1. INTRODUCTION

1.1. Background

Robots on their own have revolutionized the way any company operates, whether they are a part of software or hardware-based industries. They provide a simple, cost efficient and often time efficient as well way of automating tasks that if done by an actual person would have been tedious. These robots, also known as bots in the field of computer science have changed the way computer tasks as simple as scheduling start-up for a particular program to complex processes such as automating web scraping have been accomplished. In recent times the prevalence of bots in social media has increased rapidly due to the ease by which they can be created.

Twitter is a social media website known for its propagation of information in short messages of 280 (previously 140) characters. These tweets are broadcast 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 a 'hashtag'. This categorization of tweets allows for the generation of 'trends' which is based on the number of times a tweet contains a particular hashtag in a short span of time.

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 our purposes. 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 as in this age of information overloaded modern-day life, with content being generated every millisecond around the world, it is quite difficult to extract the most essential information in an optimal amount of time. It is a deeply explored field that as of yet still does not yield appropriate results for all cases.

1.2. Project Overview

For our project we have selected to make a Bot that helps the spread of accurate 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 certain criteria which 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 pertaining to the trend. From this list articles are selected based on certain criteria such as date of writing, type of article etc and the selected articles are sent to a summarizer. The summarizer summarizes these articles into strings of maximum 280 characters. Amongst these, the best articles are selected and tweeted onto the bots twitter account.

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, a number of researchers have defined the definition of summary from their own perspective. For instance, computer scientist Sparck Jones defines a summary as a “reductive transformation of 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 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 organised or unorganised data. It then has to extract important keywords and information from it for the summary. Separating information as important or trivial is the main challenge in text summarization.

1.4. Motivation of Problem

Short message platform Twitter finds out which topic is the talk-of-the-town at any given time. It publishes these trending topics along with hashtags. The issue with these 'trends' is that they might not always be valid. As there is no provision to validate a buzzing topic before it is made a Twitter trend, it is quite possible for information that might be partially or completely incorrect to become a trend. Among the more than a hundred million people using Twitter, many do not have the time or motivation to open and read the articles attached along with the 1-2-word trend. The aim of our project is to find a method to accurately summarize articles related to the trends from reliable news sources and post them as <280-character tweets.

1.5. Problem Statement

Currently, many approaches and algorithms exist for summarizing passages and articles related to various domains. Few of them generate summaries which are close to human generated summaries. However, there is a trade-off between accuracy of summary and the time taken by machine to analyze the text and generate 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-to-word, which seems to remain unsolved as of now.

What is required is an algorithm that will give outputs in reasonable amount of time with acceptable accuracy. 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 and/or accuracy with minimum harm to the other parameter. This summarization algorithm will be used along a web scraping tool to fetch the articles related to the trends at a given time to post grammatically correct and contextually accurate summary on the Twitter platform.

1.6.Scope of the Problem

The project will compare and utilize extractive text summarization methods, i.e. summary sentences chosen from the passage itself according to priority decided be various algorithms. Another approach, i.e. abstractive summarization, uses methods that select words based on semantic understanding, even those words that do not appear in the source documents. The methods interpret and examine the text using advanced natural language techniques in order to generate a new shorter text that conveys the most critical information. As of now, abstractive summarization remains out of the scope for the project.

It is possible that trends of a particular location are in various 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 is 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 true. If any evidence is found to suggest one or more articles contradict each other, we plan to either ignore both articles to prevent the spread of false information or use both articles to give two sides of a story.

One of the major limitations we have come across is the lack of powerful workstations. Summarization by advanced techniques involving neural networks and/or machine learning require a great deal of resources and computation power to train the system before it can actually be used.

1.8.Organization of Report

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

2. REVIEW OF LITERATURE

2.1. Approaches to Text Summarization

Automatic text summarization systems can be broadly categorized into two different types based on output type, abstract and extractive summarization [2] (Figure 2.1). The authors state that extractive summaries are produced by identifying important sentences which 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.

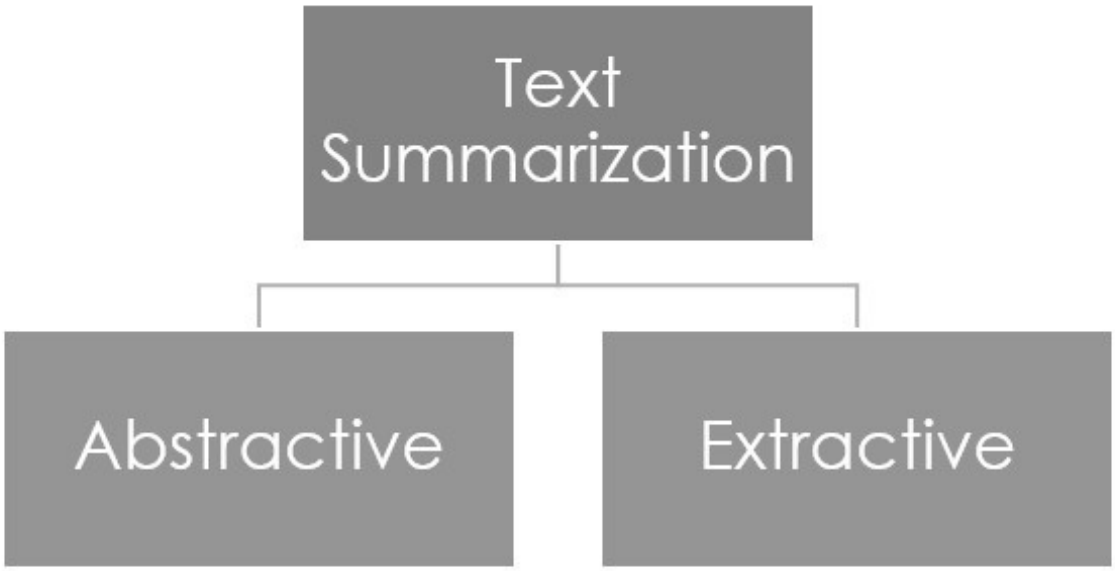

Figure 2.1: Text Summarization Approaches

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 main focus will be on creating a higher quality extractive summarization method that works for articles of various sizes and topics.

2.2.Extractive Text Summarization

Figure 2.2 shows the various algorithms for extractive text summarization such as Word Frequency, Term Frequency-Inverse Document Frequency and TextRank. There also exist approaches which use Machine Learning and Deep Learning for generating summaries [2].

