TRUMP TWEET SEARCH ENGINE

GROUP 6
Nagarathinam Ravi Sandhiya - 1036554
Surja Chaudhuri - 0977601
Camilo Montenegro - 1430963
Jorrit Bakker - 1039401

Contents

1 Introduction

At the time of writing president Donald Trump has placed over 41.000 tweets on `twitter.com`. These tweets contain a lot of information and reflect what opinion Trump has on certain topics. Because Trump has such a high function and the power to affect a whole nation it is important for people to see what he is up to and what his opinion is on certain topics. Tweets on twitter can only be viewed chronological and because of the volume, are hard to filter as a person. We want to bring a tool that allows anyone to quickly search for a topic and find all relevant thoughts about this topic that Trump tweeted. The tool will also allow the user to quickly see if the overall collection on this topic is positive or negative.
The project web application can be found at this Url: `https://irtrump.herokuapp.com/` and the repository with the components can be found in this Github: `https://github.com/ca-montenegro/TweetTrumpEngine`

2 Motivation and goal

The main goal is to design and construct a Donald Trumps Tweets search engine system that retrieves tweets that users are likely to find relevant to their queries.

3 Architecture

As is shown in figure 1 the search engine overall architecture is compound by two main parts, back and front end. Initially, the back-end consist of a NoSql database containing 36,000 tweets of Donald Trump and the index file that contain the inverted index, this information was hosted using MongoDB. At this point is important to mention that the gathered tweets are in the date range of 02/08/2011 - 25/02/2019. Furthermore, the search engine retrieve the mention database and make use as main programming language Python to perform the functionality of the four main components and return the best possible result to the user. This four important components of the search engine are Inverted Indexing `https://v2.overleaf.com/project/5c9b477bef900815bdcc0e65`, Sentiment Analysis, Clustering, and Topic Modeling. However, with the purpose of visualize the best results in a simple way we developed a ReactJS front-end component with ExpressJS as file server.

First, inverted Indexing is a common technique for building IR system, it consist of a unique word list that contains a reference to the document id in which the current word occurs. Secondly, Sentiment Analysis is a contextual mining of text that studies the data and extracts the social sentiments of the user. It is used for analyzing and classifying Trump tweets positive or negative inclination towards a topic. In addition, Clustering is a unsupervised machine learning technique that aims to group words as clusters such that the words in each cluster are similar to each other when compared to other clusters. The goal behind using this Machine Learning technique is to discover the key words that Trump make use within a same topic. Topic Modeling is a probabilistic method for analyzing the words in a database and discovering the latent topics and how these topics are related to each terms in a document. Finally, the result will be a website that make use of HTML and JavaScript. Using this website, the user can search for the query and the most relevant documents will be shown using the previously discussed components.

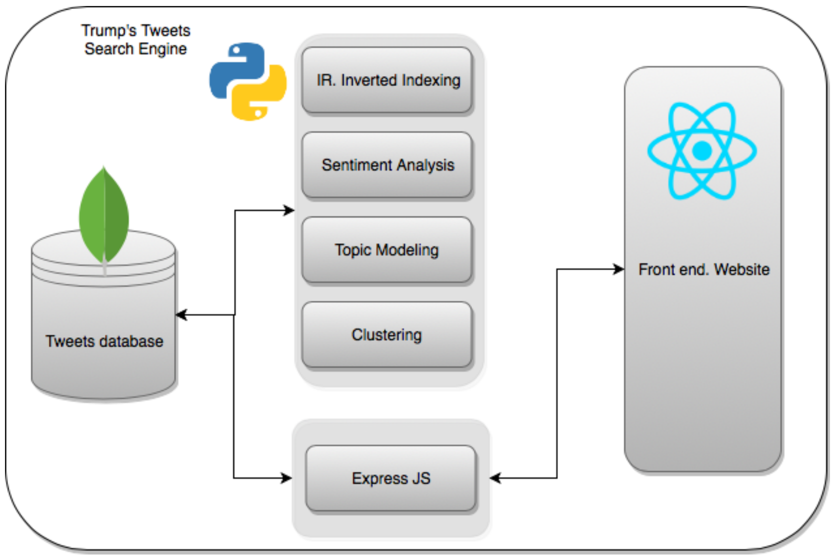


Figure 1: System Architecture overall

4 Prior work

Initially the main data source of the project is Twitter, here is where Donald Trump wrote down all the required information for the project. After searching for an easy way to scrape the tweets we found a tool with similar goals as our project. Trump Twitter Archive is a search engine that gathers all historical Trump information and show relevant data grouped by various topics this website tool can be found at `http://www.trumptwitterarchive.com/`. This website uses topics to filter tweets and gives out nice overviews. We made use of one functionality of this tool that permit to export the results obtained from any query, for this case the query was empty so that we can download all information at once.

The inverted index was constructed using a python script in which the main used libraries were Pandas and CSV.

Sentiment Analysis is a popular means of analysis with the rising popularity of social media over the past decade. People can easily express their opinion in a public platform. This data can be collected and has been collected to advance the domain of Natural Language Processing. In the work by *Pang, Lee and Vaithyanathan* on Machine learning Approaches for Sentiment Classification, interesing data mining appraches for sentiment classification are discusssed.

