TWITTER BOT FOR SUMMARIZING ARTICLES BASED ON TWITTER TRENDS

A Project Report

Submitted by

MS. SAMREEN REYAZ

MR. DISHANT BHATT

MR. GABRIEL FRANCIS

Under the guidance of

PROF. PANKTI DOSHI

in partial fulfilment for the award of the degree of

BACHELOR OF TECHNOLOGY

COMPUTER ENGINEERING

At

MUKESH PATEL SCHOOL OF TECHNOLOGY MANAGEMENT AND ENGINEERING, NMIMS, MUMBAI

MARCH 2020

DECLARATION

We, Samreen Reyaz, Dishant Bhatt and Gabriel Francis, Roll Nos. B005, B013 and B027 B.Tech (Computer Engineering), VIII semester understand that plagiarism is defined as anyone or combination of the following:

1. Un-credited verbatim copying of individual sentences, paragraphs or illustration (such as graphs, diagrams, etc.) from any source, published or unpublished, including the internet.

2. Un-credited improper paraphrasing of pages paragraphs (changing a few words phrases, or rearranging the original sentence order)

3. Credited verbatim copying of a major portion of a paper (or thesis chapter) without clear delineation of who did wrote what. (Source: IEEE, The institute, Dec. 2004)

4. I have made sure that all the ideas, expressions, graphs, diagrams, etc., that are not a result of my work, are properly credited. Long phrases or sentences that had to be used verbatim from published literature have been clearly identified using quotation marks.

5. I affirm that no portion of my work can be considered as plagiarism and I take full responsibility if such a complaint occurs. I understand fully well that the guide of the seminar/ project report may not be in a position to check for the possibility of such incidences of plagiarism in this body of work.

Signature of the Student:

Name:	Samreen Reyaz	Dishant Bhatt	Gabriel Francis
Roll No.:	B005	B013	B027
Place:		Mumbai	
Date:			

CERTIFICATE

This is to certify that the project entitled "Twitter Bot For Summarizing Articles Based On Twitter Trends" is the bonafide work carried out by Samreen Reyaz, Dishant Bhatt and Gabriel Francis of B.Tech (Computer Engineering), MPSTME (NMIMS), Mumbai, during the VIII semester of the academic year 2019-2020, in partial fulfillment of the requirements for the award of the Degree of Bachelors of Engineering as per the norms prescribed by NMIMS. The project work has been assessed and found to be satisfactory.

Prof. Pankti Doshi

Internal Mentor

Examiner 1

Examiner 2

Dean

MPSTME, NMIMS University

TABLE OF CONTENTS

LIST OF FIGURES

Evaluation of text summarization

Twitter

LIST OF TABLES

LIST OF EQUATIONS

ABBREVIATIONS

ABBREVIATION	DESCRIPTION
TF-IDF	Term Frequency – Inverse Document Frequency
BLEU	Bilingual Evaluation Understudy
BP	Brevity Penalty
ROUGE	Recall-Oriented Understudy for Gisting Evaluation
LCS	Longest Common Subsequence
WOEID	Where On Earth IDentifier
NLP	Natural Language Processing
NLTK	Natural Language Toolkit (Python)
API	Application Programming Interface
DOM	Document Object Model
GPU	Graphical Processing Unit
CPU	Central Processing Unit
RAM	Random Access Memory
XML	Extensible Mark-up Language
UML	Unified Modelling Language
JSON	JavaScript Object Notation
BBC	British Broadcasting Channel
URL	Uniform Resource Locator
TOI	Times of India

ABSTRACT

With the boom of social media in the 21st century, it has grown to impact nearly every aspect of our life. This widespread nature of social media makes it a critical platform for the spread of information. This information, more often than not, is filled with flaws such as incomplete information, uncredited sources and may even go as far as being downright false. Platforms such as Facebook, Twitter, Instagram, and many more receive more than a billion active users every day with double that active every month, making them excellent platforms for the spread of information. There have always been suspicions and even some verified examples of these platforms being used to influence the lives of people by feeding them news from biased sources in order to influence the decisions they make in real life.[14]

Twitter is a critical player in the propagation of information with 500 million tweets containing snippets of information being sent every day, proved to be the ideal platform on which we could implement a system to verify and then spread information. The decision was made to consider the trending topics on twitter to be the keywords on which more information must be found and propagated.

Summarization of the selected articles, however, proved to be the greatest challenge. We decided to focus on extractive methods of summarization where sentences are lifted from the source verbatim instead of abstractive methods, which generated sentences due to its high complexity and resource requirements. We compared a variety of methods such as word frequency, term frequency-inverse document frequency, TextRank, latent semantic analysis based on the intrinsic evaluation. We also modified word frequency to obtain better results.

Bots proved to be the ideal form with which could implement said system as it reduces the probability of human error, removes work redundancy, and is inherently unbiased. The bot would scrape data off articles from reliable sources, summarize them using the best method from those mentioned above, and then tweet them along with a link to the actual source.

CHAPTER I

1. INTRODUCTION

1.1.Background

Robots on their own have revolutionized the way many companies operate, whether they are a part of hardware or software-based industries. They provide a simple, cost-efficient, and often time-efficient way of automating tasks that, if done by an actual person, would have been tedious. Examples of these are automobile manufacturing or chatbots that need to answer the simple questions of users. These robots, also known as bots in the field of computer science, have changed the way computer tasks, from as simple as scheduling start-ups for a particular program to complex processes such as automating web scraping, are accomplished. In recent times the prevalence of bots in social media has increased rapidly due to the ease by which they can be created and deployed in the industry. They can even be combined with an intelligent agent to perform advanced tasks where a dynamic understanding of the situation is required. Such bots then, ideally perform automated tasks with nuanced understanding to handle and adapt to varying circumstances.

Twitter is a social media microblogging website known for its propagation of information in the form of short messages of 280 (previously 140) characters. These tweets are broadcasted to all the followers of the tweeter as a message that appears on their feed. Another feature of the platform is categorizing the content of a tweet by using ‘hashtag’. This categorization of tweets allows for the generation of ‘trends’ which is based on the number of tweets that contain a particular hashtag in a short period.

Web Scraping is a process by which data from one or more web pages is extracted and converted into a form that can more easily be worked on to serve one's purposes. These purposes can range from industrial applications gaining crucial pricing information from a competitor's website or even a simple system to update an individual to the current gold prices. This technique is often used to run extensive data mining operations by converting the raw HTML webpage into usable data using various libraries in python or java.

Text Summarization is required in this age of information. More often than not, in modern-day life, we are within reach of a modular device that is an endless stream of data. With content being generated every millisecond around the world, it is quite challenging to extract the essential information in a feasible amount of time. Text summarization enables us to convert large amounts of data with information sparsely placed within it into a small amount of data filled with information. It is a deeply explored field that, as of yet, still does not yield relevant results for all cases and applications.

1.2.Project Overview

For our project, we have selected to make a Bot that helps the spread of summarized information on trending topics using Twitter as a platform. The bot would first retrieve a list of trends from Twitter by using the Twitter API. It would then analyze and select the trends that fit specific criteria that deem them viable for our use case. The viable trends are converted into phrases which are then searched for on multiple news websites to yield a list of articles about the trend. From this list, articles are selected based on specific criteria such as date of writing, type of article, and the chosen articles are sent to a summarizer. The summarizer summarizes these articles into strings of a maximum of 240 characters. We then post a tweet containing the summary, a link to the original article, and the hashtag that represents the trend.

1.3.What is Text Summarization?

The purpose of automatic text summarization by machines is similar to the reason why we humans create summaries, i.e., to produce a shorter representation of the original text. Through these years, several researchers have defined the definition of summary from their perspective. For instance, computer scientist Sparck Jones describes a summary as a "reductive transformation of a source text to summary text through content reduction by selection and generalization on what is important in the source."

Natural Language Processing comes in picture when we talk about automatic text summarization. The machine first needs to understand the content, be it news, chat-rooms, email threads, discussion lists, professional medical information, voicemail, or any other organized or unorganized data. It then has to extract essential keywords and information from it for the summary. Separating information as necessary or trivial is the main challenge in text summarization.

1.4.Motivation of Problem

The microblogging platform Twitter can be used to find out what topic is the talk-of-the-town at any given time based on what the people of the area are tweeting about. These trends are usually expressed using hashtags that are made up of words or phrases. These words or phrases, more often than not, do not give any real information on why that topic is trending. In order to gain more information on a topic, people themselves would have to look up articles online that may be hundreds if not thousands of words long, and among the more than a hundred million people using Twitter, many do not have the time or motivation to open and read such articles. Our project aims to develop a method to accurately summarize articles related to the trends from reliable news sources and post them as <280-character tweets that can be read by individuals in less than 20 seconds.

1.5.Problem Statement

Currently, many approaches and algorithms exist for summarizing passages and articles related to various domains. Few of them generate summaries that are close to human-generated summaries. However, there is a trade-off between the quality of the summary, the time taken to develop the system, and the time taken by the machine to analyze the text and create a summary. As there are many trends on Twitter at a given time, summarizing multiple articles related to each of them will be a computationally-heavy and time-consuming task. There is also the issue of grammatically incorrect machine-generated summaries when the summary is generated word-by-word, which seems to remain unsolved in the entire industry as of now.

What is required is an algorithm that will give outputs in a reasonable amount of time with acceptable quality. The project will be addressing this issue by comparing the current text summarization methods and finding the best among them, which works for news articles, and then improving upon its time or quality with minimum harm to the other parameter. This summarization algorithm will be used along with web-scraping tools to fetch the articles related to the trends at a given time and post grammatically correct and contextually accurate summaries on the Twitter platform in the form of a tweet.

1.6.Scope of the Problem

The project will compare and utilize extractive text summarization methods, i.e., summary generated from sentences chosen from the passage itself according to priority decided by the algorithm used. Another approach, i.e., abstractive summarization, uses techniques that generate a summary word-by-word. This may even include words or sentences that do not appear in the source documents. These methods interpret and examine the text using advanced natural language processing techniques in order to generate a new shorter text that conveys the most critical information. Abstractive summarization remains out of the scope for the project due to the complexity involved in designing the system and the time taken by machines with consumer-grade processing power to generate summaries.

It is also possible that trends on Twitter could be in languages other than English. Processing them for summaries will require additional reference documents and corpus. Hence, the focus of this project will be on selecting trends and summarizing articles in the English language only. Translations from other languages or processing other languages directly are out of the scope of our project.

1.7.Assumptions and Limitations

An assumption is made that all articles found online from various news websites are unbiased and authentic. For this very reason, we have chosen highly reputed new sources that are internationally recognized and used by millions around the world.

One of the significant limitations we have come across is the lack of powerful workstations. Summarization by advanced techniques involving neural networks and machine learning requires a lot of resources and computation power to train the system before it can be used. Our home laptop systems do not possess the GPU, CPU, or RAM required to undertake these operations in a reasonable amount of time.

1.8.Organization of Report

In **Chapter 2**, the literature review conducted for this project has been discussed. In **Chapter 3**, a solution to the problem statement has been propounded. **Chapter 4** discusses the implementation of the project. In **Chapter 5**, the results obtained have been elucidated and analyzed. Finally, **Chapter 6** concludes this paper and discusses future directions.

CHAPTER II

2. REVIEW OF LITERATURE

2.1.Numerical Techniques to Represent Text Data [15]

Written language can be easily understood by people as we have been trained for years to know the particular meaning of words, the contextual meaning of words, and fully understand the structure of a piece of text. A computer system, however, does not have this capability and mainly only understands numbers. Thus, we have to convert our textual data into some form of numerical data that a program can understand, run operations on, and give modify the text into a new form according to our needs. We talk about the various methods for this conversion below.

2.1.1. 1-Hot Encoding

In this form of representation, each word in the document is represented in the form of a 1-hot vector, a vector where one bit is 1, and the rest are 0.

Figure 2.1 is an example of 1-Hot encoding applied to the sentence, 'the quick brown fox jumps over the lazy dog.' Each word in the sentence is given a unique vector to identify it quickly.

This forms a simple method by which each word can be represented uniquely. However, with large corpus consisting of many different words, the size and number of 1-hot vectors increase rapidly. This method also gives no way of obtaining the semantic meaning of the word or finding how each word is related to each other, making it useless for summarization applications.

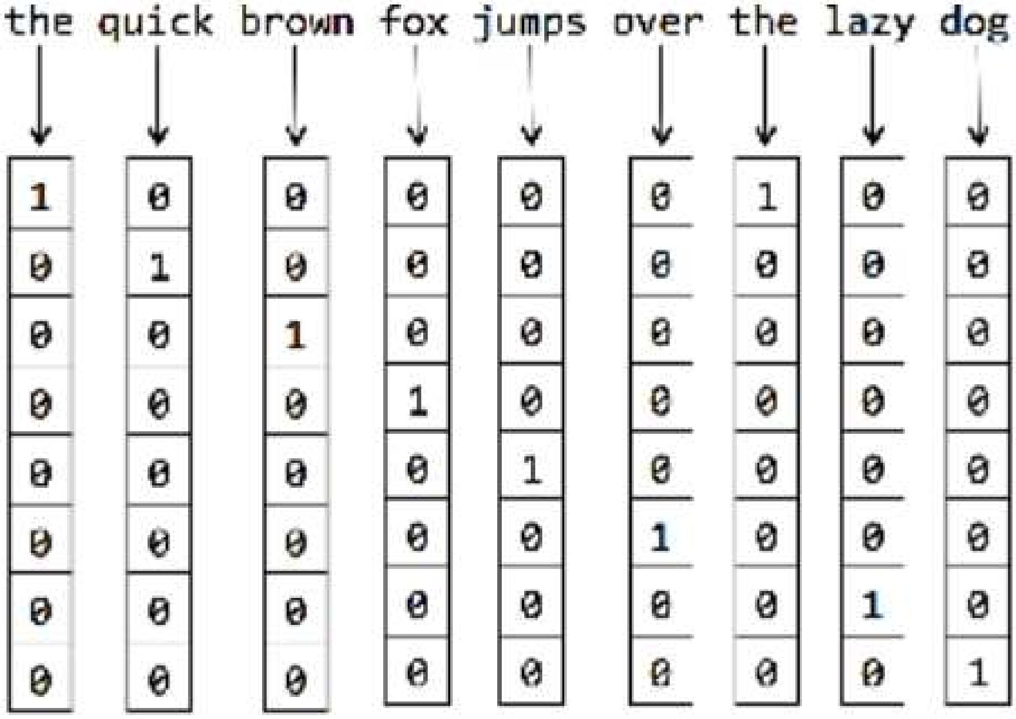
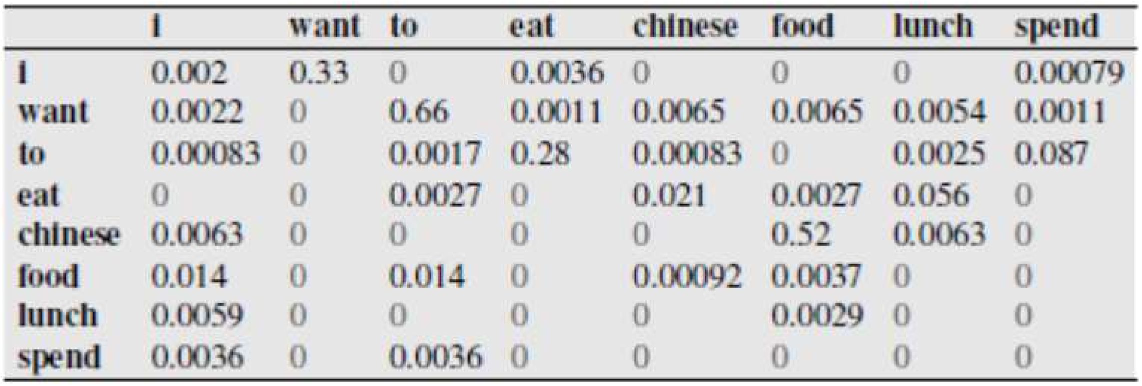


