SOCIAL NETWORK ANALYSIS USING BIG DATA APPROACH

Kiruba Dhayalan

A20435562

Manukar Bapathi

A20427621

Barani

A20434814

Vismitha

A20428844

INTRODUCTION:

The aim of this project is to build a system that would suggest the user with the most similar members who are within ‘n’ hops from him with the help of big data technologies. We mine for the user's personality traits using his tweets/retweets/likes data. Once we get personality information, we check for traits for the members who are under n hops from the user. Then cluster based on each personality trait and rank users with the highest likelihood of link using link prediction methods. Once we have ranked users for each trait we can aggregate the values and obtain top x users who are similar to the user in all aspects and recommend them.

Significance of big data technologies:

The drastic increase in the usage of social media sites resulted in the generation of large datasets. Processing and extraction of useful information from the large dataset is a tedious job. Twitter is one of the most popular microblogging fora and has become a powerful communication and information sharing tool. Analyzing tweets has become a tedious job because of volume, velocity, and variety. Big data technologies help in mining data with 3 V’s, (Volume, Velocity, Variety).

In this project, we have implemented several big data technologies based on the requirement and problem statement. Here is a brief description of the big data tools in the project.

Apache Kafka is a fast, scalable, durable, and fault-tolerant publication-subscription messaging system. It is mainly used for streaming large volumes of data. In this project, Kafka is used for streaming twitter objects from Twitter Streaming API. Producer API has been used to create a client that will fetch tweets from twitter API.

Elastic search is powerful indexing and analyzing engine. The twitter data from Kafka topic is stored and indexed in elastic search.

Kibana is a visualization platform. It is used to search, view and interact with data that is stored in elastic search. It is used to analyze and view twitter data in our project.
Spark MLlib is a Spark machine learning library. It is used in the project for feature extraction, topic modeling, and machine learning library.

Scope of the project:
The project can be divided into two major categories. One is the **Dynamic streaming twitter analysis** and Static twitter analysis. Dynamic streaming twitter analysis is where a user twitter id is given as input and feature extraction and topic modeling is done for the given twitter user and compared against the main user. The similarity score is then calculated. In static twitter analysis, we collect the second-degree user's tweets, feature extraction, and topic modeling is performed for each user and compared against the main user and output the top x users.

The initial aim of the project was to allow recommendations from all users who are at n hops from the main user (n and the main user being a configurable parameter). However, during the Data Collection phase, we realized that we have grossly underestimated the amount of data that needed to be collected. This humongous task was further complicated by the strict Rate Limits which Twitter applies on TwitterAPI. Thus we decided to choose a fixed main user (Marques Brownlee @MKBHD, a popular tech Youtuber and Tech enthusiast) and provide recommendations to him. In order to build our recommender system, we collected all the tweets of the main user @MKBHD. Then we collect 200 most recent 2nd-degree users and their corresponding tweets. Only 200 have been collected for rate limiting purposes.

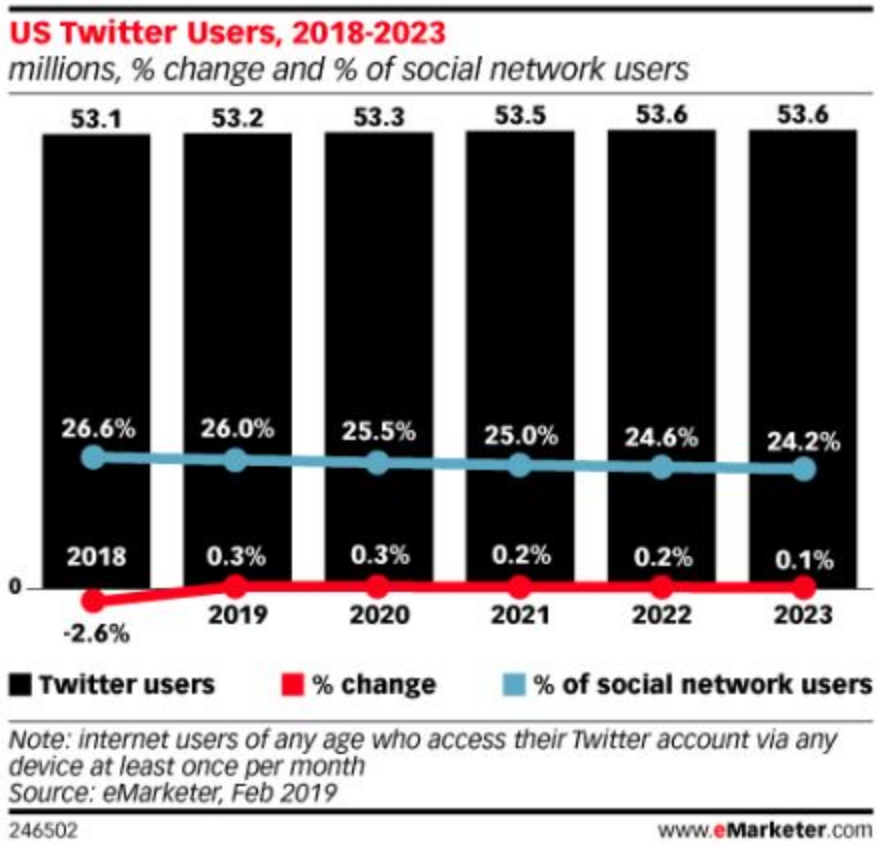
BACKGROUND AND RELATED WORK:

Online Social Networking platforms are a great place to make new connections and meet people of similar tastes. The growing popularity of such social networks has raised new areas of applications for recommender systems. The main aim of such recommender systems is to identify and suggest new friends for a user such that they have similar interests. The most popular methods utilized for making recommendations exploit the social network information or the similarity of user-generated content. In this project we try to build such a recommender system which profiles the user and generates the topics of interest of the user, Then compare his topics of interest with that of other users in the network whom the user is not friends with. Based on the similarity of the topic distribution, users are ranked and recommended to the main user.

Twitter is one of the most popular social media platforms with over 126 million users. Twitter has followers/followees social structure, where a follower will receive all the micro-blogs from the users he follows, known as followees, even though they do not necessarily follow him back. In turn, re-tweeting allows users to spread information beyond the followers of the user that post

the tweet in the first place. Big data tools are perfect for processing such huge data with the vast variety. The author in [1] used Hadoop ecosystem tools such as Hive, Pig for analyzing the relationships between the tweets posted by different users and Apache flume for data exchange. In our project, we tried implementing Apache Kafka instead of Flume and elastic search and spark instead of Hive and Pig.

Analyzing users and their behavior patterns using big data has been a topic for many related works. Network topology methods use the structure of the subgraph around the user. It uses properties of the network like the number of common neighbors between the users and ranks users based on their similarity**.** Using just the topology of the network may not be enough metric to determine if a user connects with another. So in this project, we consider the user's attributes as well to make such recommendations for better results. As people in the real world do not just connect with one another based on how many common neighbors each have but they become friends with people based on how similar they are. In order to do this, we mine for the user's personality traits using his tweets/retweets data to obtain areas of the user's interest.

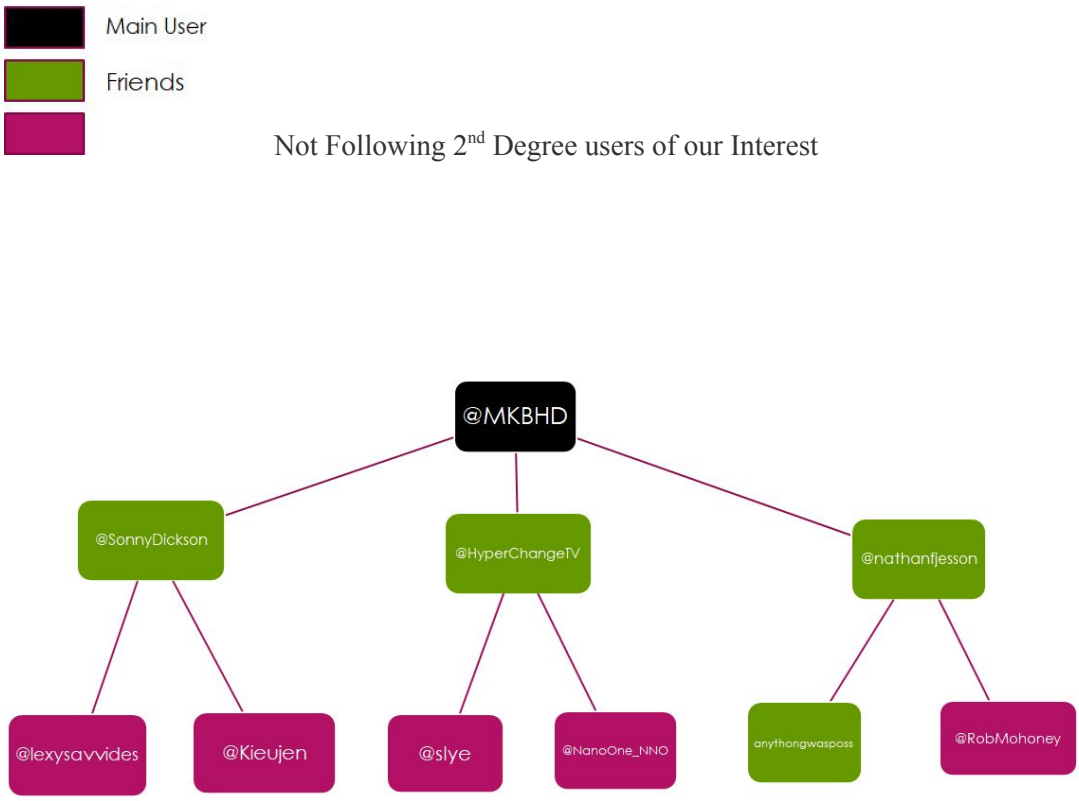


[2]

DATA COLLECTION

Twitter streaming Application Programming Interface provides a streaming API to stream real-time data and tweepy library for python. In order to access the twitter API, we should create a developer account that provides the access token which is used in authenticating the API call.