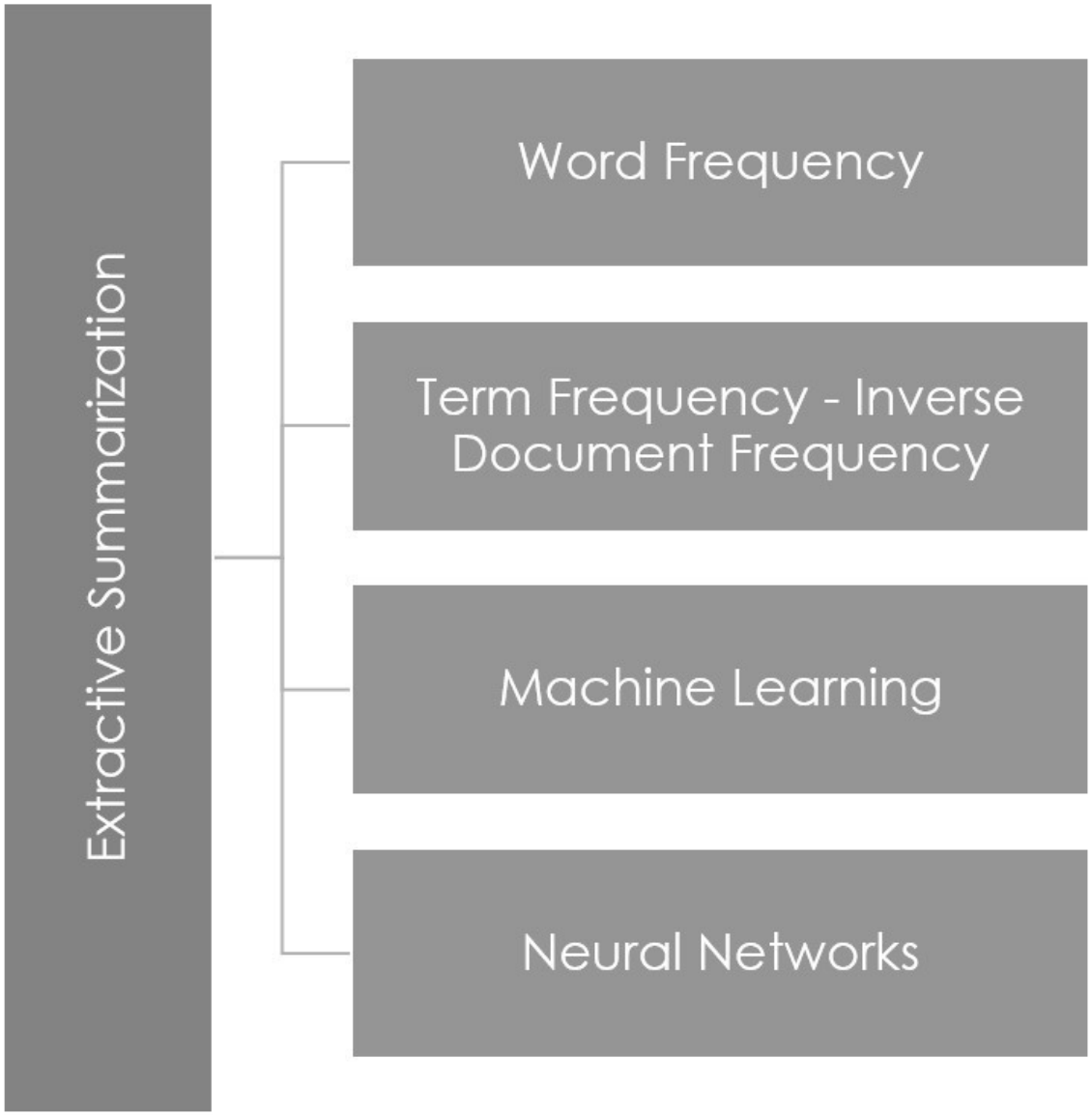

Figure 2.2: Types of Extractive Summarization

2.2.1. 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 make adjustment for the document length is by computing the word probability [2]. The probability f(w) of a word w is given by:

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

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.

2.2.2. 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 the tf-idf to be one of the universally used terminologies in extractive summarization [2].

Term frequency of a word is given as:

$$tf = n/\Sigma n \quad \text{… Equation 2.2 [2]}$$

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:

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

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:

$$tf\text{-}idf = tf*idf \quad \text{… Equation 2.4 [2]}$$

The sentences are scored based on the tf-idf values of the constituent words.

2.2.3. Machine Learning

According to [2], this approach can be applied if a set of training documents is available along with their corresponding summary extracts. Classification is used to assign a sentence either a "summary sentence" class or a "non-summary sentence" class. Bayesian Classification algorithm can be used for this method. It uses features of the sentences for the classification.

However, machine learning approaches for text summarisation take a long time to train. The models have to be trained for about a week to get good results. Also, the computation is complex which makes the overall cost high [5].

2.2.4. Neural Networks

The sentence semantics attributes can be analyzed and used for generating summaries using deep learning. Methods such as three-layered Feed-forward network, NetSum using RankNet algorithm, classifier-based approaches have been proposed and implemented [2].

A simple model consisting of one input layer, one hidden, and one output layer can be used. This model would be able to generate a summary of arbitrarily-sized documents by breaking them into fixed-size parts and feeding them recursively to the network.

Recurrent Neural Networks (Figure 2.3) and Sequence to Sequence methods have been proven to work well for summarization, but a lot of computation is needed for such models and they are fairly difficult to implement [10].

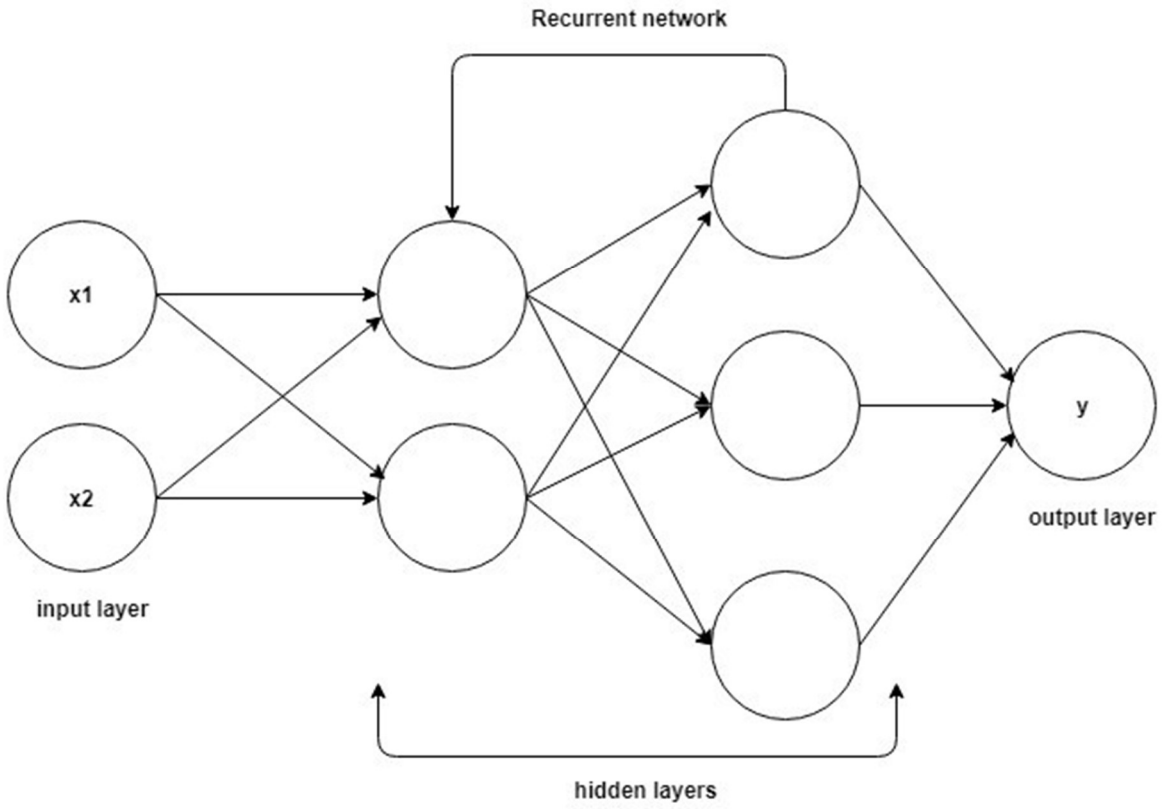

Figure 2.3 Recurrent Neural Network Architecture [13]

2.3.Evaluation of Summarization

In [6], Ani Nenkova talks about the broad classification of evaluation approaches into Extrinsic and Intrinsic evaluation (Figure 2.4). 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.