Figure 2.1: 1-Hot Encoding [15]

2.1.2. N-Gram Model

The N-gram model refers to a matrix of N-dimensions wherein each cell stores the probability of the N-words that point to that cell occurring in sequence.



	i	want	to	eat	chinese	food	lunch	spend
i	0.002	0.33	0	0.0036	0	0	0	0.00079
want	0.0022	0	0.66	0.0011	0.0065	0.0065	0.0054	0.0011
to	0.00083	0	0.0017	0.28	0.00083	0	0.0025	0.087
eat	0	0	0.0027	0	0.021	0.0027	0.056	0
chinese	0.0063	0	0	0	0	0.52	0.0063	0
food	0.014	0	0.014	0	0.00092	0.0037	0	0
lunch	0.0059	0	0	0	0	0.0029	0	0
spend	0.0036	0	0.0036	0	0	0	0	0

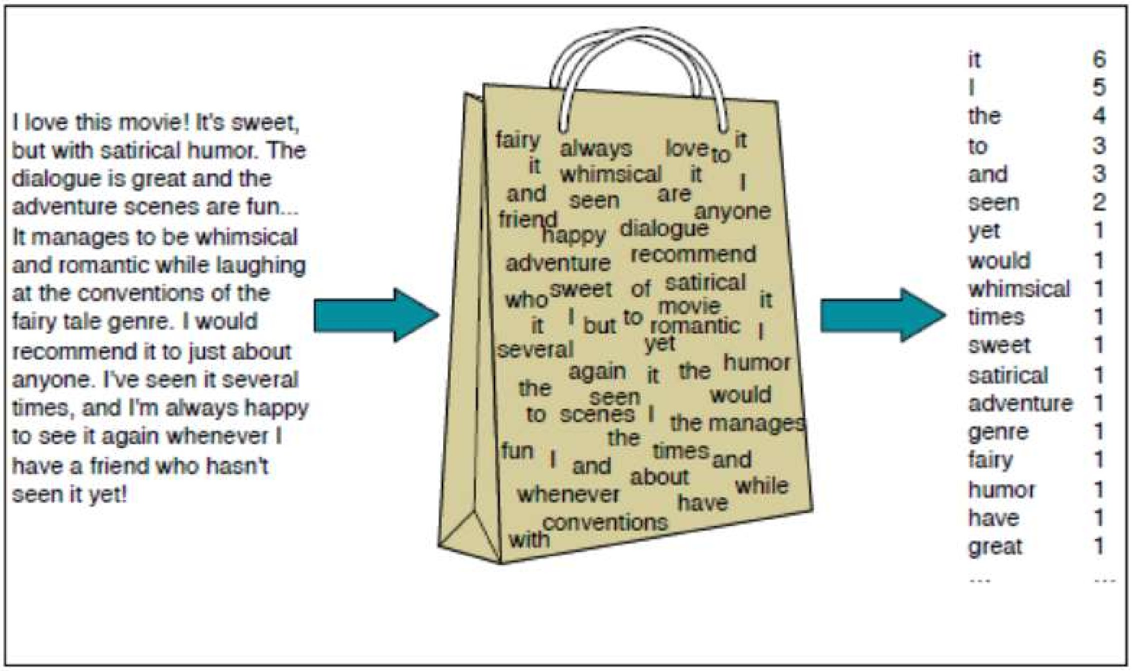
Figure 2.2: Bigram Model [15]

In Figure 2.2, the probabilities of each combination of the words related to wanting to each Chinese food in sets of 2 as it is a bi-gram model. We can see that the probability of a nonsensical phrase such as ‘eat want’ has a 0% chance of occurring.

This model greatly helps in sentiment analysis and thus helps in summarization as well. However, there are more sophisticated models that can serve this purpose.

2.1.3. Bag-of-words

The words of the corpus are taken as an unordered set. This set of words can then be worked upon using various techniques in order to extract a summary. Unlike conventional sets, a bag-of-words also contain repetitions which help us in understanding the meaning of a text.



I love this movie! It's sweet, but with satirical humor. The dialogue is great and the adventure scenes are fun... It manages to be whimsical and romantic while laughing at the conventions of the fairy tale genre. I would recommend it to just about anyone. I've seen it several times, and I'm always happy to see it again whenever I have a friend who hasn't seen it yet!

it	6
I	5
the	4
to	3
and	3
seen	2
yet	1
would	1
whimsical	1
times	1
sweet	1
satirical	1
adventure	1
genre	1
fairy	1
humor	1
have	1
great	1
...	...

Figure 2.3: Bag of Words [15]

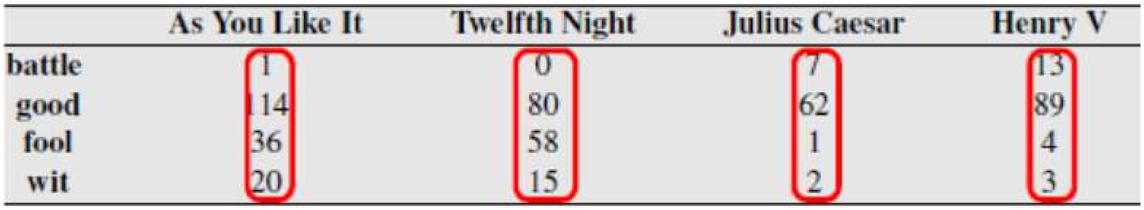
Figure 2.3 is a sample representation of a bag of words. The words from the given corpus on the left are taken one by one and placed into a set. This set acts as a bag containing the words in no specific order.

This bag-of-words representation of words is used in many basic extractive summarization techniques such as WF and TF-IDF. We shall be using this representation in our implementations of the same.

2.1.4. Vector Semantics

Vector Semantics involves defining a word by counting the number of times and what other words occur in its environment, and we can represent the word by a vector, a list of numbers, a point in N-dimensional space. Such a representation is usually called embedding.

This can be used for topic modeling as corpi with similar vector semantics can be expected to have the same meaning. This similarity can be found by using cosine similarity



	As You Like It	Twelfth Night	Julius Caesar	Henry V
battle	1	0	7	13
good	114	80	62	89
fool	36	58	1	4
wit	20	15	2	3

Figure 2.4: Vector Semantics [15]

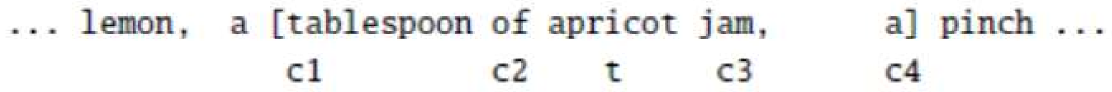
In Fig 2.4, we can see the number of times certain words occur in the works of William Shakespeare. Using this, we can say that ‘Henry V’ and ‘As You Like It’ have similar themes are the frequency of certain words is similar. While on the other hand, Twelfth Night would have a completely different theme.

In real-world applications, more advanced forms of vector semantics are typically used.

2.1.5. Word2Vec

Representing words or documents by sparse and long vectors in the methods, as mentioned above, is not practically efficient. Those vectors are typically sparse because many positions are filled by zero values.

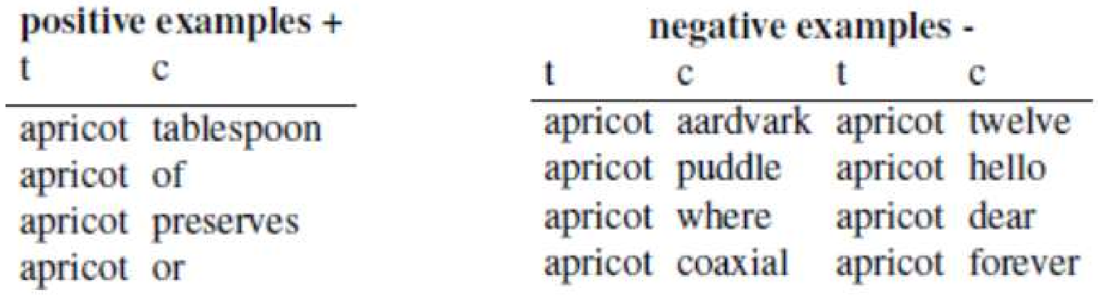
In this form of representation, we train the system by using a corpus. A matrix is used with words on both axes and the cells representing a probability those words will be together The number of times two or more words occur in the vicinity of each other will cause the probability to increase while the contrary will cause the probability to remain at whatever low amount it was previously.



```
... lemon,  a [tablespoon of apricot jam,      a] pinch ...
               c1          c2    t     c3       c4
```

Figure 2.5: Word2Vec Sample Sentence [15]

In Figure 2.5, we are taking the word apricot. We are then looking at its neighborhood of words, in this case, the two words preceding and succeeding it. We can then modify the matrix probabilities based on this information



positive examples +

t	c
apricot	tablespoon
apricot	of
apricot	preserves
apricot	or

negative examples -

t	c	t	c
apricot	aardvark	apricot	twelve
apricot	puddle	apricot	hello
apricot	where	apricot	dear
apricot	coaxial	apricot	forever

Figure 2.6: Updation Criteria of Word2Vec Matrix [15]

The updation of the value of apricot with respect to adjacent words are shown in Figure 2.6. This updation will increase the probability apricot will occur in the vicinity of the words in the pairs considered positive examples, while the negative examples imply those words will not occur near apricot.

Word2Vec is considered a potent form of representation with heavy use in advanced semantic analysis operations. However, because of the high training times and unnecessary complexity regarding summarization, we will not be using this form of representation

2.1.1. GloVe Language Model

GloVe (Global Vectors) takes a different than yet similar approach to Word2vec. Instead of extracting the embeddings from a neural network or a logistic regression that is designed to perform a classification task (predicting neighboring words), GloVe optimizes the embeddings directly so that the dot product of two-word vectors equals the log of the number of times the two words will occur near each other. This forces the embeddings vectors to encode the frequency distribution of which words occur near them.

This is generally considered an upgrade from Word2Vec as it has all the advantages while also being simpler to train and user. This, too, is too complex for our application.

Based on our analysis of the various numerical techniques for representation of text, we have chosen to use vector semantics and bag of words as they seem the most appropriate in the case of extractive text summarization.

2.2.Approaches to Text Summarization

Automatic text summarization systems can be broadly categorized into two different types based on output type, abstractive, and extractive summarization [2] (Figure 2.7). Extractive summaries are produced by identifying meaningful sentences that are directly selected from the document. Most of the summarization systems that have been developed are for extractive type summaries. In abstractive summarization, the selected document sentences are combined coherently and compressed to exclude unimportant sections of the sentences.

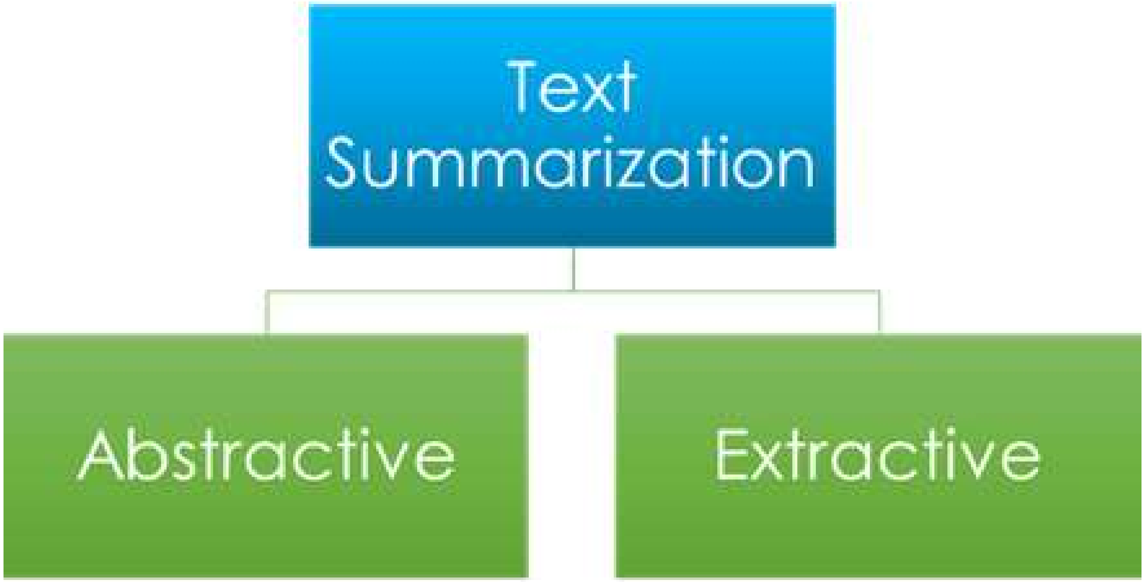


Figure 2.7: Text Summarization Approaches

However, abstractive methods take much longer to train and require a lot of data, and extractive summarization methods are faster but equally intuitive as abstractive methods [5]. Hence, as of now, our primary focus will be on creating a higher quality extractive summarization method that works for articles of various sizes and topics.

2.3.Extractive Text Summarization

Figure 2.8 shows the various algorithms for extractive text summarization, such as Word Frequency, Term Frequency-Inverse Document Frequency, TextRank, and Latent Semantic Analysis. There also exist approaches that use Machine Learning and Deep Learning for generating summaries [2].

We have implemented these methods on our own to analyze and understand the results of various algorithms.