As mentioned in the scope, we are considering fixed main users and only 200 most recent users 2nd-degree users. Below is a graph that explains the 2nd-degree user (not a friend of the main user but a friend of main user's friend)



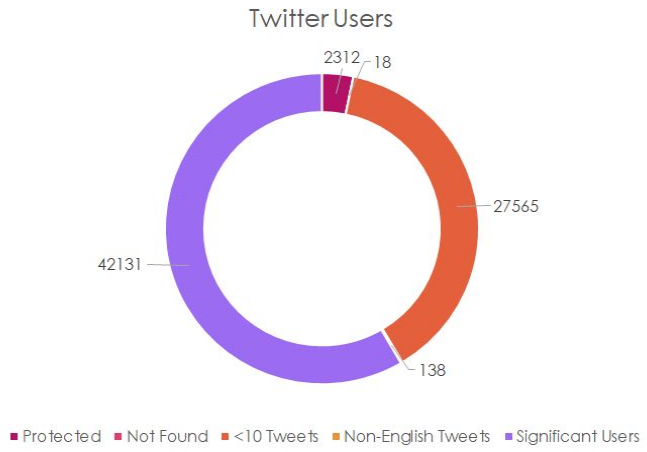
Statistics:

- Total number of friends of @MKBHD : 342
- Total number of 2nd-degree users: 72,164
  - Protected Users : 2312
  - Not Found: 18
  - User with < 10 tweets : 27,565

- ❏ Users with non-English tweets: 138

After removing protected, not found, <10 tweets, non-English twitter users, the significant workable tweets is 42131.

Number of tweets for the main user is 3204 tweets.



DATA PREPROCESSING:

Tweets data obtained is often very noisy with many acronyms, slang, non-English characters, emojis, URLs, etc. Data preprocessing was the most difficult than expected and time-consuming task as this involved many stages and needed a lot of fundamental cleaning of the data that is collected was necessary before analysis of data. Before we train any model we have to clean the tweet data by

- ➢ Detect and eliminate users with non-English tweets.
- ➢ Eliminating any url links in tweets.
- ➢ Separating hashtags and mentions from the text in a tweet.
- ➢ Using Unicode to detect emojis and eliminate them.
- ➢ Discarding any special characters or words with characters less than 3.
- ➢ Lemmatizing the text and then tokenizing it.
- ➢ Identify parts of speech and keep only nouns and verbs.

NLTK library has been used for lemmatization and tokenization. Spacy has been used to identify parts of speech. Textbob has been used for eliminating non-English words. Regular expressions are used to identify hashtags, mentions, and emojis and eliminate them.

Once we have all the text tokenized, we put together all the tokens obtained from the tweets of a single user as a single corresponding document. Then a dictionary is constructed over all the documents using the Gensim library's Dictionary. Using the tokens and the dictionary constructed we then generate a Bag of Words Corpus using doc2bow. We save this dictionary and corpus for ease use in the model building and training.

MODEL SELECTION AND TRAINING:

We can formulate finding the most similar candidates and suggesting friends as a document matching and ranking problem. Where tokens of each user is a separate document and we match it with the main user.
Two approaches can be used for this:

- **TF*ID model** which matches using the term frequency structures for each document.
- **Topic Modelling (LDA)**: This approach predicts k topics being discussed in each document and match documents with similar topics.

TF*ID Model:

In this model, we convert the documents into a vector by making use of our previously generated Bag of Words Corpus and TF-IDF vector for each document. We then create similar vector for the main user using all his 3204 tweets. Once we have the vectors, we compute the cosine similarity for all 2nd-degree users and the main user. Now we will sort the users based on the similarity score obtained.

Topic Modelling (LDA):

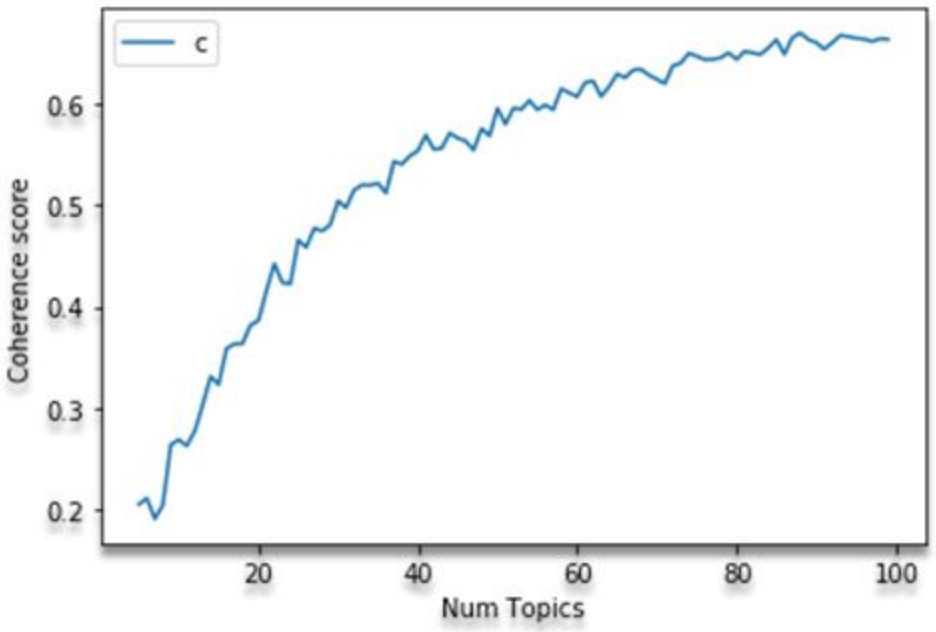
Latent Dirichlet Allocation (LDA) is an unsupervised generative model that assigns topic distributions to documents. LDA assumes that each document can be explained by the distribution of topics and each topic can be explained by the distribution of words. Once we specify the number of topics it assumes that the document is explained only by those number of topics and generates two variables:

- A distribution over topics for each document
- A distribution over words for each topic

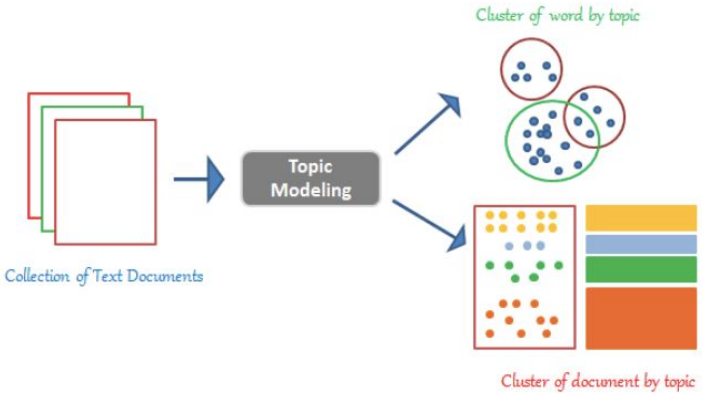
We can use the distribution over topics for each document to compare and retrieve similar documents. For comparing the documents we take the Jensen-Shannon distance between them. Jensen-Shannon distance is a method of measuring the similarity between two probability distributions. Using Jensen-Shannon we can get which documents are statistically "closer" (and therefore more similar), by comparing the divergence of their distributions.

With our model basics done we can train our LDA model on the data. However, LDA needs that we provide the number of topics in a document a-priori. The selection of number of topics is very important for the resulting topic distributions.

We measure the quality of a Topic distribution using **'coherence'**. Topic coherence measures each topic by scoring it based on calculating the degree of semantic similarity between words in the topic. It is often considered as a metric to evaluate the quality of a topic. In order to obtain a quality topic distribution, we can perform a coherence analysis. We take topics ranging from 5 to 100 and analyze the coherence scores for them.



We can see that scores flatline after 90 topics so 90 would give a good quality topic distribution. Now we train the LDA model for 90 topics on our data. Once trained we can predict the topic distributions for other user's tweets by passing the bag of words corpus for it to the model. We then compare the topic distributions of the main user and 2nd-degree users using Jensen-Shannon distance and sort them. This gives us the most similar users based on their tweets.



Overview of the LDA model.

[3]

Experimental Results:

TF*-ID:

The top 10 similar 2nd degree users recommended for the main user by TF*-ID score.

```
Most SImilar users are as follows:
Name                    score

'1. SteveLevitan     0.15528728067874908'

'2. bradybuzz     0.15149781107902527'

'3. CamAndCompany     0.15093570947647095'

'4. LoneSuspect     0.14409850537776947'

'5. maenadsnest     0.14183053374290466'

'6. tolleycasting     0.13477978110313416'

'7. LuciaFranceSays     0.127369225025177'

'8. GabbyGiffords     0.1244729682803154'

'9. shannonrwatts     0.12183833122253418'

'10. geekylonglegs     0.11530221998691559'
```

LDA:

Topics obtained can be seen to have converged well with all related terms in one topic.