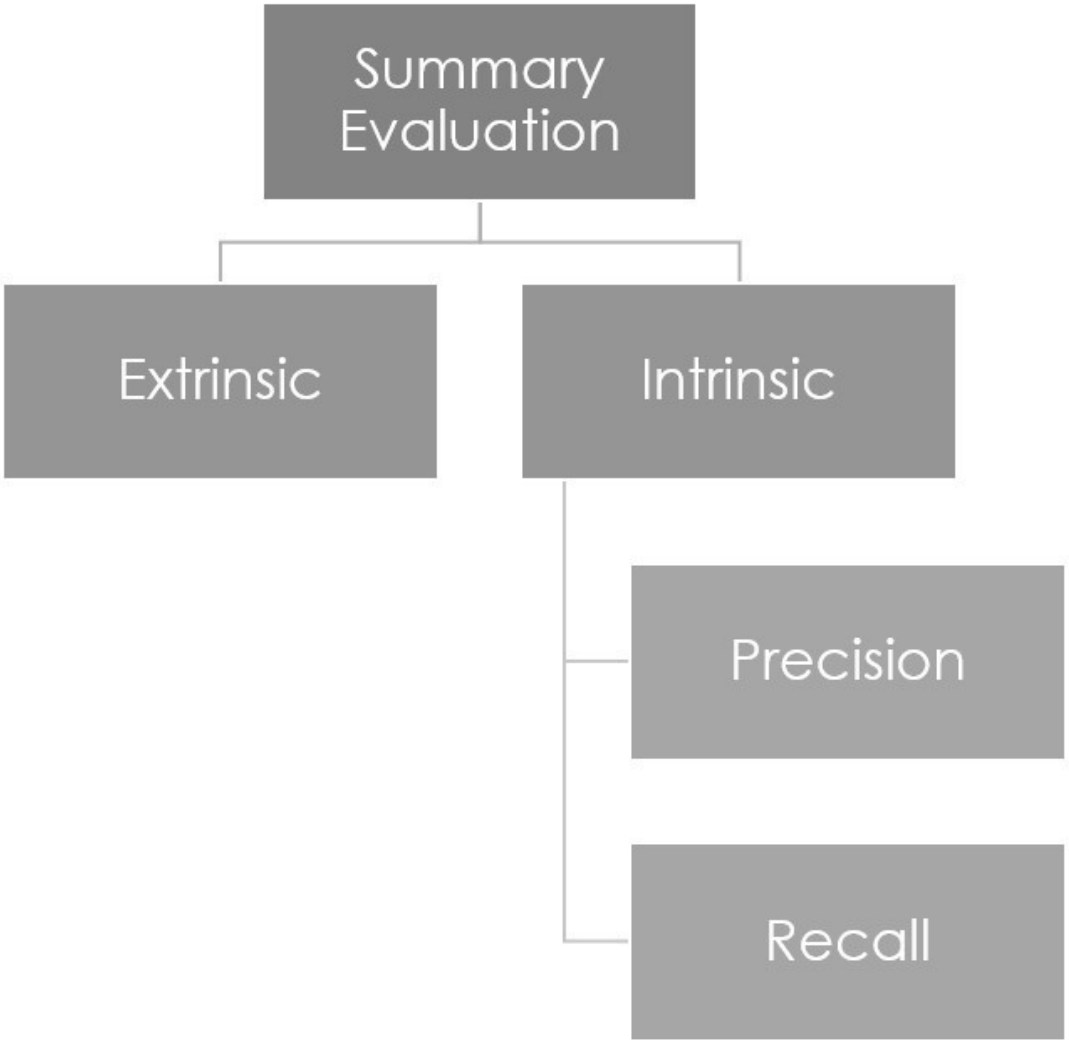

Figure 2.4: Types of Evaluations of Summaries

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.

Intrinsic Evaluation uses information retrieval metrics such as 'Precision' and 'Recall' for its measurements. 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:

Precision = |system-human choice overlap| ÷ |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 2.5, namely, Bilingual Evaluation Understudy (BLEU) [7] and Recall-Oriented Understudy for Gisting Evaluation (ROUGE) [8].

2.3.1. 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 very short summaries, BLEU combines Precision scores with 'Brevity Penalty'. It is given by:

BP = 1 ; if c > r

= e^(1-r/c) ; if c <=r … 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:

