Summarizing Daily News

Abhishek Shrestha, 390055
M.Sc Computer Science,
Technische Universität Berlin
Berlin, Germany
Email: abhishek.shrestha@campus.tu-berlin.de

Adnan Raza, 397297
M.Sc Computer Science,
Technische Universität Berlin
Berlin, Germany
Email: adnan.raza@campus.tu-berlin.de

Abstract— **Today, there are hundreds of online news sources which present news in various formats like videos, texts, images and so on. Among this overwhelming amount of information, it could be sometimes tedious for a user to figure out what the most important news of the day are. Our application tries to make it easier for the user to know what the most important news or events of the day were based on how many news sources covered the event. The application does so by collecting news items from various distinct online news sources every 24 hours (by default but configurable), processing the collected data to filter out unnecessary tokens, then performing a K-Means clustering based on the Tf-idf of the individual tokens so as to group the similar news items together. The clustered data points then are sorted to get top 20 clusters. The centroids of these clusters are then selected to be displayed as hot news or important news topics of the day. The web front then makes the selected news topics viewable to the user in a clean UI. We also implemented purity and Sum of Squared Errors (SSE) as cluster evaluation methods and used SSE to determine optimal number of clusters for K- means algorithm.**

Keywords— Feature extraction, Stop words, Stemming, K-Means, Purity, Sum of squared errors, Tf-idf

I. INTRODUCTION

With the internet being more accessible, an increasing portion of the population is turning away from traditional news sources, such as radio and newspapers, and relying on the internet to keep up-to-date with what is going on around the world. Around 49 percent of German consumers cited some sort of online publication as their main source of news in 2017 [1]. In the U.S., roughly nine-in-ten adults (93%) get at least some news online (either via mobile or desktop) [2]. Thus, since most of the consumers have transitioned to getting information online, almost all of the major news outlets now have their own websites. In addition these "legacy" news organizations, several other online only sources like Imgur, MSN, Twitter and Facebook have become very popular over the years. However, with the rise in number of news outlets which cover domestic to international news, it is getting increasingly difficult for a consumer to know what were the important news of the day just at a glance without having to go through the whole website or visit multiple websites.

The goal of our application is to provide the user with a single platform where the most important events of a day can be viewed. The application includes a service that collects news items from various news sources through their RSS feeds, Twitter handles and website automatically within defined intervals and perform clustering to group the similar items together and determine which news are important depending on how many elements there are in a cluster and a webpage displays the resulting news items.

The remainder of this report is organised as follows. In section II, a brief discussion on the related works which we studied during and before development. Section III describes in detail about our workflow and approach. Section IV presents our experiment settings leading to Section V which describes our experiments and outcomes. Section VI discusses our conclusion and future work.

II. RELATED WORK

The application development involved five main parts: Collecting data from multiple news outlets, processing the data which involved stemming and filtering tokens, clustering the collected data which also includes determining the value of K since we are using K-Means clustering, a web front which reads the results from clustering and displays it and also a module to evaluate the clusters. We took a lot of references to develop each of these parts. Fortunately, we found a lot of work related to these areas which helped us a lot during development. Some of the notable ones are referenced in this section.

[3] provides a really good explanation with sample code on how to parse RSS feeds. It uses custom API to read the feeds. We ended up using a different API but the examples were really helpful in understanding the file format. [4] provides another sample application to read and write RSS feeds. The application uses an API called Rome [5] to parse and build RSS content. We used the same API so the application was really helpful. However, it uses slightly different approach to read the input to handle certain exceptions while we directly used xml reader which worked for us for all RSS sources.

[6] presents a simple tutorial for parsing tweets from timeline. The sample code provided uses the Twitter4j API [7] to pull tweets, send direct messages and many other functionalities. We also decided on using the same API, so the application was a very good reference for us.

[8] provides a sample application that uses Part-of-Speech Tagging [9]. It also uses the OpenNLP [10] API which we also used to implement the POS Tagging for feature extraction. However, we ended up using a different training set than the one mentioned by the author in the sample snippet.

[11] provides a really good example on how to perform text clustering with K-Means and tf-idf. It presents a simple application that reads the given set of inputs and then calculates tf-idf weights and uses them to cluster text documents using K-Means. The code however is in Python while ours is in Java. Nevertheless, it was really good reference that helped us understand tf-idf weights. [12] provides another example of using tf-idf weights to represent document as vectors and then implement K-Means for clustering with cosine similarity as similarity measure. This

is very close to what we did. In fact, the first sample application we developed to try K-Means was based on this code. [13] also presents a sample application that uses cosine similarity computed using tf-idf weights for K-Means clustering. The application is easy to understand and implements representation of documents in terms of vectors using tf-idf through well defined classes. Because of this we also structured our clustering module similar to this application. [14] discusses on implementing K-Means with tf-idf. However, the paper suggests using Residual Sum of Squares (RSS) as a stopping criterion where as we used Sum of Squared Errors (SSE) for determining the optimal number of clusters.

In [15] authors discuss Silhouette and Sum of Squared Errors (SSE) method of validation clusters to find appropriate number of clusters for K-Means. We implemented only SSE but the paper was very helpful in providing a clear concept on how to compute SSE.

III. Our Approach

In this section we describe in details how we implemented different modules of our application, along with database schemas and flowcharts for clear representation.

For better overview we can divide our workflow in 5 modules:

1. Backend.

We are using MS SQL Server 2008 R2 as our database. All the links to the data sources which are used to extract data along with all the collected data are stored in the database.

Data collected from different source types (Twitter, RSS, web scraping) are stored in separate tables.

Further, while clustering, we use a global query to get the extracted data from all the tables and processing is carried out on the overall data. The final results from clustering are also stored in the database.

For more clarity the database schema is as shown in Figure 1.

