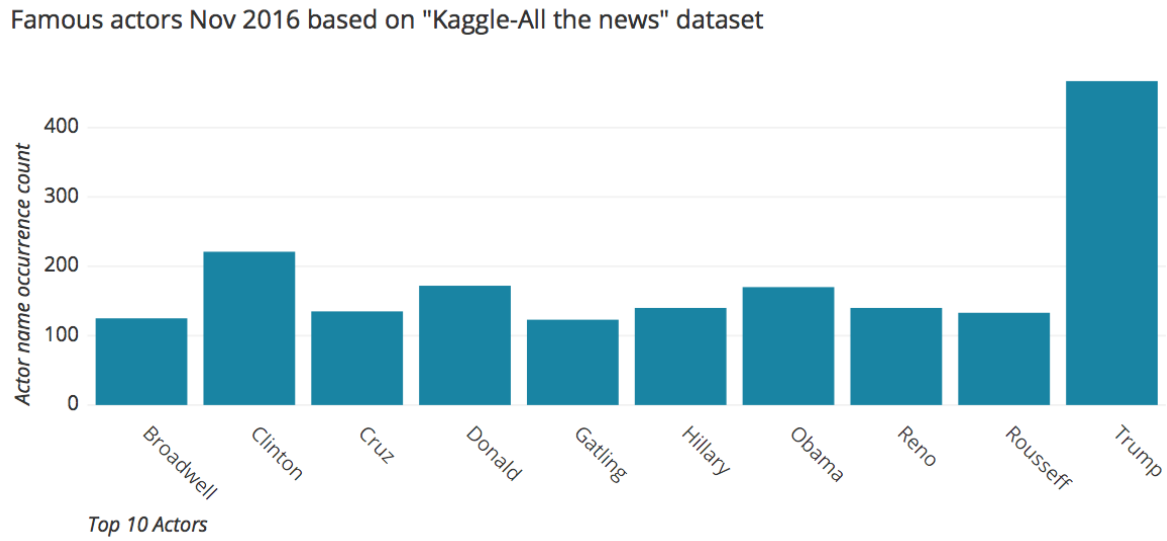
Sample results and inference

Most famous actors based on the count of occurrence of their names in various journals.