In topic id 64, we see that the topics are about cars and in topic id 4, the topics are about games, which proves the correctness of the LDA model.

```
lda.show_topic(topicid=64, topn=20)

[('car', 0.088930525),
 ('race', 0.042851936),
 ('porsche', 0.022248613),
 ('lamborghini', 0.019692246),
 ('bmw', 0.017071104),
 ('driver', 0.015411248),
 ('ford', 0.014030993),
 ('drive', 0.013471677),
 ('new', 0.013082672),
 ('suv', 0.012773525),
 ('motor', 0.011486006),
 ('vehicle', 0.010402274),
 ('racing', 0.010303565),
 ('track', 0.009814409),
 ('road', 0.009607816),
 ('wheel', 0.00953608),
 ('engine', 0.009273619),
 ('ferrari', 0.008969438),
 ('auto', 0.0089062415),
 ('sport', 0.008764437)]
```

```
lda.show_topic(topicid=1, topn=20)

[('car', 0.039846744),
 ('china', 0.03732198),
 ('flight', 0.0330994),
 ('tesla', 0.030119138),
 ('model', 0.026335724),
 ('trip', 0.025633821),
 ('travel', 0.024099456),
 ('vehicle', 0.02114598),
 ('japan', 0.020112662),
 ('driver', 0.015841782),
 ('tokyo', 0.01463205),
 ('airport', 0.014080499),
 ('electric', 0.0131783625),
 ('uber', 0.011972773),
 ('nissan', 0.011518922),
 ('korea', 0.011330447),
 ('road', 0.010881123),
 ('aston', 0.010262737),
 ('production', 0.009332807),
 ('airline', 0.009161385)]
```

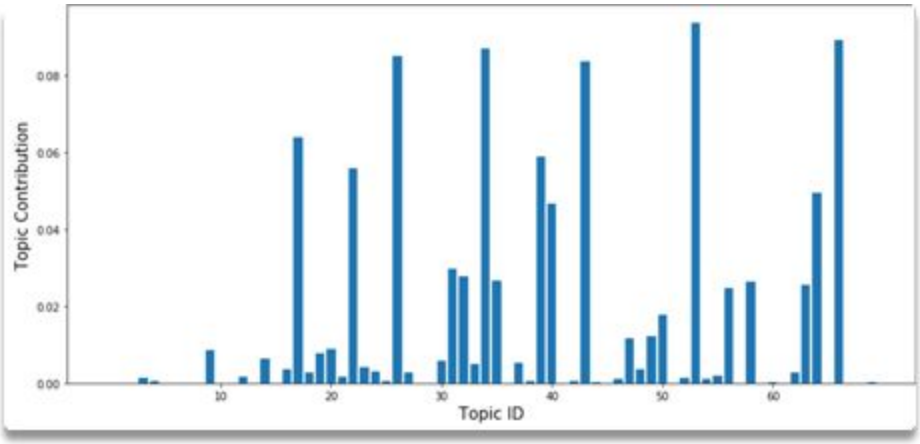
```
ctionary,corpus,lda = train_lda(df_short)

me to train LDA model on  16269 articles:

a.show_topic(topicid=4, topn=20)

'game', 0.12863763),
'stream', 0.041356992),
'play', 0.014230518),
'gaming', 0.013466893),
'xbox', 0.013353333),
'twitch', 0.013163195),
'player', 0.012194321),
'update', 0.0104355365),
'world', 0.010284799),
'pokemon', 0.010253668),
'ps4', 0.009159109),
'today', 0.008711291),
'nintendo', 0.007949925),
'streamer', 0.007242893),
'steam', 0.0071263677),
'games', 0.007113024),
'time', 0.0070354105),
'fortnite', 0.006866427),
'community', 0.006722191),
'mode', 0.0066952403)]
```

Topics distribution of the main user:



LDA Predictions:

mentions	hashtags	tokenized_text	handles	Similarity_scores
[[krishaamer], [ooliganpress], [wearehawthorne...	[[elearning, transmedia], [], [ai, artificiali...	[kinovan, focus, transmedia, storytelling, tha...	Kinovan_C	0.830906
[[], [], [], [], [], [], [], [], [], [], [], [], [...	[[], [], [], [], [], [], [], [], [], [], [], [], [...	[person, person, person, person, person, perso...	victorycini	0.828316
[[dick_in_milk, burstousobbing], [dick_in_milk...	[[], [], [], [], [], [], [], [], [], [], [], [], [...	[там, уже, вторая, вторая, еще, лучше, первой,...	kisimiaka_ururu	0.825571
[[TribLIVE, ForbesLife], [MarshProductio1, hag...	[[horsepower, musclecar], [carscoops, autoshri...	[thank, thank, thank, thank, thank, exoticar, ...	ExoticarsPgh	0.823557
[[], [], [], [], [], [], [], [], [], [], [], [], [...	[[], [FollowBack, TeamFollowBack], [], [TeamFo...	[background, page, guy, tip, follower, followe...	FocusedMark	0.823547
...	...	...	...	...
[[KuzcoLeDesma, UltimateTigers], [], [KuzcoLeD...	[[Ultimate], [], [], [], [], [], [], [], [], [...	[today, ultimate, frisbee, tomorrow, morning, ...	UltimateTigers	0.760467
[[thespunta], [], [], [ADampSandwich, MrNiceGu...	[[], [], [], [], [], [], [], [], [], [], [], [], [...	[human, water, update, mei, routine, add, bell...	RayApollo	0.760442
[[], [Asmar3313], [], [Asmar3313], [], [Asmar3...	[[], [], [checkra1n], [], [], [], [], [], [], [che...	[coupon, code, discount, app, device, apps, cy...	hacx	0.760338
[[warriorsvox, 957thegame], [epaschall, spidad...	[[Warriors], [Warriors, Jazz], [IAGLTY, Warrio...	[story, friendship, tonight, arc, childhood, f...	LaurenceScott	0.760233
[[sharkcat13], [TonyCelloTweets], [NotStephenB...	[[], [], [], [], [], [], [], [], [], [], [], [], [...	[sorry, thank, thank, canada, interest, team, ...	bublywater	0.760233

Evaluation:

To evaluate the models we add the friends of the main users to our data and predict the same number of suggestions.

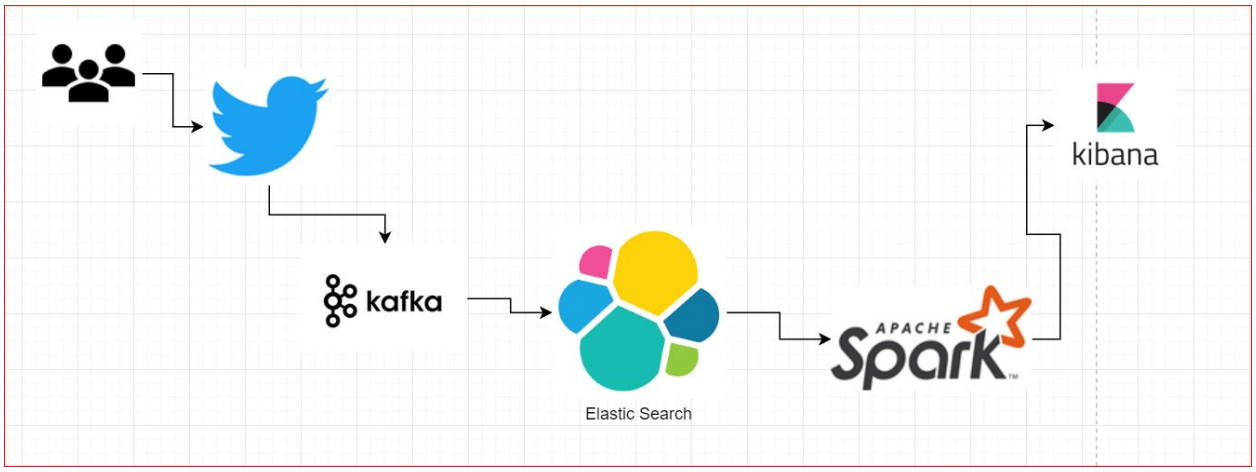
Using the TF*IDF method we added 342 and predicted 342 users of which True Positives were 5.

Using the Topic Modelling method the True Positives were 15 ~ precision of 4.61%

True positive and the precision value are low, this may be because of fewer data. Since LDA has correctly classified topics, with an increase in data, this model's performance will increase.

Dynamic Twitter Streaming Approach:

In this approach, real-time tweets are collected from Twitter. The tweets are retrieved for a given user using the Twitter Streaming API. Kafka Producer API read from the twitter stream ingests tweets into Kafka topics. The tweets from the Kafka topic is stored in elastic search with an index. Kibana can be used to analyze and view the twitter data. Now we retrieve the data from the elastic search perform data cleaning, tokenization, TF*IDF, Topic modeling(LDA) and produce LDA score as the output.



METHOD:

Create an APP on the Twitter developer account to get the consumer API keys and access token.