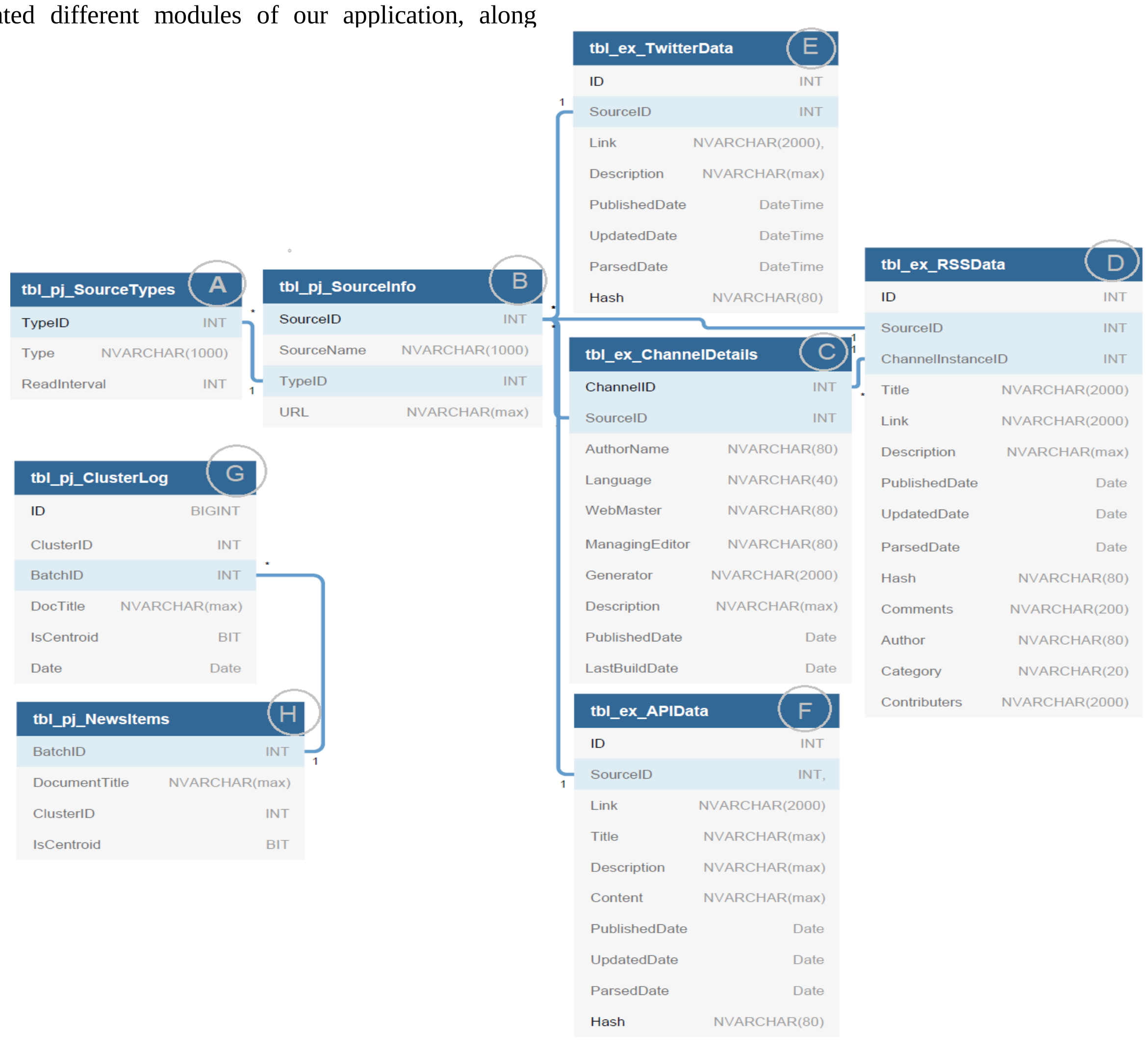


Figure 1. Database schema of the application. A: Stores source types (RSS feed, twitter, webpage). B: Stores details about each source. C: Stores channel related details of each RSS source. D: Stores new items extracted from RSS feeds. E: Stores data extracted from Tweets. F: Stores data extracted from Video source and webpage using the News API. G: Stores final results from clustering with optimal value of K. H: Stores data from top clusters that are to be displayed in the UI (only ones with IsCentroid = "true" are displayed)

2. Data collection.

We collected data from various news sources. Each sources differing in the type of format in which they present data.

The various formats of data which the application supports are discussed below.

A. RSS Feeds

RSS is acronym for “Really Simple Syndication”. It is a dialect of XML and follows XML 1.0 specification [17]. Various news services offer RSS feeds which are updated and maintained by the providers themselves. These can be read and parsed to obtain information like title of the news, a short description, a link to actual news, published date and language.

Every RSS feed has a channel element that includes details about the RSS channel like the name of the channel, link to website representing the channel and a short description of the channel. Along with this, it might also contain other information like language, copyright, managing editor’s email and so on. However, these do not change periodically and remain same for most of the channel. We capture all of the channel elements; both optional and mandatory (Figure 1.C).

Further, a channel may contain one or more “Items”, each of which represent a “Story” -- much like a story in a newspaper or magazine; if so its description is a synopsis of the story, and the link points to the full story. An item may also be complete in itself, if so, the description contains the text (entity-encoded HTML is allowed), and the link and title may be omitted. All elements of an item are optional.

To avoid duplicates to database we are using “MD5” hashvalues generated from “Description” and “Title” and checking the hash before inserting to db.

Table 1 shows the list of RSS sources we are currently using.

Table 1. List of RSS Sources

Source Name	Type	URL
BBC World News	RSS Feed	http://feeds.bbci.co.uk/news/world/rss.xml
CBN World News	RSS Feed	http://www.cbn.com/cbnnews/world/feed/
Reuters	RSS Feed	http://feeds.reuters.com/Reuters/worldNews
The Guardian World Press	RSS Feed	https://www.theguardian.com/world/rss
Yahoo News	RSS Feed	https://www.yahoo.com/news/rss/world

B. Twitter

We are using the Twitter4j API [7] to pull tweets from the timelines of the selected news sources. The twitter API was fairly straight forward to use. We created an account and then used the OAuth authentication [18] to pull tweets.

Currently, we are pulling all the tweets from a news source within the 24 hour duration.

In this case also, to avoid duplicates, we are generating MD5 hash values based on the “Description” which is the tweet itself and checking hash before inserting to database.

Table 2 shows the list of news sources whose Tweets we are collecting.

Table 2. List of Twitter Sources

Source Name	Type	URL
CCN International News	Tweeter	https://twitter.com/cnni
FOX News	Tweeter	https://twitter.com/FoxNews
RT News	Tweeter	https://twitter.com/RT_com
CGTN Official	Tweeter	https://twitter.com/CGTNOfficial

C. Video Source

Currently we are pulling the contents from only a single video web site using the News API [19]. The API returns a JSON metadata describing the videos in the website which are then parsed and stored in the database (Figure 1.F).

The list of video based sources we are currently using for data collection is listed in Table 3.