For the clustering component the programming language used for preprocessing data and executed K-means algorithm was Python with the help of some useful libraries such as Pandas, Numpy, Spacy, nltk, Scikit-learn and Matplotlib. The library Scikit-learn provide an example of K-means clustering with text documents [1] that was also helpful for the development of the final result.

For building the Latent Dirichlet Allocation component, the data was preprocessed using the python libraries ntlk and Spacy. Python's Scikit learn [2] and gensim [3]packages provide better interface for building topic models like LDA. pyLDAvis is a visualisation tool used to analyse and evaluate the number of topics in the dataset.

[1]`https://scikit-learn.org/stable/auto_examples/text/plot_document_clustering.html#sphx-glr-auto-examples-text-plot-document-clustering-py`

[2]`https://scikit-learn.org/stable/modules/generated/sklearn.decomposition.LatentDirichletAllocation.html`

[3]`https://radimrehurek.com/gensim/corpora/dictionary.html`

5 Components

5.1 Inverted boolean indexing

Inverted boolean indexing is a method which creates a dictionary based on terms present in all documents and saves documents to that term if it is present in the document. Often the frequency of term occurrence is also saved per document, this to enable page ranking based on frequency. Compared to normal boolean indexing this method is more efficient in storage usage because it only saves documents that actually include a term instead of also saving information that a certain document does not include a term. The main drawback of (inverted) boolean indexing is that it usually returns either too few or too many documents after a query. This component was developed by Jorrit Bakker.

5.1.1 Motivation

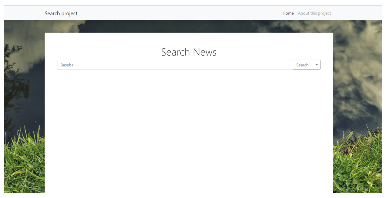
The project needs a way to convert a search term into relevant output in a reliable way. This can be achieved by building an index in which relevant tweets can be easily found.

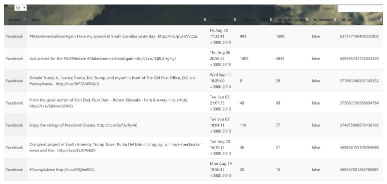
5.1.2 Problem formulation

There are about 36000 tweets that need to be searchable. They are saved in CSV or JSON format and need to be indexed. The final component should be able to take the tweets and output a dictionary of terms which also includes which tweets contain these terms and how often. This dictionary should enable our application to quickly retrieve an array of relevant tweets to a search term.

5.1.3 Approach

The data needed to be cleaned, a dictionary needed to be made and the tweets had to be indexed to this dictionary. The tweets were scraped of the website `http://www.trumptwitterarchive.com/archive` on wednesday Feb 27 2019 as a CSV and JSON file. A github repository was setup in which the data became available to the rest of the group. A temporary frontend was created to play around with the data as seen in figure **??**. Sandhiya cleaned the CSV file

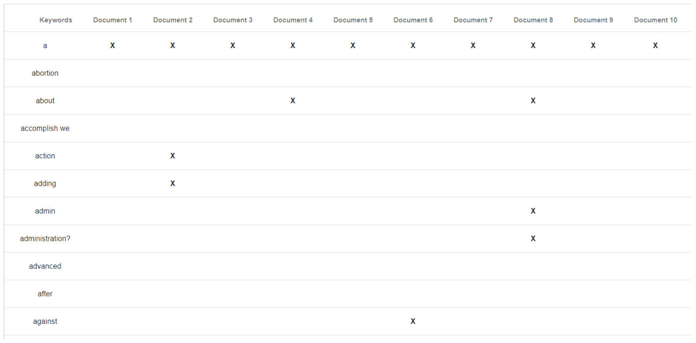




source	text	created_at	retweet_count	favorite_count	is_retweet	id_str
Facebook	#MakeAmericaGreatAgain From my speech in South Carolina yesterday- http://t.co/pv8slVxCzu	Fri Aug 28 17:33:41 +0000 2015	993	1698	false	637317140400332802
Facebook	Just arrived for the #GOPdebate #MakeAmericaGreatAgain http://t.co/rQ8LZHgHyJ	Thu Aug 06 20:56:53 +0000 2015	1409	4835	false	629395741732024320
Facebook	Donald Trump Jr., Ivanka Trump, Eric Trump, and myself in front of The Old Post Office, D.C. on Pennsylvania... http://t.co/WTJZA0R6UG	Wed Sep 11 18:29:00 +0000 2013	9	29	false	377861346551140352
Facebook	From the great author of Rich Dad, Poor Dad-- Robert Kiyosaki-- here is a very nice article. http://t.co/QbkwrUdW6z	Tue Sep 03 21:07:35 +0000 2013	49	69	false	375002156568694784
Facebook	Enjoy the ratings of President Obama. http://t.co/Gx7AvfroA6	Tue Sep 03 18:04:11 +0000 2013	119	77	false	374955998576136192
Facebook	Our great project in South America, Trump Tower Punta Del Este in Uruguay, will have spectacular views and the... http://t.co/ELJCPeK8hi	Tue Aug 20 16:19:13 +0000 2013	36	21	false	369856155109294080
Facebook	#TrumpAdvice http://t.co/f0Sjhe80OL	Mon Aug 19 19:50:45 +0000 2013	20	15	false	369547001265786881

Figure 2: First version of the frontend of the trump search engine

for the LDA part of the project and posted this cleaned version in the repository. Github was searched for earlier inverted boolean indexing projects. The project inverted-index (`https://github.com/oredavids/inverted-index`) was used but after getting it to work it seemed that it produced a regular boolean index instead of an inverted index as shown in figure 5.1.4. The search on github