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

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

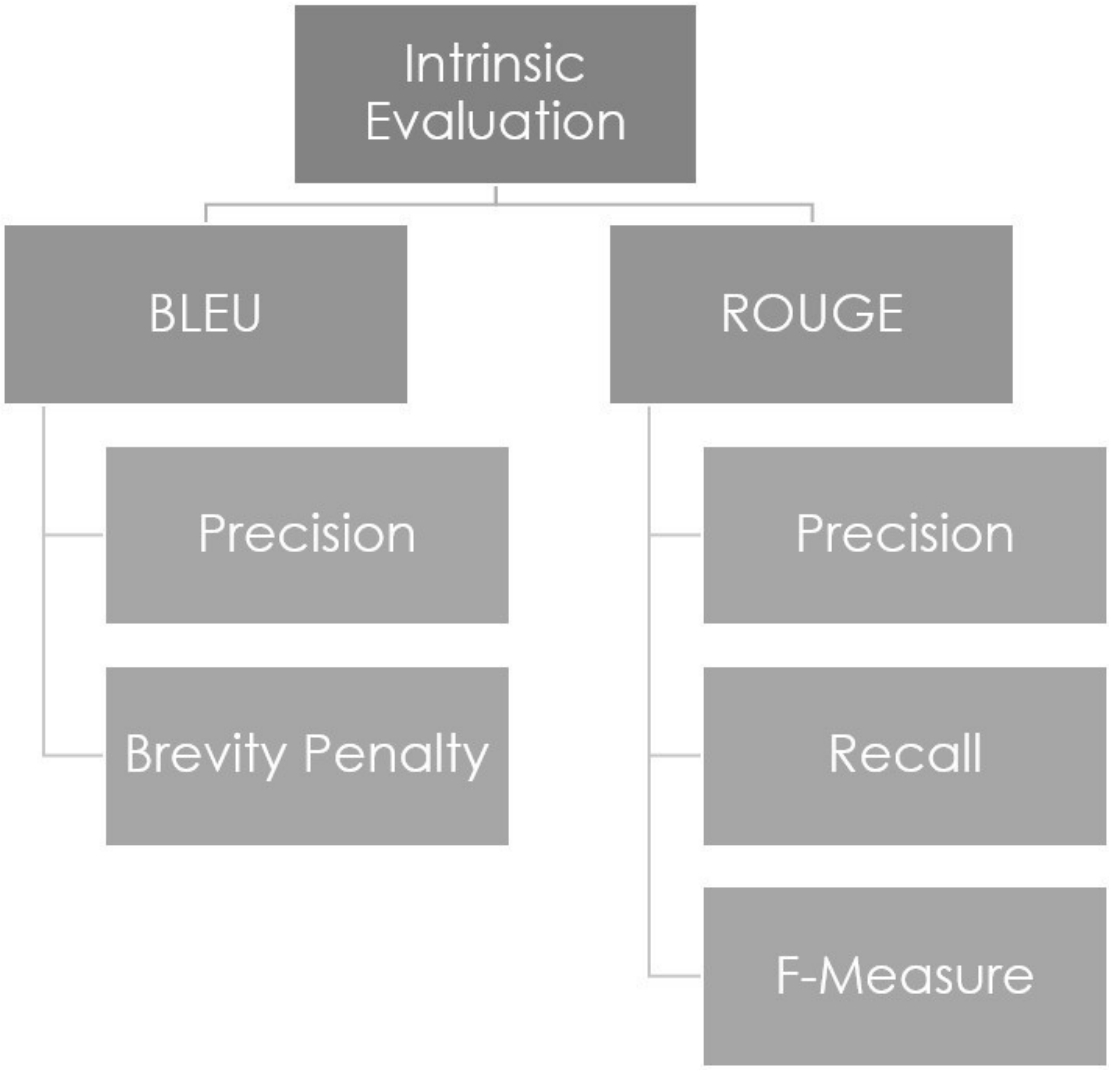

Figure 2.5 Algorithms for Intrinsic Evaluation

2.3.2. 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, 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 judgements for evaluation of text summarization [6].

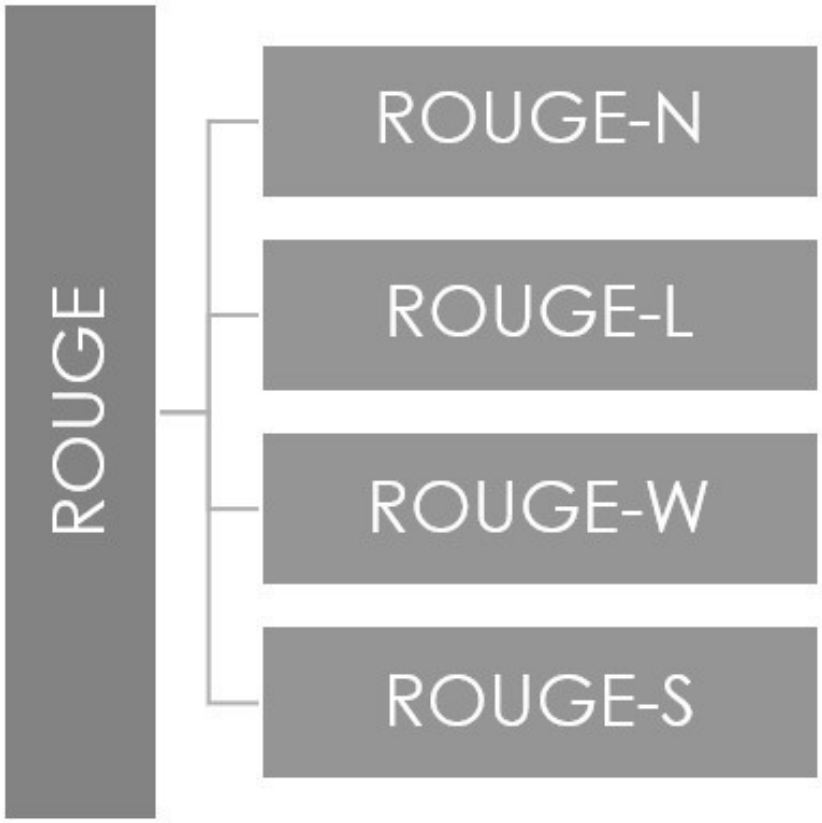

Fig 2.5: Types of ROUGE Measures

2.4. Twitter Trends

2.4.1. Fetch Twitter Trends

An essential aspect of our project is the task of fetching live and updated twitter trends from various sources these trends will then be used while web scraping for articles, thus it is of the utmost importance that the fetched trends are updated and authentic.

To ensure the same we have reviewed and researched multiple ways of getting these trends: -

- We can get twitter trends by using python packages like ‘tweepy’ or ‘twitter’.
- We can use the official Twitter APIs that provides us with trends and the tweet volume of all top trends. Using these methods, we can efficiently fetch live trends.

2.4.2. Twitter API

Twitter API call gives its response as a JSON file. Json needs to be filtered and cleaned to get useful trends while removing unwanted data. API requires authentication in the form of consumer keys and tokens provided by twitter. We can call this API for a finite number in a given time limit. API needs WOEID to specify location of trends API gives cached responses for repeated calls within 5 minutes.

2.4.3. Twitter Web App

To use any official twitter resources or put out our own tweet with our bot we first have to make a twitter web app on the twitter developer website. This is a necessary step as after creating a web app twitter provides us with the API keys and tokens that are required for us to use twitter API.

2.4.4. Posting New Tweets

The above-mentioned resources can also be used to write our own tweet and post it via the Bot account or the twitter developer profile. Either way, we can write our own string in python and it can be posted live. The constraint here is that the string should be less than 280 words as that it the max word limit for a tweet.

2.4.5. WOEID

Where on earth ID or WOEID is the location identifier created by yahoo and used primarily by twitter to identify locations. WOEID is a required parameter of the Twitter API as it helps twitter identify the country and city whose trends are being requested.

2.4.6. Trend Manipulation

Once we have the Trends it is imperative that we must manipulate and edit them fit our needs normal trends contain many unnecessary information such as hashtags and tweet volume. we must remove all this unnecessary data and keep only the trend strings as a list this will help us make the web scraping more effective and more efficient.

2.4.7. Drawbacks

- Twitter API uses WOEID to identify locations instead of latitude and longitude.
- Twitter API has a limited request rate which when exceeded does not give any useful results
- Twitter API returns cached information for repeated API calls within a time interval of 5 minutes

2.5. Web Scraping

2.5.1. BeautifulSoup

BeautifulSoup is a python library that is used to pull data out of HTML and XML files. It can work with different parsers to navigate, search and modify the parse tree according to our use case. It is one of the most prominently used libraries in the field of web scraping as it can use the parsers to easily and quickly generate a parse tree for the HTML pages taken off the web.

There exist a variety of functions to extract the exact tag(s) that we need and these functions return objects that have attributes for each part of the tag such as class, id, name, 'href', etc. which make it ideal for dealing with web pages from which we need to extract only a very specific piece of information.

2.5.2. Selenium

Selenium is the most popular library for web automation that while mainly used for testing can also be used for automating simple web tasks that if done by an individual would take long periods of time. It can also be used to abstract data from web pages dynamically. Selenium is mainly used using its java and python libraries

Selenium can only work with the use of drivers which are browser and version specific. These drivers are responsible for directly interacting with the browser and hence the web page. Drivers are only available for Chrome, Firefox and Internet Explorer and do not support any other browsers.

It has a wide variety of power classes and functions that allow it to send keys, enable clicks, focus, selection from a dropdown menu etc along with getting tags in the form of web elements from which data can be extracted.