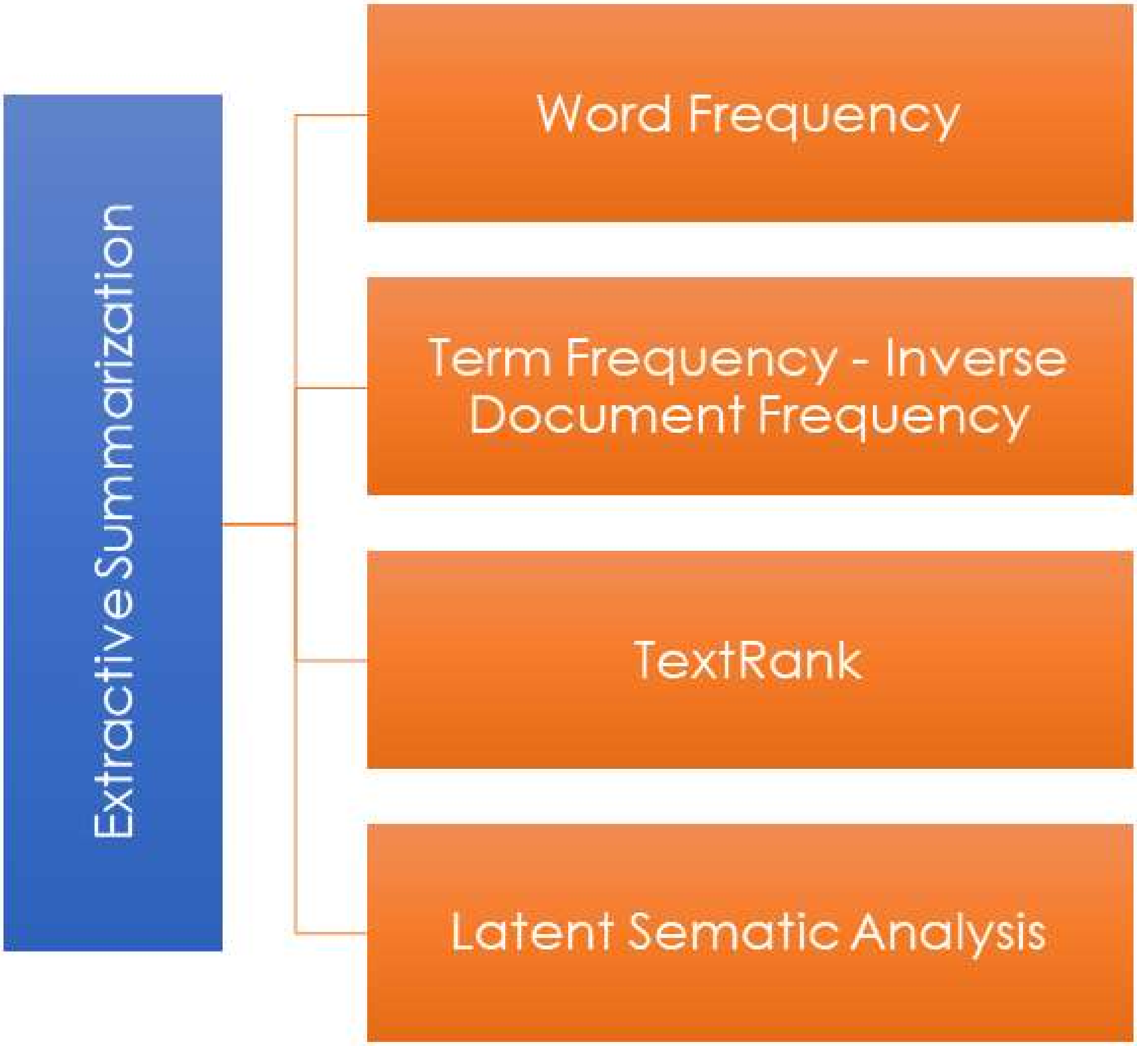


Figure 2.8: Types of Extractive Summarization

2.3.1. Word Frequency

The simplest way of using frequency is by counting each word occurrence in the document. However, the document length can greatly influence this approach. One way to adjust for the document length is by computing the word probability [2]. The probability f(w) of a word w is given by:

$$f(w) = n(w)/N \qquad \text{… Equation 2.1 [2]}$$

where
n(w) = the count of the word w in the document
N = total number of words in the document

This word probability is used for scoring sentences in the document, and then selecting a certain number of best sentences for the summary.

2.3.2. Term Frequency – Inverse Document Frequency

This approach solves the issue of word frequency approach of selecting a sentence with a non-essential word repeated many times. It is a mixture of two algorithms, term frequency, and inverse document frequency.

It is originally used for selecting a document from a group of documents. The idea of TF-IDF is to reduce the weightage of frequently occurring words by comparing its proportional frequency in the document collection. This property has made TF-IDF to be one of the universally used terminologies in extractive summarization [2].

Term frequency of a word is given as:

$$TF = n/\Sigma n \qquad \text{… Equation 2.2 [2]}$$

where n = the frequency count of the word in its document

Each word is then normalized by the total number of words in the document. The inverse document frequency is given by:

$$IDF = \log(D/d) \qquad \text{… Equation 2.3 [2]}$$

where

D = total number of documents

d = number of documents containing the word

The Term Frequency - Inverse Document Frequency for each word is given by:

$$TF\text{-}IDF = TF*IDF \qquad \text{… Equation 2.4 [2]}$$

The sentences are scored based on the TF-IDF values of the constituent words.

2.3.3. TextRank

TextRank algorithm for text summarization is based on the PageRank algorithm used by Google’s search engine. PageRank is one of the methods Google uses to determine a page’s relevance or importance. The PageRank algorithm calculates the probability that a person randomly clicking on links will arrive at any particular page.

We can take all web pages as a big directed graph (Figure 2.9). In this graph, a node is a webpage. If webpage A has the link to web page B, it can be represented as a directed edge from A to B.

After we construct the whole graph, we can assign weights for web pages, which are then used for finding the relevance for web pages.

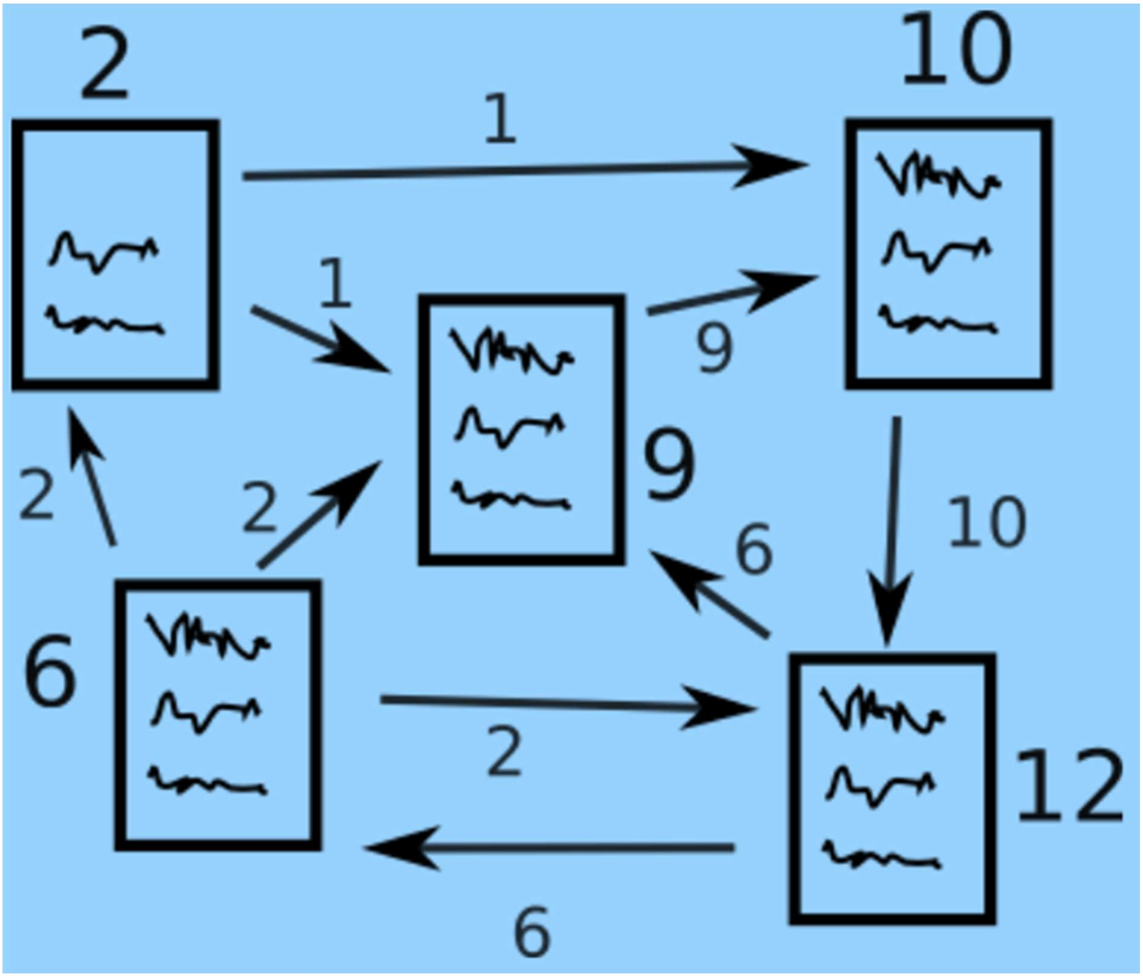


Figure 2.9: PageRank Nodes and Edges [22]

In the TextRank algorithm [22],

- We use sentences in place of web pages
- We find similarity between all sentences with each other to score them
- Similarity scores are stored in a matrix

2.3.4. Latent Semantic Analysis

Latent Semantic Analysis is an unsupervised technique that attempts to find the context around the words to capture the hidden concepts, also known as topics.

Topic Modeling

Topic Modeling is used to discover the main themes across documents automatically. It is an unsupervised learning algorithm that is used for finding the group of words representing a topic from the given document or multiple documents. For example, a group of words such as 'bank,' 'money,' 'cash,' 'cheque,' represent the topic 'finance.'

Topic modeling helps in exploring large amounts of text data, finding clusters of words, the similarity between documents, and discovering abstract topics.

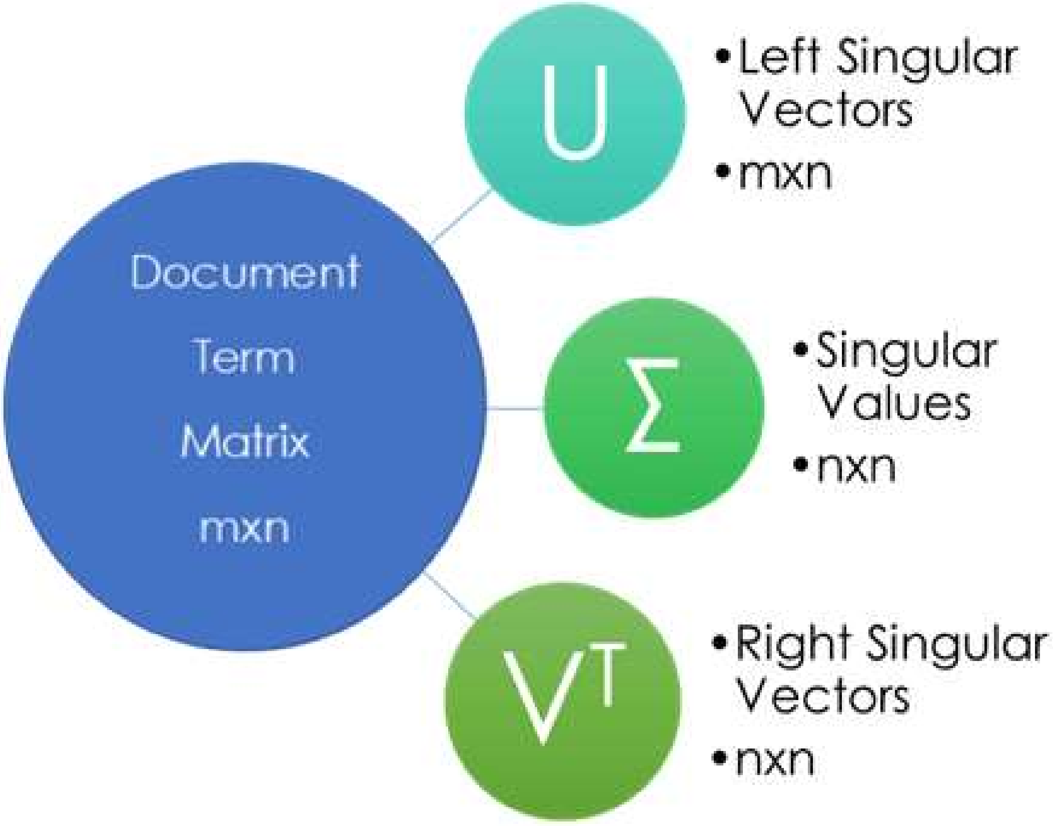


Figure 2.10: Singular Value Decomposition

Singular Value Decomposition

Singular Value Decomposition is used in the LSA algorithm. It breaks down a matrix into three other matrices, namely, matrix U, matrix S, and $V^{T,}$ i.e., transpose of matrix V (Figure 2.10). U is the document-term matrix, S is the topic importance matrix, and V is the term-topic matric. The values in matrix V^{T} denote the strength of each word in a sentence or document.
So, SVD gives us vectors for every document and term in our data. The cosine similarity method can be used on these documents to find similar words and similar documents.

Latent Semantic Analysis

The English language has words that can have multiple meanings based on the context of the sentence. It can be challenging for machines to understand such sentences. For example, “I ate an apple for lunch today” and “I bought the latest Apple mobile today” refer two entirely different “Apples,” but it is challenging for the computer to understand this. Latent Semantic Analysis is a type of topic modeling algorithm that aims to understand the context of the sentences.

2.4.Evaluation of Summarization

In [6], Ani Nenkova talks about the broad classification of evaluation approaches into Extrinsic and Intrinsic evaluation (Figure 2.11). In extrinsic evaluation, summarization results need to be assessed in a task-based setting, determining their usefulness at fulfilling a purpose whereas intrinsic evaluations work either by soliciting human judgments on the goodness and utility of a given summary or by a comparison of the summary with a human-authored gold-standard.

Extrinsic evaluations are time-consuming, expensive, and require a humongous amount of planning. [6] Hence, intrinsic evaluations are easier and more cost-effective when compared to extrinsic evaluation.

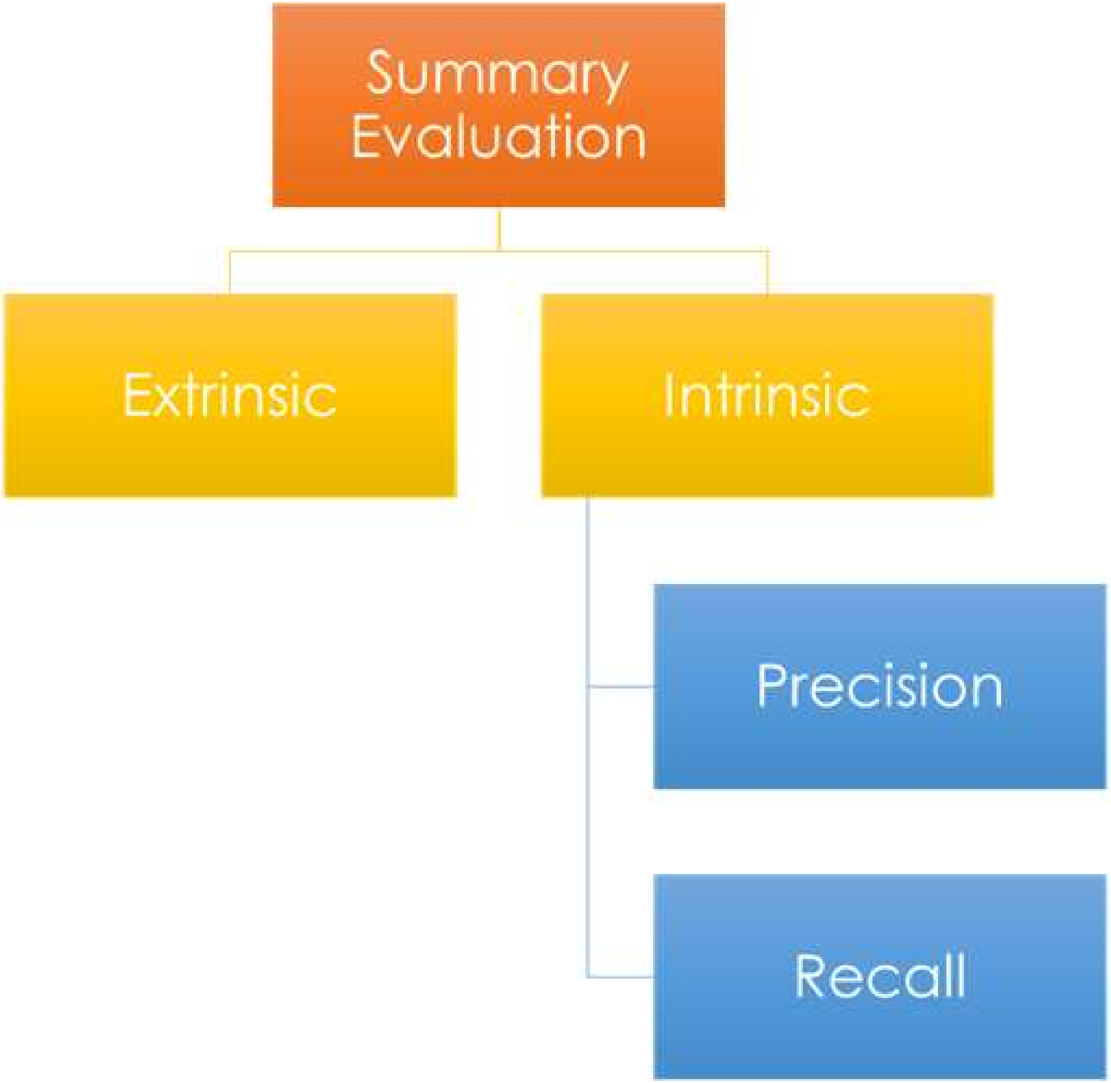


Figure 2.11: Types of Evaluations of Summaries

Intrinsic Evaluation uses information retrieval metrics such as ‘Precision’ and ‘Recall’ for its measurements. The recall is the fraction of sentences chosen by the person that were also correctly identified by the system. It is given by:

Recall = |system-human choice overlap | ÷ |sentences chosen by human|

… Equation 2.5 [6]

Whereas precision is the fraction of system sentences that were correct. We calculate it as:

$$\text{Precision} = |\text{system-human choice overlap}| \div |\text{sentences chosen by system}|$$

… Equation 2.6 [6]

For intrinsic evaluation purposes, generally, ‘n-grams’ are used. They are sentences of length n taken at a time sequentially for comparison. Unigrams compare one word at a time, bigrams compare two words at a time, and so on.

