News summarization

Dyuthi Vinod, Sai Prasanth Duvvuri, Sana Jahan, Shreya Singh

Khoury College of Computer and Information Science

Northeastern University

Boston, MA

April 29, 2021

Objectives and significance

Strategies for summarizing have been a subject of study for a long time, however, we are in a time in which more powerful tools are at our disposal than ever before. If we consider the huge amount of data involved in summarization (pre-trained models, neural networks, word embeddings) plus the need for an algorithm that can predict semantic variations, we naturally think of using NLP Techniques.In particular, given that the data is most likely unlabeled, reinforcement learning can be of aid.
Our work in this project was incremental, in the sense that we first spent some time researching and understanding the various algorithms, then we proceeded to implement them and verify their results. We used various summarization techniques on News Articles and compared the performance of each of these models.

Background

Summarization is the task of condensing a piece of text to a shorter version, reducing the size of the initial text while at the same time preserving key informational elements and the meaning of content. There are important applications for text summarization in various NLP related tasks such as text classification, question answering, legal texts summarization, news summarization, and headline generation. Moreover, the generation of summaries can be integrated into these systems as an intermediate stage which helps to reduce the length of the document. In general, there are two different approaches for automatic summarization: **extraction** and **abstraction**. We have decided to follow the extractive approach where sentences are directly picked up from the document based on a scoring function to form a coherent summary.This method works by identifying important sections of the text cropping out and stitching together portions of the content to produce a condensed version.

Our work

We have traversed through 4 different algorithms in order to compare and analyse the summaries created by each of them. The four algorithms explored were:

- Text Rank
- Word frequency
- Seq2Seq
- CNN

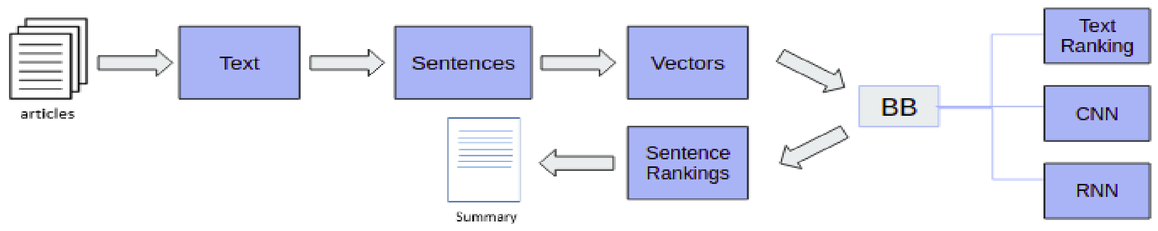


Figure 1: Flowchart for extractive Text Summarization

Methods Explored

Text summarization using word frequency

This is an extractive approach where summaries are printed with the help of word frequencies and by ranking sentences by giving them weights. Here, after the preprocessing stage, a dictionary containing the count of each word is created. This process is carried out to give weight to words that aim to determine whether the word has an effect or not. The weight of each sentence is calculated which is carried out to determine which sentence is considered to best represent the whole story. Finally, the top n sentences with the highest weight are outputted as the summary. The basic workflow of word frequency algorithm can be observed as follows:
Input document → sentences similarity → weight sentences → select sentences with higher rank.

Text summarization using Text Rank algorithm

TextRank algorithm goes through three major steps:
1. **Intermediate representation**: For any simplest summarizer, intermediate representation of the text to be summarized is done to identify the important content. We used word embeddings (Glove by Stanford) which gives vector representations for words.
2. **Score sentences**: After converting the input text document into intermediate representation, score (or weight) is assigned to each sentence of the given document to identify the important sentences. The weight of each sentence is determined by examining the different parameters used in the method. Here, cosine similarity was used to determine the weights of each sentence.
3. **Selecting summary sentences**: This is the final and last step to construct a relevant summary. Generally, it selects the best combination of sentences that gives the important information of the original text and the desired summary is the combination of top n important sentences. The optimal and best collection of sentences is selected to maximize overall importance, minimize redundancy, or maximize coherence in global selection procedure.We use PageRank algorithm to do the selection. In PageRank, important Web pages are linked with other important Web pages. Similarly, in our approach we assume that the important sentences are linked (similar) to other important sentences of the input document, where the sentences act as the node and the cosine score are the edges between those nodes.

Text summarization using Seq2Seq2

Seq2Seq2 models are used in problems where the length of input and output sentences vary. When given an input, the encoder-decoder seq2seq model first generates an encoded representation of the model, which is then passed to the decoder to generate the desired output. In this case, the input and output vectors need not be fixed in size.

Text summarization using CNN

This experimentation was to understand the performance of Convolutional Neural Networks on extractive text summarization. CNN was used to learn sentence features and thus rank the sentences. We have used Pre-trained word2vec embeddings to enhance the performance of the model. We have evaluated the idea on the DUC 2002 News article dataset. So far the experimentation has given results comparable to that of the other works for the extractive text summarization task.