3. PROPOSED SOLUTION

3.1. Selection of Summarization and Evaluation Algorithms

After reviewing 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.

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

Among intrinsic evaluation measures, **ROUGE-2** was and will be used for evaluating the summaries as it is close to human judgements.

We use **F-Measure**, i.e. harmonic mean of the 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

The algorithms with highest accuracy for text summarization use neural networks for analysing the semantic structure of text and then forming a gist. Hence, the project will focus more on **Neural Network** approaches.

3.2.Components

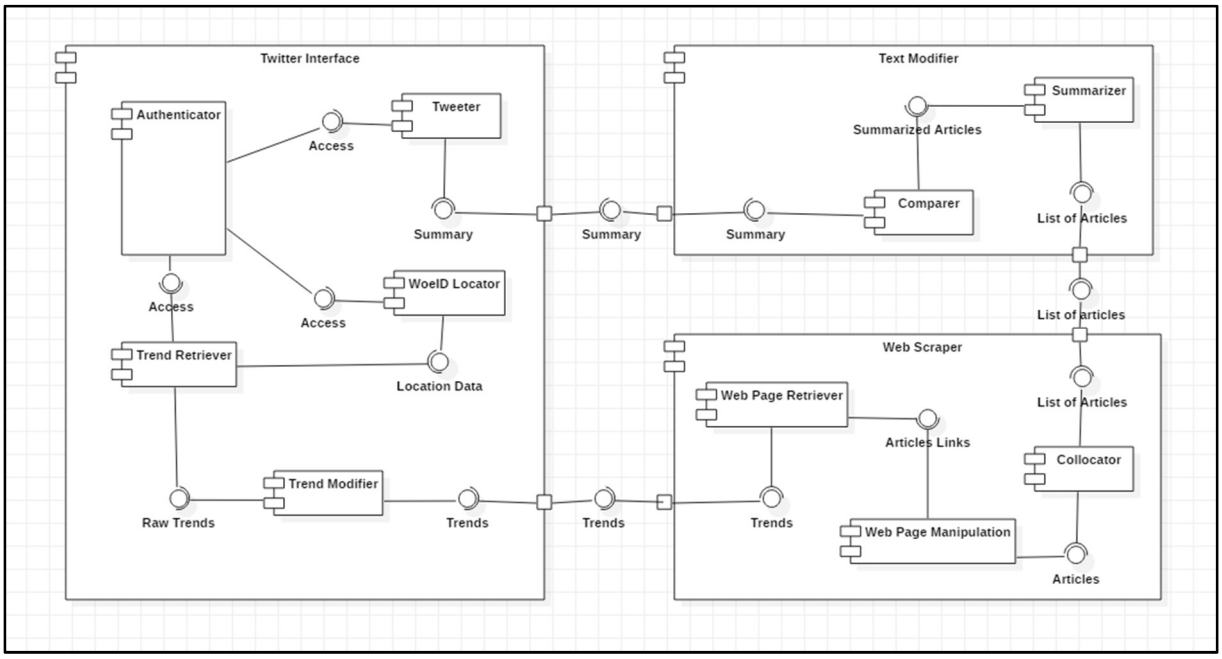

Figure 3.1 Component Diagram

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.

3.2.1. Twitter Interface

Twitter interface is the primary medium of the bot operation.

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 tweet.

3.2.2. Web Scraper

The web scraper is the second major 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.

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 picture, video 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 required 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

3.2.3. Text Modifier

This module is responsible for generating the <280-character Tweet from articles received from the Web Scraper module.

Summarizer

The list of articles received from 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 generated by Summarizer component will be sent to the Twitter Interface module.

3.3.General Flow of Execution

The bot has three main actors involved in its proper execution. If anyone 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 website we will be connected to using the Twitter API.

News websites refers 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, whether they have emoji etc. All viable trends are place in a list and the rest are ignored
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 pertaining to 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 certain extractive summarization techniques to obtain summaries for each.
8. The various summaries are then compared using simple techniques to check for contradiction.
9.
    a) If no contradiction is found, the best summary is tweeted
    b) If a contradiction is found, a comparison of two tweets is tweeted