There are two widely used algorithms under intrinsic evaluation shown in Figure 2.12, namely, Bilingual Evaluation Understudy (BLEU) [7] and Recall-Oriented Understudy for Gisting Evaluation (ROUGE) [8].

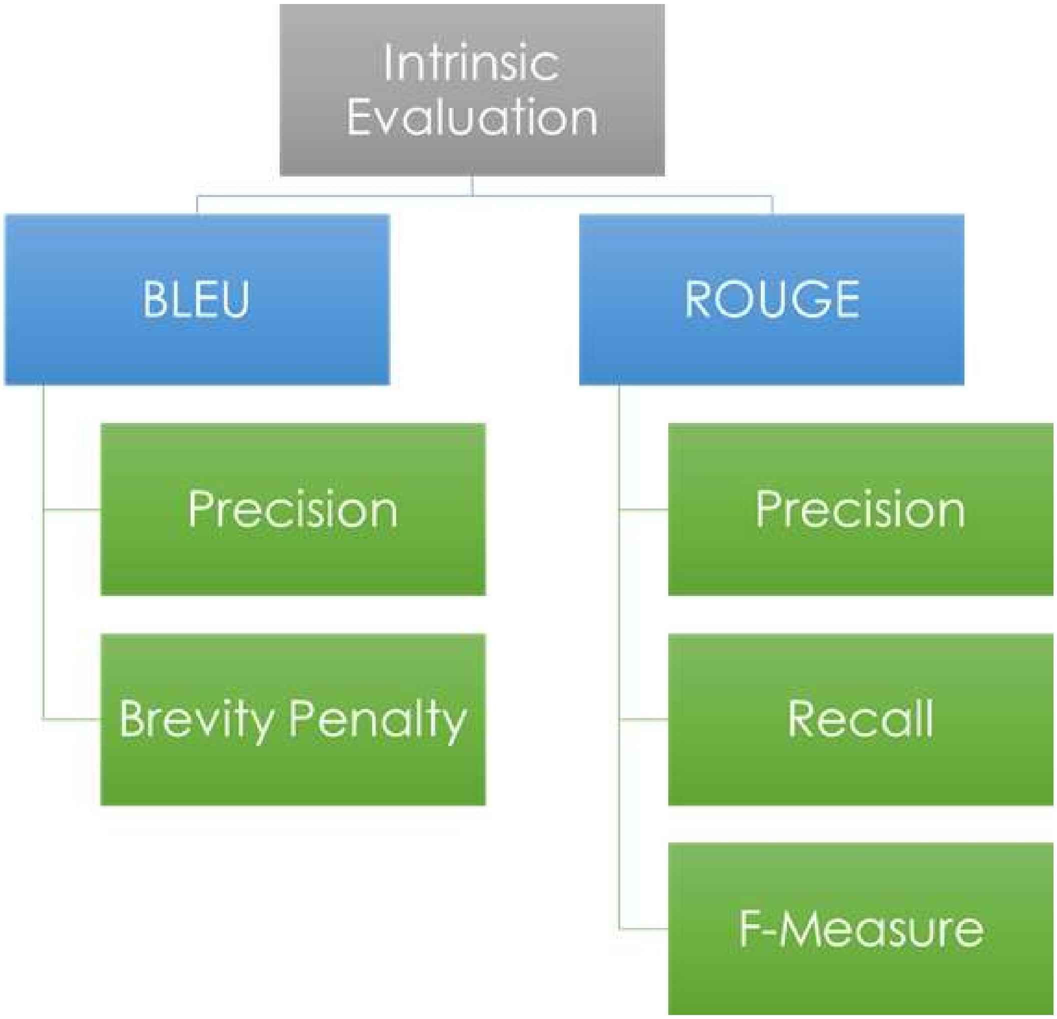


Figure 2.12 Algorithms for Intrinsic Evaluation

2.4.1. Bilingual Evaluation Understudy (BLEU)

A BLEU implementer compares n-grams of the candidate with the n-grams of the human selected reference translation and counts the number of matches. These matches are position-independent. The higher the number of matches, the better the candidate translation is. [7]
It uses the Precision (Equation 2.6) metric to evaluate summaries. To reduce scores of concise summaries, BLEU combines Precision scores with 'Brevity Penalty.' It is given by:

$$\begin{aligned} BP &= 1 \quad ; \text{ if } c > r \\ &= e\text{^}(1-r/c) \; ; \text{ if } c <= r \end{aligned} \qquad \text{… Equation 2.7 [7]}$$

where,
BP = Brevity Penalty
c = length of candidate summary
r = length of reference summary

This Brevity Penalty is multiplied by the geometric mean of the Precision scores of the test corpus. The BLEU score is then given by:

$$BLEU = BP*\exp(\textstyle\sum_{n=1}^{N} w_n \log(p_n)) \qquad \text{… Equation 2.8 [7]}$$

The issue in using BLEU for evaluation is that it does not map well with human judgments and is not suitable for the evaluation of text summarization [15].

2.4.2. Recall-Oriented Understudy for Gisting Evaluation (ROUGE)

ROUGE is a set of evaluation measures for text summaries. There exist four measures in the original ROUGE evaluation package (Figure 2.13), namely, ROUGE-N, ROUGE-L, ROUGE-W, and ROUGE-S [8].

ROUGE-N: These are n-gram co-occurrence statistics.

ROUGE-L: Longest matching sequence of words is measured using LCS (longest common subsequence).

ROUGE-W: Weighted longest common subsequence is measured. The length of consecutive matches encountered at any time is stored in a two-dimensional table.

ROUGE-S: Skip-gram occurrence statistics. Any pair of words in a sentence in order are compared, with arbitrary gaps in between.

ROUGE-2 measure, i.e., bigram ROUGE-N is known to best match with human judgments for evaluation of text summarization [6].

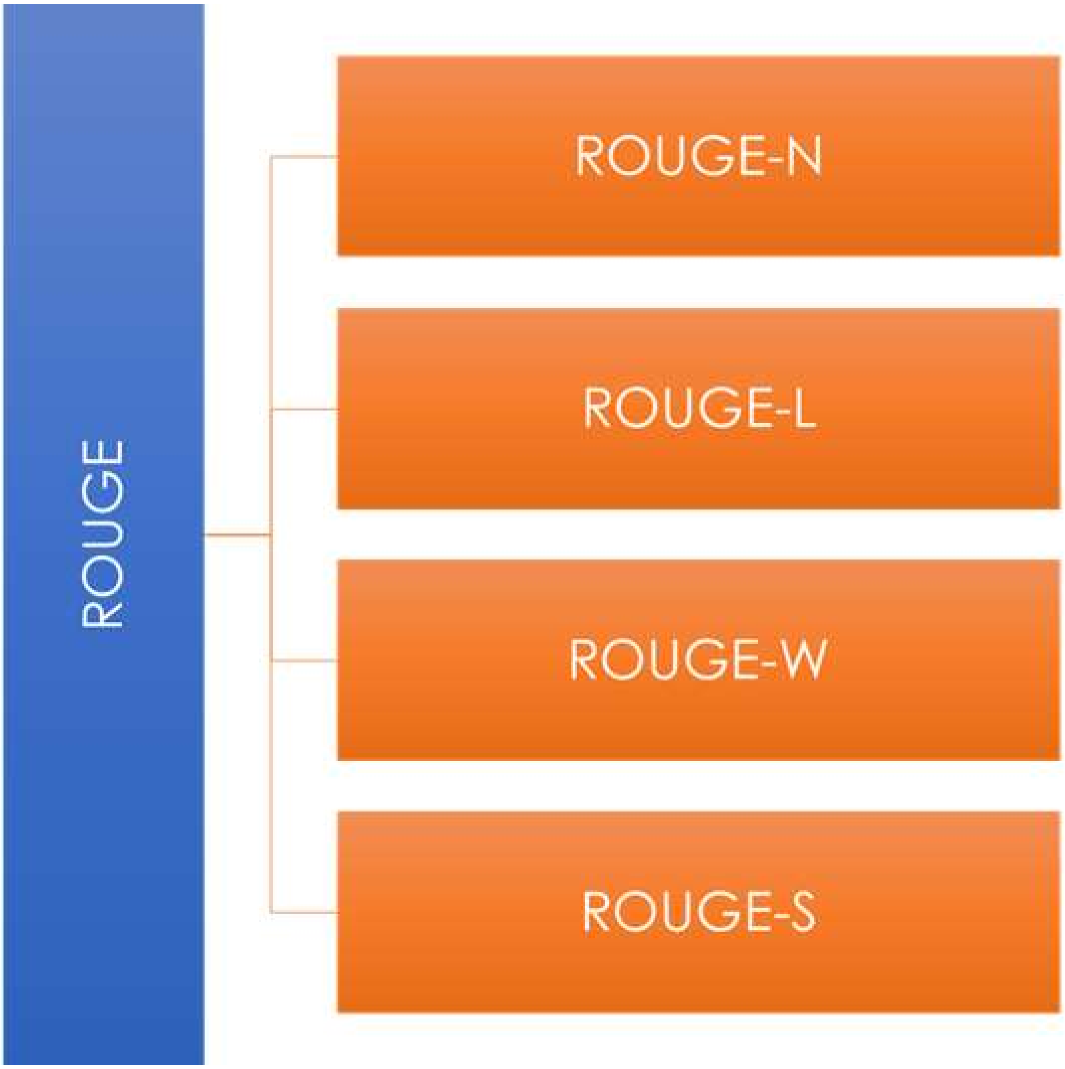


Figure 2.13: Types of ROUGE Measures

2.5.Web Scraping

Web Scraping (also referred to as Screen Scraping, Web Data Extraction, Web Mining, Web Harvesting.) is a collection of techniques used to extract data from websites on the internet. This data can be saved, processed, and used for various other purposes at a later time.

The data on most websites are designed to be viewed for a web browser in a form that is both aesthetically pleasing to see and such that the user can have a delightful user experience. They do not directly provide the functionality to save the data for personal use. The only direct option is to copy-paste the data from the browser onto a file manually, which is an extremely tedious job that can take hours or days to complete and may even be a task that needs to be done daily and thus never ends.

Web scraping is the process of automating this process so that instead of needing to copy-paste the data from the website manually, a program can automatically do this. A web scraping program will automatically load and extract data from a webpage, a website, or even multiple websites based on the way it is designed. These systems can be built in highly specified manners such that they work for only one site or can be configurable to work with a variety of websites. An advantage that web scraping has over conventional copy-pasting is that it can even obtain data that is not visible to the user, such as metadata in tags and attribute information. Most generic web scraping software is costly and requires lots of configuration time.

2.5.1. Techniques for Web Scraping

There are multiple methods when it comes to web scraping. We discuss a few of those methods below (Figure 2.14).

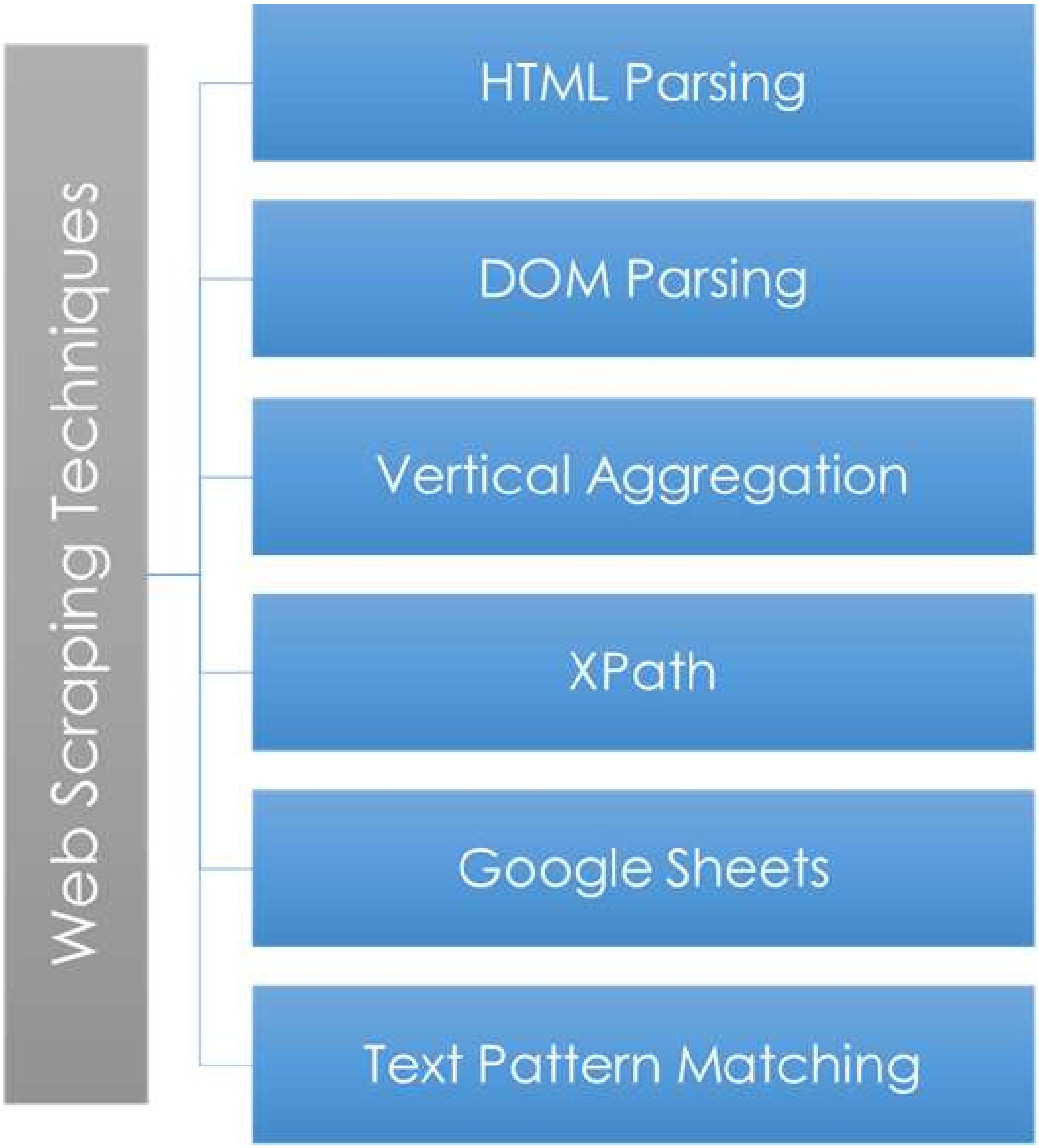


Figure 2.14: Techniques for Web Scraping

HTML Parsing

HTML parsing can be done using many different languages such as Java, Javascript, or Python, using different packages such as selenium and beautifulsoup and targets HTML pages. This fast and robust method is used for text, link extraction (such as nested links or email addresses), resource extraction, and so on. It can primarily obtain any data that is present on the web page.

