

Corpus of Abstractive Text Summarization

Final Report

Prepared by

Aniket Sangwan (180001005)

Sarthak Jain (180001047)

Shah Miten (180001049)

Sundesh Gupta (180001057)

Indian Institute of Technology Indore
Software Engineering (CS258)

Table of Contents

1	Introduction	2
2	User Documentation	3
2.1	Installation guide	3
2.1.1	Installing Dataset	3
2.1.2	Installing Web Server	4
2.2	User Manual	4
2.3	Use Cases	4
3	System Documentation	6
3.1	System Architecture	6
3.2	Dataset	8
3.2.1	Web Crawling	8
3.2.2	Analyzing the Dataset	11
3.3	Machine Learning	12
3.3.1	Abstractive Model	12
3.3.2	Extractive Model	12
3.3.3	Rouge Scores	12
3.4	Web Interface and API	13
3.4.1	Summarization	13
3.4.2	Visualization	13
3.4.3	Datasets	14
3.5	Verification and Testing	14
3.5.1	Comparing and Testing Dataset	14
3.5.2	Comparing ROUGE Scores	14
3.5.3	Comparing Summarization Results	15

Chapter 1

Introduction

Now-a-days, research in the field of Natural Language Processing and Machine Learning is expanding at a great pace. To facilitate this research, we need improved datasets to train our model. Our aim is to create a dataset for Abstractive Text Summarization and also to test some state of the art model for the our dataset. Almost all the available datasets have specific writing style. Moreover, data with higher levels of abstraction is required for abstractive text summarizers. In this project, we present two different datasets- Hindustan Times dataset and freepressjournal dataset containing over 2k and 5k articles respectively. They involve articles from human authors with a lot of difference in their writing styles. The articles span a wide range of topics and therefore represent high diversity styles. We then calculate various characteristics such as compression ratio and extractive fragment coverage for each created dataset along with several pre-existing datasets.

To determine the usefulness of our datasets, they are evaluated on pre-trained summarization systems. The systems used are TextRank and Pointer-generator which use extractive and abstractive technique respectively. Rouge scores are calculated for each dataset to evaluate them. A web interface is developed which can be used to obtain both abstractive and extractive summary for input provided by the user. It also provides options to visualise the datasets by showing various characteristic graphs for every dataset.

Chapter 2

User Documentation

2.1 Installation guide

2.1.1 Installing Dataset

We have uploaded our dataset on github. You can find the dataset here:

- Free Press Journal Dataset: <https://github.com/sundeshgupta/Corpus-abstractive-text-summarization/blob/master/DataSet/freepressjournal.xml>
- Hindustan Times Dataset: <https://github.com/sundeshgupta/Corpus-abstractive-text-summarization/blob/master/DataSet/HindustanTimes.xml>

The dataset are available in the xml format, and contains the following information for each article-summary pair.

- url
- title
- body
- date

The code for parsing the data in python is given below:

```
1
2 import xml.etree.ElementTree as ET
3 tree = ET.parse('freepressjournal.xml')
4 root = tree.getroot()
5
6 for i in range(0, len(root), 4):
7     url = root[i].text
8     title = root[i+1].text
9     article = root[i+2].text
10    date = root[i+3].text
11    % processing title, article and date
```

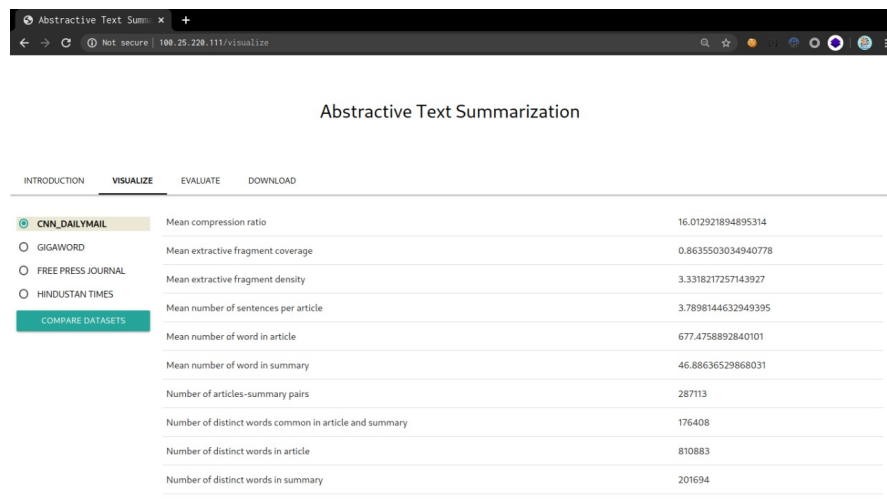
2.1.2 Installing Web Server

Follow the following instructions along with the instructions provided on [SummarizeModel](#).