Download and install Kafka in the local machine. Start Zookeeper and Kafka from the Kafka directory. Kafka client will be running in localhost:9092 and zookeeper in localhost:2181

bin/zookeeper-server-start.sh config/zookeeper.properties

bin/kafka-server-start.sh config/server.properties

Create kafka topic “bigdataproject”.

bin/kafka-topics.sh --create --zookeeper localhost:2181 --replication-factor 1 --partitions 1 --topic bigdataproject

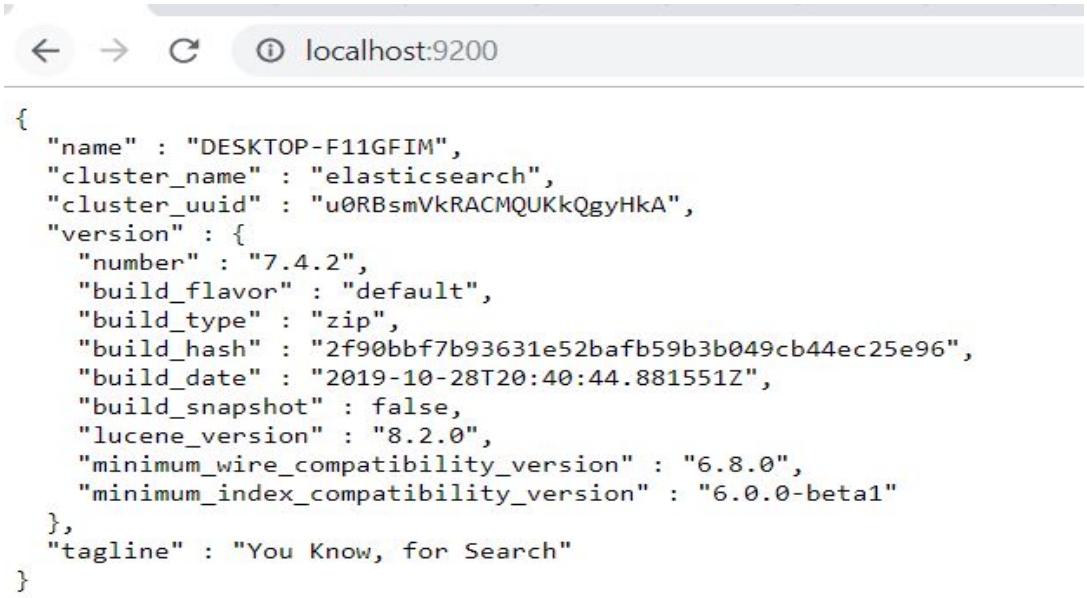
```
MINGW64:/c/data/kafka_2.12-2.3.0/kafka_2.12-2.3.0

Kiruba Dhayalan@DESKTOP-F11GFIM MINGW64 /c/data/kafka_2.12-2.3.0/kafka_2.12-2.3.
0
$ bin/kafka-topics.sh --create --zookeeper localhost:2181 --replication-factor 1 --partitions 1 --topic bigdataproject
log4j:ERROR Could not read configuration file from URL [file:/c/data/kafka_2.12-2.3.0/kafka_2.12-2.3.0/bin/../config/tools-log4j.properties].
java.io.FileNotFoundException: \c\data\kafka_2.12-2.3.0\kafka_2.12-2.3.0\bin\..\config\tools-log4j.properties (The system cannot find the path specified)
        at java.io.FileInputStream.open0(Native Method)
        at java.io.FileInputStream.open(Unknown Source)
        at java.io.FileInputStream.<init>(Unknown Source)
        at java.io.FileInputStream.<init>(Unknown Source)
        at sun.net.www.protocol.file.FileURLConnection.connect(Unknown Source)
        at sun.net.www.protocol.file.FileURLConnection.getInputStream(Unknown Source)
        at org.apache.log4j.PropertyConfigurator.doConfigure(PropertyConfigurator.java:557)
        at org.apache.log4j.helpers.OptionConverter.selectAndConfigure(OptionConverter.java:526)
        at org.apache.log4j.LogManager.<clinit>(LogManager.java:127)
        at org.slf4j.impl.Log4jLoggerFactory.<init>(Log4jLoggerFactory.java:66)
        at org.slf4j.impl.StaticLoggerBinder.<init>(StaticLoggerBinder.java:72)
        at org.slf4j.impl.StaticLoggerBinder.<clinit>(StaticLoggerBinder.java:45)
        at org.slf4j.LoggerFactory.bind(LoggerFactory.java:150)
        at org.slf4j.LoggerFactory.performInitialization(LoggerFactory.java:124)
        at org.slf4j.LoggerFactory.getILoggerFactory(LoggerFactory.java:412)
        at org.slf4j.LoggerFactory.getLogger(LoggerFactory.java:357)
        at com.typesafe.scalalogging.Logger$.apply(Logger.scala:48)
        at kafka.utils.Log4jControllerRegistration$.<init>(Logging.scala:25)
        at kafka.utils.Log4jControllerRegistration$.<clinit>(Logging.scala)
        at kafka.utils.Logging.$init$(Logging.scala:47)
        at kafka.admin.TopicCommand$.<init>(TopicCommand.scala:45)
        at kafka.admin.TopicCommand$.<clinit>(TopicCommand.scala)
        at kafka.admin.TopicCommand.main(TopicCommand.scala)
log4j:ERROR Ignoring configuration file [file:/c/data/kafka_2.12-2.3.0/kafka_2.12-2.3.0/bin/../config/tools-log4j.properties].
log4j:WARN No appenders could be found for logger (kafka.utils.Log4jControllerRegistration$).
log4j:WARN Please initialize the log4j system properly.
log4j:WARN See http://logging.apache.org/log4j/1.2/faq.html#noconfig for more info.
Created topic bigdataproject.
```

Download and install elastic search https://www.elastic.co/downloads/elasticsearch and kibana https://www.elastic.co/downloads/kibana.