Table 3. List of video based sources

Source Name	Type	URL
FOX Video News	Video	http://video.foxnews.com

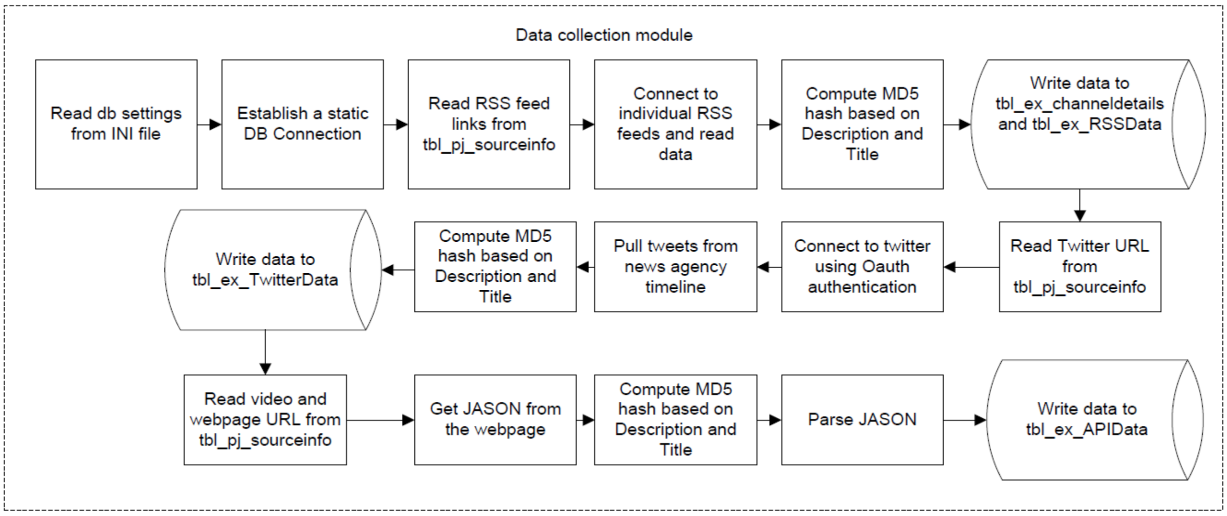
D. Web Source

We are using CNN US web site to collect news data via web scraping. We are using the News API for scraping. Like in case of video sources, the API reads the metadata and other details from the website and returns a JSON text which the application then parses to get the relevant data to store in the database (Figure 1.F).

The list of websites we are using directly to gather news are listed in Table 4.

Table 4. List of scraped websites

Source Name	Type	URL
CNN News	Website	http://us.cnn.com



Data collection module
Read db settings from INI file
Establish a static DB Connection
Read RSS feed links from tbl_pj_sourceinfo
Connect to individual RSS feeds and read data
Compute MD5 hash based on Description and Title
Write data to tbl_ex_channeldetails and tbl_ex_RSSData
Read Twitter URL from tbl_pj_sourceinfo
Connect to twitter using Oauth authentication
Pull tweets from news agency timeline
Compute MD5 hash based on Description and Title
Write data to tbl_ex_TwitterData
Read video and webpage URL from tbl_pj_sourceinfo
Get JASON from the webpage
Compute MD5 hash based on Description and Title
Parse JASON
Write data to tbl_ex_APIData

Figure 2. A flowdiagram of data collection module

3. *Feature extraction*

In this section, we describe the feature extraction models we used before sending the data for clustering.

We are using a global SQL query to get all the data in a single table and since we are performing clustering based on the "Description" field of the news sources (Figure 1), we are using the feature extraction on only the "Description" field.

We are using simple white space analyzer to tokenize the text from the selected field. To improve clustering and remove unnecessary tokens we experimented with few of the feature extraction models which are described below.

A. *Stop word filter*

Sometimes, some extremely common words which would appear to be of little value in documents matching are excluded from the vocabulary entirely. These words are called stop words [22]. Filtering stop words is more of a feature selection process rather than a feature extraction process.

We are using NLTK's predefined list of stop words [23]. Apart from the standard list we also added name of days to the list since these are also not significant. So the stop words list we are currently using is slightly modified version of the standard NLTK list.

The input token stream is compared with this list and if any tokens are matched with the stop words then they are completely ignored in the output tokens list.

```
'the' is stopword: yes
'Scott' is stopword: no
'earth' is stopword: no
'went' is stopword: no
'crawled' is stopword: no
'monday' is stopword: yes
```

Figure 3. Output from the stopword checker.

B. *Stemming*

For grammatical reasons, documents are going to use different forms of a word, such as organize, organizes, and organizing. However, all of these terms stem from a single root word "organize". While comparing documents if any of these words are detected then the two words will be detected as dissimilar, however if we only consider the root word then the two words can be detected as similar. Stemming usually refers to a crude heuristic process that chops off the ends of words in the hope of achieving this goal correctly most of the time [20].

We are using the Java version of snowball stemmer API [21] for stemming the tokens obtained after filtering the stop words.

```
'organise' stem: 'organis'
'organisation' stem: 'organis'
'exploded' stem: 'explod'
'explosion' stem: 'explos'
'search' stem: 'search'
'research' stem: 'research'
'buses' stem: 'buse'
```

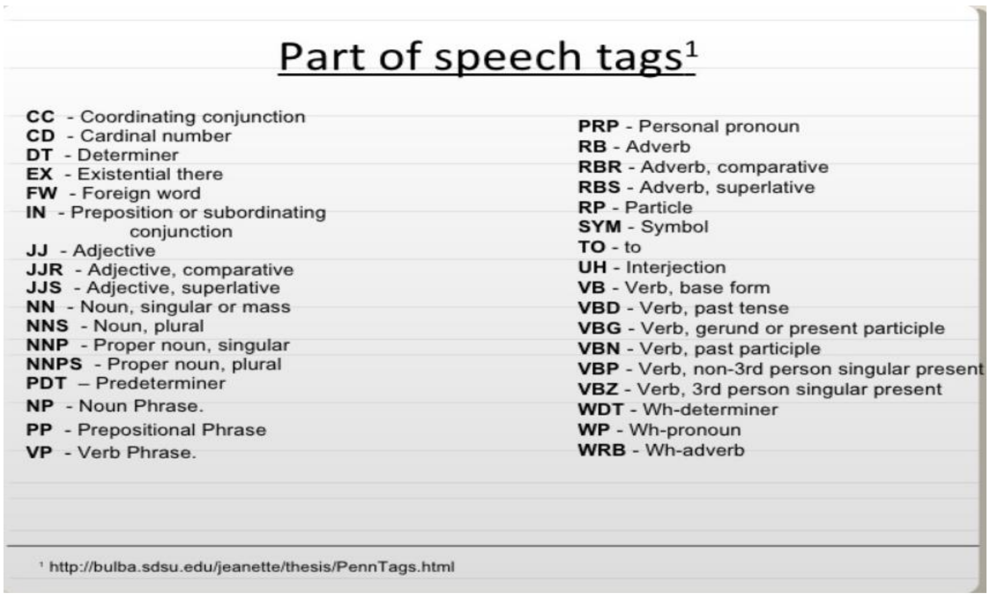
Figure 4. Example of how stemmer converts tokens.

C. *POS Tagging.*

Part-Of-Speech Tagging involves reading the given text in some language and assigning parts of speech to each word, such as noun, verb, adjectives etc. We implemented POS Tagging in our application because in a sentence in almost all cases only nouns, adjective and verbs are more important while other parts of the sentence like conjunctions, pronouns and determiners are not important and can be ignored. So we can improve document matching during clustering by removing unnecessary tokens.

We implemented OpenNLP POS Tagger API [24] for POS Tagging with en-pos-maxent.bin as a pre-trained model with tag dictionary [10].

The output from the sentence detector [25] was fed as input to the tagger and from the output stream, we ignored tokens tagged as Conjunctions (CC), determiner(DT), particle (RP), preposition (IN), interjection (UH), wh-determiner (WDT), wh-pronoun (WP), predeterminer, to (TO) and wh-adverb (WRB) (Figure 5.).