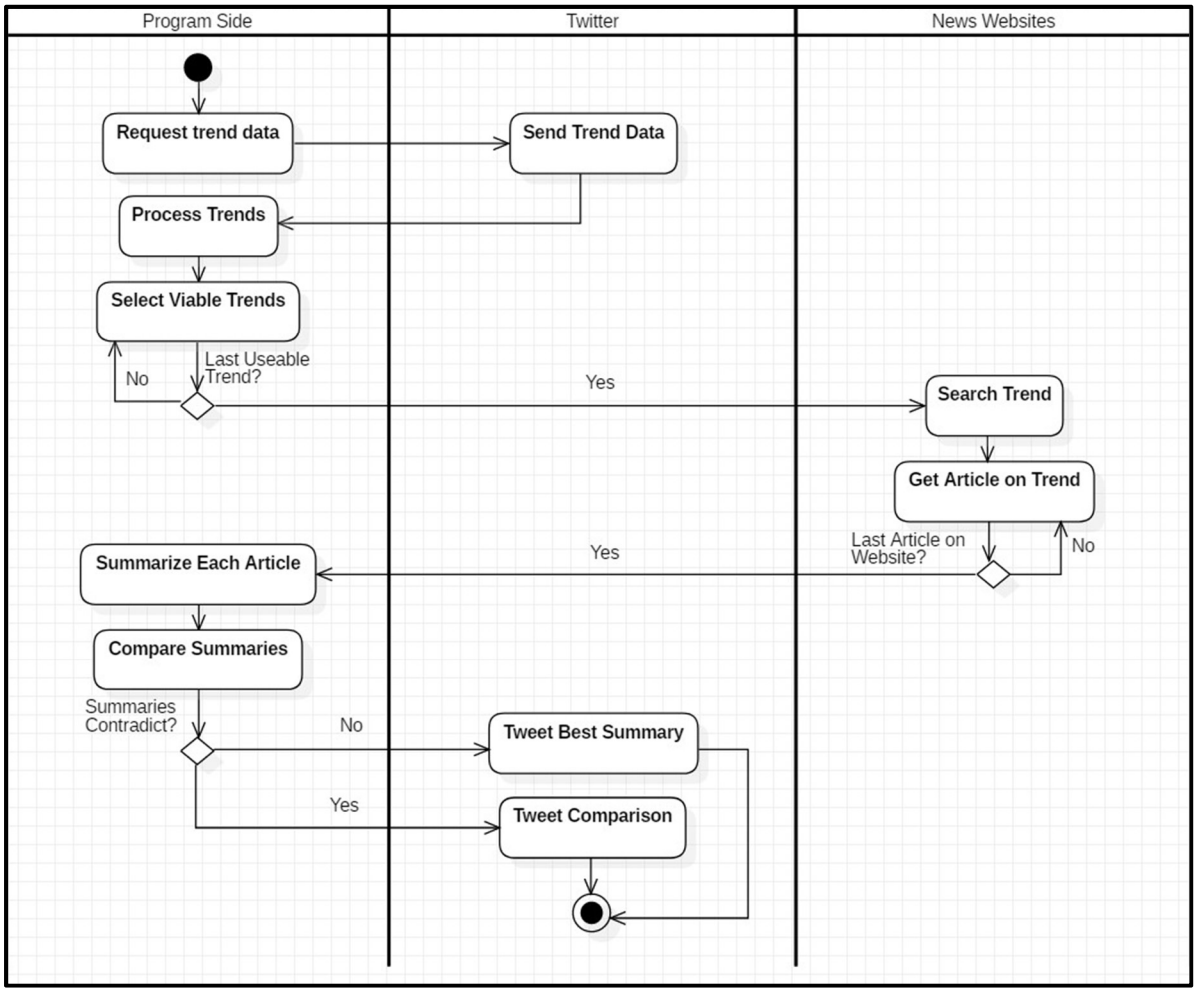

Figure 3.2 Swimlane Diagram

4. IMPLEMENTATION

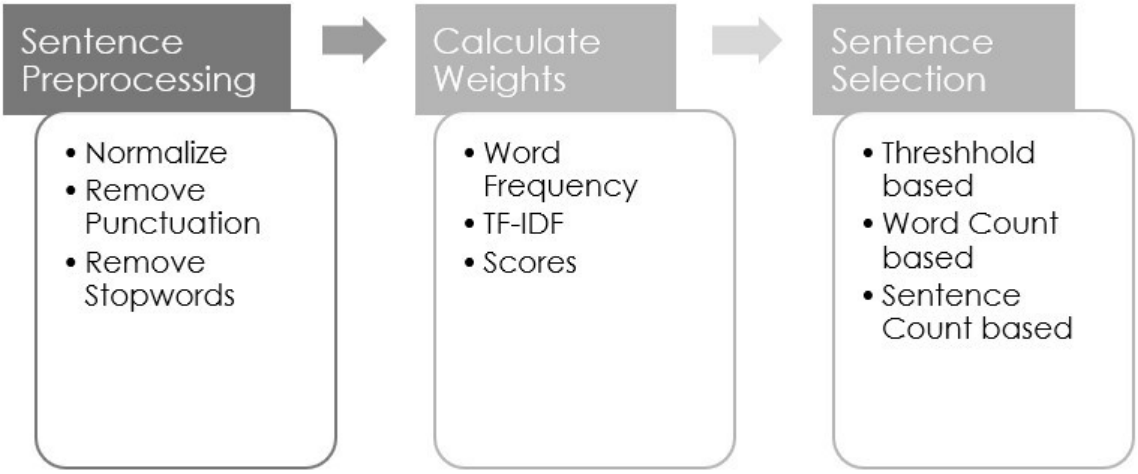

Figure 4.1: Extractive Text Summarization

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.

4.1.Summarization – 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 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 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 they 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 removal of stop words from test 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 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.

4.2.Summarization – 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 together two algorithms – the term frequency and the inverse document frequency algorithms.

We first used 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 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 3.). To calculate inverse document frequency (Equation 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 an 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 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.

4.3. Twitter – Fetching and Manipulating Trends

To process starts when our bot will initiate the Twitter API call on its own. The bot will send its credentials that is 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 etc 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.

We faced many challenges while doing this trend manipulation

- The first challenge was the language processing as the Trends were in multiple languages.
- Secondly, we found that many times multiple Trends talked about the same topic we needed to identify such trends out so as to make our web scraping search more efficient.
- Finally, we have to handle the abbreviations used in trends as abbreviations make it difficult for us to find understand the context of the trend.

4.4. Twitter – 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 successful match, the string is posted via that particular twitter account.

4.5. Web Scraping – BeautifulSoup and Selenium

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.

4.5.1. 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 *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 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 together 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 which 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.

4.5.2. Al-Jazeera

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

We begin but starting up the browser using the browser and version specific Selenium driver. The browser is run normally 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 *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 certain criteria. These links are placed in 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 together 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 which 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.

5. RESULTS AND DISCUSSION

The Twitter, Web Scraping and Summary generating modules were put together and the accuracy of 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.

For each keyword shown in tables 5.1 and 5.2, the Web Scraping module fetched 10 articles from websites and sent it to the summarizing module. The summaries were generated and were sent for evaluation. The average of Recall (Equation 2.5), Precision (Equation 2.5) and F-Measure (Equation 3.1) scores were taken for each keyword and are shown in tables 5.1 and 5.2. For comparison purposes, we also used Gensim's TextRank [4] summarizer module in Python on articles from all keywords mentioned before. The results are shown in table 5.3.

Algorithm	**Keyword**	**Average ROUGE-2 Precision**	**Average ROUGE-2 Recall**	**Average ROUGE-2 F-Measure**
Word Frequency	'huawei'	0.0417	0.2954	0.0712
	'brexit'	0.2246	0.2613	0.2004
	'blizzard'	0.1579	0.4992	0.2400
	'syria'	0.1880	0.4670	0.1838
	'chandrayaan'	0.0263	0.0571	0.1031
	'iphone'	0.0084	0.0434	0.0140
	'microsoft'	0.0145	0.0928	0.0242
	'turkey'	0.2109	0.5149	0.1894
	'google	0.0740	0.0212	0.0330
	'climate'	0.1800	0.0320	0.0290
Average		**0.1126**	**0.2284**	**0.1088**

Table 5.1: Word Frequency Evaluation

Table 5.1 shows the average scores for Word Frequency Algorithm for summarization. We can see that the accuracy of the algorithm is about 10%, which is very low when compared to other methods.

Algorithm	Keyword	Average ROUGE-2 Precision	Average ROUGE-2 Recall	Average ROUGE-2 F-Measure
Term Frequency – Inverse Document Frequency	'huawei'	0.1158	0.3242	0.1626
	'brexit'	0.4131	0.3983	0.3685
	'blizzard'	0.1813	0.4771	0.2546
	'syria'	0.3578	0.6013	0.4469
	'chandrayaan'	0.2512	0.3805	0.2850
	'iphone'	0.1056	0.1619	0.1109
	'microsoft'	0.1683	0.1918	0.2184
	'turkey'	0.3104	0.5726	0.3380
	'google	0.2460	0.3570	0.2350
	'climate'	0.2430	0.5610	0.3120
Average		**0.2392**	**0.4026**	**0.2732**

Table 5.2: TF-IDF Evaluation

The average scores for Term Frequency – Inverse Document Frequency are shown in table 5.2. The accuracy is close to 30%, which is better than Word Frequency, but is still far from the ideal score.

The scores from the Gensim's TextRank summarizer are shown in table 5.3. The average scores are less than but close to the TF-IDF algorithm.