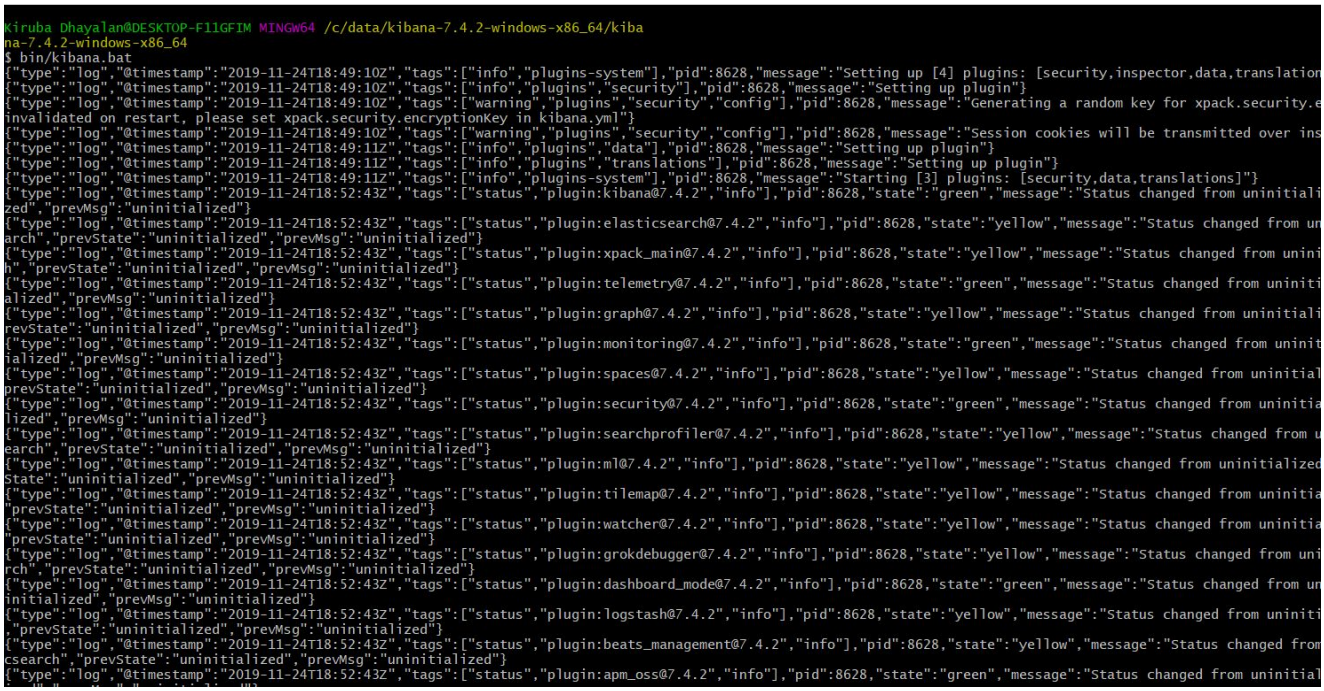
bin/elasticsearch.bat

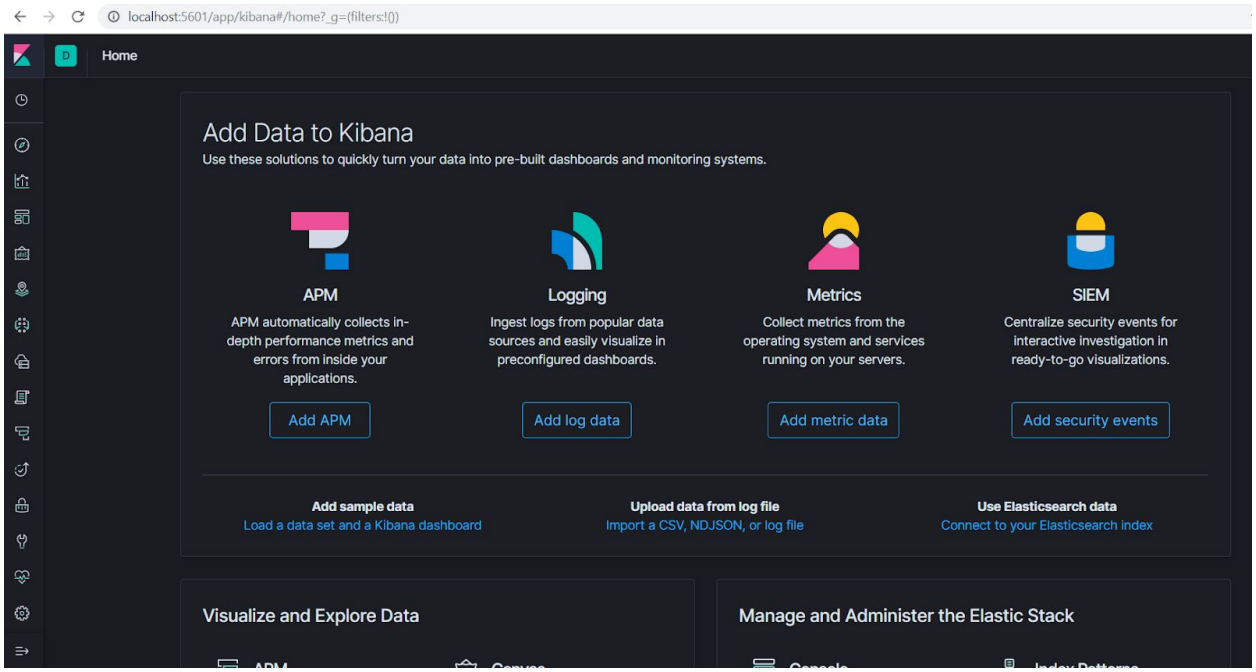
```
Kiruba Dhayalan@DESKTOP-F11GFIM MINGW64 /c/data/elasticsearch-7.4.2
$ bin/elasticsearch.bat
OpenJDK 64-Bit Server VM warning: Option UseConcMarkSweepGC was deprecated in version 9.0 and will likely be removed in a future release.
[2019-11-24T12:42:57,263][INFO ][o.e.e.NodeEnvironment    ] [DESKTOP-F11GFIM] using [1] data paths, mounts [[Windows (C:)]], net usable_space [370.9gb], net total_space [506gb], types [NTFS]
[2019-11-24T12:42:57,298][INFO ][o.e.e.NodeEnvironment    ] [DESKTOP-F11GFIM] heap size [990.7mb], compressed ordinary object pointers [true]
[2019-11-24T12:42:57,713][INFO ][o.e.n.Node               ] [DESKTOP-F11GFIM] node name [DESKTOP-F11GFIM], node ID [oG5zmCYwSpyfoyw29FufPA], cluster name [elasticsearch]
[2019-11-24T12:42:57,714][INFO ][o.e.n.Node               ] [DESKTOP-F11GFIM] version[7.4.2], pid[13440], build[default/zip/2f90bbf7b93631e52bafb59b3b049cb44ec25e96/2019-10-28T20:40:44.881551Z], OS[Windows 10/1
0.0/amd64], JVM[AdoptOpenJDK/OpenJDK 64-Bit Server VM/13/13+33]
[2019-11-24T12:42:57,716][INFO ][o.e.n.Node               ] [DESKTOP-F11GFIM] JVM home [C:\data\elasticsearch-7.4.2\jdk]
[2019-11-24T12:42:57,717][INFO ][o.e.n.Node               ] [DESKTOP-F11GFIM] JVM arguments [-Xms1g, -Xmx1g, -XX:+UseConcMarkSweepGC, -XX:CMSInitiatingOccupancyFraction=75, -XX:+UseCMSInitiatingOccupancyOnly, -
Des.networkaddress.cache.ttl=60, -Des.networkaddress.cache.negative.ttl=10, -XX:+AlwaysPreTouch, -Xss1m, -Djava.awt.headless=true, -Dfile.encoding=UTF-8, -Djna.nosys=true, -XX:-OmitStackTraceInFastThrow, -Dio.n
etty.noUnsafe=true, -Dio.netty.noKeySetOptimization=true, -Dio.netty.recycler.maxCapacityPerThread=0, -Dio.netty.allocator.numDirectArenas=0, -Dlog4j.shutdownHookEnabled=false, -Dlog4j2.disable.jmx=true, -Djava
.io.tmpdir=C:\Users\KIRUBA~1\AppData\Local\Temp\elasticsearch, -XX:+HeapDumpOnOutOfMemoryError, -XX:HeapDumpPath=data, -XX:ErrorFile=logs/hs_err_pid%p.log, -Xlog:gc*,gc+age=trace,safepoint:file=logs/gc.log:utct
ime,pid,tags:filecount=32,filesize=64m, -Djava.locale.providers=COMPAT, -Dio.netty.allocator.type=unpooled, -XX:MaxDirectMemorySize=536870912, -Delasticsearch, -Des.path.home=C:\data\elasticsearch-7.4.2, -Des.p
ath.conf=C:\data\elasticsearch-7.4.2\config, -Des.distribution.flavor=default, -Des.distribution.type=zip, -Des.bundled_jdk=true]
[2019-11-24T12:43:13,394][INFO ][o.e.p.PluginsService     ] [DESKTOP-F11GFIM] loaded module [aggs-matrix-stats]
[2019-11-24T12:43:13,394][INFO ][o.e.p.PluginsService     ] [DESKTOP-F11GFIM] loaded module [analysis-common]
[2019-11-24T12:43:13,396][INFO ][o.e.p.PluginsService     ] [DESKTOP-F11GFIM] loaded module [data-frame]
[2019-11-24T12:43:13,396][INFO ][o.e.p.PluginsService     ] [DESKTOP-F11GFIM] loaded module [flattened]
[2019-11-24T12:43:13,397][INFO ][o.e.p.PluginsService     ] [DESKTOP-F11GFIM] loaded module [frozen-indices]
[2019-11-24T12:43:13,398][INFO ][o.e.p.PluginsService     ] [DESKTOP-F11GFIM] loaded module [ingest-common]
[2019-11-24T12:43:13,398][INFO ][o.e.p.PluginsService     ] [DESKTOP-F11GFIM] loaded module [ingest-geoip]
[2019-11-24T12:43:13,398][INFO ][o.e.p.PluginsService     ] [DESKTOP-F11GFIM] loaded module [ingest-user-agent]
[2019-11-24T12:43:13,399][INFO ][o.e.p.PluginsService     ] [DESKTOP-F11GFIM] loaded module [lang-expression]
[2019-11-24T12:43:13,399][INFO ][o.e.p.PluginsService     ] [DESKTOP-F11GFIM] loaded module [lang-mustache]
[2019-11-24T12:43:13,399][INFO ][o.e.p.PluginsService     ] [DESKTOP-F11GFIM] loaded module [lang-painless]
[2019-11-24T12:43:13,401][INFO ][o.e.p.PluginsService     ] [DESKTOP-F11GFIM] loaded module [mapper-extras]
[2019-11-24T12:43:13,401][INFO ][o.e.p.PluginsService     ] [DESKTOP-F11GFIM] loaded module [parent-join]
[2019-11-24T12:43:13,402][INFO ][o.e.p.PluginsService     ] [DESKTOP-F11GFIM] loaded module [percolator]
[2019-11-24T12:43:13,402][INFO ][o.e.p.PluginsService     ] [DESKTOP-F11GFIM] loaded module [rank-eval]
[2019-11-24T12:43:13,402][INFO ][o.e.p.PluginsService     ] [DESKTOP-F11GFIM] loaded module [reindex]
[2019-11-24T12:43:13,403][INFO ][o.e.p.PluginsService     ] [DESKTOP-F11GFIM] loaded module [repository-url]
[2019-11-24T12:43:13,404][INFO ][o.e.p.PluginsService     ] [DESKTOP-F11GFIM] loaded module [search-business-rules]
[2019-11-24T12:43:13,405][INFO ][o.e.p.PluginsService     ] [DESKTOP-F11GFIM] loaded module [spatial]
[2019-11-24T12:43:13,406][INFO ][o.e.p.PluginsService     ] [DESKTOP-F11GFIM] loaded module [transport-netty4]
```



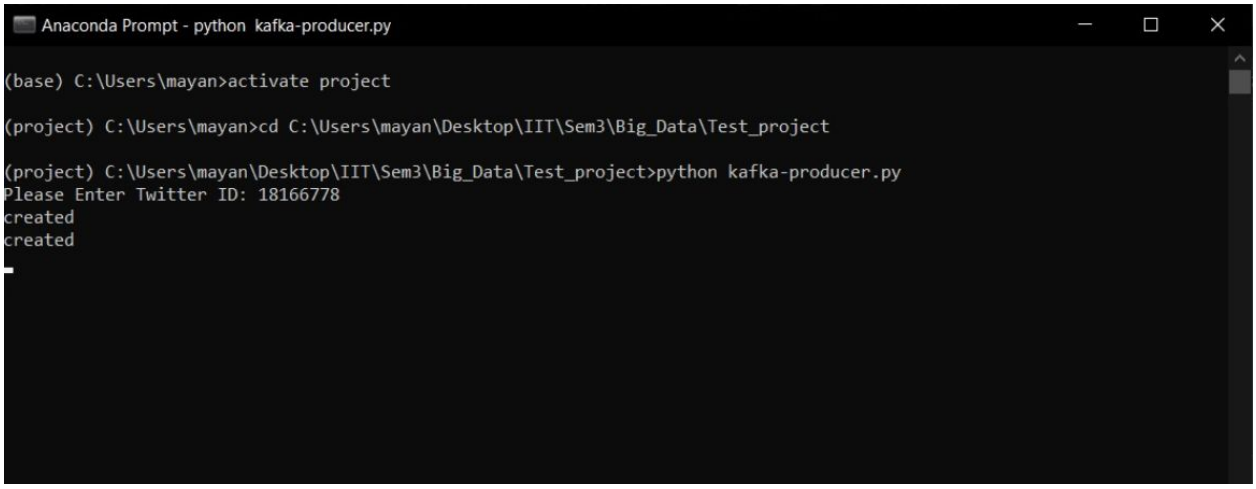
In kibana.yml configure elastic search host to http://localhost:9200/