Part of speech tags[1]

CC	Coordinating conjunction
CD	Cardinal number
DT	Determiner
EX	Existential there
FW	Foreign word
IN	Preposition or subordinating conjunction
JJ	Adjective
JJR	Adjective, comparative
JJS	Adjective, superlative
NN	Noun, singular or mass
NNS	Noun, plural
NNP	Proper noun, singular
NNPS	Proper noun, plural
PDT	Predeterminer
NP	Noun Phrase.
PP	Prepositional Phrase
VP	Verb Phrase.
PRP	Personal pronoun
RB	Adverb
RBR	Adverb, comparative
RBS	Adverb, superlative
RP	Particle
SYM	Symbol
TO	to
UH	Interjection
VB	Verb, base form
VBD	Verb, past tense
VBG	Verb, gerund or present participle
VBN	Verb, past participle
VBP	Verb, non-3rd person singular present
VBZ	Verb, 3rd person singular present
WDT	Wh-determiner
WP	Wh-pronoun
WRB	Wh-adverb

[1] http://bulba.sdsu.edu/jeanette/thesis/PennTags.html

Figure 5. Parts of speech tags applicable for the maxent model.

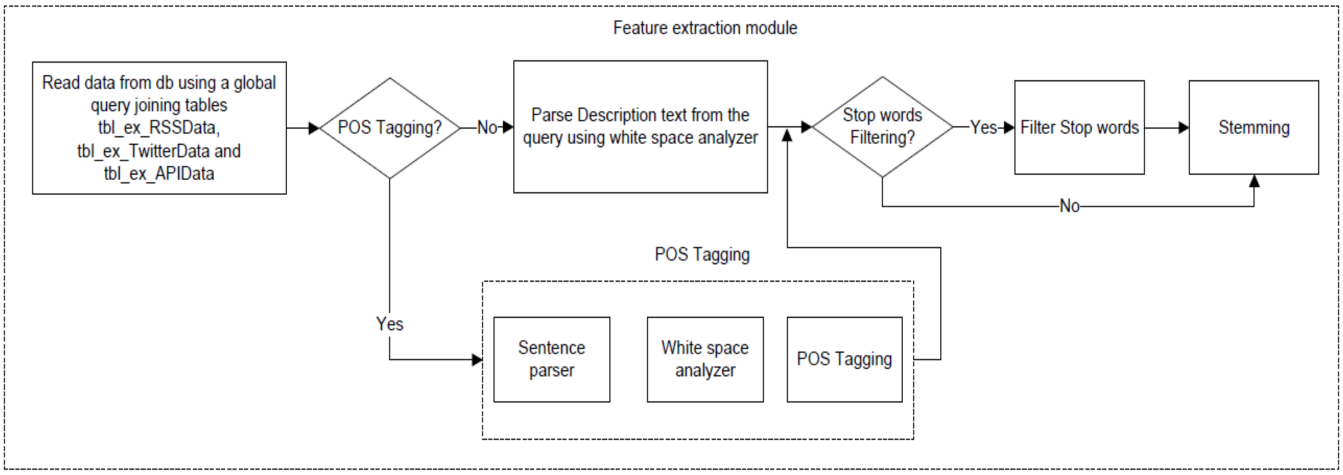

Feature extraction module
Read data from db using a global query joining tables tbl_ex_RSSData, tbl_ex_TwitterData and tbl_ex_APIData
POS Tagging?
No
Parse Description text from the query using white space analyzer
Stop words Filtering?
Yes
Filter Stop words
Stemming
No
POS Tagging
Yes
Sentence parser
White space analyzer
POS Tagging

Figure 6. A flow diagram showing feature extraction module.

4. Clustering model

We selected K-Means clustering as our clustering model. The k-Means clustering algorithm is an unsupervised hard clustering method. The algorithm aims to partition a set of objects, based on their attributes/features, into k clusters, where k is a predefined or user-defined constant.

The main idea is to define k centroids, one for each cluster. The centroid of a cluster is formed in such a way that it is closely related (in terms of similarity function; similarity can be measured by using different methods such as cosine similarity, Euclidean distance, Extended Jaccard) to all objects in that cluster [14].

To compute the similarity between the documents we must first represent each document in terms of document vectors. For this we computed Tf-idf for each term in the document.

Tf-idf stands for term frequency-inverse document frequency. It is a numerical statistics which reflects how important a word is to a document in a collection or corpus.

For computing Tf-idf we calculated the term frequency for each term in each document and maintained a hash table of term and its frequency for each document. So, for any term t, in document d and $f_{t,d}$ be the frequency of term t in document d then, the term frequency can be represented Figure 7.

$$\mathrm{tf}(t,d) = f_{t,d}.$$

Figure 7. Term frequency

The inverse document frequency was then obtained by dividing the total number of documents by the number of documents containing the term, and then taking the logarithm of that quotient, as shown in Figure 8.

$$\mathrm{idf}(t, D) = \log \frac{N}{|\{d \in D : t \in d\}|}$$

with

- N: total number of documents in the corpus $N = |D|$
- $|\{d \in D : t \in d\}|$: number of documents where the term t appears (i.e., $\mathrm{tf}(t, d) \neq 0$).

Figure 8. Inverse document frequency.

Thus, from the formula we can see that idf is the measure of whether a term is rare or common across the whole corpus. The more the term is common the less the idf.

Multiplying tf and idf gives us tf-idf for each term in a document. We maintain a hash table of each document and its tf-idf for each term.

Then, from the available news data we select K number of centroids randomly. Since, we already have tf-idf representation for each terms in a document, we then calculate similarity of each document with each centroid.

The similarity measure we are using is Cosine Similarity, which can be computed as shown in Figure 9.

$$\text{similarity} = \cos(\theta) = \frac{\mathbf{A} \cdot \mathbf{B}}{\|\mathbf{A}\|\|\mathbf{B}\|} = \frac{\sum\limits_{i=1}^{n} A_i B_i}{\sqrt{\sum\limits_{i=1}^{n} A_i^2} \sqrt{\sum\limits_{i=1}^{n} B_i^2}},$$

where A_i and B_i are components of vector A and B respectively.

Figure 9. Cosine Similarity.

Then based on the Cosine similarity with the centroids, documents are placed on clusters.

A. *Optimal value of K*

One of the main problems with the K-Means algorithm is that the value of K, which is the number of centroids or the number of clusters, has to be provided manually. For an application like ours, it might not be possible to always specify K manually since it cannot be predetermined. So we used Sum of Squared Errors (SSE) to determine the optimal value of K.

