**News Summarization and Clustering**

Noah Raphael David, C0846073

AIMT

Toronto Canada.

Jebin George, C0850509

AIMT

Tonroto Canada

Akash Guje, C0835384

AIMT

Tornoto Canada

# ABSTRACT

The most valuable resource in the world is time and in today's hectic environment people frequently lack the time to read the news, even when it is of interest to them. A full newspaper is supposed to take about 60 minutes to read. People often struggle to find time to read the news because of their demanding jobs and demanding family obligations. But everyone is aware of how important it is to stay current with news. This initiative aims to address and resolve this issue. By delivering the major news story in the smallest amount of time feasible, our goal is to make reading the news simple and quick. The concept of text summarizing differs slightly from that of topic modeling's key phrase extraction. According on how long we would want the summary to be, the final product in this scenario takes the form of a paper but only contains a few sentences. This is comparable to an executive summary or abstract in a research report. The primary goal of automated document summarizing is to carry out this summarization using only computer programs and no human involvement.

Building and automating the process of summarizing papers by looking at their content and context will be made easier with the aid of mathematical and statistical models. In this study, extractive-based summarizing has been used to examine the text summarization challenge.

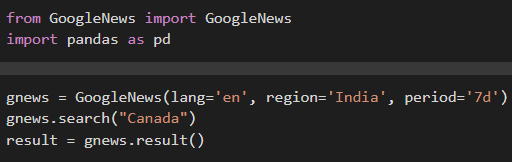
# INTRODUCTION

We are creating a news summarizing API with this problem in mind, where you may enter the topic, you're interested in learning about and receive the condensed version of lengthy and tiresome news items from Google news. Additionally, this project will gather the top headlines by clustering the news documents and getting the best headline from each cluster. Data compression and information understanding, which are both essential to information science and retrieval, are intimately related to summarization. Text summarization technology can enhance information extractions systems *and enable readers to quickly scan through a huge number* of texts for crucial information. Automatic summarization is one of the most significant natural language processing jobs, but it is also one of the least addressed ones, according to recent research. There are primarily two methods for text summarizing, namely Extractive Summarization and Abstractive Summarization. Extractive techniques are useful for locating the most pertinent data and that is what we have used in our project. Further we have used K-means algorithm to cluster similar news and leveraged the K-means to extract top 5 headlines by picking up headlines from each cluster.

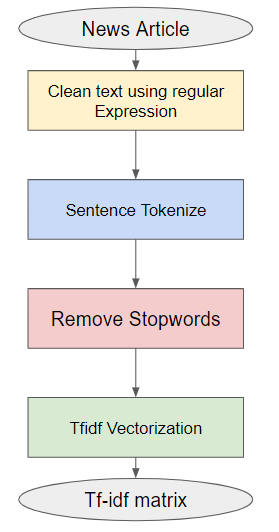
III. EXPERIMENT

This project involved using various packages, tools and API to get to our desired output. The experiments are as follows:

**Google news:** Googlenews package within python is an API used to scrape and retrieve news from google news platform. Using this package, we can extract news based on topic for last N number of days. First and foremost, we need to install Googlenews package. This was achieved using Pip command. Then we created an object for Googlenews. To instantiate the object, the language and time was be passed. Period is a parameter that represents news from last N days. So, we might write the previous three days as "3d" or the previous day as "1d." In our case we have used “7d” to extract news for past 7 days. Apart from period, we can mention the region and the topic. For the purpose of experiment and trying out parameters ability we have tried giving “India” as the region and “Canada” as the topic. We were able to extract news links from Indian media, news related to Canada in English language.