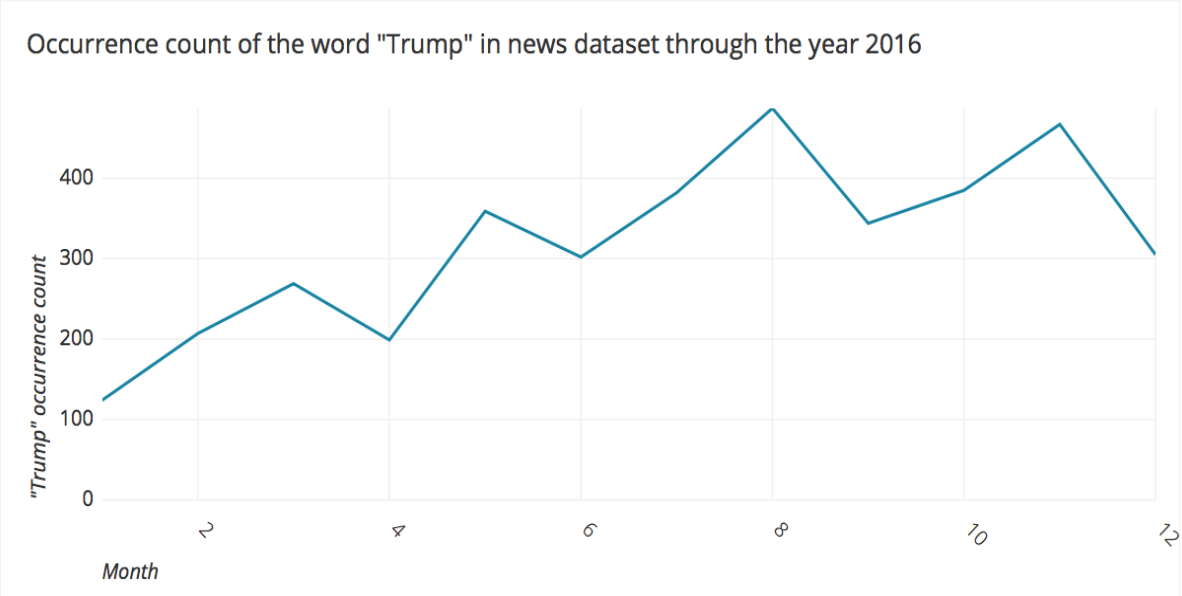
Month –Feb 2016

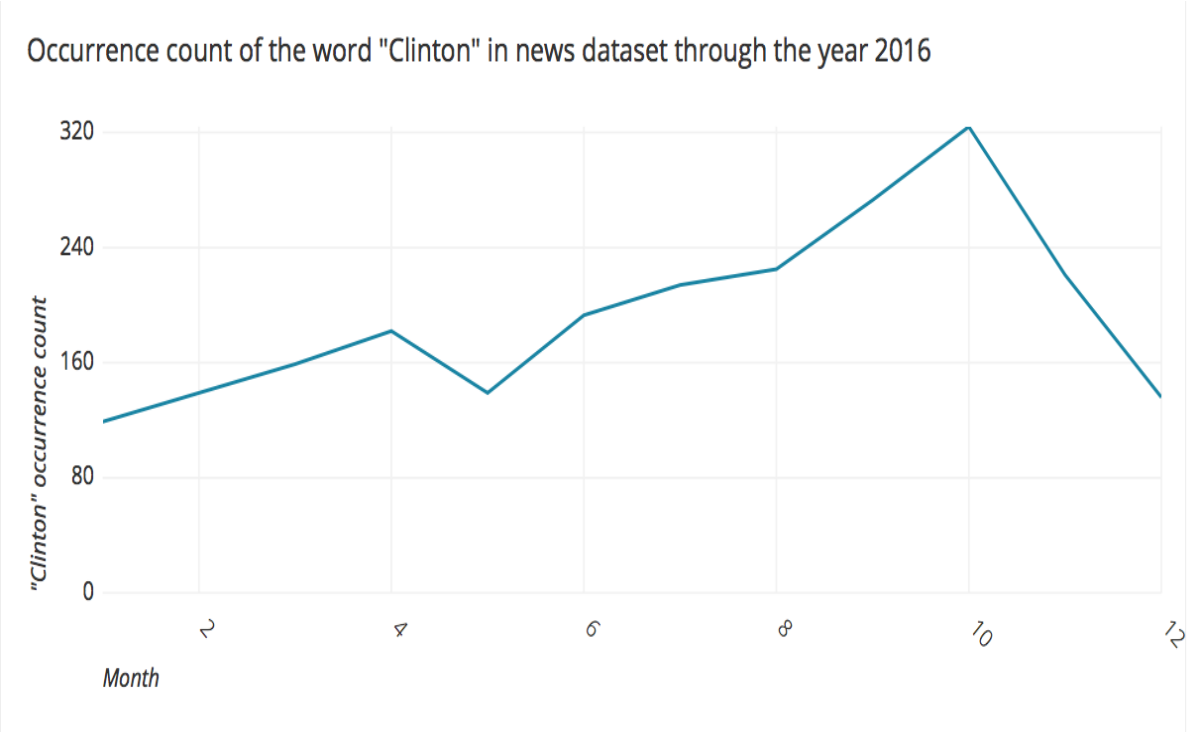


As you can see in the below graph, “Trump” has the maximum count for Nov 2016- might be due to the election results.