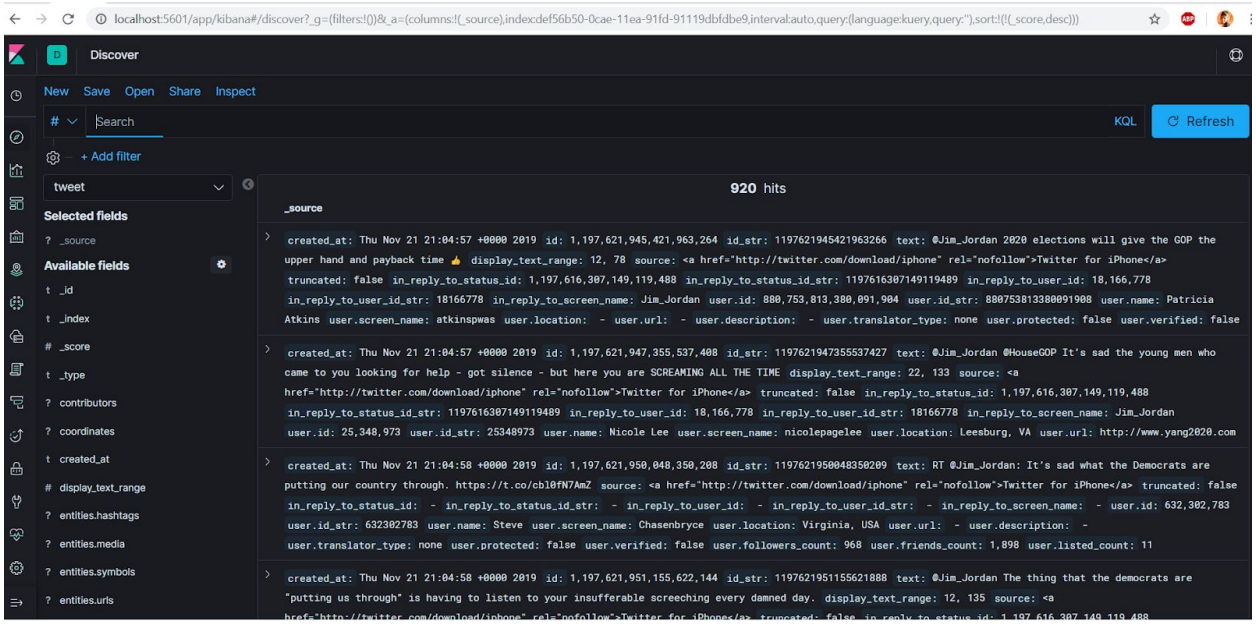
bin/kibana.bat



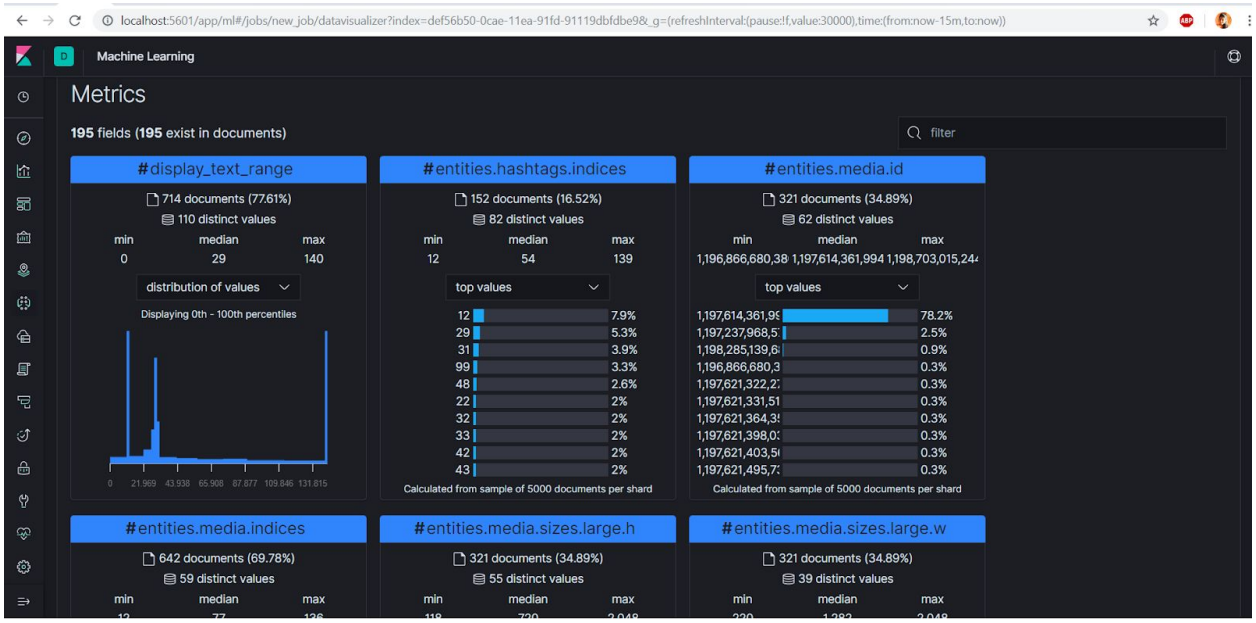


Once the above setup is done, we execute the kafka producer which created the kafka client, ingest tweet from Twitter stream and stores them in elastic search. We can view the data stored in elastic search through kibana by searching the index in which the tweet objects are stored.

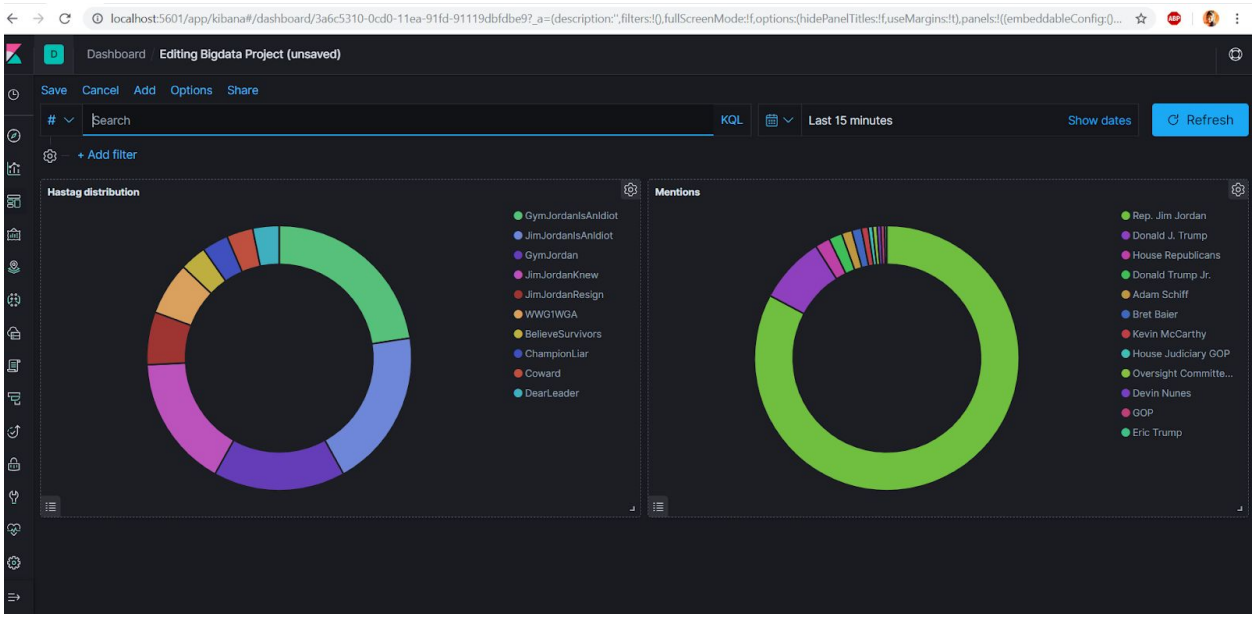




In addition to viewing data in elastic search we can perform analysis on how each field in the object is distributed. We can also perform statistical analysis like mean, median, average of each field by using the machine learning tool in kibana.



The dashboard can be created to save the visualization results for later reference. Below are the top 10 hashtag and top 12 user mentions used by twitter id “18166778”



Lessons Learnt and Future Work

- ➢ It can be seen that using Topic modeling methods provide friends suggestions but does not perform too well. However, if more access to data is obtained the model is bound to perform much better.
- ➢ Also, the scores obtained can be used in tandem with topological link prediction methods to obtain strong predictions.
- ➢ Sentiment Analysis can be done on the LDA, which will give better predictions.
- ➢ Rate limit restriction of Twitter API has to be considered and have to allocate enough time for data collection.