*Figure 1: Gnews object declaration* *and extracting news links.*

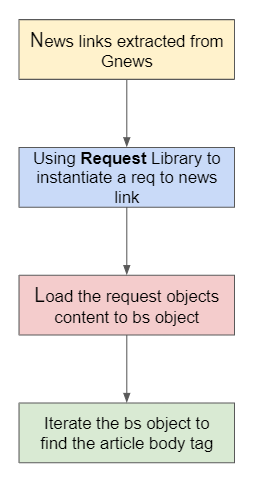
**Beautiful soup:** Beautifulsoup is package within python by which we can parse the HTML and XML documents. Since we have already extracted news links using Googlenews based on various factors, now we will parse through each URL to extract the entire news article.



We also used **request** library present in python to instantiate a request to the given news article. By this we were able to collect the HTML skeleton of the article webpage. Now, this request object’s content attribute is passed to the beautiful soup object to extract the news.

This entire functionality is encapsulated in the **extract**

**\_news(link) function**. In this function we also collect the HTML script tag with the attributes

 “type='application/ld+json', text=True”. Then from the retrieved HTML script tags we look for the article body tag or p tag to find the entire news article content.

*Figure 2: Web scraping News article Flow –*

*bs is beautiful Soup object and req is request object*

Now that we have extracted the description of the various news, we can now move onto the next phase of project, i.e. Data preprocessing.

**Data preprocessing:** The process of text preprocessing involves cleaning the text data and getting it ready to be fed into the model. Noise in text data comes in many different forms, including emotions, punctuation, and different cases of text. Text preparation is handled by numerous libraries and techniques. Regular expression (re) is one of the most common libraries for text cleaning. The next-level libraries are used for natural language tasks including stopword removal, named entity identification, phrase matching, part of speech tagging, etc., these tasks are achieved with help of different libraries such as NLTK and spacy. In text preprocessing, the following tasks were performed.

*Figure 3: Data-Preprocessing Flow*

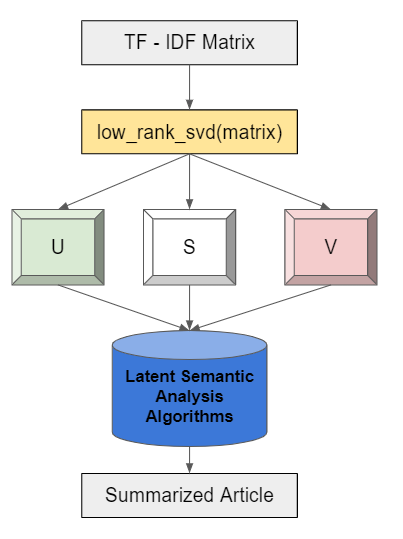
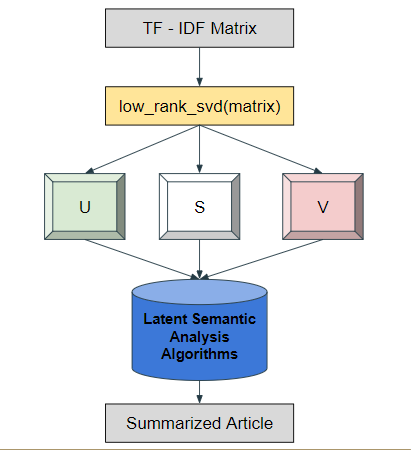
**Cleaning News Articles: -** we used the regular expression library to remove the below noise from the data

* HTML Tags <tag>.
* Markdown Url’s.
* Text or Code in brackets.
* Standalone sequences for specials, punctuations and newline characters.

This entire functionality is include in the clean(article) function.

**Tokenization:** Tokenization is a process of breaking a document into sentence or sentence into words. As per our use case, we have used Sentence tokenization from NLTK library.so that we shouldn’t lose any data from the news article.

**Stopwords:** Stopwords are a list words that needs to be filtered out before passing the text to the models as these words are repetitive and no not add meaning to the documents (insignificant words). This function is encapsulated in removestopwords(finaltext) function.