SSE can be computed by the relation as shown in Figure 10.

$$SSE(X, \Pi) = \sum_{i=1}^{K} \sum_{x_j \in C_i} ||x_j - m_i||^2$$

where

$X = \{x_1, \dots, x_i, \dots x_N\}$, $x_i \in \Re^M$

$\Pi = \{C_1, C_2, \dots C_K\}$ $, \forall_{i \neq j} C_i \cap C_j = \emptyset, \cup_{i=1}^{K} C_i = X, \forall_i C_i \neq \emptyset.$

$|| \cdot ||$ = Euclidean distance and m_i is centroid of cluster C_i

Figure 10. Sum of Squared Errors

So basically, SSE is the sum of squared difference between each data point and its centroid or mean point [15].

So beginning with K =1 we gradually increased the value of K in the steps of 5, we computed SSE till the maximum value of K. Obviously, SSE decreases as the value of K increases because of the thinning number of members in each clusters. So we find a point where there is the maximum decrease in SSE from the previous value. And the value of K for which this occurs is the optimal value of K.

Then using this value of K we get the final clusters. The results are then inserted into the database as well (Figure 1 G.)

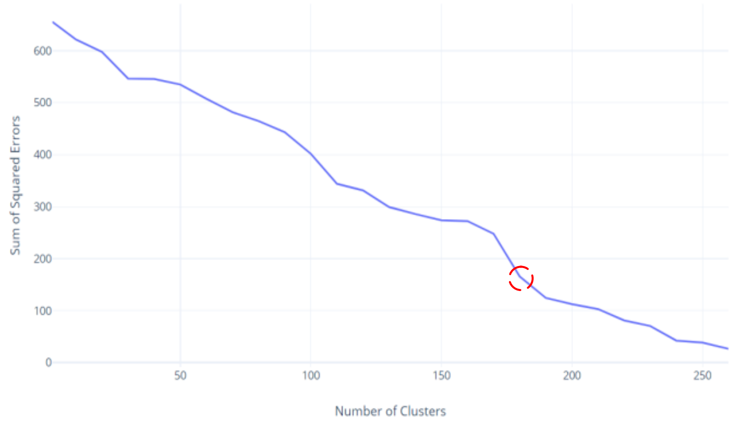


Figure 11. Plot between varying number of clusters to SSE. The red mark represents the point at which there is maximum decrease in SSE as compared to previous SSE, which is the optimal value of K.

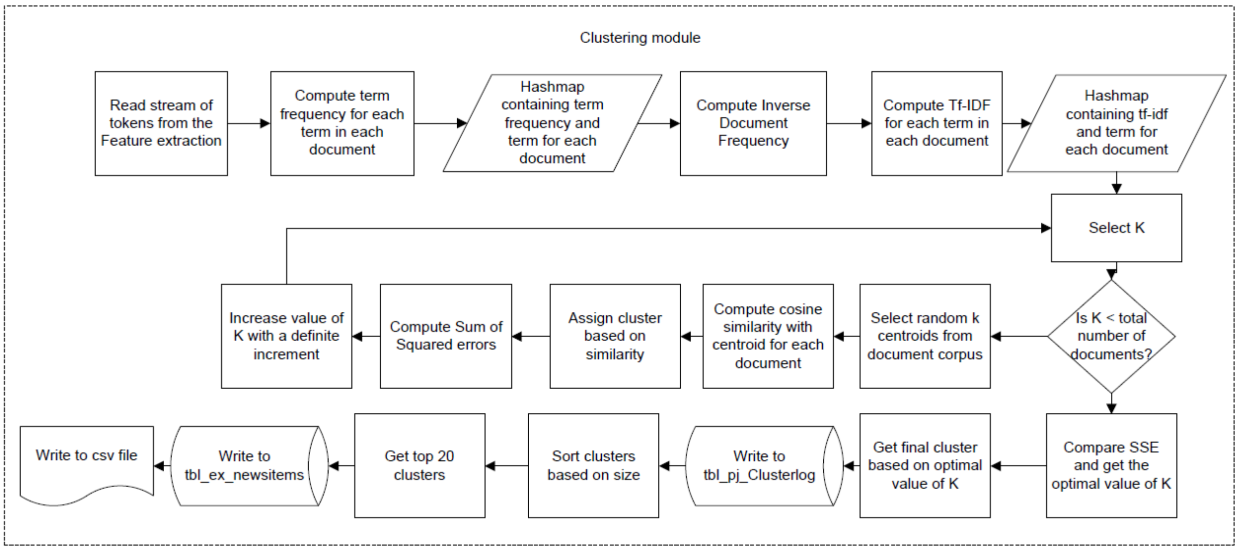

Clustering module
Read stream of tokens from the Feature extraction
Compute term frequency for each term in each document
Hashmap containing term frequency and term for each document
Compute Inverse Document Frequency
Compute Tf-IDF for each term in each document
Hashmap containing tf-idf and term for each document
Select K
Is K < total number of documents?
Select random k centroids from document corpus
Compute cosine similarity with centroid for each document
Assign cluster based on similarity
Compute Sum of Squared errors
Increase value of K with a definite increment
Compare SSE and get the optimal value of K
Get final cluster based on optimal value of K
Write to tbl_pj_Clusterlog
Sort clusters based on size
Get top 20 clusters
Write to tbl_ex_newsitems
Write to csv file

Figure 12. Flow diagram depicting the working of the clustering module.

5. *The final output and webfront.*

From our observations, the usual number of news items overall was about 600 – 900 and for this number of news items the number of clusters were really large (about 200-300). And most of these clusters contained only 1- 2 elements. So it is not feasible to show all the clustered results in the UI even when we select just 1 element from each cluster. Further, if a cluster contains very less number of elements then it cannot be considered as hot or important news.

So to solve this problem, we collected data for three days taking the optimal value of K (as calculated using SSE) and studied the size of clusters. We found out that there were very less number of clusters which had more than 4 or 5 elements.

From the observations, there were at most about 20 news elements that had more than 5 elements. This is also depicted in the Figure 13.