- Run the command **pip install -r requirements.txt**
- Change the working directory accordingly in this [line](#).
- Run the flask app with the command **python app.py**
- The IP Address and the Port for running the app can be changed [here](#)

2.2 User Manual

2.3 Use Cases



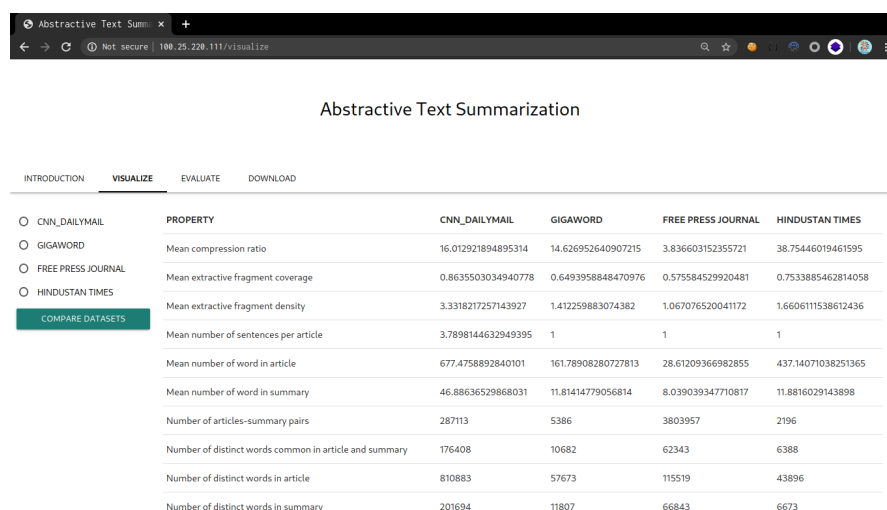
Abstractive Text Summarization

INTRODUCTION **VISUALIZE** EVALUATE DOWNLOAD

☒ CNN_DAILYMAIL

PROPERTY	CNN_DAILYMAIL
Mean compression ratio	16.012921894895314
Mean extractive fragment coverage	0.8635503034940778
Mean extractive fragment density	3.3318217257143927
Mean number of sentences per article	3.7898144632949395
Mean number of word in article	677.4758892840101
Mean number of word in summary	46.88636529868031
Number of articles-summary pairs	287113
Number of distinct words common in article and summary	176408
Number of distinct words in article	810883
Number of distinct words in summary	201694

Figure 2.1: Visualization Page of Web App



Abstractive Text Summarization

INTRODUCTION **VISUALIZE** EVALUATE DOWNLOAD

☐ CNN_DAILYMAIL
☐ GIGAWORD
☐ FREE PRESS JOURNAL
☐ HINDUSTAN TIMES

COMPARE DATASETS

PROPERTY	CNN_DAILYMAIL	GIGAWORD	FREE PRESS JOURNAL	HINDUSTAN TIMES
Mean compression ratio	16.012921894895314	14.626952640907215	3.836603152355721	38.75446019461595
Mean extractive fragment coverage	0.8635503034940778	0.6493958848470976	0.575584529920481	0.7533885462814058
Mean extractive fragment density	3.3318217257143927	1.412259883074382	1.067076520041172	1.660611538612436
Mean number of sentences per article	3.7898144632949395	1	1	1
Mean number of word in article	677.4758892840101	161.78908280727813	28.61209366982855	437.14071038251365
Mean number of word in summary	46.88636529868031	11.81414779056814	8.039039347710817	11.8816029143898
Number of articles-summary pairs	287113	5386	3803957	2196
Number of distinct words common in article and summary	176408	10682	62343	6388
Number of distinct words in article	810883	57673	115519	43896
Number of distinct words in summary	201694	11807	66843	6673