**Vectorization:** This method is one of the most important steps in NLP where we convert the raw texts into numbers, this is because the machine can only understand numbers and not text. This process is known as Vectorization. This can be achieved using various techniques and for our use case we have used TFIDF, which stands for Term Frequency – Inverse document frequency. TfidfVectorizer is a module that is a part of sklearn.feature\_extraction.text that convert the words into numbers based on term frequency and inverse document frequency. This process involves with creating an object of TfidfVectorizer and then fitting the model to our data. The final outcome of Data preprocessing is the tfidf-matrix.

**Modeling:** In the modeling part, We have performed two main experiments.

* Text Summarization – gets news articles in summarized form.
* Document Clustering – gets the top headlines.

**Text Summarization:** Summarization is the method of condensing lengthy texts so that the summary contains all of the crucial information from the original text. There are 2 types of approach that can be used for summarization namely Extraction based and Abstraction based. For this project we have chosen the extractive based approach where this approach involves selecting out the key words and sentences from the texts is one strategy. The summary is then created by combining all the key lines. In this instance, every word and line in the summary genuinely comes from the source material that is being summarized.

In brief, this technique gives us the summary by selecting the subset of the sentences from original content. The idea behind using extractive summarization is that we want to display the news as it is but in an efficient manner where the reader gets to read only the crucial points that the news would want to convey. This way we can achieve our goal of reducing the time spent on reading the entire news.

We have employed latent semantic analysis to do text summarization. The fundamental tenet of latent semantic analysis (LSA) is that there exists a latent structure in each document among terms that are contextually related and ought to be corelated in the same single space. Singular Value Decomposition is a linear algebra technique that is the foundation of our approach. Factorizing a real or complex matrix is what SVD does. Formally speaking, SVD is defined as follows. Consider the dimensions of a matrix M, where m stands for the number of rows and n for the number of columns. Mathematically, SVD can be used as a factorization to represent the matrix M as follows:

Mm\*n = Um\*m \* Sm\*n \* VTn\*n

* U is a m\*m unitary matrix such that UTU = Im\*m where I is the identity matrix.
* S is a diagonal m\*n matrix with positive real numbers on the diagonal of the matrix.
* VT is a n\*n unitary matrix. Such that VTV = In\*n, where I is the identity matrix

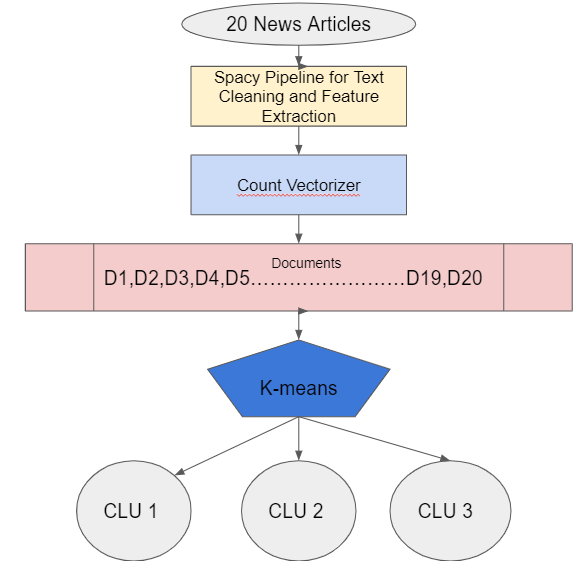
*Figure 4: Latent Semantic Analysis Architecture*

We have written a function named low\_rank\_svd(matrix) to get the low rank approximation matrixes using scipy. sparse.linlag package. Then we pass our tfidf matrix to the low\_rank\_svd() function to get the term\_topic\_matrix (U), singular\_values(S), topic\_document\_matrix (V) with a singular count value K which denotes number od topics is set to the considering.