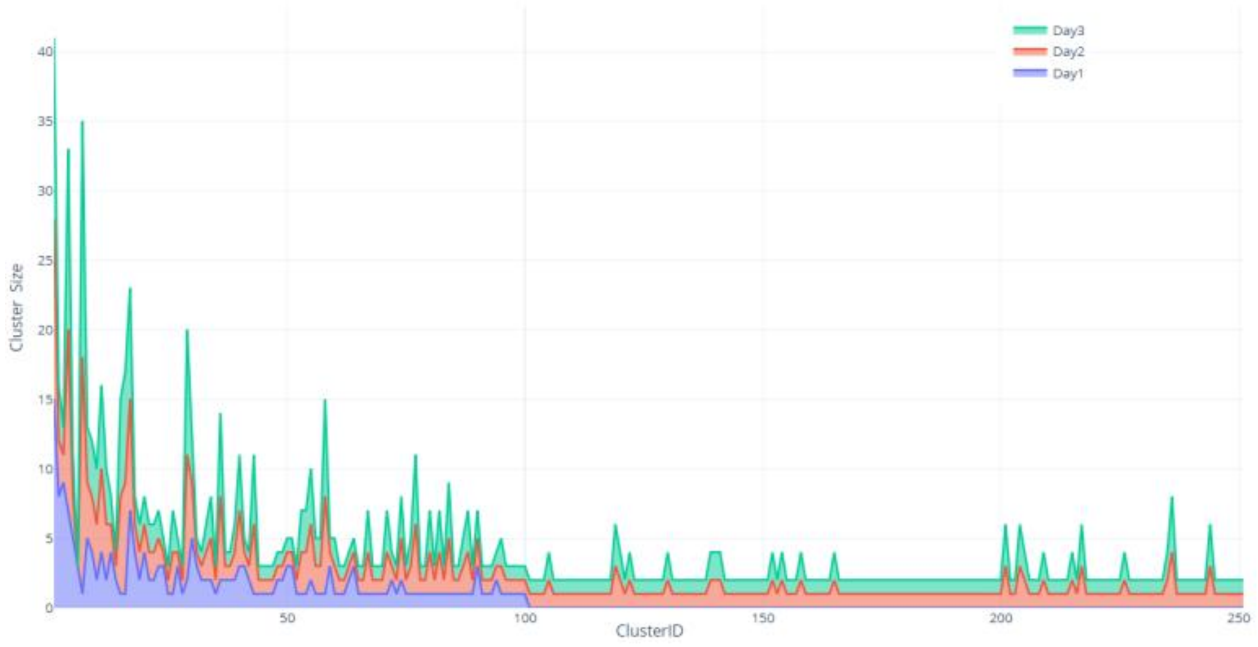
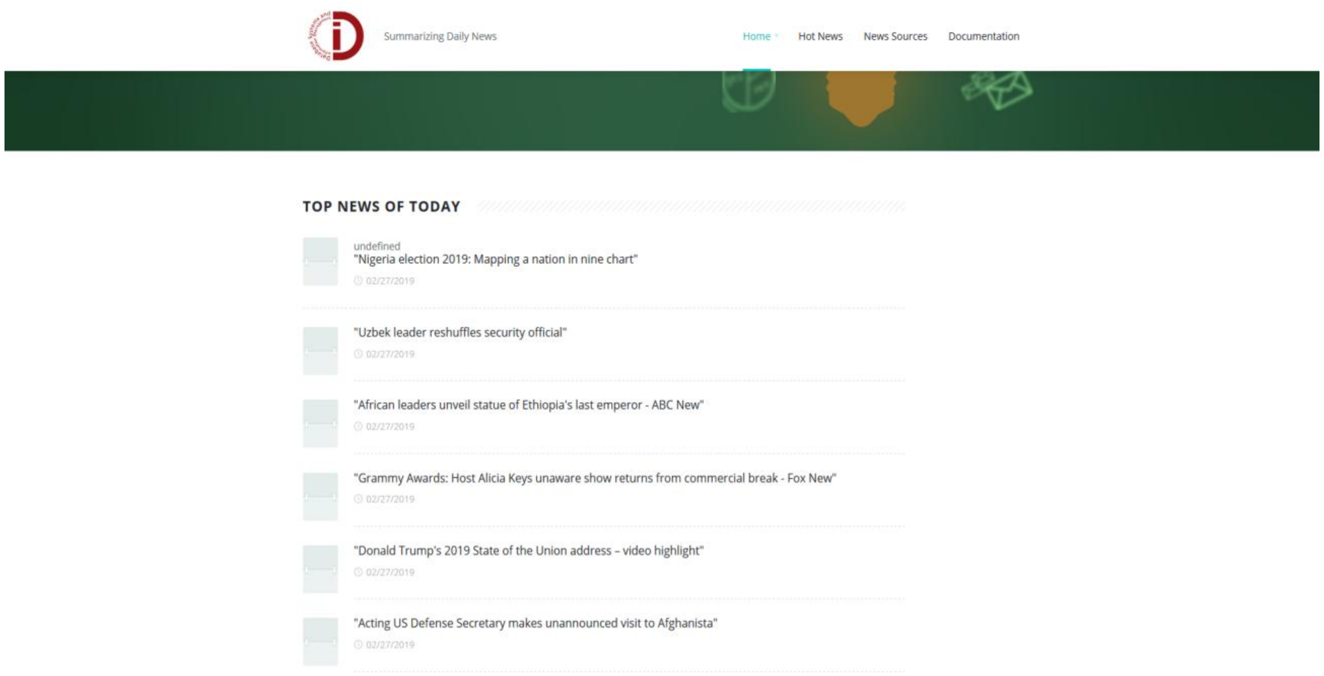


Figure 13. Plot between cluster size and the corresponding cluster id for three days.

So, to show only relevant news, we sort all the clusters based on their size and pick 20 clusters (this is configurable in the ini file to accommodate for later changes). The data points in these clusters are then also written in a csv file as well as in the database (Figure 1 H.)

The web front reads the csv file and from the 20 clusters, selects the centroids i.e. 20 news items and shows them in the UI.



TOP NEWS OF TODAY

undefined
"Nigeria election 2019: Mapping a nation in nine chart"
02/27/2019

"Uzbek leader reshuffles security official"
02/27/2019

"African leaders unveil statue of Ethiopia's last emperor - ABC New"
02/27/2019

"Grammy Awards: Host Alicia Keys unaware show returns from commercial break - Fox New"
02/27/2019

"Donald Trump's 2019 State of the Union address – video highlight"
02/27/2019

"Acting US Defense Secretary makes unannounced visit to Afghanista"
02/27/2019

Figure 14. Only top 20 selected news are displayed in the UI.

The whole application runs as a service so the entire process starting from data collection is again started in every 24 hours by default. This duration is controllable from a setting in the INI file.

IV. Experiment Settings

We performed various experiments to determine which feature extraction process gave better clusters. For this, we implemented various cluster evaluation methods.

This section describes the cluster evaluation methods we implemented and how we prepared the data required for the experiments we performed.

For cluster evaluation, we implemented two evaluation methods where are discussed below:

A. *Purity*

In cluster analysis, Purity is one of the external evaluation criterion of cluster quality. It measures how close the resulting clusters were by comparing the results with a ground truth [26].

Purity can be computed using the relation as shown in Figure 15.

$$\text{purity}(\Omega, \mathbb{C}) = \frac{1}{N} \sum_k \max_j |\omega_k \cap c_j|$$

where $\Omega = \{\omega_1, \omega_2, \ldots, \omega_K\}$ is the set of clusters and $\mathbb{C} = \{c_1, c_2, \ldots, c_J\}$ is the set of classes.

Figure 15. Purity

Thus, purity can be computed by looping through each of the computed clusters against the entire ground truth and counting the number of common elements in each cluster and the clusters in the ground truth and finding the maximum value for each cluster and then summing them together and finally dividing by the number of data points.

It is fairly easy to implement which is one of the reasons why we selected this evaluation measure.

Purity can vary from 0 to 1 and is greater for more number of clusters.