References:

[1] Rodrigues, Anisha P., and Niranjan N. Chiplunkar. “**Real-Time Twitter Data Analysis Using Hadoop Ecosystem.**” *Cogent Engineering* 5, no. 1 (2018). https://doi.org/10.1080/23311916.2018.1534519.

[2] P. Cooper, “25 Twitter Statistics All Marketers Should Know in 2020,” *Hootsuite Social Media Management*
https://blog.hootsuite.com/twitter-statistics/

[3] A. Naskar, “Latent Dirichlet Allocation for Beginners: A high-level overview,” *Think Infi* https://www.thinkinfi.com/2019/02/lda-theory.html.

GitLink : https://github.com/KirubaDhayalan/CSP554-Big_data_project

Appendix:

Method for extracting text:

#The raw tweet is sent as argument to this method

```
def extract_text(tweet):
    regex = r"http\S+"
    subset = ""
    emoji_pattern = re.compile("["
        u"\U0001F600-\U0001F64F"  # emoticons
        u"\U0001F300-\U0001F5FF"  # symbols & pictographs
        u"\U0001F680-\U0001F6FF"  # transport & map symbols
        u"\U0001F1E0-\U0001F1FF"  # flags (iOS)
                           "]+", flags=re.UNICODE)
    clean = re.sub(regex, subset, tweet)
    clean = emoji_pattern.sub(subset, clean).strip()
    return clean
```

Method for extracting mentions:

#The raw tweet is sent as argument to this method

```
def extract_mentions(tweet):
    mentions = []
    for k,v in tweet.items():
        if(k == 'user_mentions' and v != None):
            for i in v:
                mentions.append(i['screen_name'])
    return mentions
```

Method for extracting hashtags:

```
def extract_hashtags(tweet):
    hashtags = []
    for k,v in tweet.items():
        if(k == 'hashtags' and v != None):
            for i in v:
                hashtags.append(i['text'])
    return hashtags
```