Figure 2.2: Visualization Page of Web App

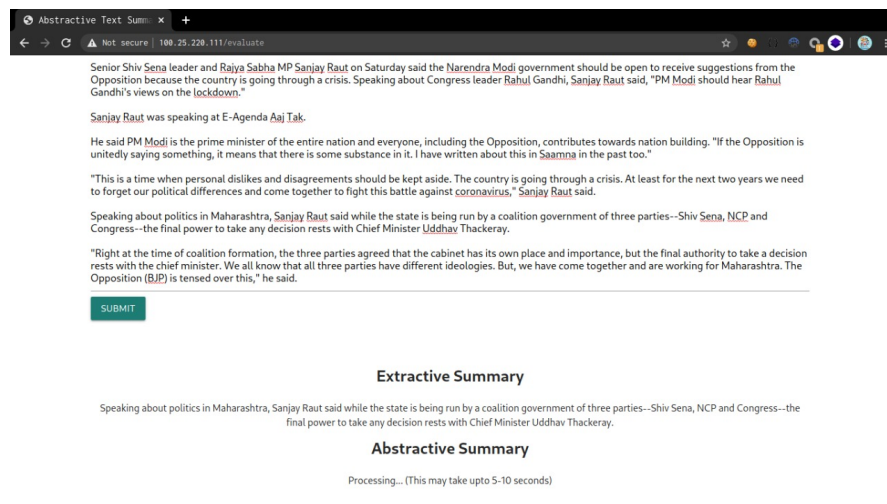


Figure 2.3: Summarization Page of Web App

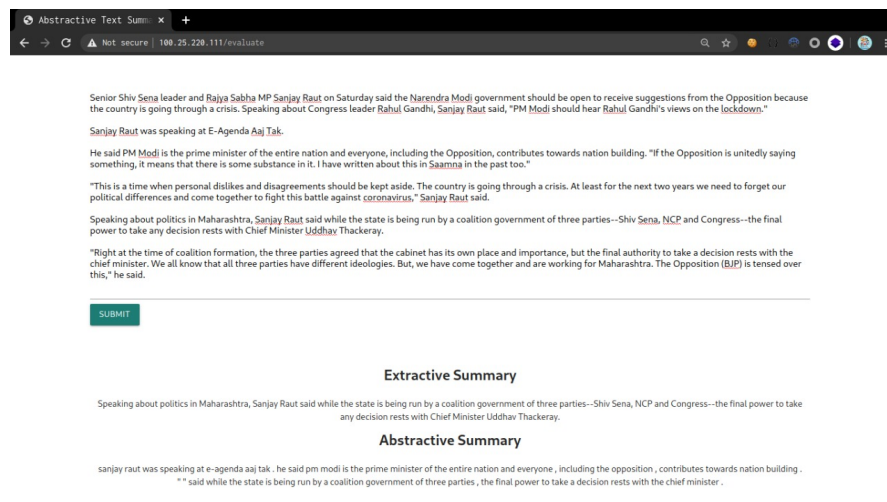


Figure 2.4: Summarization Page of Web App

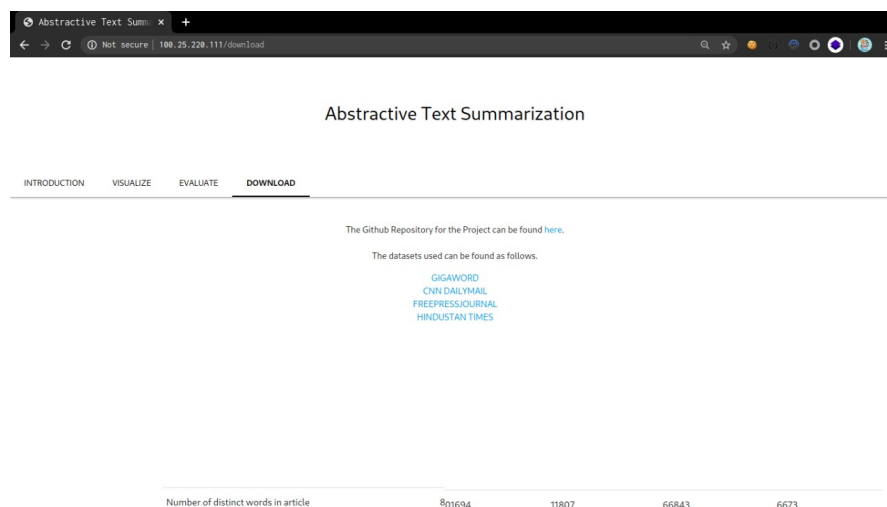


Figure 2.5: Dataset download Page of Web App

Chapter 3

System Documentation

3.1 System Architecture

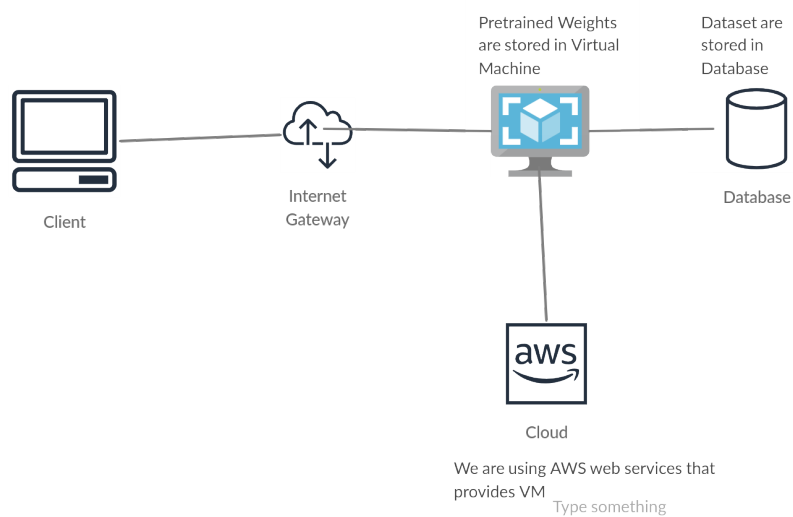


Figure 3.1: System Architecture

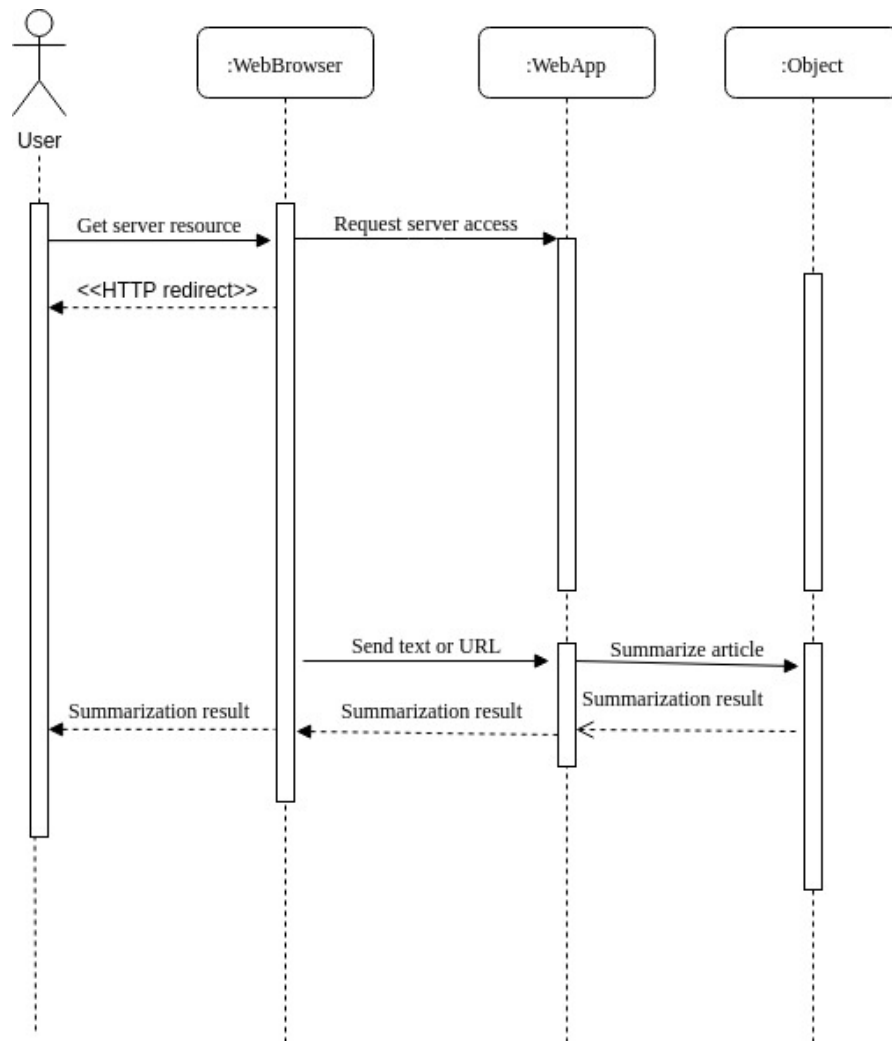


Figure 3.2: Web Application Sequence Diagram