1. Get the sentence vector from the matrix V
2. Get the top k singular values from S.
3. Apply threshold-based approach to remove singular values that are less than half of the largest singular values if any exists. This a heuristic and you can play around with this value if you want
4. Multiple each term sentence columns from V squared with its corresponding singular value from S also squared to get sentence weights for topics
5. Compute the sum of the sentence wights across the topics and take the square root of the final score to get the salience score for each sentences in the document.
6. Once got the salience score we will be displaying the sentences based on their salience score.

This entire text Summarization concept is encapsulated in the function get\_summarized\_article(article) in our project.

**Document Clustering:**  Clustering is a process of grouping similar documents together and dissimilar document far. In Document clustering we will be clustering documents based on the words present in each document. To perform this we have to do some text cleaning and feature engineering.

To perform text cleaning, we have created a spacy pipeline which perform the below operations

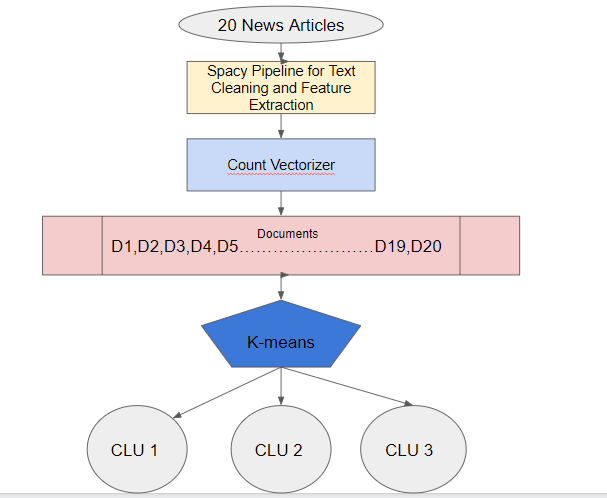
* Cleaning the data using clean() function written for text summarization
* Tokenizing the text using WhiteSpcactokenizer from NLTK
* Remove stopwords using the remove\_stopwords function() written for text summarization.

These functions are given as estimators for the spacy pipeline by leveraging the Function Transformer() method from sklearn.preprocessing. This Complete functionality is encapsulated in the pre\_process(list(news)) function. Now, we transform the news articles to the list of sentences for each article by using the above pre\_process(list(news)) function. The next part is the Feature Engineering.

Feature engineering is a section where extract features which will be useful for our clustering. We achieved this by using CountVectorizer from sklearn.feature\_extraction.text. Count vectorizer basically returns us a the word frequency matrix similar to Bag-of-Words model where rows are the documents and columns are the words. That words extracted using Count Vectorizer are based on unigrams and bigrams such that each feature occurs at least in 1 document and at most 4 documents. Now that we have our features and documents ready we can start the clustering analysis.

We used K-means clustering algorithm which is based on centroid clustering which tries to cluster different data points into groups or clusters with equal variance. The main intuition behind our model is to minimize the inertia which is also know as within clusters sum of squares. The challenge that we encountered is to select the right number of clusters in advance. However, this clustering algorithm is very popular due to its ease of use and its capability to being scalable with huge amount of data.

We are collecting 20 news articles for every search and clustering them into 3 groups. The main objective here is to get the top news from each cluster, as each individual cluster has same features which apparently means that it represents the same news. So, we collect the title of one document from each cluster and present it as Headlines for our project.



*Figure 4: Generating Top Headlines using Clustering*

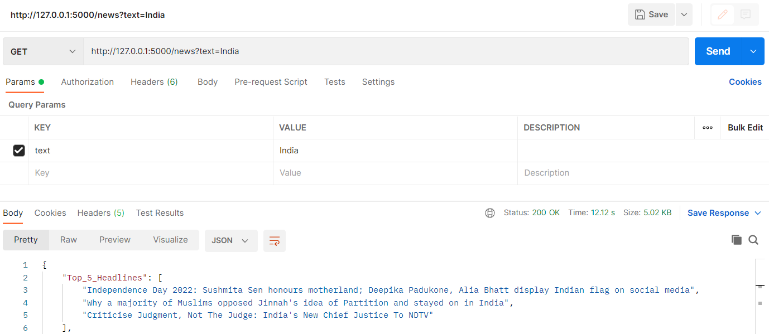
This complete clustering process is defined in the cluster() function of our project.