Keywords	Document 1	Document 2	Document 3	Document 4	Document 5	Document 6	Document 7	Document 8	Document 9	Document 10
a	x	x	x	x	x	x	x	x	x	x
abortion										
about				x				x		
accomplish we										
action		x								
adding		x								
admin								x		
administration?								x		
advanced										
after										
against						x				

Figure 3: Regular Boolean Index of a test tweet collection

became more specific and a project was found that focused on indexing tweets `https://github.com/brandonsorensen/TwitterIQ`. This project generates a inverted index and spit it out in a json file. The project also includes a search function which returns tweet id numbers with the frequency of the term in the tweet. For example for the search word colombia it returns:

```
{'colombia': [{'docId': 5156, 'frequency': 1},{'docId': 15078, 'frequency': 1},
{'docId': 21322, 'frequency': 1}]}
```

33955 unique terms were indexed to a JSON file of about 10mb. This was a bit unexpected because the original dataset was only around 7mb. Camilio integrated the generated JSON index into the final search program.

5.1.4 Evaluation

The distribution of appearance of terms in tweets was as expected. Figure 4 shows the amount of tweets that contained a specific term for the 500 most frequent terms. Only 38 out of the 33955 terms were present in more than 500 tweets. This leads to the steep decrease seen in figure 4. The overall term distribution is shown in figure 5 . Because the biggest differences are between terms with 1 to 10 tweets this distribution is shown with more detail in the second part of figure 5.

The table underneath shows the data for the plots of figure 5.

1	2	3	4	5	6	7	8	9	10	50	100	500
33955	10804	7172	5701	4777	4200	3761	3446	3188	2950	846	386	38

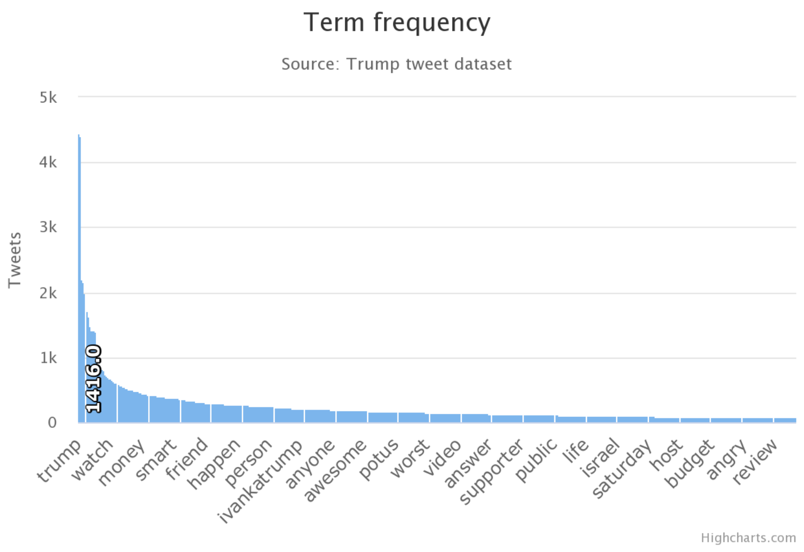


Figure 4: Term frequency in the dataset for 500 most common terms

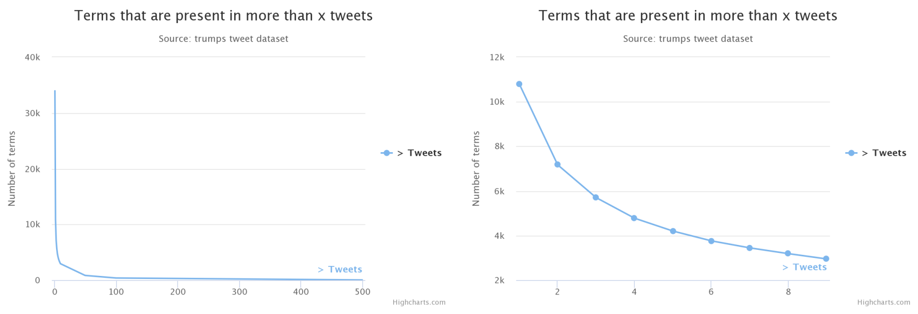


Figure 5: Terms by frequency for all and for 1 to 10 tweets

The most common 38 terms that were present in more than 500 tweets collectively saved 50192 tweets to them. On the other side, 23151 out of the 33955 terms were unique terms and only appeared once in the entire dataset. If we dropped all terms with less than 3 or more than 500 connected tweets we would be left with 7134 terms. This would make for a decrease in file size of 42,7 percent! (From 9775kb to 5597kb). This would make the index database smaller than the original dataset. (Which it wasn't before the outtake)

However, this would mean that populair term topics as ex: obama, hillary, foxnews and border but also specific terms as ex: offshore, lithium, terminal and honeymoon can no longer be found by the index. This is too big of a cost compared to the benefits that the decreased file size brings us. So the full index is kept in the final product.

5.2 Supervised learning

Supervised learning is a data mining technique that can be used for classification. In supervised learning a classifier learns from a given data which already has the labels for the classification. After this the classifier is used on the unlabeled data that needs to be assigned labels. This data mining technique can be used for many types of classification for example binary classification where data items are given either one of the two labels. Other types of classification can have more than two labels. In this project, binary classification is done for sentiment analysis. This component was developed by Surja Chaudhuri.