Experimental Setup and Implementation

All the algorithms were implemented in Python. We discuss in each section the packages used to implement each model.
For evaluation we used libraries like rouge_metric, fuzzywuzzy, python-Levenshtein.
Our workflow was incremental and it can be divided into 5 different stages that we describe below:

Stage 1: Choosing right dataset

BBC News Summarization: The dataset from the BBC consisted of 2225 articles, each labeled under one of 5 categories: business, entertainment, politics, sport or tech.The dataset is broken into 1490 records for training and 735 for testing.
The BBC articles were also used to compare the fuzzy scores of the algorithms i.e TextRank and word-frequency, which was then used to compare their performance. For the actual summarization service,users could simply provide a huffingtonpost url and they would be able to get a summary for the same.
News Summary Data from Indian News sources: The news articles are from The Hindu, Indian times and Guardian and can be found at https://github.com/sunnysai12345/News_Summary. It has about 4515 samples and the average length of each article is 100 words and the summaries/headlines are trained on 15 words.

Stage 2: Preprocessing Dataset

After aggregating the data from multiple sources, we noticed that the text property has text which can be used to contains the stop words. In addition to that, there were symbols like “\n,\t” which can generate a whole different feature that we do not want as a part of our features. For example, “firstaid” and “first\taid” were same words, but together they would be considered as two different features which will increase the feature space even further. So, Text preprocessing was done to remove such special characters (“@,$”) and symbols(“\n,\t)
We also reduced the embedded URLs in the article to only Domain names and removed extra spaces from between words.
For the TextRank Algorithm, we also removed the stopwords from the articles.

Stage 3: Exploring TextRank and Sentence scoring using Word Frequency

We chose TextRank and word-frequency as two unsupervised algorithms for extractive summarization.
The reason we chose TextRank is because implementations tend to be lightweight and can run fast, even with limited memory resources. The algorithm was directed by adding semantic relations using word embeddings (Glove by Stanford). Another benefit of TextRank is that it is an unsupervised algorithm that does not require any training on datasets. Before we apply the algorithm, the text data goes through a few preprocessing steps, then the sentences are converted to sentence vectors which are then used in the algorithm to rank the sentences.

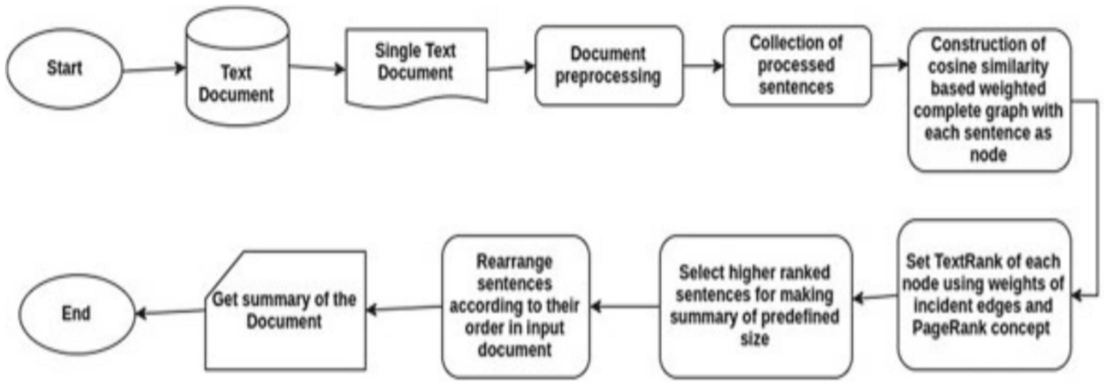
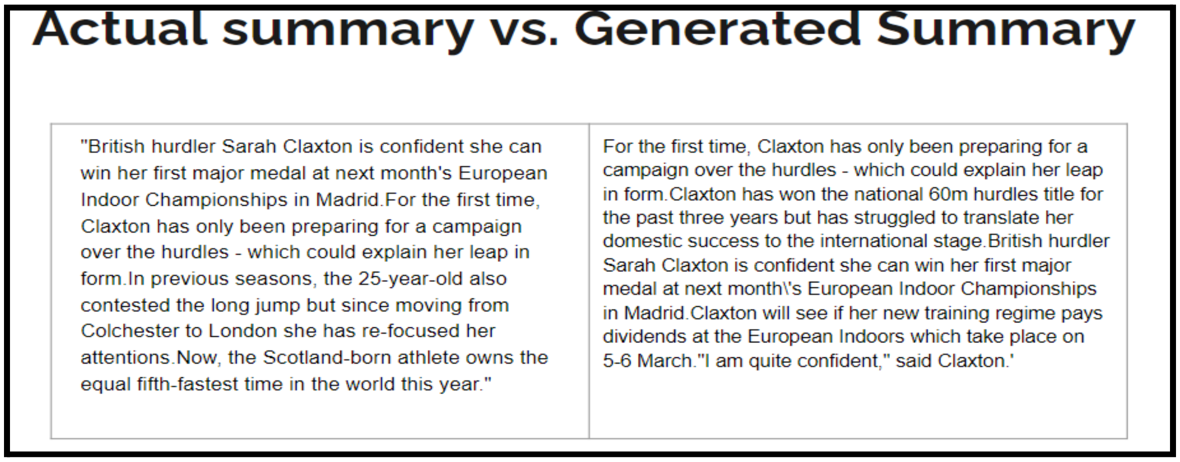