**API Endpoint Generation:** To present the use-case that our project is going to serve we choose API is the best option as it is easy to build and scale anywhere. To generate an API we have used the flask framework in python.

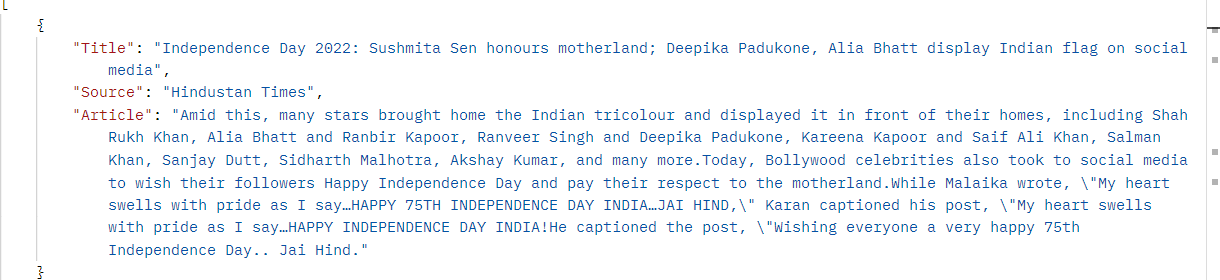
Flask is a web framework where you can generate a API endpoint by using python decorators. Firstly we installed flask and declared a flask app which supports our project. Then assigned a decorator named app.route(‘/news’) to the function get\_news(). This get\_news function handles all our text summarization and clustering code flow.

# RESULT

On successful completion of the project, the final product is an API endpoint which deduce the news into simplest form possible and gets the top headlines. The below screen shots are the news about the topic India. Among the two screenshots one is top headlines return in json format and other is the Summarized news article with the title and the publisher information



*Figure 5: Top Headlines Output*



*Figure 6: Summarized article*

# DISCUSSION

Through this project, we gained more expertise in the areas of site design, text preprocessing, text summarization, clustering, and constructing API endpoints. The project's analysis clarifies how headlines are created and how effectively they convey the content of news articles. This research provides a rough sense of the ease and efficacy with which an artificial intelligence system may carry out a human work.

# CONCLUSION

This project cuts the average reader's time reading the news by 50%. Additionally, the reader can quickly research a topic of interest by typing a term or heading into the search bar. The creation of an Android app and its playstore publication are the project's future objectives.

# References

[1] Danang Tri Massandy, M. L. (2014). Guided summarization for Indonesian news articles. *IEEE*.

[2] Ekaterina Zolotareva, T. M. (2020). *Abstractive Text Summarization using Transfer Learning.* Budapest: CEUR Workshop.

[3] Hujia Yu, C. Y. (2016). News Article Summarization with Attention-based. *stanford*, 1-7.

[4] Josef Steinberger, K. J. (2004). Using Latent Semantic Analysis in Text Summarization and Summary Evaluation. *researchgate*.

[5] Roy, A. (2020). Understanding Automatic Text Summarization-1: Extractive Methods. *towardsdatascience*.

[6] Sarkar, D. (2019). *Text Analytics with Python.* Bangalore: Apress.

[7] Woojin, G. (2022). How to read Google News RAPIDLY using a short python script. *medium*.

[8] Yue Dong, Y. S. (2018). *BanditSum: Extractive Summarization as a Contextual Bandit.* Brussels: Aclanthology.

# ACKNOWLEDGEMENTS.

[1] The research has been carried by 3 students from Lambton college, Toronto, Canada. The author would like to thank Parisa Naraei for giving the guidance to complete this project.