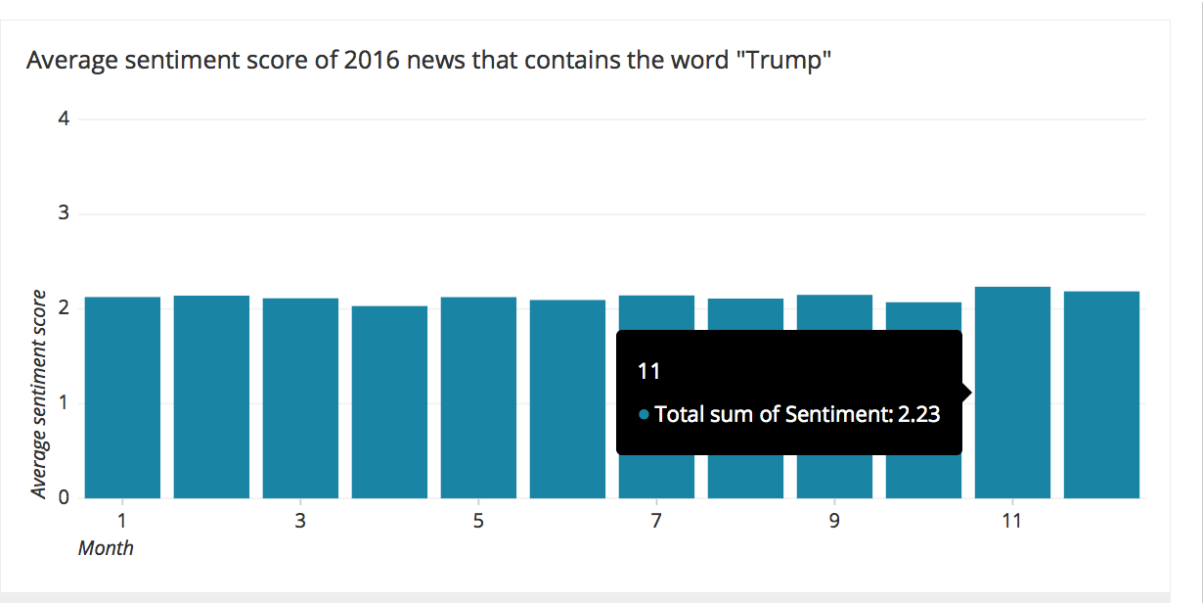


Following are the occurrence patterns for the words “Trump” and “Clinton” for the year 2016.