To compute purity we have to cluster the news items manually so as to create a ground truth. So, we isolated the data for a single day and then manually clustered the data and created a csv file which can be read by the application. We then added an evaluation class in the application which compares the clusters created by application with the clusters from the ground truth to perform our experiments involving Purity.

B. *Sum of Squared Errors (SSE)*

SSE is one of the internal measures to evaluate a cluster and does not require a ground truth. As mentioned (Figure 10) SSE is the Euclidian distance between each data point and its centroid.

So SSE is simple to compute as we already represented each document in terms of document vectors. So we computed the Euclidian Distance between each point in the cluster and its centroid.

One of the disadvantages with Purity was that it required ground truth and since there can be large of news items in a day, it becomes tedious to compute ground truth each time we have to validate cluster. So we had to implement an internal measure and SSE is simple to understand and implement. Thus, we selected this measure to validate clusters.

V. Experiments and Outcomes

We implemented various feature extraction methods to eliminate unnecessary tokens using stop word filters and POS Tagging as well as modified certain tokens using the snowball stemmer in the hopes that we would get better clusters and better performance overall. To verify which of the feature extraction methods worked and which did not, we carried out few experiments taking individual filters as well as combination of filters to perform clustering.

In this section we describe those experiments and discuss the result of those experiments.

A. *Effect of feature extraction on purity*

We used various combinations of feature extraction models and computed purity for those combinations for different values of K. The results are as in tables 5.

Table 5: Effect of feature extraction models on purity

Number of clusters	Purity			
	No filter	Stop word filter	word stemming and Stop word filter	POS Tagging and Stop word filter
10	0.116	0.116	0.116	0.116
50	0.356	0.4	0.393	0.325
100	0.543	0.547	0.546	0.524
150	0.704	0.685	0.663	0.681
200	0.824	0.817	0.801	0.808
250	0.944	0.984	0.94	0.948

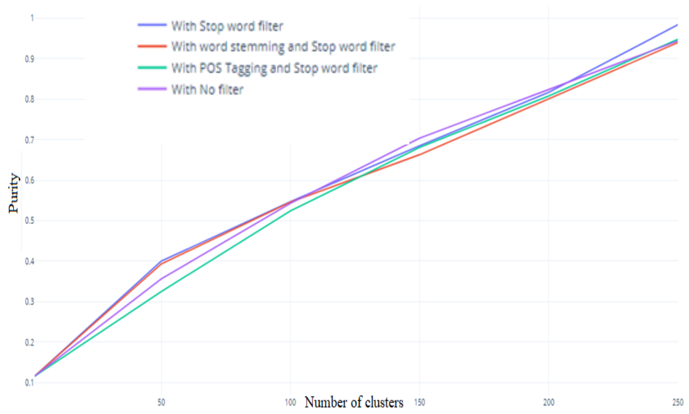


Figure 16. Effect of feature extraction models on purity

B. *Effect of feature extraction models on Sum of Squared Errors.*

Similarly, we computed SSE for various combinations of feature extraction models and for different values of K. The result is as shown in table 6.

Table 6: Effect of feature extraction models on SSE

Number of clusters	Purity			
	No filter	Stop word filter	word stemming and Stop word filter	POS Tagging and Stop word filter
1	568.82	631.84	655.3	655.34
10	552.03	601.11	621.5	628.17
20	542.09	574.33	597.7	587.66
30	517.15	545.61	546.4	564.99
40	478.9	498.53	545.7	553.75
50	426.6	477.44	534.9	534.99
60	391.8	454.84	507.5	462.65
70	361.5	445.34	481.7	460.89
80	331.3	435.12	464.6	423.39
90	311.7	400.46	443.2	407.07
100	297.6	358.88	401.6	407.96
110	271.6	329.83	344	380.78
120	224.5	305.52	331.2	372.71
130	191.1	295.71	299.2	343.82
140	186.1	283.6	285.8	321.69
150	170.6	267.1	273.8	271.24
160	155.3	205.81	272.2	268.84
170	129.4	197.92	247.9	246.86
180	115.43	161.42	165.6	195.65
190	89.39	137.72	124.4	189.09
200	97.23	134.08	112.4	152.31
210	76.62	163.3	102.9	117.39
220	58.85	75.707	81.06	104.01
230	57.51	81.947	70.45	87.1
240	31.22	61.433	42.25	78.771
250	25.73	64.136	38.4	68.408

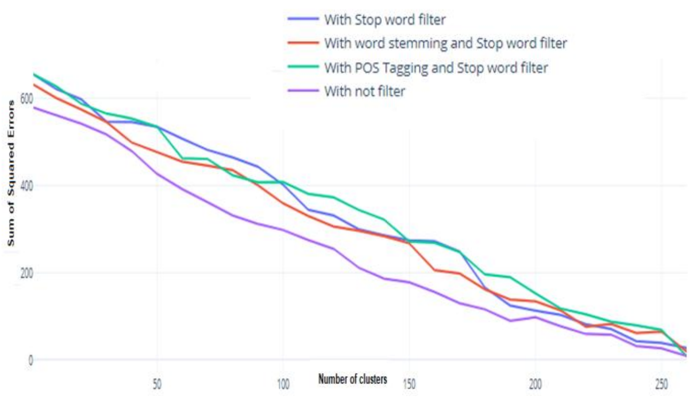


Figure 17. Effect of feature extraction models on SSE

Data from table 5 suggests that the feature extractions did not help in improving the results in anyway. Purity for all number of clusters did not seem to vary much while using the feature extraction methods and when not using any kind of filters. This result was a bit unexpected as the feature extractions should have helped in improving the purity.

So to diagnose this, we looked into the tokens while not applying any filter and after applying the filters. The result is as shown in Figure 18.

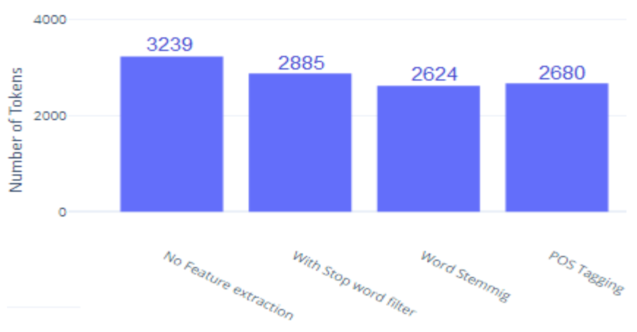


Figure 18. Plot showing number of tokens after using different feature extraction models

From the data (Figure 18), it seems that the number of tokens did not actually change much after applying filters and this is true for all types of filters we were using. This might be because of the nature of data we are clustering. Our main sources of data are description fields from RSS, Tweets, videos and web scraping all of which do not have long texts. The description in RSS is hardly two sentences, tweets by nature are very short and videos do not have long description in them so even when filters are applied, since the text themselves are too short the filters do not make much impact on the data and this is why purity did not change.

We see similar results in case of SSE also (Table 6). In this case also since the tokens nearly remain same even with filters, SSE does not change.