Algorithm	Keyword	Average ROUGE-2 Precision	Average ROUGE-2 Recall	Average ROUGE-2 F-Measure
TextRank	'huawei'	0.1758	0.3108	0.2006
	'brexit'	0.1864	0.2626	0.1369
	'blizzard'	0.3032	0.4051	0.3420
	'syria'	0.2269	0.4351	0.2872
	'chandrayaan'	0.1559	0.0998	0.2699
	'iphone'	0.2066	0.2339	0.2192
	'microsoft'	0.2168	0.3357	0.2634
	'turkey'	0.3245	0.2481	0.2617
	'google	0.2750	0.5840	0.3740
	'climate'	0.2400	0.3400	0.2779
Average		**0.2311**	**0.3255**	**0.2633**

Figure 5.3: Gensim's TextRank Evaluation

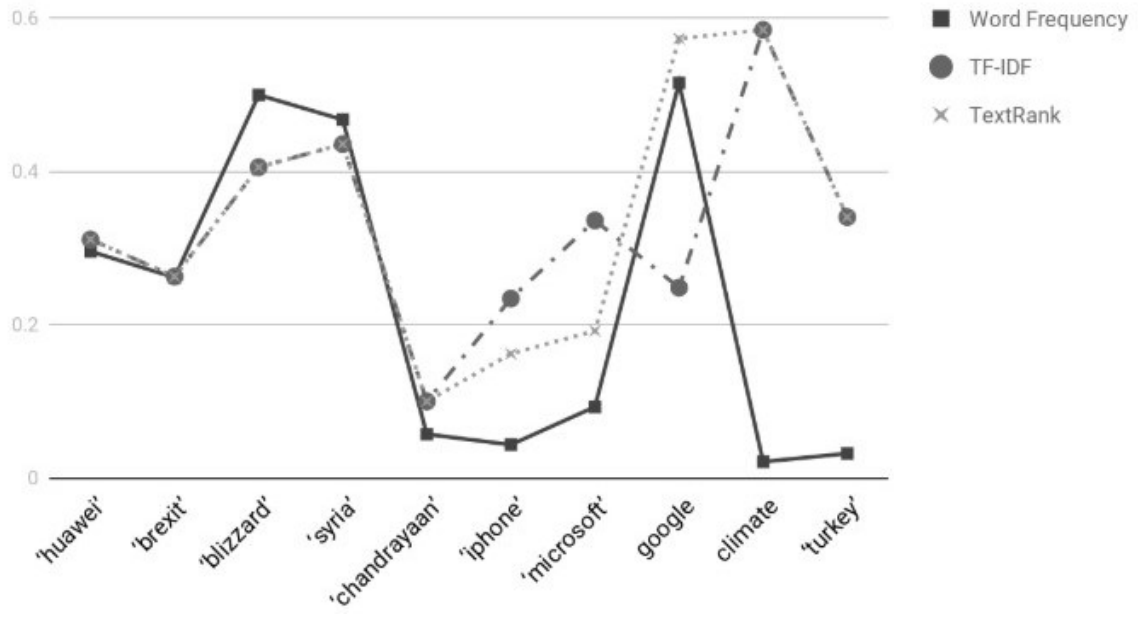

Figure 5.1: Average ROUGE-2 Recall Scores

Figures 5.1, 5.2 and 5.3 show comparison of scores of all keywords for the three algorithms. As we can see from figure 5.1, Recall alone is not an accurate metric for comparing algorithms, as it gives fluctuating scores, with no clear indication of which is better. Precision chart (Figure 5.2) and F-Measure chart (Figure 5.3) elucidate that word frequency is an all-time bad performer, and TF-IDF has the highest scores for most keywords.

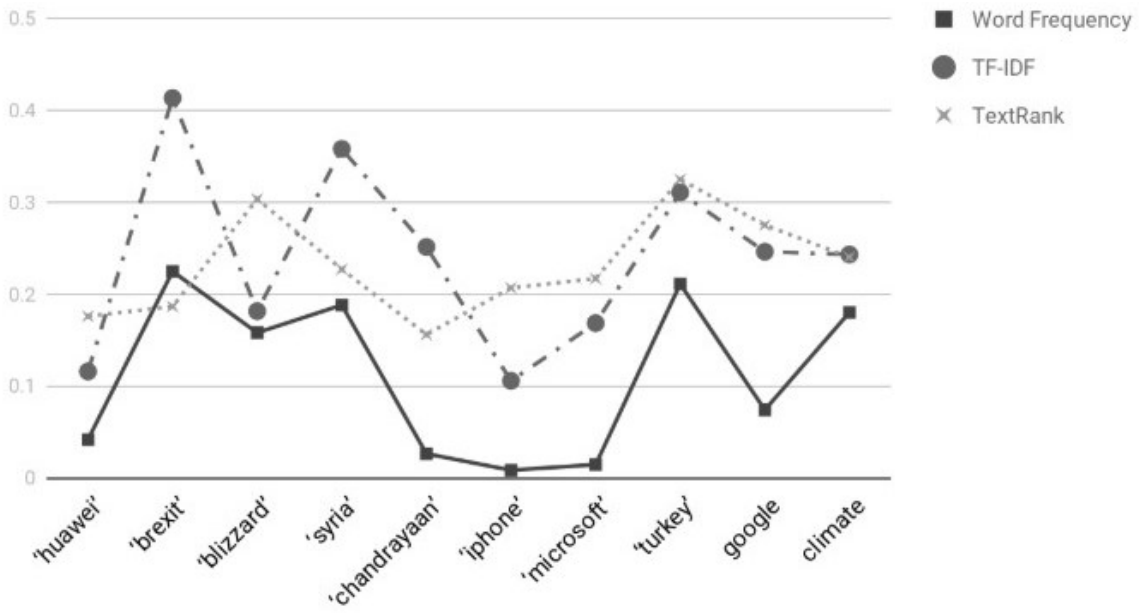

Figure 5.2: Average ROUGE-2 Precision Scores

Figure 5.4 concludes that TF-IDF algorithm has the highest accuracy among the three algorithms, with Gensim's TextRank close behind it. The accuracies of all three algorithms is still very low as some methods using deep learning have managed to reach an accuracy of more than 90 percent.