Data cleaning and LDA analysis in pyspark shell

```
      ____              __
     / __/__  ___ _____/ /__
    _\ \/ _ \/ _ `/ __/  '_/
   /__ / .__/\_,_/_/ /_/\_\   version 2.2.0.2.6.4.0-91
      /_/

Using Python version 2.6.6 (r266:84292, Aug 18 2016 15:13:37)
SparkSession available as 'spark'.
>>> spark
<pyspark.sql.session.SparkSession object at 0x13b0350>
>>> import pyspark
>>> import string
>>> from pyspark import SparkContext
>>> from pyspark.sql import SQLContext
>>> from pyspark.mllib.util import MLUtils
>>> from pyspark.sql.types import *
>>> from pyspark.ml.feature import CountVectorizer, CountVectorizerModel, Tokenizer, RegexTokenizer, StopWordsRemover
>>> data_df = spark.read.csv('/user/maria_dev/part-00003-e30fd2db-bd23-40e8-9f52-040c4dd8ad74-c000.csv')
>>> data_df.printSchema()
root
 |-- _c0: string (nullable = true)
 |-- _c1: string (nullable = true)
 |-- _c2: string (nullable = true)

>>> data_df.show()
+--------------------+--------------------+--------------------+
|                 _c0|                 _c1|                 _c2|
+--------------------+--------------------+--------------------+
|[[], [], [], [], ...|[[Jon4Lakers, Hey...|[ Jon4Lakers HeyM...|
+--------------------+--------------------+--------------------+

>>> data_df = data_df.selectExpr("_c0 as hashtags", "_c1 as mentions","_c2 as text")
>>> data_df.show()
+--------------------+--------------------+--------------------+
|            hashtags|            mentions|                text|
+--------------------+--------------------+--------------------+
|[[], [], [], [], ...|[[Jon4Lakers, Hey...|[ Jon4Lakers HeyM...|
+--------------------+--------------------+--------------------+

>>> labels=["hashtags","mentions","text"]
>>> data_df.printSchema()
root
 |-- hashtags: string (nullable = true)
 |-- mentions: string (nullable = true)
 |-- text: string (nullable = true)

>>> _
```

```
>>> data_df.printSchema()
root
 |-- hashtags: string (nullable = true)
 |-- mentions: string (nullable = true)
 |-- text: string (nullable = true)

>>> tweets = data_df.rdd.map(lambda x : x['text']).filter(lambda x: x is not None)
>>> tweets
PythonRDD[18] at RDD at PythonRDD.scala:48
>>> tokenizer = Tokenizer(inputCol="text", outputCol="words")
>>> wordsDataFrame = tokenizer.transform(data_df)
>>> wordsDataFrame.show()
+--------------------+--------------------+--------------------+--------------------+
|            hashtags|            mentions|                text|               words|
+--------------------+--------------------+--------------------+--------------------+
|[[], [], [], [], ...|[[Jon4Lakers, Hey...|[ Jon4Lakers HeyM...|[[, jon4lakers, h...|
+--------------------+--------------------+--------------------+--------------------+

>>> #remove 20 most occuring documents, documents with non numeric characters, and documents with <= 3 characters
... cv_tmp = CountVectorizer(inputCol="words", outputCol="tmp_vectors")
>>> cv_tmp_model = cv_tmp.fit(wordsDataFrame)
>>> cv_tmp_model
CountVectorizer_42fcb3b57059fe770b58
>>> top20 = list(cv_tmp_model.vocabulary[0:20])
>>> more_than_3_charachters = [word for word in cv_tmp_model.vocabulary if len(word) <= 3]
>>> contains_digits = [word for word in cv_tmp_model.vocabulary if any(char.isdigit() for char in word)]
>>> #Add additional stopwords
... stopwords = []
>>> #combine three stop words
... stopwords = stopwords + top20 + more_then_3_charachters + contains_digits
Traceback (most recent call last):
  File "<stdin>", line 2, in <module>
NameError: name 'more_then_3_charachters' is not defined
>>> stopwords = stopwords + top20 + more_than_3_charachters + contains_digits
>>> stopwords
[u'', u',', u'the', u'i', u'a', u'to', u'it', u's', u'and', u'is', u'rt', u'this', u'of', u'in', u'that', u'you', u'for', u'new', u'video', u'on', u'', u',', u'the', u'i', u'a', u'to', u'it', u's', u'and', u'is', u'rt', u'of', u'in', u'you', u'for', u'new', u'on', u't', u'but', u'my', u'so', u'be', u'pro', u'm', u'can', u'all', u'if', u'not', u'we', u'at', u'was', u'are', u'up', u'get', u'one', u're', u'me', u'an', u'no', u'has', u'now', u've', u'don', u'out', u'll', u'as', u'2', u'see', u'or', u'by', u'do', u'got', u'1', u'd', u'4', u'11', u'3', u'man', u'how', u'10', u'any', u'way', u'5', u'too', u'car', u'day', u'vs', u'7', u'lot', u'it,', u'he', u'yet', u'yes', u'had', u'6', u'mac', u'ok', u'oh', u'use', u's10', u'go', u'our', u'why', u'who', u'let', u'buy', u'hah', u'off', u'100', u'top', u'us', u'did', u'hey', u'0', u'fun', u'tim', u'app', u'red', u'far', u'amp', u'big', u'say', u'000', u'e', u'c', u'20', u'bad', u'12', u'8', u'nah', u'4k', u'x', u'old', u'5g', u'o', u'ram', u'him', u'ios', u'try', u'7t', u'yup', u'may', u'p30', u'y', u'its', u'hot', u'am', u'lg', u'ai', u'3a', u'add', u'set', u'his', u'9', u'gt', u'few', u'max', u'13', u'guy', u'usb', u'g', u'won', u'win', u'ces', u'ad', u'til', u'run', u'30', u'isn', u'h', u'box', u'p', u'rog', u'lol', u'50', u'wow', u'bit', u'r', u'99', u'put', u'16', u'etc', u'end', u'60', u'low', u'cut', u'15', u'hi', u'own', u'nyc', u'per', u'she', u'n', u'rip', u'bro', u'70', u'b', u'iam', u'ran', u'via', u'pay', u'q', u'ads', u'ask', u'k', u'jon', u'is,', u'ago', u'ii', u'os', u'1,', u'em', u'24', u'htc', u'god', u'due', u'saw', u'ain', u'tv', u'3,', u'855', u'yo', u'two', u'ny', u'her', u'xdr', u'tho', u'g8', u'boy', u'gps', u'mic', u'25', u'hd', u'bar', u'gas', u'7th', u'28', u'6pm', u'pre', u'4,', u'heh', u'ha,', u'me,', u'fly', u'23', u'k20', u'air', u'40', u'on,', u'18', u'ui', u'eye', u'6t', u'bet', u'mix', u'm
```

```
>>> #The corpus of tweets
... cv_tmp_model.vocabulary
[u'', u',', u'the', u'i', u'a', u'to', u'it', u's', u'and', u'is', u'rt', u'this', u'of', u'in', u'that', u'you', u'for', u'new', u'video', u'on', u't', u'but', u'my', u'with', u'just', u'so', u'be', u'pro', u'mkbhd', u'm', u'can', u'all', u'have', u'if', u'not', u'what', u'we', u'they', u'at', u'tesla', u'was', u'are', u'apple', u'up', u'get', u'from', u'one', u're', u'there', u'me', u'like', u'iphone', u'an', u'no', u'time', u'more', u'has', u'now', u'pixel', u've', u'don', u'uploading', u'your', u'out', u'll', u'youtube', u'lol,', u'will', u'as', u'2', u'see', u'still', u'about', u'tech', u'snazzyq', u'elonmusk', u'phone', u'camera', u'galaxy', u'tldtoday', u'people', u'first', u'than', u'or', u'google', u'good', u'review', u'think', u'by', u'oneplus', u'do', u'got', u'1', u'd', u'here', u'samsung', u'been', u'love', u'4', u'11', u'right', u'back', u'better', u'3', u'year', u'know', u'2019', u'austinnotduncan', u'man', u'coming', u'gonna', u'want', u'when', u'make', u'also', u'these', u'much', u'never', u'every', u'some', u'how', u'would', u'actually', u'only', u'10', u'any', u'way', u'5', u'thanks', u'too', u'yeah', u'need', u'next', u'car', u'great', u'videos', u'day', u'today', u'really', u'same', u'watch', u'vs', u'7', u'uravgconsumer', u'display', u'pretty', u'well', u'over', u'literally', u'lot', u'impressions', u'jon4lakers', u'super', u'thing', u'definitely', u'last', u'real', u'should', u'other', u'ever', u'model', u'it,', u'android', u'take', u'linustech', u'most', u'everything', u'best', u'he', u'them', u'detroitborg', u'shot', u'yet', u'ready', u'already', u'gotta', u'even', u'yes', u'could', u'samsheffer', u'had', u'superscientific', u'6', u'mac', u'ipad', u'caseyneistat', u'ijustine', u'things', u'ok', u'then', u'getting', u'those', u'hah,', u'oh', u'seen', u'damn', u'into', u'battery', u'casey', u'everyone', u'airpods', u'incredible', u'cameras', u'supersaf', u'use', u'absolutely', u's10', u'their', u'live', u'go', u'mean', u'because', u'matte', u'our', u'why', u'phones', u'andymanganelli', u'update', u'twitter', u'who', u'happy', u'looks', u'whatsinside', u'look', u'black', u'zacksjerryrig', u'week', u'let', u'makes', u'many', u'buy', u'finally', u'sure', u'wait', u'exactly', u'nope', u'hours', u'fold', u'shen', u'where', u'hah', u'haha,', u'congrats', u'though', u'something', u'off', u'event', u'artemr', u'100', u'name', u'reneritchie', u'top', u'note', u'whoa', u'us', u'did', u'hey', u'keep', u'huawei', u'brandonjhavard', u'mode', u'heymarkl', u'game', u'0', u'unboxtherapy', u'full', u'dave2d', u'soon', u'little', u'years', u'didn', u'haven', u'fun', u'made', u'canoopsy', u'might', u'electric', u'verge', u'before', u'yup,', u'down', u'2020', u'tim', u'true', u'which', u'quality', u'app', u'minutes', u'photo', u'maybe', u'couple', u'red', u'always', u'probably', u'saradietschy', u'screen', u'far', u'looking', u'color', u'smartphone', u'fingers', u'themrmobile', u'alright', u'test', u'believe', u'amp', u'does', u'internet', u'going', u'long', u'less', u'hard', u'jonyiveparody', u'wvfrm', u'big', u'airpower', u'nothing', u'mrwhosetheboss', u'starting', u'bensullins', u'studio', u'give', u'say', u'enough', u'vyyyper', u'000', u'dope', u'e', u'trying', u'feel', u'almost', u'c', u'photos', u'mine', u'product', u'night', u'kinda', u'charging', u'favorite', u'dude', u'shoutout', u'20', u'both', u'anything', u'since', u'podcast', u'wireless', u'very', u'gets', u'this,', u'after', u'software', u'bring', u'doesn', u'bad', u'second', u'12', u'8', u'making', u'life', u'works', u'took', u'ultimate', u'between', u'design', u'nah', u'thank', u'waiting', u'range', u'huge', u'4k', u'marques', u'whole', u'talk', u'x', u'point', u'doing', u'future', u'around', u'times', u'sometimes', u'jacksonhvisuals', u'wouldn', u'markgurman', u'empireultimate', u'else', u'work', u'old', u'someone', u'price', u'world', u'worth', u'birthday', u'nbt88yt', u'same,', u'being', u'plus', u'fact', u'honestly', u'now,', u'miles', u'vladsavov', u'2018', u'were', u'5g', u'o', u'faster', u'ram', u'fast', u'case', u'roadster', u'porsche', u'least', u'quick', u'left', u'perfect', u'him', u'another', u'official', u'ios', u'months', u'that,', u'working', u'start', u'drive', u'days', u'tell', u'entire', u'jack', u'thought', u'imagine', u'show', u'please', u'features', u'current', u'amazing', u'mkbhd,', u'spotify', u'try', u'anyone', u'folding', u'again', u'episode', u'found', u'dark', u'7t', u'nobody', u'yup', u'backlon', u'version', u'talking', u'news', u'tonight', u'may', u'p30', u'team', u'weird', u'y', u'mate', u'cook', u'part', u'straight', u'its', u'starts', u'refresh', u'said', u'cool', u'hope', u'needs', u'through', u'dropping', u'anywhere', u'livestream', u'theaudl', u'done', u'find', u'person', u'stuff', u'hot', u'hands', u'power', u'editing', u'am', u'interior', u'either', u'testing', u'lg', u'wish', u'king', u'ai', u'empire', u'behind', u'3a', u'add', u'wrong', u'free', u'set', u'lens', u'goes', u'piamuehlenbeck', u'hate', u'wanted', u'awesome', u'smartphones', u'his', u'upload', u'light', u'9', u'stage', u'crazy', u'exist', u'season', u'apollo', u'gt', u'final', u'seconds', u'come', u'service', u'headphone', u'couldn', u'microsoft', u'today,', u'few', u'high', u'actual', u'speed', u'original', u'guess', u'wide', u'triple', u'problem', u'plan', u'change', u'used', u'word', u'max', u'elon', u'tweet', u'major', u'taking', u'dream', u'course', u'tiny', u'unboxing', u'hardware', u'shots', u'early', u'13', u'dbrand', u'time,', u'guy', u'one,', u'rewind', u'hell', u'usb', u'without', u'g', u'late', u'clean', u'instagram', u'open', u'won', u'explained', u'different', u'glass', u'cars', u'bigger', u'win', u'shoot', u'while', u'ces', u'forward', u'ad', u'universeice', u'read', u'hoping', u'true,', u'card', u'improvements', u'using', u'feature', u'hear', u'york', u'ultrawide', u'special', u'til', u'dual', u'thinking', u'retro', u'run', u'music', u'asus', u'30', u'updates', u'officially', u'month', u'weekend', u'isn', u'performance', u'having', u'drop', u'brownlee', u'sheesh', u'h', u'laptop', u'white', u'box', u'yes,', u'seems', u'site', u'focus', u'truly', u'tho,', u'break', u'hand', u'listen', u'p', u'minute', u'went', u'seeing', u'hour', u'weeks', u'shows', u'120hz', u'build', u'space', u'pro,', u'home', u'rest', u'snbrownlee', u'rog', u'past', u'face', u'yet,', u'lol', u'50', u'touch', u'wow', u'bit', u'april', u'soon,', u'wallpaper', u'welcome', u'lease', u'r', u'imac', u'99', u'tried', u'thoughts', u'support', u'multiple', u'oled', u'must', u'put', u'tailosivetech', u'incoming', u'level', u'experience', u'selfie', u'sick', u'biggest', u'fair', u'difference', u'wojespn', u'wins', u'wants', u'watching', u'guys', u'link', u'16', u'until', u'extra', u'later', u'september', u'etc', u'xiaomi', u'bunch', u'aren', u'happens', u'footage', u'mrbeastyt', u'randomfrankp', u'luck', u'remember', u'end', u'iliasvant', u'knows', u'sorry', u'subtle', u'side', u'burn', u'60', u'surface', u'questions', u'launch', u'door', u'mobile', u'leaks', u'low', u'extremely', u'wasn', u'cut', u'phonebuff', u'store', u'appreciat
```

```
>>> tokens = tweets.map( lambda document: document.strip().lower()).map( lambda document: re.split(" ", document)).map( lambda word: [x for x in word if x.isa
lpha()]).map( lambda word: [x for x in word if len(x) > 3] ).map( lambda word: [x for x in word if x not in stopwords]).zipWithIndex()
>>> df_txts = sqlContext.createDataFrame(tokens,["list_of_words",'index'])
>>> df_txts.show()
+--------------------+-----+
|       list_of_words|index|
+--------------------+-----+
|[heymarkl, within...|    0|
+--------------------+-----+
```

```
  File "/usr/hdp/current/spark2-client/python/pyspark/sql/utils.py", line 79, in deco
    raise IllegalArgumentException(s.split(': ', 1)[1], stackTrace)
pyspark.sql.utils.IllegalArgumentException: u'requirement failed: The vocabulary size should be > 0. Lower minDF as necessary.'
>>> cv = CountVectorizer(inputCol="list_of_words",outputCol="raw_features", vocabSize=100, minDF=1.0)
>>> cvmodel = cv.fit(df_txts)
>>> from pyspark.ml.feature import CountVectorizer , IDF
>>> result_cv = cvmodel.transform(df_txts)
>>> idf = IDF(inputCol="raw_features", outputCol="features")
>>> idfModel = idf.fit(result_cv)
>>> result_tfidf = idfModel.transform(result_cv)
>>> _
```