DOM Parsing

The Document Object Model, or DOM, defines the style, structure, and content within XML or HTML files. DOM parsers are generally used by scrapers that want to get an in-depth view of the structure of a web page that is not visible to regular users on a web browser. Full-fledged web browsers like Google Chrome or Firefox can be driven to extract the entire web page or just parts of it, even if the content generated is dynamic.

Vertical Aggregation

Vertical aggregation refers to platforms that are created by companies with access to large-scale computing power to target specific verticals. Some companies even run these data harvesting platforms on the cloud on a remote server where the cloud service provider must bear the computation load. The creation and monitoring of bots for specific verticals is carried out by these platforms with virtually no human intervention.

XPath

XML Path Language, is a query language that is designed to work on XML documents. Since XML documents are structured, XPath can be used to navigate across the tree by selecting nodes based on a variety of parameters. XPath functionality is also provided by selenium and other tools so that the functionality of XPath can easily be extended to other languages.

Google Sheets

Google Sheets can be used as a scraping tool, and it is quite popular among scrapers. A scraper can use the IMPORTXML() function to scrape data from websites. This technique is useful when the scraper wants specific data or patterns to be extracted from a website. This tool can also be used to check if a given site is scrape proof or not.

Text Pattern Matching

This is a regular expression-matching technique that can be implemented using Perl or Python. On its own, this method cannot do much as it does not have means of retrieving data. When paired with techniques such as HTML or DOM parsing, it can help us process the data.

2.6.Twitter

Twitter is a social media platform in which one can tweet his/her ideas for everyone to listen to. One can type a message of up to 280 characters with attached multimedia if necessary.

In Twitter, a word, phrase, or topic that is mentioned at a higher rate than others is said to be a "trending topic" or merely a "trend." Trending topics become popular either through a concerted effort by users or because of an event that prompts people to talk about a specific topic. So, a trend is a topic that many people are tweeting or talking about. (Figure 2.15). We will use these trending topics or trends to understand what is happening at a particular location and use that for further

Twitter Developer provides tools and necessary resources required to develop a Bot on twitter that can put out tweets on its own, among other things, without explicit manual instructions from a human. We can use this bot for fetching trending topics from twitter and also post summarized articles for others to read.

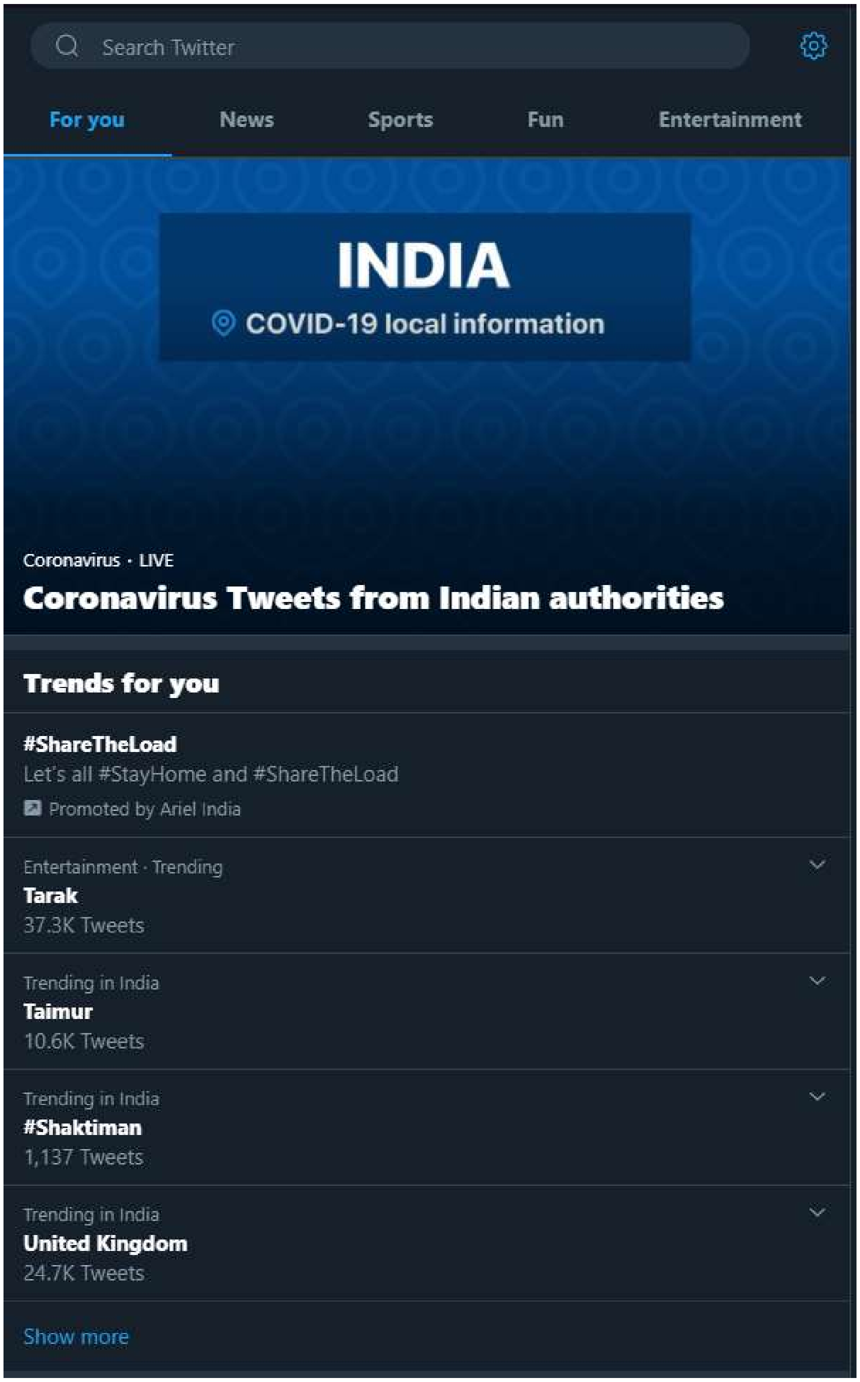
2.6.1. Fetching Trends from Twitter

Figure 2.15 shows the Twitter web page, where we can see the topics trending at that time. We will obtain the top ten current trends of a particular geographic location using the official Twitter API and use them to find relevant news articles. We will use WOEID or where on earth ID to locate or identify a specific geographic region.

We can do this using: -

- Official Twitter API
- Python Twitter packages (Twitter, Tweepy)

Twitter provides its official API in which it gives data like the trending topics, tweets of an account, the tweet volume, etc. These resources are refreshed every 5 minutes, and repeat calls within that duration will return only cached data.
The same resources will be used to post the summarized updates on twitter as tweets.



Coronavirus · LIVE

Coronavirus Tweets from Indian authorities

Trends for you

#ShareTheLoad
Let's all #StayHome and #ShareTheLoad
Promoted by Ariel India

Entertainment · Trending
Tarak
37.3K Tweets

Trending in India
Taimur
10.6K Tweets

Trending in India
#Shaktiman
1,137 Tweets

Trending in India
United Kingdom
24.7K Tweets

Show more

Figure 2.15: Trends as shown on Twitter on 27-03-2020

2.6.2. Analyzing Trends and Context

The data provided by the twitter API is in a raw JSON file that cannot be used in that form; thus, a series of operations and manipulations are needed to be done.

We will manipulate and format obtained Trends to convert them into a desirable format. This includes operations like removing hashtags, generalizing capitals, and changing the spacing between words.

We are using regex and python string manipulation techniques to filter data. This will help the web scraper module to quickly locate the main keywords of the trend and make its article search more efficient. We will also receive tweet volume, which can be used to rank topics and assign priority.

CHAPTER III

3. ANALYSIS AND DESIGN

3.1.Functional Requirements

a. The bot must receive only the latest trends for a given location
b. The bot must reference multiple new websites for acquiring the articles that will be summarized to ensure an unbiased view
c. Web scraping must take place in a reasonable amount of time and must not miss out on any relevant information
d. Summarization must take place in a fair amount of time with proper quality

3.2.Non-functional Requirements

a. A server must be continuously running the bot
b. A continuous internet connection is required
c. It must be able to handle many different kinds of errors without failure
d. Followers of the Twitter account must be informed in advance in case of any maintenance-caused downtime of the server
e. Users must not be able to influence the bot

3.3.Selection of Summarization and Evaluation Algorithms

After reviewing the literature, it was concluded that as abstractive methods require lots of computation and costs, and still sometimes fail to produce grammatically correct sentences, the project will be comparing and using **Extractive Summarization** algorithms.

When spending time reading various documents, we found that articles are generally built following the principle of having an introduction and conclusion. This introduction generally gives us an outline of the article, and the conclusion gives us a better idea of what the entire article is about. On the other hand, the main body generally provides detailed supplemental information that may not be adequate for a summary. Thus, we decided it would be best to formulate a method to give more weightage to the start and end of the article. Hence, we use **Improved Word Frequency** as our selected algorithm for text summarization.

For evaluation of the summaries, **Intrinsic Methods** were selected as Extrinsic Methods are known to be time-consuming, expensing, and require a considerable amount of planning.

Among intrinsic evaluation measures, **ROUGE-2** was and will be used for evaluating the summaries as it is close to human judgments. We use **F-Measure**, which uses Recall (Equation 2.5) and Precision (Equation 2.6) of ROUGE-2 to evaluate machine-generated summaries.

F-Measure = 2*Recall*Precision / (Recall + Precision) …Equation 3.1[6]

3.4.Components

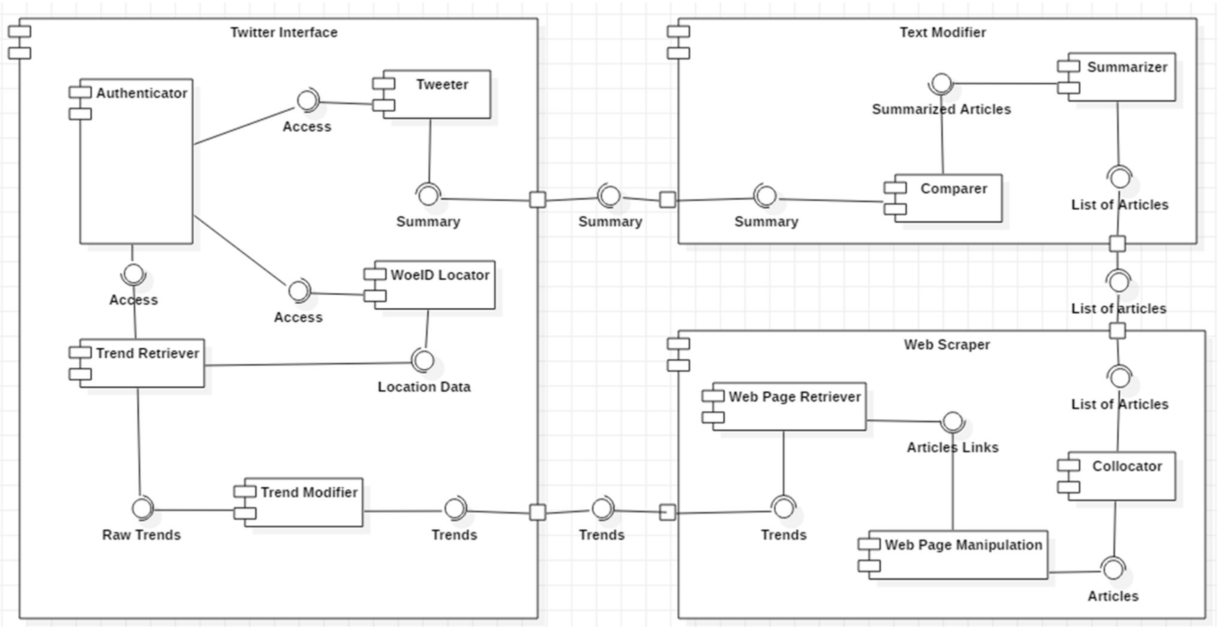


Figure 3.1 Component Diagram

The main program is divided into three main components (Figure 3.1), which are further divided into multiple smaller ones. These are all interconnected to work together seamlessly.

Twitter Interface Components

The Twitter interface is the primary medium of the bot operation and is responsible for all interactions with twitter. Mentioned below are the various components of the twitter interface component

- **Authentication**
  This is the component that verifies and validates the twitter web app authentication credentials like authentication_token,consumer_key, etc. This ensures that the correct bot account is connected to twitter.

- **WOEID Locator**

  WOEID or where on earth ID is the primary id used by Twitter to identify and categorize various geographic locations in the world. This is a required parameter of the Twitter API that we are using.

- **Trend Retriever**

  This component is responsible for fetching the top trends of a particular region. This will use the twitter API to request trends from twitter.

- **Trend Modifier**

  This is the final component of the twitter module used to manipulate the trends that we received from twitter. This includes removing the hashtags and cleaning the response from the API to keep only the required trends and removing waste data.

- **Tweeter**

  This is the main component that our bot will use for actually posting the summarized news update in the form of a tweet.

Web Scraper

The web scraper is the second principal component in the program. It is responsible for gathering relevant articles from multiple sources, manipulating and collocating them to be sent to the text modifier. It receives a list of trends from the twitter interface and sends out a list of articles to the text modifier. It contains the modules mentioned below.

- **Web Page Retriever**

  The component first accepts a list of trends on which articles need to be found. The web page retriever queries multiple news websites to obtain a list of links that pertain to a particular trend. This component is also responsible for filtering out articles that may be pictures, videos, or other kinds of articles from which text cannot be easily extracted.

Once an appropriate set of links are found, they are provided to the web page manipulator.

- **Web Page Manipulator**

  This component is provided with a list of links from the web page retriever. It must then visit each of these links, correctly identify the required content while also doing smaller checks to ensure the main content of the webpage is the article. It may even be necessary to do basic automation tasks to handle pop-ups. Once these articles are found, they are sent to the collocator.

- **Collocator**

  The collocator is a small component whose task is only to label the article list with the appropriate trends and then provide an interface to the text modifier from which it can be easily accessed

Text Modifier

This module is responsible for generating the <280-character Tweet from articles received from the Web Scraper module. It uses the Improved Word Frequency algorithm for summarizing the articles. It has subparts which have been mentioned below.

- **Summarizer**

  The list of articles received from the Web Scraper module will be summarized using the selected algorithm and will not exceed 280 characters, which is the character limit for posts on Twitter.

- **Comparer**

  The summaries of the list of articles will then be compared to check for any discrepancies between them. If there are any, a comparison of the two is used to generate the final Tweet. Otherwise, the summary produced by the Summarizer component will be sent to the Twitter Interface module.

3.5.General Flow of Execution

The bot has three main actors involved in its proper execution. If any one actor is down or performs poorly, the program will cease to function optimally. They are:

The program refers to the bot that we will implement. It will interact with both twitter and the various new websites. It will be written entirely in python, consist of three main modules and use various third-party modules

Twitter is the social media micro blogging website we will be connected to using the Twitter API.