5.2.1 Motivation

Textual information is often categorized into being subjective or objective. Twitter is a platform for people to voice their opinions. Donald Trumps twitter also contains tweets that are quite subjective. The search engine for Donald Trumps twitter allows us to retrieve tweets regarding certain topics. Building a sentiment analyzer will allow to see the positive and negative opinions of the US president regarding these different topics.
This type of analysis can be interesting to different types of users. For example, new reporters may be interested in questions such as change in opinion of the president about various topics. For example if there are topics that the president has changed his opinion about before and after him being elected. Other interesting analysis can be based on how Donald Trump uses his words, for example the ratio between subjective and objective words.
In the search engine, the tweets can be searched with key words and the tweets displayed has an associated sentiment which is either positive or negative.

5.2.2 Problem formulation

The problem is to classify the tweets from the twitter account @realDonaldTrump into positive and negative sentiment. The tweets are normally a small piece of text between 0 and 280 characters. The challenge is to use natural language processing to tokenize the tweet to extract the words to properly extract the sentiment of the tweet.
The exact problem formulation for this component is the following: Given a set of tweets categorise them to being positive or negative. The component takes the csv file containing the tweets as input and outputs a csv file with new columns containing the sentiment. There will be additional columns to determine the polarity and subjectivity of tweets.

5.2.3 Approach

In order to solve the above problem the first step that needs to be taken is to process the tweets such that the words can be extracted from the tweets. The csv file taken from the source, stated in the previous section, first needs to be processed to remove error lines and encoded as a string to be readable by the

natural language processing library. The python library Textblob was used for tokenizing and polarity and subjectivity analysis of the tweets. Textblob is a library for processing textual data in a consistent way. It provides an API for natural language processing for tokenizing and identifying parts of speech and also sentiment. Textblob uses the natural language toolkit (NLTK) to work with human language data. For the sentiment analysis part, the user of this library may choose to train their own classifier.
NLTK provides a corpus for twitter data. This corpus contains a collection of positive tweets and a collection of negative tweets. These are json files with various tweets that have a label as positive or negative. This data was was downloaded using NLTK and used as training data for the Naive Bayes Classifier. The collection of tweets was split into training and test data and the test data was used to calculate the accuracy of the classifier. This classifier was then used on the trump tweets to provide them with a positive or negative label.
Textblob has its own classifier that is also capable of assigning sentiment to a piece of text. If the sentiment function is used, Textblob assigns a polarity score (between -1 and +1) and a subjectivity score (between 0 and 1) for the text. The polarity score of -1 represents a negative polarity for the text and +1 represents a positive polarity. Similarly the subjectivity score attempts to identify how subjective a piece of text, where 0 is least subjective and 1 is highly objective. This function is also used to provide a polarity and subjectivity score.

5.2.4 Evaluation

The evaluation of the classifier is done through checking the accuracy of training data on the test data. The training data and test data is created by splitting the twitter corpus provided by NLTK. The accuracy of the trained classifier was tested on the test data. The following table shows the accuracy when different amounts of training and test data was used. For the final results the training

Test Set	**Train Set**	**Accuracy**
2000	8000	0.69
4000	6000	0.76
8000	2000	0.78
9400	800	0.79

data set with 6000 points was used. This is because using a huge training data set might lead to overfitting of the model.
The results obtained using the sentiment analyzer can be summarized in the following diagrams. The first figure Figure 6 the count of positive and negative tweets. This result is based on the Naive Bayes Classifier. As it shows the counts are quite comparable however there are more positive tweets.

The figures for polarity and subjectivity Figure 7 show that more tweets are positive as the histogram is left skewed. This confirms the results produced by the classifier. The subjectivity shows a high count of low subjective tweets.