Although, the feature extractions did not make much difference in the results, we wanted to continue with the implementation of POS Tagging feature since it provided us with more control over what types of tokens we want to use in clustering. However, there is a significant decrease in performance while using POS Tagging.

The time taken to complete tagging for different number of documents is shown in table 7.

Table 7. Tagging time for POS Tagger for different number of documents

Number of documents	Time (Mins)
267	2.1
688	4.5
946	6.52

As seen on Table 7, the time taken to finish tagging increases linearly with increase in number of documents and it takes about 2 minutes just to tag 267 documents. This is because the API compares each token in a sentence with its internal dictionary for tagging which takes very long time. Since, the run time is not acceptable, we completely discarded the POS Tagging in our workflow. The application thus only uses word stemmer and stop word filter as feature extraction models.

VI. Conclusion and Future Work

Through our project we tried to understand and implement various data collection strategies, feature extraction methods to filter the data or clean the data and implement a clustering algorithm to group the similar data together and show the user only the portion of data that the application sorted out to be important. Basically what we did was news aggregation, which is not a new idea in itself and there are already several websites which do it. But the main goal for us, was to learn how to collect, clean or enhance data, how to use machine learning to cluster the data and most importantly we learned how to analyse our results with a more methodical approach.

Although, we completed much of what we planned, we really tried to include another clustering model besides K-Means. We looked into several other clustering models and tried to implement DBSCAN model. We were using an API called Weka but we could not get proper results. So we were not able to include it. We would like to incorporate another clustering model and compare it to the K-Means.

Further, the UI currently is static. The web front just reads the data from a file created periodically by the application and displays it in the UI. We would like to make the website more dynamic with the ability to view hot news from past days as well.

Our project is available publicly at:

https://gitlab.tubit.tu-berlin.de/mandir123/DataScienceApplication.git

The database scripts which create tables and populate default values in tables along with the source code for the application are in location:

https://gitlab.tu-berlin.de/mandir123/DataScienceApplication/tree/master/Source

Acknowledgment

Although hard work and proper management are the key to success of any project, there are several other crucial factors like guidance, motivation, support and feedbacks that are significant for any kind of endeavour. The success and final outcome of our project would not have been possible without the proper supervision and assistance of our supervisor Mr. Mohammad Mahdavi. We would also like to thank Prof. Dr. Ziawash Abedyan for his valuable suggestions and helpfulness throughout the duration of the project. The weekly meeting to review progress and discuss the problems during the development really kept us motivated and on track during the development process. We are also grateful to all the advisors for listening to us and giving us suggestions during our presentations. All the supervisors and professor were always easy to reach and always helpful which kept us inspired throughout the course duration so we will always be thankful for that.

Finally we would also like to thank BigDaMa department for organising this course.

References

[1] (2019) A statistics portal. [Online]. Available: https://www.statista.com/topics/1640/news/

[2] (2019) Pew research center official website. [Online]. Available: http://www.journalism.org/fact-sheet/digital-news/

[3] Lars Vogel (2017, Oct.) "RSS feeds with Java - Tutorial". [Online]. Available: https://www.vogella.com/tutorials/RSSFeed/article.html

[4] Adrian Matei (2013, Sept.) "Reading/Parsing RSS and Atom feeds in Java with Rome" . [Online]. Available: http://www.codingpedia.org/ama/reading-parsing-rss-and-atom-feeds-with-rome/

[5] (2017) Rome tools website. [Online]. Available: https://rometools.github.io/rome/

[6] Eugen Paraschiv (2019, Jan) Twitter4j Tutorial. [Online]. Available: https://github.com/eugenp/tutorials/tree/master/Twitter4J

[7] (2019) Twitter4J Documentation. [Online]. Available: http://twitter4j.org/javadoc/twitter4j/Twitter.html

[8] Dhiraj Ray (2017, July) "Open NLP POS Tagger Example". [Online]. Available: https://www.devglan.com/artificial-intelligence/open-nlp-pos-tagger-example

[9] (2019) Stanford NLP Group, Stanford Log-linear Part-Of-Speech Tagger. [Online]. Available: https://nlp.stanford.edu/software/tagger.shtml

[10] (2019) Opennlp pretrained models. [Online]. Available: http://opennlp.sourceforge.net/models-1.5/

[11] Mihail salnikov (2018, Aug.) "Text clustering with K-means and tf-idf". [Online]. Available: https://medium.com/@MSalnikov/text-clustering-with-k-means-and-tf-idf-f099bcf95183

[12] Jonathan Zong (2019, Feb.) "K Means Clustering with Tf-idf Weights". [Online]. Available: http://jonathanzong.com/blog/2013/02/02/k-means-clustering-with-tfidf-weights

[13] Debashishc (2016, Aug.) "Clustering documents using k-means algorithm with cosine similarity and tf-idf". [Online]. Available: https://github.com/debashishc/doc-clustering

[14] Sanjivaini Tushar Deokar, "Text Document Clustering using K-Means Algorithm", International Journal of Technology and Engineering Science [IJTES] vol (4), pp. 282- 286, July 2013

[15] Tippaya Thinsungnoena, Nuntawut Kaoungkub, Pongsakorn Durongdumronchaib,Kittisak Kerdprasopb, Nittaya Kerdprasopb, "The Clustering Validity with Silhouette and Sum of Squared Errors", Proceedings of the 3rd International Conference on Industrial Application Engineering, pp. 44- 51, 2015

[16] (2019) Microsoft SQLServer official website. [Online]. Available: https://www.microsoft.com/en-us/sql-server/sql-server-2008

[17] (2019) W3C markup validation service. [Online]. Available: https://validator.w3.org/feed/docs/rss2.html

[18] (2019) Oauth official website. [Online]. Available: https://oauth.net/2/

[19] (2019) News API official website. [Online]. Available: https://newsapi.org/

[20] (2019) Stanford NLP Group, Stemming and lemmatization. [Online]. Available: https://nlp.stanford.edu/IR-book/html/htmledition/stemming-and-lemmatization-1.html

[21] (2019) Snowball system website. [Online]. Available: http://snowball.tartarus.org/

[22] (2019) Stanford NLP Group, Dropping common terms: stop words. [Online]. Available: https://nlp.stanford.edu/IR-book/html/htmledition/dropping-common-terms-stop-words-1.html

[23] (2019) NLTK's Stop words list. [Online]. Available: https://gist.github.com/sebleier/554280

[24] (2019) Apache OpenNLP Documentation. [Online]. Available: https://opennlp.apache.org/docs/1.8.0/manual/opennlp.html

[25] Dhiraj Ray (2017, July) "Apache Open NLP Maven Eclipse Example". [Online]. Available: https://www.devglan.com/artificial-intelligence/apache-opennlp-maven-eclipse-example

[26] (2019) Stanford NLP Group, Evaluation of clustering. [Online]. Available: https://nlp.stanford.edu/IR-book/html/htmledition/evaluation-of-clustering-1.html