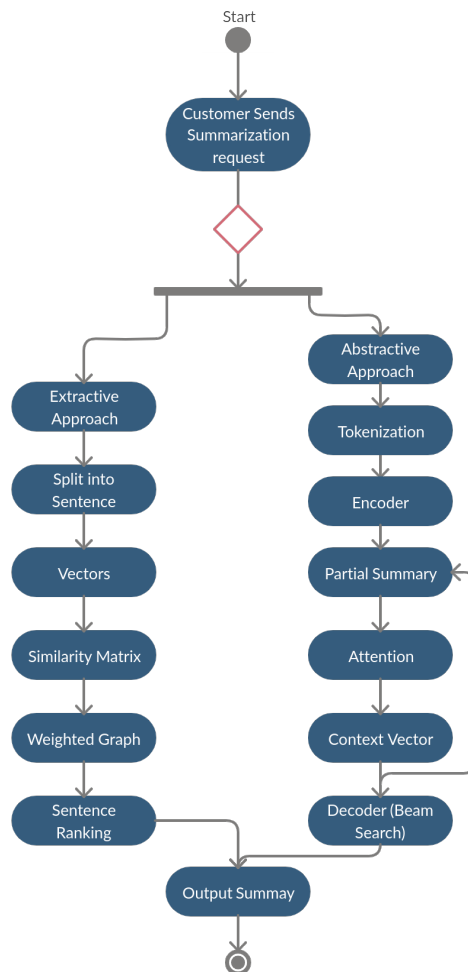


Figure 3.3: Activity Diagram for Summarization

3.2 Dataset

3.2.1 Web Crawling

- To begin the process of web scraping, we first specify the start URL. This is the URL which is used to begin scraping. A set is used to keep track of all the visited URLs.

```

1  URL = "https://www.hindustantimes.com/business-news/now-microsoft-eyes-stake-in-jio
2      /story-AFLHYwozmBPP0L6Y3ffzyI.html"
3  domain = "https://www.hindustantimes.com/"
4  http = 'http://'
5  https = 'https://'
6  visited = set()