One of the few ways to improve the accuracy of the results would be to better

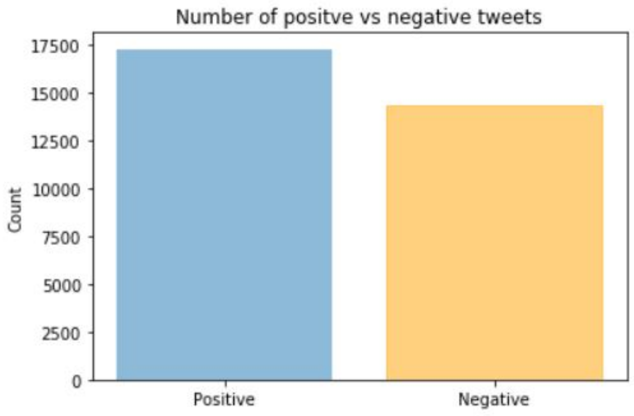


Figure 6: Count of positve and negative tweets

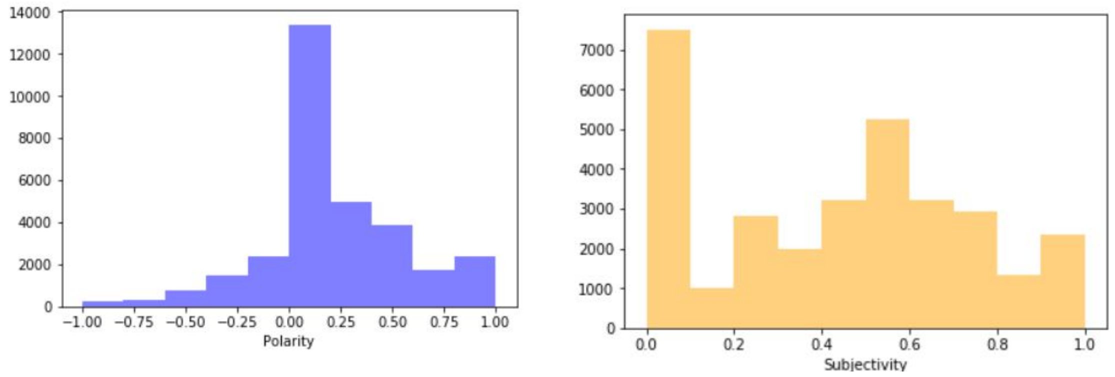


Figure 7: Frequency of different polarity and subjectivity

pre process the data. Removing words that hinder the analysis of tweets could be helpful. For example in the current implementation the terms that are for the twitter handler name and other key words of twitter were not removed.
Another improvement could be to implement a sentiment analyser based on targets. Most sentiments in a piece of text are directed towards a target. A tweet maybe classified as neutral because the amount of words that are not actually part of the tweet.

5.3 K-means clustering

The Machine Learning technique K-means clustering is a type of unsupervised learning, where the input is an unlabeled data set(i.e., data without defined categories or groups). The main goal of this algorithm is to find groups in the data based on their features similarity, the amount of groups that the algorithm will find is represented by the variable K. [4] This component was developed by Camilo Montenegro.

5.3.1 Motivation

The raw data set of this project is natural text phrases no longer than 200 characters. This text contains rich information useful to get insights but if we ask to a computer what a sentence means or which topics are within the phrase, the computer has no idea they only see 1's and 0's. For this reason is why we made use of k-means in this search engine, the purpose is to discover relevant keywords that could represent a similar meaning to each tweet, so when the user input any query the system will return as a result some useful tweets that do not have the exact word match but the words meaning is similar to another keyword.

5.3.2 Problem formulation

From what we have defined and mention before is possible to formulate the main problem to solve. Using raw text from tweets as input we want to obtain at least 'k' groups in which each group is represented by some main keywords. The group will be compose by at least 1 and at most n-k tweets that could or could not contain the main keywords that describe it.

5.3.3 Approach

The previous problem will be solved using the well know Machine Learning Technique for unsupervised learning named K-Means a more in deep explanation can be found in section 5.3.
The figure 8 show a high-level view of the k-means pipeline implemented in this

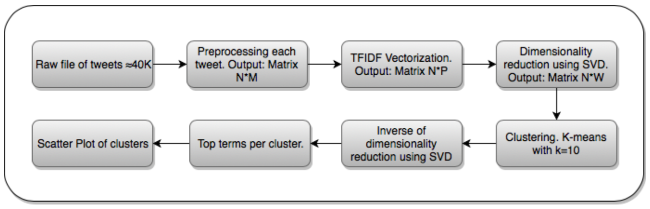

Raw file of tweets ≈40K
Preprocessing each tweet. Output: Matrix N*M
TFIDF Vectorization. Output: Matrix N*P
Dimensionality reduction using SVD. Output: Matrix N*W
Clustering. K-means with k=10
Inverse of dimensionality reduction using SVD
Top terms per cluster.
Scatter Plot of clusters

Figure 8: K-means clustering Pipeline

project. The pipeline starts with the input of the raw tweets text, afterward

[4]For more information: `https://www.datascience.com/blog/k-means-clustering`

and using some Python libraries such as Nltk and Spacy the pre-processing of the data take place. In this step, each tweet text goes through the process of removing stop words, symbols, and URLs. Finally, the result is a matrix N*M where N is the number of documents and M is the total amount of unique words in the whole data set. Later, the TFIDF Vectorization is performed. TfidfVectorizer uses an in-memory vocabulary (a python dictionary) to map the most frequent words to features indices and hence compute a word occurrence frequency (sparse) matrix. The word frequencies are then reweighted using the Inverse Document Frequency (IDF) vector collected feature-wise over the corpus.[5]Having around 36,000 documents means to have a close amount of unique words, that leads to increasing the computation and resources consumption, but with SVD (Singular Value Decomposition) the processing time and complexity are decreased. In predictive analytics, more columns normally mean more time required to build models and score data. If some columns have no predictive value, this means wasted time, or worse, those columns contribute noise to the model and reduce model quality or predictive accuracy.[6]With this data set that has passed through various preprocessing steps the K-means clustering algorithm can be executed using as parameter $K = 10$. Finally, after returning to the original range of values is possible to obtain the top terms or keywords per cluster and afterward construct a Scatter plot that represents in a visual way the clustering process.

5.3.4 Evaluation

For supervised learning is well know that there is a solid ground truth to evaluate the performance of the model, in this case, we are working with unsupervised learning and for that reason, there is no solid ground truth evolution metric that can be used to evaluate the outcome of the clustering algorithm. Furthermore, as was mentioned before, the k-means algorithm requires a value of k so it can start working, this is because it doesn't learn the value from the data, there will not be a right answer of the k-value but there is an optimal value to work on. Even though, there are two commonly used metrics that permit get a good value of k or at least compare against other possible k-values, Elbow method, and Silhouette Analysis.