News websites refer to a variety of news websites such as BBC, Al-Jazeera, Times of India, and a few others which will be scraped to get more data on the trends.

The flow of execution will be as follows (Figure 3.2):

1. The program connects to Twitter via the twitter API, authenticates itself, and then requests the trends of a particular location.
2. Twitter then verifies the request and sends the requested data in the form of a JSON file.
3. The trends need to then be appropriately extracted from the JSON response.
4. These trends are then each tested for viability, i.e., whether they are in English, words can be adequately extracted, have emoji, etc. All viable trends are placed in a list, and the rest are deleted as they cannot be used. The viable trends then undergo minor processing in order to turn them into phrases that can be queried online.
5. The list of trends is then iterated through, and each trend is searched for on multiple news websites. These websites give a list of links to various articles about the trend.
6. These articles are then accessed using its respective link, and various web scraping tools are used to retrieve the required data.
7. These articles are sent to a summarizer which uses specific extractive summarization techniques to obtain summaries for each.

8. The various summaries are then compared using simple techniques to check for usefulness. The most useful one is uploaded.
9. If both have equal usefulness, some other metadata is considered.

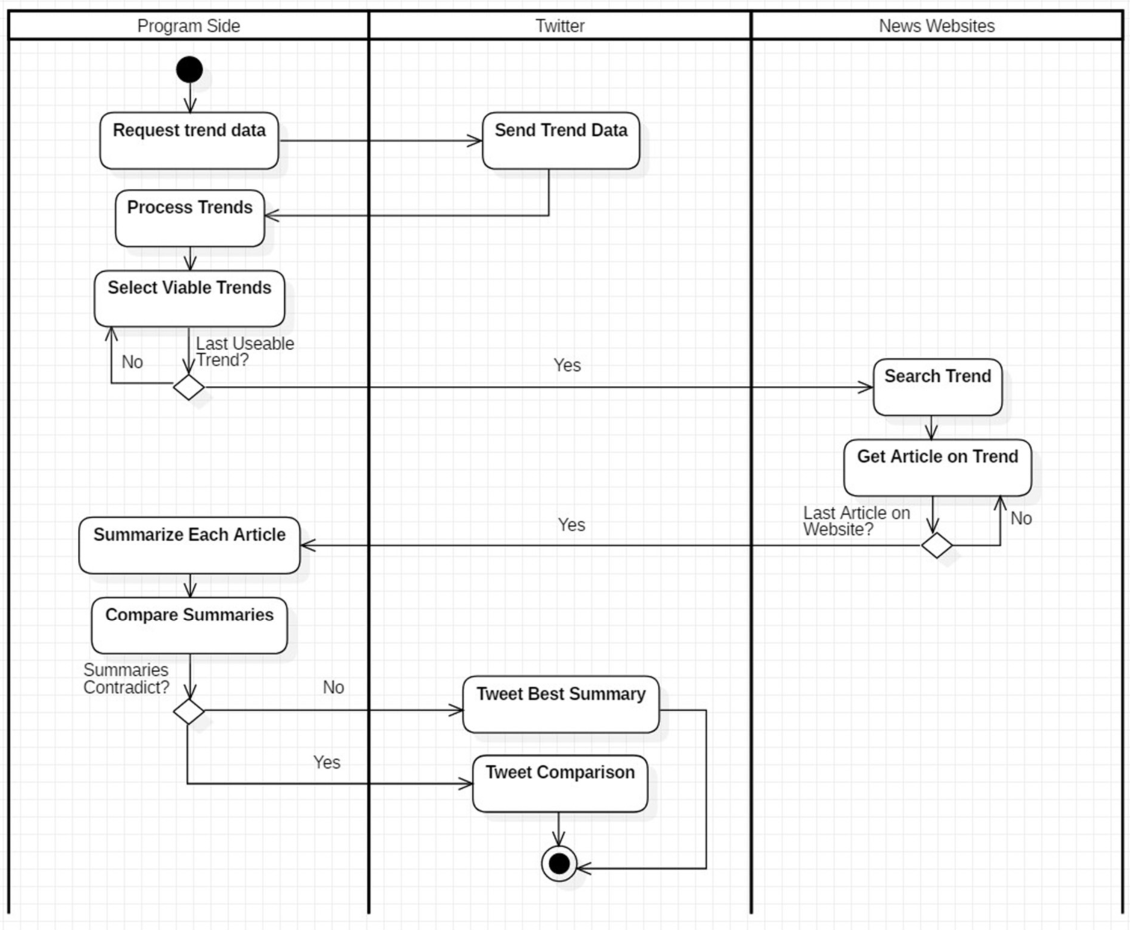


Figure 3.2 Swimlane Diagram

CHAPTER IV

4. IMPLEMENTATION DETAILS

4.1.Software Requirements

a. Python Module installed (v3.6 or higher)
b. Twitter Developers account
c. Text editor (Sublime Text/Atom/IDLE)
d. Google Chrome v80.0.xx
e. Windows OS (7 or higher)
f. Selenium driver

4.2.Hardware Requirements

a. Full computer system with all standard peripherals
b. RAM 4 GB or higher
c. Storage 16 GB or higher

4.3.Libraries and Functions

a. NLTK: Natural Language Toolkit library available in Python. We have used the "sent_tokenise" to split documents into sentences. To remove stopwords from the sentences, we used the "stopwords" module. For ridding the sentence of punctuation, the "RegexpTokenizer" module was used.
b. Gensim: This library in Python provides the TextRank algorithm, which we have used for our project. It gives us the option to select the maximum word count for the summary.

c. Scikit-Learn: The “TfidfVectorizer” provided by the package was used to create the document-term matrix for Latent Semantic Analysis. It also offers a module for the decomposition of the document-term matrix into singular value decomposition, which has also been used to rank the sentences in Latent Semantic Analysis.

d. Timeit: This library provides functions to record time. The “default_timer” has been used to find out the computation time taken by each algorithm for summarizing each article.

e. Twitter: We have used the Twitter package to access twitter resources The “twitter_api.trends.place(_id=WOEID)” function was used to fetch trends determined by location.WOEID is used to pinpoint a location. Lastly, we have used "twitter_api.statuses.update(status=summary)" to post our customized strings to twitter in the form of tweets.

f. Requests: This module is used to retrieve web pages from the internet. The URL is provided to the relevant function, and it returns the web page in the form of a string. However, this package only returns the initial static view of the webpage and cannot work with dynamic web pages

g. BeautifulSoup: This module is designed to allow easy parsing of HTML pages based on its structure. It does not have the functionality to retrieve documents from the web on its own. It can parse documents based on tags, classes, IDs, names, or attributes. This enables it to get extremely powerful and allows the user to find the exact piece of information we require as long as we know specific information about the tag it is in. In this project, it is used to find the links to every article that we require, and then later, it is used to extract the exact article and the relevant metadata along with it.

h. Selenium: A powerful library for web automation, selenium can be used for web scraping and web automation. Certain dynamic websites require a small degree of automation in order to ensure we get the data that we want. We use this module in order to wait for a web page to load before we scrape it for the data we need. It works similar to beautifulsoup such that it allows easy parsing of the HTML page. It includes support for Xpath, a language specifically designed to handle web pages

4.4.Steps Followed for Individual Modules

The various steps involved in the multiple algorithms for summarization; web scraping; fetching, and posting the tweets have been discussed below.

4.4.1. Summarization

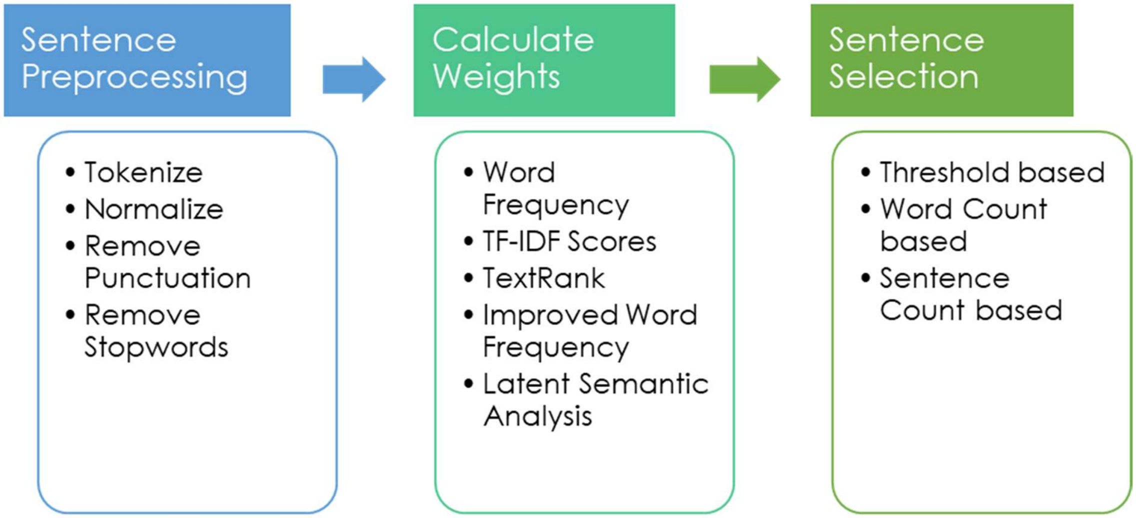


Figure 4.1: Extractive Text Summarization

The general algorithm of an extractive summarization method is shown in Figure 4.1. The sentences are separated and stored in a list. Before applying the summarization algorithms, the sentences had to be pre-processed, then an algorithm is used to calculated weights for all sentences, which are then sorted according to those weights. We discuss the multiple algorithms we have implemented and evaluated for our project below. These include Word Frequency, Term Frequency-Inverse Document Frequency, TextRank, Improved Word Frequency, and Latent Semantic Analysis.

Word Frequency

The simplest method in automatic extractive summarization is based on counting the number of times a word appears in a document.

The articles sent from the Web Scraping module were received in a list. For each article, all sentences were split and stored in another list. The sentences were then normalized by making all into the lower case to avoid the same word mentioned in both capital and small case to be taken as two different words for counting. They were then tokenized by removing punctuation so that only the words remained in every sentence. In NLP, stop words are removed from sentences before using they are used in any algorithm. NLTK package in Python was used for the removal of stop words from text sentences.

Then every distinct word was assigned with its word count. To calculate weights of words, the count of each word was divided by the maximum count. For each sentence, the score was calculated by taking the sum of counts of its constituent words. These scores were then used to sort the list of sentences according to the priority of importance in descending order. To keep the character limit in check, a sentence was appended to the summary only if adding it would not make the character count more than 280.

Term Frequency – Inverse Document Frequency

Used generally for document selection, this algorithm can also be used for text summarization by considering each sentence to be an individual document. It works by combining two algorithms – the term frequency and the inverse document frequency algorithms.

We first used the NLTK package to tokenize the articles into a list of sentences. We used the same module to eliminate stop words from the sentences. They were normalized by converting each word to lowercase. For ease in processing, we stemmed the words using the PorterStemmer module.

A frequency matrix was created, which stored the stemmed word along with the count of the original word. This was used to calculate term frequency (Equation 2.2). To calculate inverse document frequency (Equation 2.3.), we needed to know how unique a word is. To do this, we found out how many sentences contain a particular word.

After getting the TF and IDF values for each word, we found the TF-IDF values by multiplying the TF and IDF values together. The sentence weights were then calculated by adding values of TF-IDF of their constituent words. These values were used to arrange the sentences in ascending order. In TF-IDF, usually, a threshold value is selected as the average scores of all sentences, and this value, along with an arbitrary factor, is used to determine if the sentence will be selected for the summary or not. We experimented with different values of threshold and found out that there exists an upper limit, which is about 1.5x – 2x of the threshold value, after which we do not get any sentence in the summary.

TextRank

TextRank is an unsupervised algorithm based on PageRank, which is used for finding web pages on a search. TextRank can be used in keyword extraction and text summarization.

Working of TextRank:

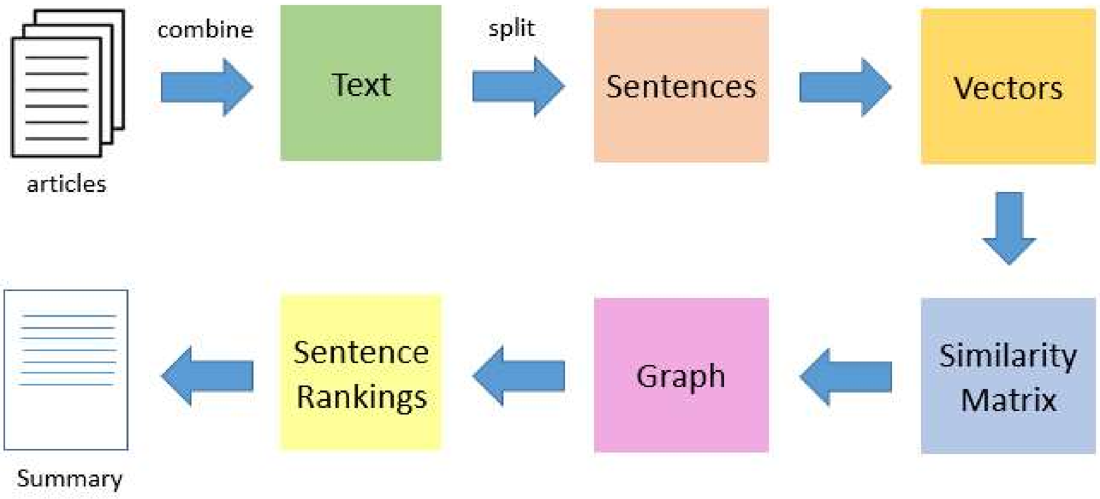


Figure 4.2: TextRank Algorithm [22]

1. Concatenate all the text contained in the documents
2. Tokenize, i.e., split the text into individual sentences
3. Find vector representation for each sentence
4. Calculate similarities between sentence and store in a matrix
5. Make a graph with sentences as vertices and similarities as weights to the edges
6. Select a few top-ranking sentences in the summary

Improved Word Frequency

An improvement can be made on the existing word frequency method based on the fact that the introduction and conclusion contain more relevant information than the body in terms of generating a summary.

We follow all the same steps, followed by traditional word frequency, until generating the weights of each sentence. We then apply Equation 5.1 on each line of the article.

New_weight = weight*(1+0.2*abs((num_sen/2)-sen_pos)/(num_sen/2)   … Equation 5.1

This equation is designed to give a small boost based on a specific factor to sentences based on its distance from the middle sentence. The middle sentence weight remains completely unchanged while the values of the first and last sentence change the greatest.

Latent Semantic Analysis

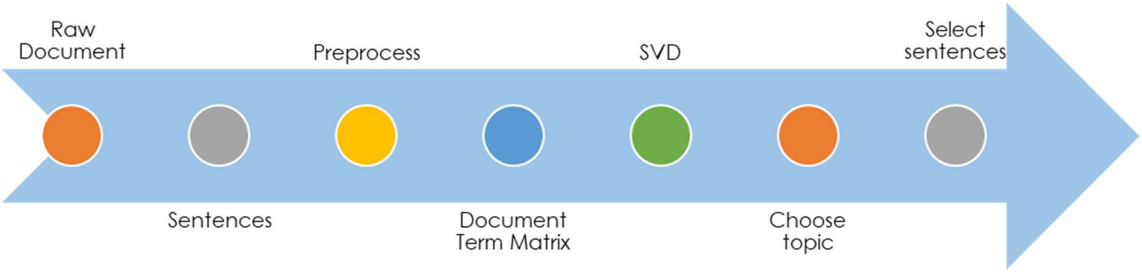


Figure 4.3: Latent Semantic Analysis Algorithm

As shown in Figure 4.3, we can see the steps involved in text summarization using Latent Semantic Analysis.