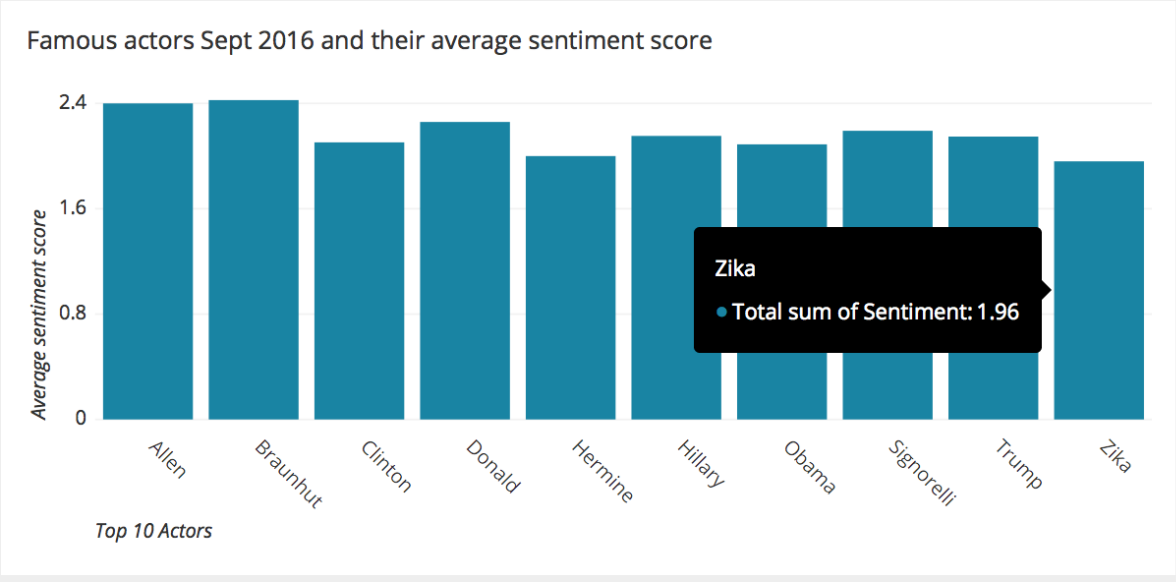


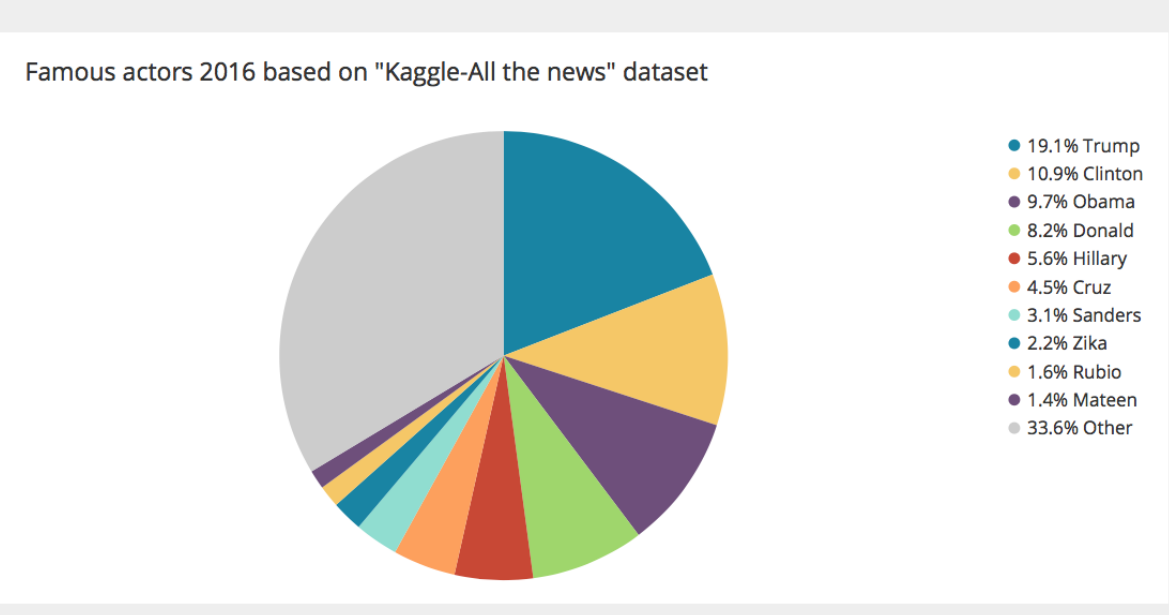


Following graph represents the change in the sentiment score of Donald Trump for the year 2016 based on “All the news” dataset. You can notice the maximum sentiment score the word Trump is in November 2016 and the value is 2.23.



Actors need not be personalities, in the following case, Zika virus is one among the top 10 actors and has a sentiment score of 1.96 (NEGATIVE). Hermine is an actor because most of the news in September 2016 talked about tropical storm Hermine, it has a sentiment score of 2(NEGATIVE).





From the pie chart above, the most talked about actors for the year is “Trump”, about 19.1% of the news contains the word “Trump”. 10.9% of news was about Clinton. The above graph can be used to infer the major actors for the year 2016.

Points to note

- Stanford NLP library identifies each space separated word as a different actor, hence most the graphs might contain the same person’s first and last name in the output. (For example, you might find entries for Donald and Trump represented as different actors)

- The sentiment score from Stanford NLP library ranges from 0 (Very NEGATIVE) to 4 (VERY POSITIVE). As you can see most of the values lies between 2 to 2.5, because the source dataset contains news from wide variety of publications which publish positive news and negative news about the same actor, so the average sentiment score falls in the NEGATIVE to NETURAL range.

Limitations

Scala support for Stanford NLP is very limited. Stanford NLP and Databricks Core NLP are compatible only with Scala Version 2.10.6, Spark 2.0.0.

Contribution of team members

- Topic modelling on news content to get most talked about topics and its evaluation - **Kruthika Vishwanath & Eswar Chowdary Ganta**
- News Content Clustering as per Top Most Headlines using cosine similarity – **Poonam Purushottam Pathak**
- Actor identification and sentiment analysis using Stanford NLP – **Radhika Kalaiselvan**

References

- Cosine similarity: https://venukanaparthy.wordpress.com/2015/08/16/simple-document-classification-using-cosine-similarity-on-spark/
- Learning about evaluation in clustering problems: https://jaceklaskowski.gitbooks.io/mastering-apache-spark/content/spark-mllib/spark-mllib-KMeans.html
- http://blogs.quovantis.com/k-means-clustering-algorithm-through-apache-spark/
- https://databricks-prod-cloudfront.cloud.databricks.com/public/4027ec902e239c93eaaa8714f173bcfc/957490241968034/2278296410500454/8369728141520448/latest.html
- https://towardsdatascience.com/multi-class-text-classification-with-pyspark-7d78d022ed35
- Paper on understanding topic modelling in a deeper way: http://papers.nips.cc/paper/3700-reading-tea-leaves-how-humans-interpret-topic-models.pdf
- ML library: https://spark.apache.org/mllib/
- Understanding on cosine similarity: http://blog.christianperone.com/2013/09/machine-learning-cosine-similarity-for-vector-space-models-part-iii/
- Plot graphs: https://plot.ly/scala/line-and-scatter/
- To understand IDF calculation: https://github.com/deanwampler/spark-scala-tutorial/blob/master/src/main/scala/sparktutorial/solns/InvertedIndex5bTfIdf.scala