```

- We specify the size of dataset required and begin BFS to visit every URL present in the queue. A request is sent to the address of current URL.

```

1  while len(q)>0:
2      URL=q.popleft()
3      req=request.Request(URL,headers = {"User-Agent": "Mozilla/5.0"})

```

```

4
5     if cnt>10000:
6         break

```

- If the request returns error, then we ignore the current address and continue further

```

1     try: request.urlopen(req)
2     except error.URLError as e:
3         continue

```

- Beautiful soup is used to pull data out of HTML address. We extract the hyperlinks present on the page and in next steps, insert only those URLs which belong to the same domain. We define the minimum length of URL to skip through the links which don't contain any article. We observed that a lot of visited URLs had no article. So we created a list with common words in those URLs to filter them. This resulted in a huge speed-up. If any of the extracted text is below minimum length, the current URL is ignored and the process is continued.

```

1     a=request.urlopen(req).read()
2     soup = BeautifulSoup(a, 'html.parser')
3     l=[]
4     for i in soup.find_all('a'):
5         l.append(i.get('href'))
6     if len(summary)<25:
7         return None, None, None

```

- We analyzed the general HTML format for each domain and selected classes which contain headline, article text and date of the article. Certain methods were used to get rid of HTML tags and non-ASCII characters from the extracted text. The below code shows how the article was extracted.

```

1     article=""
2     for row in table.find_all_next('p'):
3         text = row.text
4         article += text.strip() + " "
5
6     article=article.replace('<','')
7     article=article.replace('>','')
8     article=article.replace('&','')
9
10    if len(article)<200:
11        return None, None, None

```

- The extracted text is stored in an XML file in fixed format. This XML file acts as our database, once the BFS process is completed.

```

1     s=("<url >"+URL+"</url>")
2     s+='\n'
3     s+=("<title>"+summary+"</title>")
4     s+='\n'
5     s+=("<body >"+article+"</body>")
6     s+='\n'
7     s+=("<date >"+date+"</date>")
8     s+='\n'