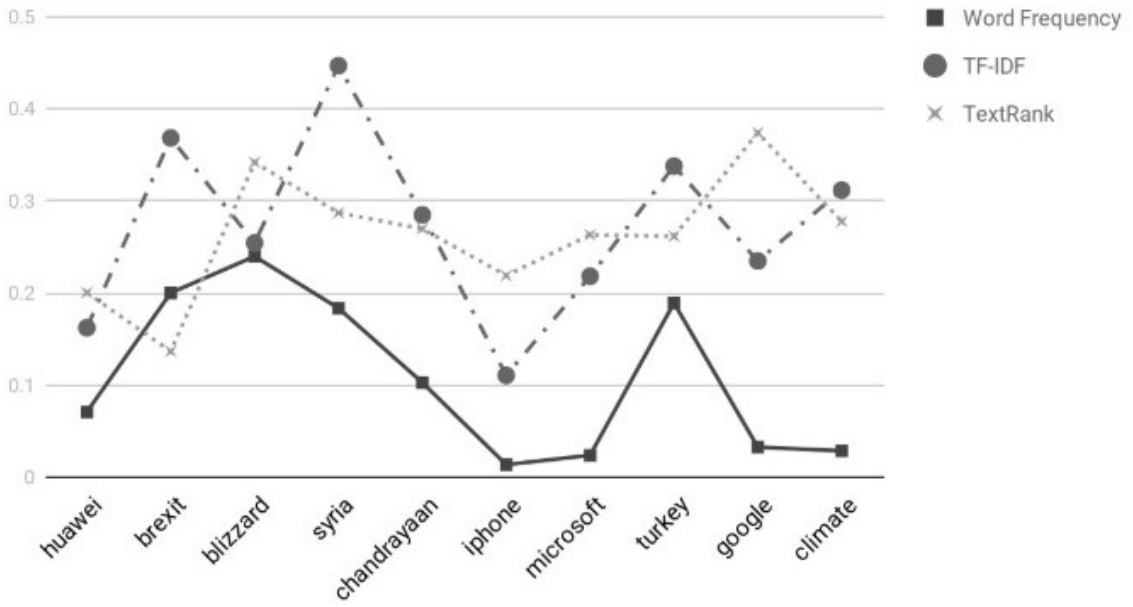

Figure 5.3: F-Measure Comparison

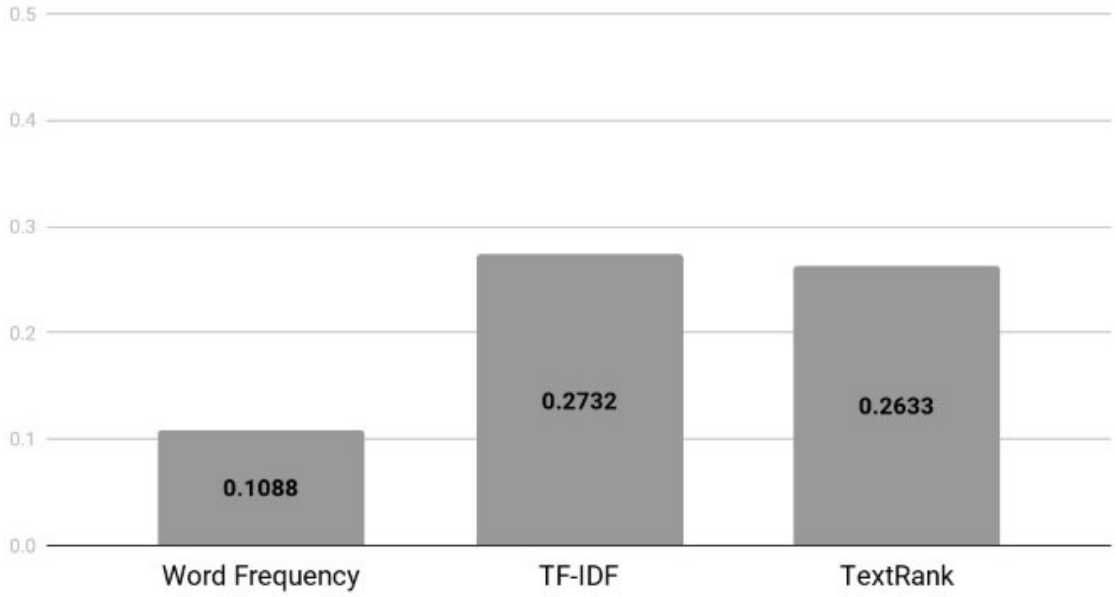

Figure 5.4: Average F-Scores

Web scraping as a whole has proved to be extremely time consuming to design and also proves to be one of the main aspects involved in the amount of time required for the program to complete one cycle of execution. This is primarily because of network speed and server-side queries that take time to give a result.

Along with that, the implementation of web scraping is extremely time consuming due to the varied nature of web pages on different web sites. Extensive bug testing has to be done to ensure a simple web scraping function works.

6. CONCLUSION AND FUTURE SCOPE

Thus, we have outlined our intent and implementation for our final year project in this report.

We have thus begun the implementation our Twitter bot that will provide its followers with instant and unbiased summarised news on the trending topics in their vicinities. Official Twitter resources are being used to fetch trends and post our tweets and various text summarisation algorithms summarise a news article into a tweet of less than 280 words based on articles that will be retrieved using web scraping

We continue to evaluate and test various summarization algorithms and by analysing and comparing those results we will select our final summarizing technique. The accuracy of Word Frequency algorithm is as low as 10 percent, whereas TF-IDF and Gensim's TextRank perform slightly better with an accuracy closer to 30percent. For now, we have chosen to focus our efforts on using a Recurrent Neural Network for summarization as our research has indicated towards it.

In the future, on successful implementation of our bot, we plan to focus on fetching trends and summarising articles in multiple languages, to try to bring about abstractive summarization instead of extractive to improve the quality of our content and to be able to do more advanced techniques of fact checking to prevent the spread of false information.

REFERENCES

[1] Gaikwad Deepali Kailash and C. Namrata Mahender (2016). *"A Review Paper on Text Summarization".*

[2] Kumar Yogan Jaya, Ong Sing Goh, Halizah Basiron, Ngo Hea Choon and Puspalata C Suppiah (2016). *"A Review on Automatic Text Summarization Approaches"*, JCS 12.

[3] Sinha, Aakash, Abhishek Yadav and Akshay Gahlot (2018). "*Extractive Text Summarization using Neural Networks"*, ArXiv abs/1802.10137.

[4] Mihalcea, Rada and Paul Tarau (2004). *"TextRank: Bringing Order Into Texts"*, EMNLP.

[5] Richa Bathija (2018). *"Data Scientist's Guide to Summarization"*.

[6] Nenkova, Ani (2006). *"Summarization Evaluation for Text and Speech: Issues and Approaches"*, Ninth ICSLP.

[7] Papineni, Kishore & Roukos, Salim & Ward, Todd & Zhu, Wei Jing (2002). *"BLEU: a Method for Automatic Evaluation of Machine Translation"*, ACL Workshop.

[8] Lin, Chin-Yew (2004). *"ROUGE: A Package for Automatic Evaluation of Summaries."*, Proceedings of the ACL Workshop

[9] Khosrow Kaikhah (2004). *"Text Summarization Using Neural Networks"*, WSEAS Transactions on Systems.

[10] Sinha, Aakash, Abhishek Yadav and Akshay Gahlot (2018). *"Extractive Text Summarization using Neural Networks"*, ArXiv abs/1802.10137

[11] BY Stefan Wojcik, Solomon Messing, Aaron Smith, Lee Rainie, and Paul Hitlin, (2018). *"Bots in the Twittersphere".*

[12] Haustein, S., Bowman, T. D., Holmberg, K., Tsou, A., Sugimoto, C. R. and Larivière, V (2016). *"Tweets as Impact Indicators: Examining the Implications of Automated "bot" Accounts on Twitter".*

[13] Debarko (2018). *"RNN or Recurrent Neural Network for Noobs"*.

[14] J. Fernquist, L. Kaati and R. Schroeder (2018). *"Political Bots and the Swedish General Election",* IEEE International Conference on ISI.

[15] Sulem, Elior & Abend, Omri & Rappoport, Ari. (2018). *"BLEU is Not Suitable for the Evaluation of Text Simplification"*.

[16] Atindra Bandi (2018). *"Web Scraping Using Selenium — Python".*

[17] Gilbert Tanner (2018). *"Introduction to Web Scraping with BeautifulSoup".*

APPENDIX

MODULE	DESCRIPTION
bs4	A popular web scraping module to parse html and xml documents
gensim	Python library designed to automatically extract semantic topics from documents.
nltk	Python package for NLP
– PorterStemmer	Module that stems words
– stopwords	Contains all the stopwords defined in the English language
os	Used to specify the directory in which the web driver is stored
requests	A function to retrieve the html web page of a specified url Selenium
selenium	
– webdriver	Class used to interact with the driver used to drive the web browser
– options	Provides the ability to set certain parameters to the browser such as running it headless (without GUI)
– keys	Used to send keys to the web browser which can fill text boxes and other text-based inputs
time	
– sleep	Stops execution of all processes in the given thread for a given amount of time
twitter	Package to access Twitter