The Elbow method is a good approximation to get some possible values that can be helpful to use them as the k-value. What this method basically do is to calculate the sum of squared distance (SSE) between data points and each assignment clusters centroid. The theory of this method tells us that we have to choose the value of k where the SSE starts to flatten out and an elbow is formed. The figure 9 doesnt show clear where the elbow is created because in this case, the curve is monotonically decreasing so there is no obvious point where the curve starts flattening out. For the following evaluation we choose k

[5]For more information: `https://scikit-learn.org/stable/auto_examples/text/plot_document_clustering.html#sphx-glr-auto-examples-text-plot-document-clustering-py`

[6]For more information: `https://blogs.oracle.com/r/using-svd-for-dimensionality-reduction`

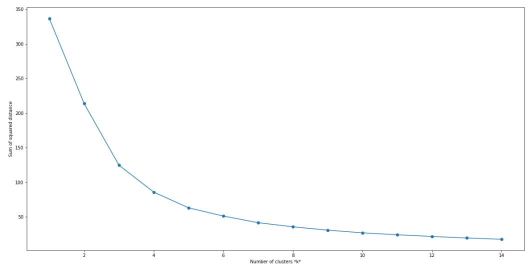
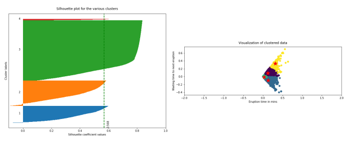
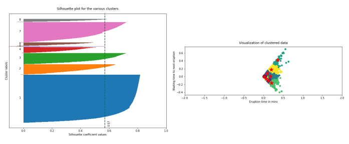


Figure 9: Elbow method using k ϵ [1,15]

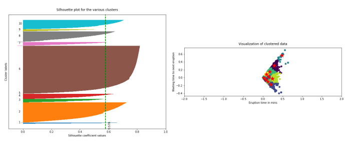
= 4,8,10,12.
Regarding the Silhouette Analysis, this is a method that is used to determine the degree of separation between two clusters. The possible cases are if it is 0 so the sample is close to the neighbor clusters, if it is 1 means the sample is far from the neighbor cluster and if it is -1 means that the sample is in the wrong cluster. In consequence what we will prefer is that the average value will be closer to 1. According to the figure 10 the method was executed for different values of k (4,8,10,12). In the figure 10a the average value is 0.568, for figure 10b is 0.57, figure 10c shows 0.578 and in figure 10d the value is 0.477. With this evaluation method performed with can expose that 10 is a good threshold value where the Silhouette value stop increasing and start to decrease.



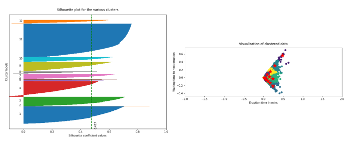
(a) Silhouette and kmeans plot for k=4



(b) Silhouette and kmeans plot for k=8



(c) Silhouette and kmeans plot for k=10



(d) Silhouette and kmeans plot for k=12

Figure 10: Silhouette Analysis for k = 4,8,10,12

5.4 Latent Dirichlet Allocation

Latent Dirichlet Allocation is an unsupervised machine learning technique that classifies the tweets in the document to a specific topic. It models each document has a multi-nominal distribution of topics (alpha) and each topic as the multi-nominal distribution of words (beta). These alpha and beta parameters can be tuned to get a better topics for the raw data. The model assumes that the tweets are a mixture of topics and annotates the most relevant keywords to the topics based on probability. This component was developed by Nagarathinam Ravi Sandhiya.

5.4.1 Motivation

The main goal of the IR system is to retrieve relevant tweets by comparing the user query with tweet documents. The data is increasing day by day due to the demand of software and hardware technology, and we need tools to organize and retrieve the vast amount of information. The data storage is not tedious, but extracting useful data is a big challenge. For building the IR systems, Topic Modeling can be used for understanding and summarizing the tweets by extracting the most relevant topics. From these topics, the user can search for relevant tweets in an intuitive way. For instance, consider a NetFlix dataset containing the movies and the user is interested in finding fantasy movies. In such cases, topic modeling can be used and the best example is *Latent Dirichlet Allocation* which views each document as a mixture of topics and assigns the words based on the probability to their dominant topics in the document.

5.4.2 Problem formulation

In our project, we are focused on filtering the topics from a dataset containing 40K tweets of Trump and other labels like date of creation, retweet counts, the id of the tweet and the source of the tweet. Let's say the user query is Hillary, and we need to retrieve the relevant tweets. For instance, LDA model will classify the raw text and extract the topics Hillary, China, and politics and the topic distributions for a document may be 40% Hillary, 30% China and 50% politics. This distribution of topics may vary for every document and again for each topic, the words from the corpus are mapped to their specific topics based on probability. This means words which have high probability like Clinton will be mapped to the topic Hillary. So based on this classification of topics, the users will have better insights for searching the queries.

The inputs for the model are a dictionary, the document term matrix and the number of topics. The gensim package in Python can be used for creating unique identifiers for each word and to map the words with the frequencies. The number of topics were initially tested for few values to find the optimal one. After building the model, the results will be topic per document model and word per topic model. The document-topic model represents the number of topics in the documents while topic-word highlights the keywords for each topic.

5.4.3 Approach

The data was preprocessed using the python libraries ntlk and spacy. From figure 11, the first step was to tokenize the tweet text and to normalize the words in the text. The stop words are removed which has no meaning while discovering the hidden patterns and lemmatisation was performed.