```

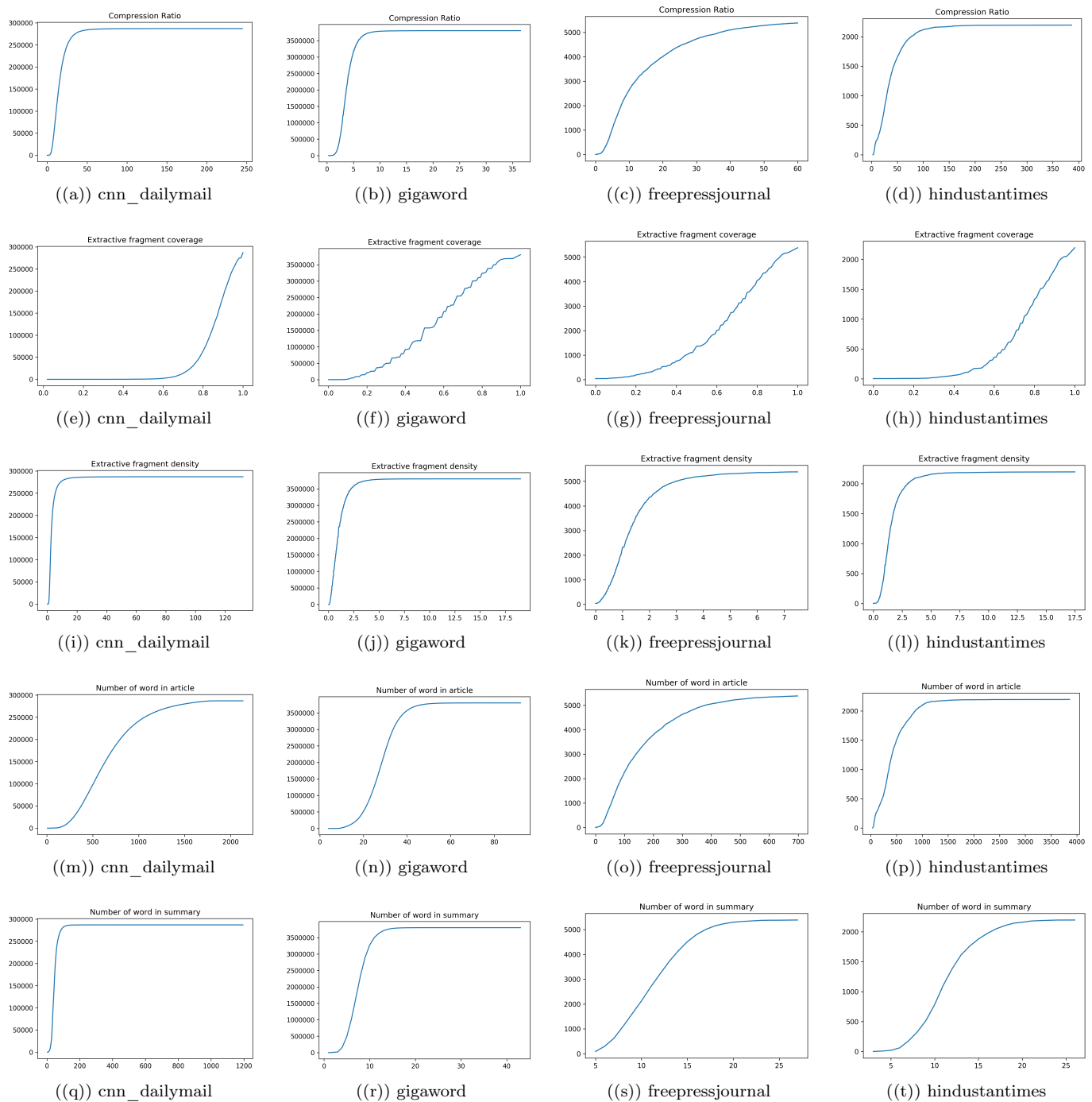


Figure 3.4: Statistical Information about the dataset

3.2.2 Analyzing the Dataset

We analyze our dataset with various standard summarization strategies that capture the degree of text overlap between the summary and article, and the rate of compression of the information conveyed. We also compared our dataset with standard summarization dataset, `cnm_dailymail` and `gigaword`, that are available on tensorflow. We define a set of extractive fragments $F(A, S)$, as the set of shared sequences of tokens in A and S . This was defined in the [NEWSROOM: A Dataset of 1.3 Million Summaries with Diverse Extractive Strategies](#). They used a greedy method to find this set. The python code for the method, is given below.

```

13 def extractive_fragment(article, summary):
14     F = []      % Declaration of Extractive Fragments
15     i = 0
16     j = 0
17     p = len(article)
18     q = len(summary)
19     while i < q:
20         f = []
21         while j < p:
22             if summary[i] == article[j]:
23                 tmp_i = i
24                 tmp_j = j
25                 while tmp_i < q and tmp_j < p and summary[tmp_i] == article[tmp_j]:
26                     tmp_i += 1
27                     tmp_j += 1
28                 if len(f) < (tmp_i - i):
29                     f = summary[i:tmp_i]
30                 j = tmp_j
31             else:
32                 j += 1
33             i = i + max(len(f), 1)
34             j = 0
35             F.append(f)
36     return F

```

Using $F(A, S)$, we compute two measures:

- **Extractive Fragment Coverage:** The coverage measure quantifies the extent to which a summary is derivative of a text. $COVERAGE(A, S)$ measures the percentage of words in the summary that are part of an extractive fragment with the article:

$$COVERAGE(A, S) = \frac{1}{|S|} \sum_{f \in F(A, S)} |f|$$

- **Extractive Fragment Density:** The density measure quantifies how well the word sequence of a summary can be described as a series of extractions. We define $DENSITY(A, S)$ as the average length of the extractive fragment to which each word in the summary belongs

$$DENSITY(A, S) = \frac{1}{|S|} \sum_{f \in F(A, S)} |f|^2$$

We use a simple dimension of summarization, **Compression Ratio**, to further characterize summarization strategies. We define $COMPRESSION$ as the word ratio between the article and summary:

$$COMPRESSION(A, S) = \frac{|A|}{|S|}$$

We have further, compared our dataset in a tabular format in Section 3.5.5.

3.3 Machine Learning

We evaluate our proposed datasets with popular summarization systems and compare with well-known datasets such as CNN/Daily Mail and Gigaword datasets. We use pre-trained summarization systems to understand the challenges of our Hindustan Times and Freepressjournal datasets and their usefulness for training systems. We evaluate based on two systems, each using a different summarization strategy. First is the TextRank, that uses extractive technique for text summarization and second is the Pointer-generator, that uses abstractive technique to achieve text summarization. Finally, for evaluation, we calculate, for a subset of all four datasets, ROUGE-1, ROUGE-2 and ROUGE-L scores using both summarization systems.

3.3.1 Abstractive Model

The pointer-generator model (See et al., 2017) uses abstractive token generation and extractive token copying using a pointer mechanism, keeping track of extractions using coverage. We used @becxer's code from the github repository, which is @abisee's (the author of pointer-generator paper) code but updated for Python3. The pre-trained model used is trained on CNN and Daily Mail dataset. Input article text and reference summaries are tokenized using the Stanford CoreNLP tokenizer and decoding is done using beam search.

3.3.2 Extractive Model

TextRank is a sentence-level extractive summarization system. The system was originally developed by Mihalcea and Tarau (2004) and was later further developed and improved by Barrios et al. (2016). TextRank picks a sequence of sentences from a text for the summary up to a maximum allowable length. While this maximum length is typically preset by the user, in order to optimize ROUGE scoring for our proposed datasets, we tuned this parameter to optimize ROUGE scores. We used summa, a Python package for extractive summarization. Summa is implementation of TextRank that uses optimisations from Barrios et al. research paper.