Unlike typical LSA, where multiple documents are taken as input, and a given number of topics are extracted, here we take a single document, and consider each sentence within it as a document for topic modeling.

The document is tokenized, i.e., sentences are extracted for further processing. Just like any other data, text data also has noise and undesired things. Hence, sentences are normalized, stop-words are removed, punctuations are removed, and then the sentences are ready for further processing.

The text is converted into its numerical representation in the form of a document term matrix, which can be used for dealing with the text data mathematically. For LSA, we decompose the mentioned matrix into its components using singular value decomposition. From this, we can find out how much a sentence is relevant to a topic. We select the most critical topic in our document and take the top sentences most related to it. These form our final summary.

4.4.2. Web Scraping

BeautifulSoup has been used extensively either by itself or along with Selenium to accomplish the web scraping of each of the different websites. The methods implemented for each website are below.

BBC

We first build up a URL of which the HTML file must be retrieved. This URL is of the format "https://www.bbc.co.uk/search?q=*search*&filter=news" where the *search* is the phrase previously generated from the trend, that we wish to retrieve articles on. The HTML file is then extracted using the requests package and then converted to a 'soup,' which can be easily parsed by the BeautifulSoup package. We then use the soup to get the links to the articles displayed on the web page one by one by using HTML parsing and then place them into a list.

This list is iterated through, and for each link, we once again use the requests package to retrieve the webpage for each article, convert it to a soup and find the tag with the article body class under which the required data is stored in p tags. As there are many p tags, the data from all of them must be merged to get the necessary data. This is done for each article link in the list, and the result of each is placed into a list that is sent to the text modifier.

We also run a few minor checks on the web page by searching for the existence of certain tags such as video and audio to ensure that the page consists of a proper article.

Al-Jazeera

The website used by Al-Jazeera is highly dynamic, and thus, just using the BeautifulSoup package proved insufficient, and minor automation techniques using the Selenium package are required.

We begin but starting up the browser using the browser and version-specific Selenium driver. The browser is run normally in testing, but in final implementation, it will be run headless.

A URL of which the HTML file must be retrieved is first built up. This URL is of the format “https://www.aljazeera.com/Search/?q=*search*” where the *search* is the phrase previously generated from the trend, that we wish to retrieve articles on. We then navigate to this URL using Selenium and wait a few seconds as the dynamic web page takes time to load the required content. The web page is then parsed through using Selenium itself to get the required article links based on specific criteria. These links are placed on a list.

This list is iterated through, and for each link, we once again use the requests package to retrieve the webpage for each article, convert it to a soup and find the tag with the article body class under which the required data is stored in p tags. As there are many p tags, the data from all of them must be merged to get the required data. This is done for each article link in the list, and the result of each is placed into a list that is sent to the text modifier.

We also run a few minor checks on the web page by searching for the existence of certain tags such as video and audio to ensure that the page consists of a proper article.

The Times of India

We first build up a URL of which the HTML file must be retrieved. This URL is of the format "https://timesofindia.indiatimes.com/topic/*search*/news" where the *search* is the phrase previously generated from the trend that we wish to retrieve articles on. The HTML file is then retrieved using the requests package and then converted to a ‘soup,’ which can be easily parsed by the BeautifulSoup package. We then use the soup to get the links to the articles displayed on the web page one by one by using HTML parsing and then place them into a list.

This list is iterated through, and for each link, we once again use the requests package to retrieve the webpage for each article, convert it to a soup. The Times of India news website can, however, redirect to one of the various other TOI websites. A separate scraping algorithm had to be written for each website as some contained the required text in span tags, some in div tags, and some in p tags. Along with the article data, we also retrieve metadata, such as upload time.

We also run a few minor checks on the web page by searching for the existence of certain tags such as video and audio to ensure that the page consists of a proper article.

4.4.3. Twitter

The process of retrieving trends from Twitter and posting them after summarization is complete has been given below.

Fetching and Manipulating Trends

The process starts when our bot initiates the Twitter API call on its own. The bot will send its credentials that are customer and authentication token and the other required parameters to twitter via python.

Twitter API then verifies the input, and on successful verification of the credentials, it will search its database and fetch the results according to the parameters given as input.

Twitter replies to the API call in the form of a JSON file. This JSON file will contain all the details about the top Trends of a particular location. Using Python tools, we will thoroughly filter and edit this JSON file to remove all unnecessary information; hashtags will be removed. The output thus will be a list of clean string of trends. These trends will further undergo more manipulation to ensure that we understand their context more efficiently.

Tweeting Summary

In this process, the bot uses the twitter resources to post a string of length less than 280 words to twitter as a tweet. This tweet can now be seen by everyone who follows the bot's twitter profile.

This is achieved by using the same Twitter API we used earlier. Here we take the summarized string from the summarizer module and use the POST method of the API to send this string to twitter servers. Twitter then matches the user credentials with the bot twitter account, and thus on a successful match, the string is posted via that particular twitter account.

CHAPTER V

RESULTS AND DISCUSSION

Results of Comparison of Summarization Algorithms

The Twitter, Web Scraping and Summary generating modules were put together, and the accuracy of the generated summary was evaluated by using different summarization algorithms. For this purpose, we have and will be using ROUGE-2 F-Measure. F-Measure scores range from 0 to 1, with 1 being the ideal score.

As there existed no dataset that fit our requirements. We decided to create our own. The dataset consists of **119** articles and human-generated summaries. The articles were sourced from the internet from various web sites. The human-generated summaries were carefully created by us upon reading the article multiple times. These summaries were also designed to be lesser than 240 characters as twitter can only accept summaries of this length and the methods; we will test have also been modified to satisfy this requirement.

As we can see in Figure 5.1, the improved word frequency method gives us the best average F-Measure with text rank close behind. Advanced methods like TF-IDF and LSA do not give as good mean results.

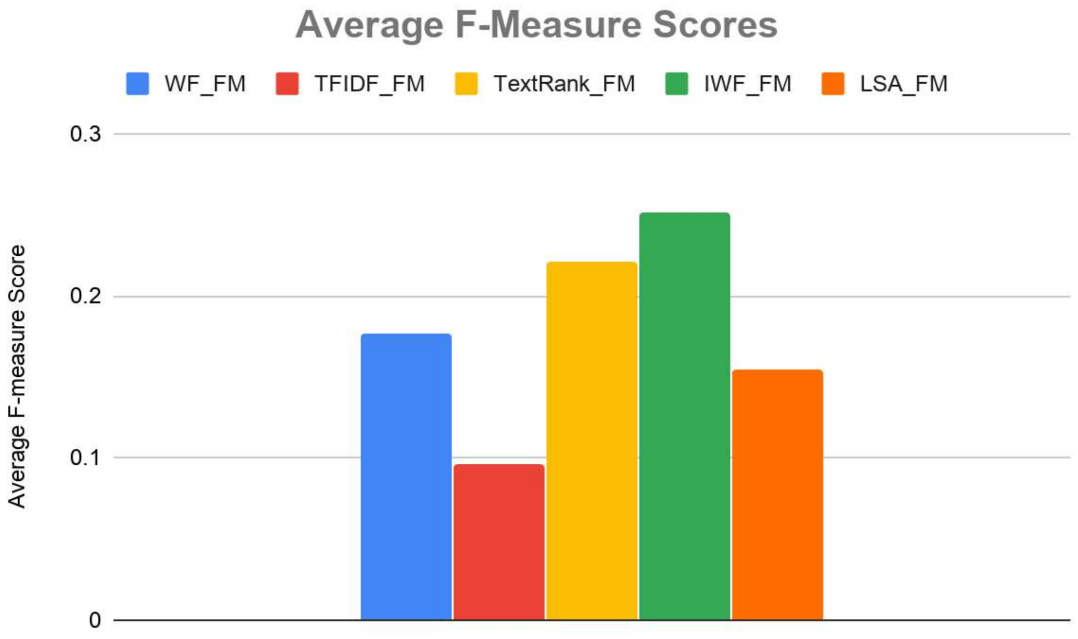


Figure 5.1: Average F-Measure Scores

The unexpected results above can be explained by a large number of zero values for f-measure that were found in those complicated methods.

Algorithm	Word Frequency	Term Frequency - Inverse Document Frequency	TextRank	Improved Word Frequency	Latent Semantic Analysis
Number of times 0 was obtained as F-measure	30	50	2	30	42

Table 5.1: Zero Value Count in each Method

Table 5.1 clearly shows us that both the highly complex methods, i.e., TF-IDF and LSA, have a more significant number of zero values. This more significant number of these could be due to the algorithm choosing sentences utterly different from the user-chosen ones. This can be because of human error or because there were multiple similar sentences that explained the article in reasonable detail.

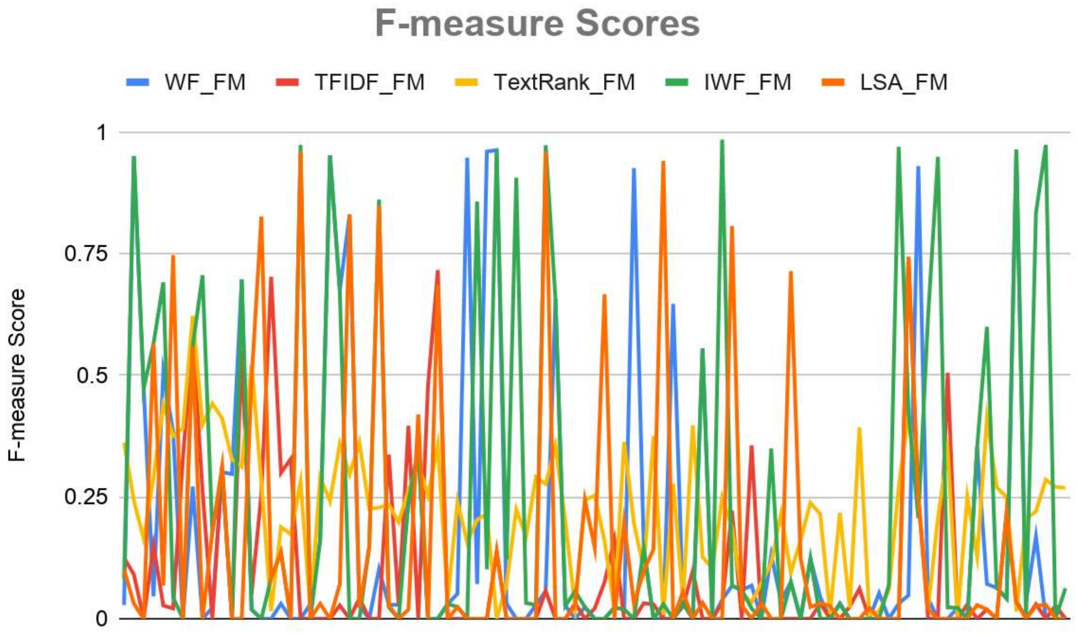


Figure 5.2: F-Measure Scores Comparison

Figure 5.2 shows us the values of F-measure for each method on all 114 articles. As we can see, generally, the IWF can give near-perfect results.   The few zero values are probably satisfactory results similar to the methods like LSA and TF-IDF.

We can also see that the TextRank algorithm gives us nearly no zero values but has a consistently low f-measure that implies it generally yields low-quality summaries.

Algorithm	Word Frequency	Term Frequency - Inverse Document Frequency	TextRank	Improved Word Frequency	Latent Semantic Analysis
Frequency of scores greater than 0.9	8	0	0	**11**	3
Frequency of scores greater than 0.8	9	0	0	**14**	7

Figure 5.2: High F-Measure Probability per Method

The claim of IWF having the best results can be verified Table 5.2 that shows that it most frequently has F-measure greater than 0.8 and 0.9.

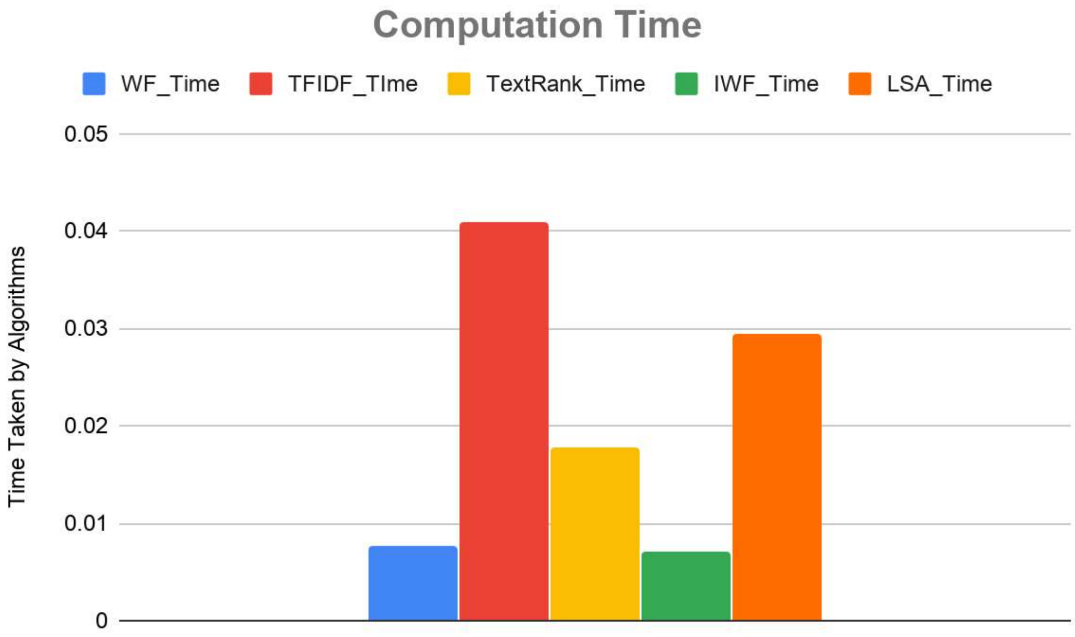


Figure 5.3: Average Computation Time

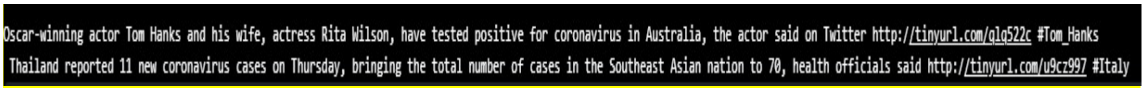
As seen in Figure 5.3, IWF and WF take a near equal amount of time to process an article on average. This amount of time is far less than the more sophisticated methods and is more than four times as fast as TF-IDF and twice as fast as LSA

The fact the IWF gave us the best results on average in terms of quality of summary while also taking the least amount of time to process is what convinced us that it would be the ideal method for our application.