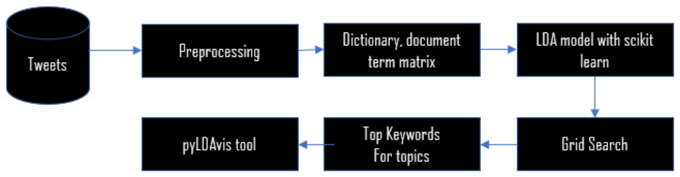
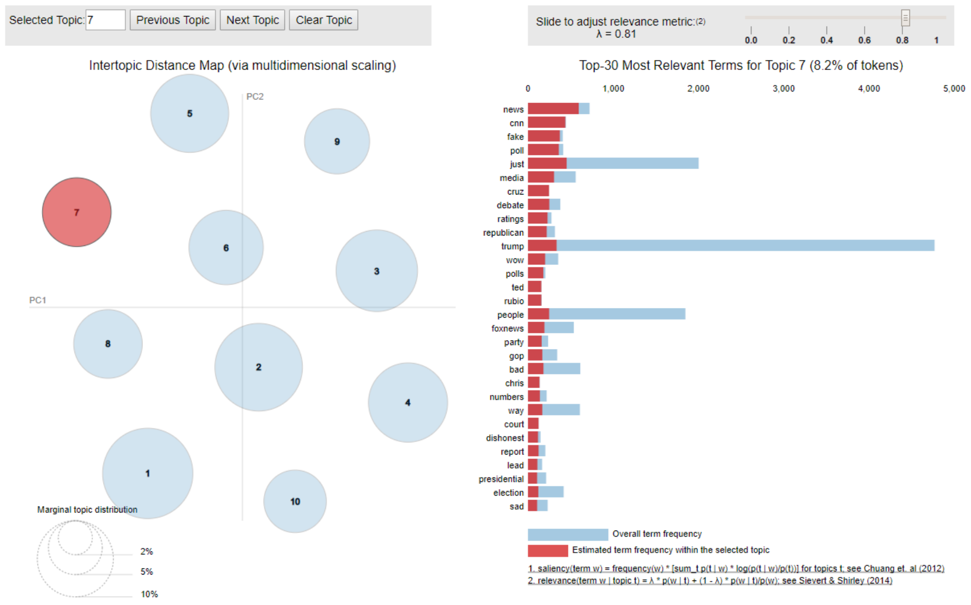


Figure 11: Approach

After preprocessing, a dictionary is created for the unique words in a corpus and the index is assigned. The tweet documents in the corpus are then converted to the document term matrix using gensim package. The LDA model is build using the input and hyperparameters alpha and beta. Alpha controls the per document topic distribution while beta controls the per topic word distribution. If the value of alpha is high, the documents have a large number of topics and higher the beta the topics are composed of more keywords. These hyperparameters were tuned in the LDA model. Alternatively, we also build an LDA model with sklearn and performed a grid search to analyze the results. These results were providing dominant keywords for each topic and there was no overlap over the topics while visualizing in pyLDAvis. LDA is a widely used topic model that models the topics as the bag of words. It is a generative approach that works efficiently for the unseen documents and can be embedded well for different models. From the research papers, it's clear that LSA provides good summarisation for a large group of documents, while LDA is focused on smaller ones. It is a statistical model that determines the top terms for the topics that provide a better way for clustering the documents when compared to LSA.

5.4.4 Evaluation

As topic models are used for extracting the number of topics from textual data, there is no guarantee that the final result is optimal. For evaluating the LDA models, we used coherence models which assesses the quality of topics being extracted. From figure 13 it is clear that the Latent Dirichlet Allocation model has the highest coherence score when compared to other topic models. Coherence is one of the best metric to evaluate the performance of Topic Models. Topic coherence is applied to the top N words extracted by the Latent Dirichlet Allocation model. These models generate coherent topics that can use for



Overall term frequency

Estimated term frequency within the selected topic

1. saliency(term w) = frequency(w) * [sum_t p(t | w) * log(p(t | w)/p(t))] for topics t; see Chuang et. al (2012)

2. relevance(term w | topic t) = λ * p(w | t) + (1 - λ) * p(w | t)/p(w); see Sievert & Shirley (2014)

Figure 12: pyLDAvis for the Twitter dataset

predicting the short topic labels. The optimal number of topics for the dataset can be decided based on the highest coherence score. From figure 14 we observe a sharp decline between 10 to 20, so 10 can be chosen as the optimal value for the topics. If the number of topics picked for a model is large, the keywords or the phrases start repeating for certain topics. The component performs well and provides coherent topics for the dataset. From figure 12, we see that the topics are evenly distributed across the quadrants and there is no overlap between the topics. This means that we have picked a good topic model. Hence, the coherence measure can be used as a metric for comparing and evaluating different topic models based on human interpretability..