3.3.3 Rouge Scores

ROUGE, or Recall-Oriented Understudy for Gisting Evaluation is a set of metrics used for evaluating automatic summarization and machine translation software in natural language processing. The metrics compare an automatically produced summary or translation against a reference or a set of references (human-produced) summary or translation. It helps to characterize the sensitivity of automatic summarization systems to different levels of extractiveness in reference summaries, thus help us understand the usefulness of our datasets for training systems. We use ROUGE-1, ROUGE-2 and ROUGE-L variants of ROUGE score for evaluation purpose.

- ROUGE-1 refers to the overlap of unigram (each word) between the system and reference summaries.
- ROUGE-2 refers to the overlap of bigrams between the system and reference summaries.
- ROUGE-L refers to Longest Common Subsequence (LCS) based statistics and takes into account sentence level structure similarity naturally and identifies longest co-occurring in sequence n-grams.

We use pyrouge, a python package to run ROUGE evaluation. The experimental results are available in Section 3.5.3

3.4 Web Interface and API

3.4.1 Summarization

API

The Summarization API runs on the endpoint `/api/summarize`. All data is sent and received as JSON. A given block of text can be summarized by hitting a POST request on the mentioned endpoint. The user has an option to either request for Extractive Summarization or Abstractive Summarization. The format for the request and response is mentioned below.

POST Request

```
1 {
2   "text" : "I want to summarize this text",
3   "type" : "abstractive or extractive"
4 }
```

Response

```
1 {
2   "text" : "I want to summarize this text",
3   "summary" : "This is the summary you wanted"
4 }
```

Web Frontend

The Web Frontend for Summarization runs on the endpoint `/evaluate`. The interface is self-explanatory. It requests the above API for the summary and embeds it in the document. The user can select whether they want the summary to be abstractive/extractive. (Note: Abstractive Summarization may take time upto 15 seconds)

3.4.2 Visualization

API

The Visualization API runs on the endpoint `/api/visualize`. This part deals with the visualization of the datasets generated by Crawling Webpages along with some pre-generated dataset. Similar to summarization, all data for this endpoint is sent and received as JSON. The options for datasets are :

- 'cnn' : Dataset description for [CNN Dailymail](#) Dataset.
- 'giga' : Dataset description for [Gigaword](#) Dataset.
- 'free' : Dataset description for [Free Press Journal](#). This was generated using the Crawler.
- 'free' : Dataset description for [Hindustan Times](#). This was generated using the Crawler.
- 'all' : Dataset description for all the datasets nested together.

POST Request

```
1 {
2   "dataset" : "Dataset Type"
3 }
```

Response

```

1 {
2   "Number of articles-summary pairs": <number>,
3   "Number of distinct words in article": <number>,
4   "Number of distinct words in summary": <number>,
5   "Number of distinct words common in article and summary": <number>,
6   "Mean number of word in summary": <number>,
7   "Mean number of word in article": <number>,
8   "Mean number of sentences per article": <number>,
9   "Mean compression ratio": <number>,
10  "Mean extractive fragment coverage": <number>,
11  "Mean extractive fragment density": <number>
12 }

```

Web Frontend

The Web Frontend for Visualization can be found at the endpoint `/visualize`. It hits the above API in background and tabulates the response. The user also has an option for comparing all the datasets.

3.4.3 Datasets

This runs on the endpoint `/download`. It provides links to the Github Repository along with the datasets used for the project.

3.5 Verification and Testing

3.5.1 Comparing and Testing Dataset

To ensure that our dataset is proper, we have compared the datasets in a tabular manner.

	cnn_dailymail	gigaword	freepressjournal	hindustantimes
Number of articles-summary pairs	287113	3803957	5386	2196
Number of distinct words in article	810883	115519	57673	43896
Number of distinct words in summary	201694	66843	11807	6673
Number of distinct words common in article and summary	176408	62343	10682	6388
Mean number of word in summary	46.88	8.03	11.81	11.88
Mean number of word in article	677.47	28.61	161.78	437.14
Mean number of sentences per article	3.78	1.0	1.0	1.0
Mean compression ratio	16.01	3.83	0.57	38.75
Mean extractive fragment coverage	0.86	0.57	0.64	0.75
Mean extractive fragment density	3.33	1.06	1.41	1.66

3.5.2 Comparing ROUGE Scores

To understand the performance of Text Summarization Models, we have calculated ROUGE scores. We took a random sample of 50 article-summary pairs, and calculated F1, Recall and Precision metrics for each of the R1, R2 and RL scores.