Results of the Bot

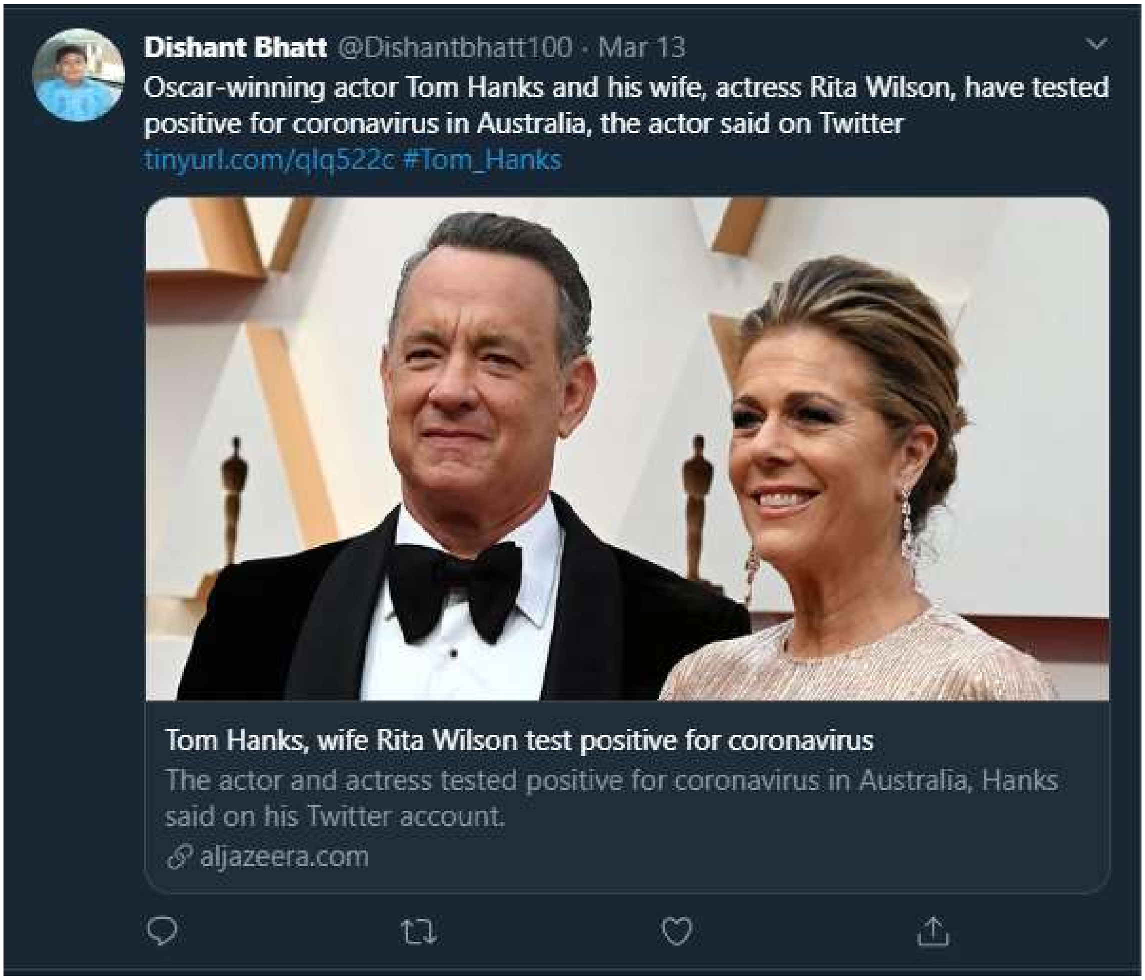
The bot was run on March 13th and March 28th.

The bot provides a basic console output (Figure 5.4) that lets the administrators view what tweets are being sent. This allows a little more control over what is being spread and allows verification of the relevancy and understandability of the tweet being sent out.



```
Oscar-winning actor Tom Hanks and his wife, actress Rita Wilson, have tested positive for coronavirus in Australia, the actor said on Twitter http://tinyurl.com/q1q522c #Tom_Hanks
 Thailand reported 11 new coronavirus cases on Thursday, bringing the total number of cases in the Southeast Asian nation to 70, health officials said http://tinyurl.com/u9cz997 #Italy
```

Figure 5.4: Bot Results on System Console on 13-3-2020


Dishant Bhatt @Dishantbhatt100 · Mar 13

Oscar-winning actor Tom Hanks and his wife, actress Rita Wilson, have tested positive for coronavirus in Australia, the actor said on Twitter tinyurl.com/qlq522c #Tom_Hanks

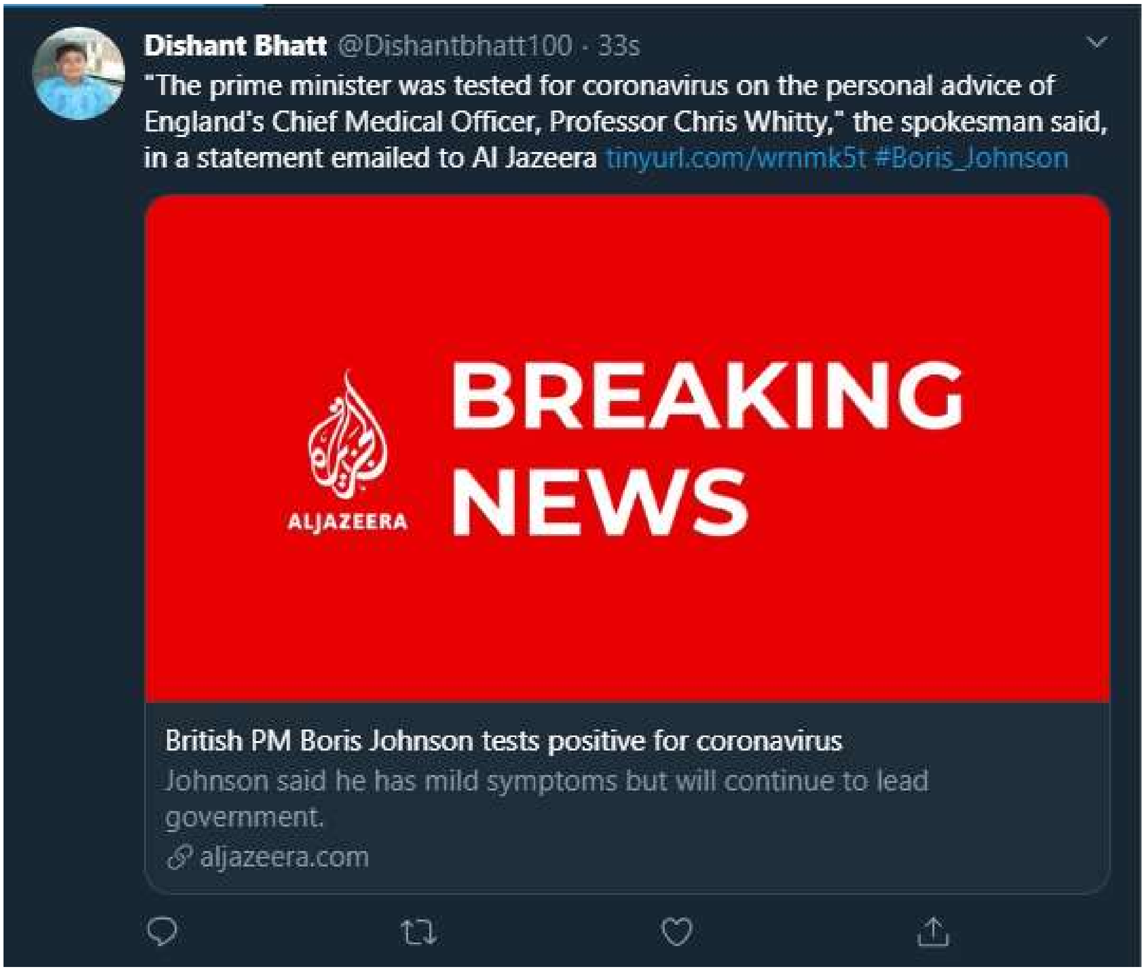
Tom Hanks, wife Rita Wilson test positive for coronavirus

The actor and actress tested positive for coronavirus in Australia, Hanks said on his Twitter account.

aljazeera.com

Figure 5.5: Bot Results on Twitter on 13-03-2020

The trend of March 13th was related to Tom Hanks and his wife contracting coronavirus in Australia. This trend was a huge new story that also managed to sensationalize the media, making it both a direct trend and an accessible article to find. (Figure 5.5)



Dishant Bhatt @Dishantbhatt100 · 33s

"The prime minister was tested for coronavirus on the personal advice of England's Chief Medical Officer, Professor Chris Whitty," the spokesman said, in a statement emailed to Al Jazeera tinyurl.com/wrnmk5t #Boris_Johnson

British PM Boris Johnson tests positive for coronavirus

Johnson said he has mild symptoms but will continue to lead government.

aljazeera.com

Figure 5.6: Bot Results on Twitter on 27-03-2020

The trend of March 27th was related to Boris Johnson, the Prime Minister of the United Kingdom catching COVID-19, which is caused by the coronavirus. (Figure 5.6)

CHAPTER VI

CONCLUSION AND FUTURE SCOPE

We evaluated multiple text summarization algorithms using the ROUGE-2 evaluation metric and based on the analysis of those results and data, and we realized that our modification to Word Frequency to have a weighted gradient gave the best results in impressive time. Hence, we conclude that Improved Word Frequency is the best method among all the other extractive methods we evaluated.

A Twitter developer's account was created to access Twitter's resources and data. The account will be used to regularly fetch the trends, which will be queried on websites such as Al-Jazeera and BBC, and the most relevant article will be summarized by our algorithm and then posted on Twitter along with the link to the article and the trend.

In the future, we plan to move on to fetching trends and summarising articles in multiple languages. We also aim to make use of abstractive summarization methods for our summaries. Another feature that can be added to our project is to check the coherence of articles with each other to reduce the spread of fake news

REFERENCES

[1] Gaikwad, D. K., & Mahender, C. N. (2016). A review paper on text summarization. *International Journal of Advanced Research in Computer and Communication Engineering*, *5*(3), 154-160.

[2] Basiron, H., Jaya Kumar, Y., Ong, S. G., Ngo, H. C., & C Suppiah, P. (2016). A review on automatic text summarization approaches. *Journal of Computer Science*, *12*, 178-190.

[3] Sinha, A., Yadav, A., & Gahlot, A. (2018). Extractive text summarization using neural networks. *arXiv preprint arXiv:1802.10137*.

[4] Mihalcea, R., & Tarau, P. (2004, July). Textrank: Bringing order into text. In *Proceedings of the 2004 conference on empirical methods in natural language processing* (pp. 404-411).

[5] Nenkova, A. (2006). Summarization evaluation for text and speech: issues and approaches. In *Ninth International Conference on Spoken Language Processing*.

[6] Papineni, K., Roukos, S., Ward, T., & Zhu, W. J. (2002, July). BLEU: a method for automatic evaluation of machine translation. In *Proceedings of the 40th annual meeting on association for computational linguistics* (pp. 311-318). Association for Computational Linguistics.

[7] Lin, C. Y. (2004, June). Looking for a few good metrics: Automatic summarization evaluation-how many samples are enough?. In *NTCIR*.

[8] Kaikhah, K. (2004). Text summarization using neural networks.

[9] Sinha, A., Yadav, A., & Gahlot, A. (2018). Extractive text summarization using neural networks. *arXiv preprint arXiv:1802.10137*.

[10] Wojcik, S., Messing, S., Smith, A., Rainie, L., & Hitlin, P. (2018). *Bots in the Twittersphere: An Estimated Two-thirds of Tweeted Links to Popular Websites are Posted by Automated Accounts-Not Human Beings*. Pew Research Center.

[11] Haustein, S., Bowman, T. D., Holmberg, K., Tsou, A., Sugimoto, C. R., & Larivière, V. (2016). Tweets as impact indicators: Examining the implications of automated “bot” accounts on T witter. *Journal of the Association for Information Science and Technology*, *67*(1), 232-238.

[12] Fernquist, J., Kaati, L., & Schroeder, R. (2018, November). Political bots and the swedish general election. In *2018 IEEE International Conference on Intelligence and Security Informatics (ISI)* (pp. 124-129). IEEE.

[13] Sulem, E., Abend, O., & Rappoport, A. (2018). Bleu is not suitable for the evaluation of text simplification. *arXiv preprint arXiv:1810.05995*.

[14] Michel Kana (2019). "Representing Text in Natural Language Processing", https://towardsdatascience.com/representing-text-in-natural-language-processing-1eead30e57d8
Last accessed: 18/11/19

[15] Prateek Joshi (2018). "A Stepwise Introduction to Topic Modeling using Latent Semantic Analysis", https://www.analyticsvidhya.com/blog/2018/10/stepwise-guide-topic-modeling-latent-semantic-analysis/
Last accessed: 20/10/19

[16] Pulkit Sharma (2018). "The Ultimate Guide to Dimensionality Reduction Techniques", https://www.analyticsvidhya.com/blog/2018/08/dimensionality-reduction-techniques-python/
Last accessed: 2/10/19

[17] Steinberger, J., & Jezek, K. (2004). Using latent semantic analysis in text summarization and summary evaluation. *Proc. ISIM*, *4*, 93-100.

[18] Atindra Bandi (2018). "Web Scraping Using Selenium — Python", https://towardsdatascience.com/web-scraping-using-selenium-python-8a60f4cf40ab
Last accessed: 20/08/19

[19] Gilbert Tanner (2018). "Introduction to Web Scraping with BeautifulSoup", https://towardsdatascience.com/introduction-to-web-scraping-with-beautifulsoup-e87a06c2b857
Last accessed: 18/08/19

[20] Richa Bathija (2018). "Data Scientist's Guide to Summarization", https://towardsdatascience.com/data-scientists-guide-to-summarization-fc0db952e363
Last accessed: 05/09/19

[21] Omkar S Hiremath (2020), "A Beginner's Guide to learn web scraping with python!", https://www.edureka.co/blog/web-scraping-with-python/
Last accessed: 28/09/19

[22] Prateek Joshi (2018), “An Introduction to Text Summarization using the TextRank Algorithm (with Python implementation)”, https://www.analyticsvidhya.com/blog/2018/11/introduction-text-summarization-textrank-python/

[23] Last accessed: 6/09/2019

APPENDIX

Improved Word Frequency Python (3.7) Code:

```
def word_freq_improved_summarize(text):
    sen = text.split('.')
    #normalise
    small = [s.lower() for s in sen]
    #remove punctuation
    punc_free = []
    for p in small: punc_free.extend(token.tokenize(p))
    #remove stopwords
    stop_words = set(stopwords.words('english'))
    words = []
    for x in punc_free:
        if x not in stop_words: words.append(x)
    #weighted frequency
    wgt = {}
    for x in words: wgt[x] = words.count(x)
    max_freq = max(wgt.values())
    for x in wgt.keys(): wgt[x] = wgt[x]/max_freq
    #replace with weighted_frequency
    order = {}
    avg = len(sen)/2
    for i in range(len(sen)):
        sum = 0
        wrd = sen[i].split()
        for w in wrd:
            current = (str(token.tokenize(w))[2:-2]).lower()
            if current in wgt:
                sum += wgt[current]
        order[sen[i]] = sum*(1+0.1*abs(avg-i)/avg)
    sorted_sen = dict(sorted(order.items(), key = lambda x:x[1], reverse=True))

    final_summary = ""
    while True and len(sorted_sen)>0:
        summ = max(sorted_sen, key=lambda x:sorted_sen[x])
        if (len(final_summary)+len(summ))<240:
            final_summary += summ
            del sorted_sen[summ]
        else:
            if len(final_summary)<1:
                del sorted_sen[summ]
                continue
            else:
                break

    return final_summary
```

ACKNOWLEDGMENT

For the successful accomplishment of this project, many people have bestowed upon us their blessings and heart-pledged support. We want to thank all those people who have been directly or indirectly involved in this project.

We are highly indebted to our mentor Prof. Pankti Doshi for her guidance and constant supervision as well as for providing necessary information regarding the project & also for her support in completing the project. She spent a lot of her time and effort guiding us. Her constant support, invaluable feedback, and instructions enabled us to move forward in the right direction and overcome every obstacle that we came across.

We would also like to express our gratitude towards our college Mukesh Patel School of Technology Management and Engineering, NMIMS, for their kind cooperation and encouragement, which helped us in the completion of this project.