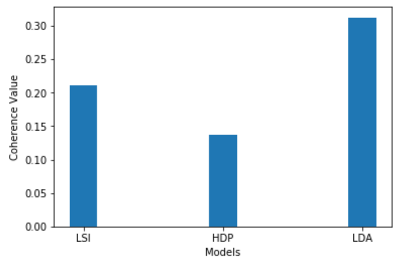


Figure 13: Comparison of different models

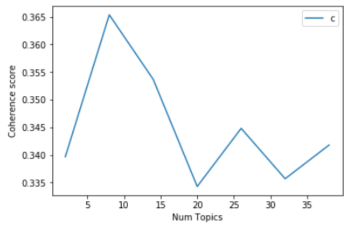
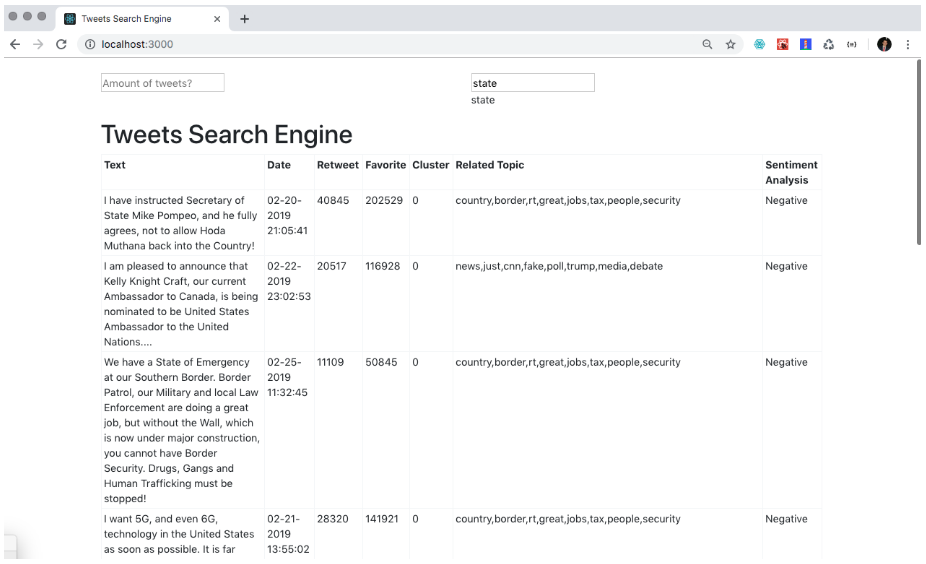


Figure 14: Coherence score for LDA

6 System as a whole



state

state

Tweets Search Engine

Text	Date	Retweet	Favorite	Cluster	Related Topic	Sentiment Analysis
I have instructed Secretary of State Mike Pompeo, and he fully agrees, not to allow Hoda Muthana back into the Country!	02-20-2019 21:05:41	40845	202529	0	country,border,rt,great,jobs,tax,people,security	Negative
I am pleased to announce that Kelly Knight Craft, our current Ambassador to Canada, is being nominated to be United States Ambassador to the United Nations....	02-22-2019 23:02:53	20517	116928	0	news,just,cnn,fake,poll,trump,media,debate	Negative
We have a State of Emergency at our Southern Border. Border Patrol, our Military and local Law Enforcement are doing a great job, but without the Wall, which is now under major construction, you cannot have Border Security. Drugs, Gangs and Human Trafficking must be stopped!	02-25-2019 11:32:45	11109	50845	0	country,border,rt,great,jobs,tax,people,security	Negative
I want 5G, and even 6G, technology in the United States as soon as possible. It is far	02-21-2019 13:55:02	28320	141921	0	country,border,rt,great,jobs,tax,people,security	Negative

Figure 15: Front End Search Engine

Recalling the figure 1, our final system consist of a front-end view for the user and a back-end side dedicate to process the queries and the information retrieved. In figure 15 is possible to see the initial version of our search engine made using ReactJS Framework, is a simple UI that pretends only to receive two parameters from the user: amount of tweets to display and the query. The query results are display in matter of seconds, the web-application returns a table in which each row represent the tweet and its main information such as tweet content, date posted, amount of retweets and favorites, cluster that this tweet belongs to, the related keywords identified using LDA and the sentiment analysis per tweet content.

Regarding the performance of the web application, this analysis will be divided in two parts: fetching the index information (10MB) and fetching the results about one query. On the first hand, fetching the index information, and taking into account the architecture of the system in which we developed a file server that fetch all the index information from a remote MongoDB, this takes around 11931ms. On the other hand, fetching the result for one query is the sum of the time that takes to get 'm' documents, where 'm' is the amount of docsID that the index had previously indexed for a certain word. The average time it take to fetch one document is 270ms, that means that the total time to fetch all documents based on a query will be m*270ms. In order to control that m value, we add the functionality of limit the query results based on the user preferences. By default this value is 10, so the query time will be 2700ms or 2 seconds. This value is not that big and is in the range of time that an user will normally accept.

For improvements we would like to upgrade the user interface and user experience by adding automated informational graphs to the interface, suggested related terms to search using the results obtained from topic modeling and k-means and automated updates to the database if new tweets are created by Trump. Trump also regularly tweets out images. We could index these images and suggest similar images and tweets.

7 Conclusions

The main objective was to design a search engine for Donald Trumps tweets which retrieves relevant information from these for users. The final system is efficient in processing requests and has speeded up lookup tasks by using an inverted index and having the classifications pre-saved to all tweets. The users are given a table with relevant information about their searched topic. The hardest part of the assignment was to fit our work together into a single solution. This required us to rethink how we would setup things and made us all use the same programming languages which were not known to all of us beforehand. In the future a project like this would need more alignment from the start to make the merging at the end easier. Overall we are happy with the search engine we produced.