- F1 score

	CNN/Dailymail			Gigaword			Free Press Journal			Hindustan Times		
	R-1	R-2	R-L	R-1	R-2	R-L	R-1	R-2	R-L	R-1	R-2	R-L
Abs	35.25	12.59	32.34	14.23	4.62	13.22	16.45	5.07	13.82	18.54	6.58	16.55
Ext	30.40	8.19	20.40	4.47	1.50	4.04	17.27	6.06	13.89	24.88	10.09	20.51

- Recall

	CNN/Dailymail			Gigaword			Free Press Journal			Hindustan Times		
	R-1	R-2	R-L	R-1	R-2	R-L	R-1	R-2	R-L	R-1	R-2	R-L
Abs	40.07	14.02	36.58	60.07	20.11	55.12	44.72	15.29	37.92	51.57	19.31	45.76
Ext	33.51	8.52	22.69	10.91	3.88	9.77	40.54	15.87	33.31	59.98	25.56	50.19

- Precision

	CNN/Dailymail			Gigaword			Free Press Journal			Hindustan Times		
	R-1	R-2	R-L	R-1	R-2	R-L	R-1	R-2	R-L	R-1	R-2	R-L
Abs	32.97	12.02	30.36	8.26	2.68	7.68	10.38	3.12	8.71	11.52	4.03	10.29
Ext	29.99	8.51	20.01	2.86	0.95	2.60	11.50	3.89	9.19	16.06	6.44	13.16

3.5.3 Comparing Summarization Results

To understand the performance of Text Summarization Models, we have given example summaries

Dataset: CNN/Daily Mail
Example Summary: 1. Police: Jose Antonio Acosta Hernandez says he targeted police, officials, rivals 2. Acosta has been a reputed top leader in La Linea, the armed branch of the Juarez drug cartel 3. Mexican authorities say they worked with the DEA to capture him 4. Federal police link him to some of the most notorious violence in Juarez
Start of Article: Mexico City (CNN) – A suspected leader of the Juarez drug cartel told authorities he had ordered the deaths of about 1,500 people, a Mexican federal police official said Sunday. Federal police detailed accusations against Jose Antonio Acosta Hernandez, known as "El Diego," a day after authorities announced his capture [...].
Generated Summary using Pointer-Generator: federal police detailed accusations against jose antonio acosta hernandez . he was one of the country 's most wanted criminals . he is accused of being a leader of the drug gang known as la linea .
Generated Summary using TextRank: mexico city (cnn) – a suspected leader of the juarez drug cartel told authorities he had ordered the deaths of about 1,500 people, a mexican federal police official said sunday.

Dataset: Gigaword
Example Summary: arcelormittal sees profit up
Start of Article: arcelormittal , the world 's largest steel company , reported on wednesday an annual net profit of 1.23 billion u.s. dollars [..]
Generated Summary using Pointer-Generator: the world 's largest steel company reported on wednesday an annual net profit of ## billion u.s. dollars in #### . ## , the world 's largest steel company , reported on wednesday an annual net profit of ## . ## , the world 's largest steel company , reported on wednesday .
Note: Since numbers are not present in vocabulary, they are represented by #.
Generated Summary using TextRank: arcelormittal , the world 's largest steel company , reported on wednesday an annual net profit of ## . ## billion u.s. dollars in #### .

Dataset: Free Press Journal
Example Summary: Watch: Karan Johar's mother Hiroo undergoes sanitation procedure after 2 staff members test positive for COVID-19
Start of Article: Filmmaker Karan Johar on Monday revealed that two members of his household staff have tested positive for COVID-19. Although, the filmmaker and his family tested negative for the novel coronavirus, Karan had said that they will observe self-isolation as precautionary measure. In a video that is doing rounds on the internet, his mother Hiroo Johar can [..]
Generated Summary using Pointer-Generator: karan johar revealed that two members of his household staff have tested positive for covid - 19 . although the filmmaker and his family tested negative for the novel coronavirus , karan had said that they will observe self - isolation as precautionary measure . in a video that is doing rounds on the internet .
Generated Summary using TextRank: filmmaker karan johar on monday revealed that two members of his household staff have tested positive for covid - 19 . in a video that is doing rounds on the internet , his mother hiroo johar can be seen undergoing a sanitation process.the video shows hiroo johar standing in front of a machine that sprays disinfectant .

Dataset: Hindustan Times
Example Summary: Mohun Bagan duo hail fathers contributions in I-League win
Start of Article: Young Mohun Bagan footballers Subha Ghosh and Sheikh Sahil have hailed their fathers contributions in helping them develop mental strength as the Kolkata giants won the I-League this year. My father is my first hero. He ignited the zeal within me to play football and follow my passion. He had a dream of playing football professionally but unfortunately, that couldnt happen. He always supported me, guided me whenever I felt low. Ill always admire him as my hero from the bottom of my heart, Subha told www.i-league.org . His three crucial goals helped [..]
Generated Summary using Pointer-Generator: mohun bagan and sheikh sahil hailed their fathers contributions in helping them develop mental strength . his three crucial goals helped mohun bagan in their surge to the i - league crown . he smacked six more in the state youth league to propel the green and maroons to the tournament final .
Generated Summary using TextRank: young mohun bagan footballers subha ghosh and sheikh sahil have hailed their fathers contributions in helping them develop mental strength as the kolkata giants won the i - league this year . while subha thanked his father for backing him since he laced his first football boots , his teammate sheikh sahil labelled his father as his luckiest charm .