Start
Text Document
Single Text Document
Document preprocessing
Collection of processed sentences
Construction of cosine similarity based weighted complete graph with each sentence as node
Set TextRank of each node using weights of incident edges and PageRank concept
Select higher ranked sentences for making summary of predefined size
Rearrange sentences according to their order in input document
Get summary of the Document
End

Figure 2: Flowchart for TextRank Algorithm



Actual summary vs. Generated Summary

"British hurdler Sarah Claxton is confident she can win her first major medal at next month's European Indoor Championships in Madrid.For the first time, Claxton has only been preparing for a campaign over the hurdles - which could explain her leap in form.In previous seasons, the 25-year-old also contested the long jump but since moving from Colchester to London she has re-focused her attentions.Now, the Scotland-born athlete owns the equal fifth-fastest time in the world this year."	For the first time, Claxton has only been preparing for a campaign over the hurdles - which could explain her leap in form.Claxton has won the national 60m hurdles title for the past three years but has struggled to translate her domestic success to the international stage.British hurdler Sarah Claxton is confident she can win her first major medal at next month\'s European Indoor Championships in Madrid.Claxton will see if her new training regime pays dividends at the European Indoors which take place on 5-6 March."I am quite confident," said Claxton.'

Figure 3 : One of the generated results for the TextRank algorithm

The word frequency Model is also a simple model. With just a few steps like creating a word frequency table, tokenizing the sentences and scoring them, we can create summaries by giving in sentence thresholds.

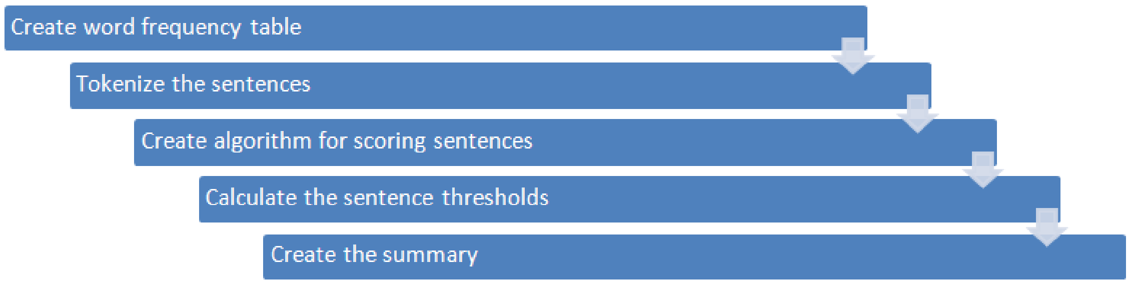
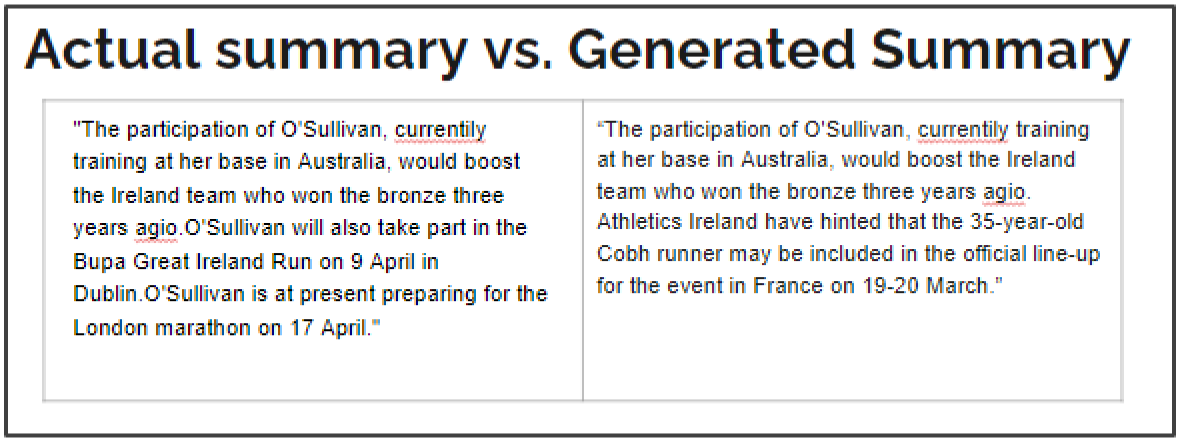

Create word frequency table
Tokenize the sentences
Create algorithm for scoring sentences
Calculate the sentence thresholds
Create the summary

Figure 4: Flow chart for Word Freq Sentence Scoring Model



Actual summary vs. Generated Summary

"The participation of O'Sullivan, currentily training at her base in Australia, would boost the Ireland team who won the bronze three years agio.O'Sullivan will also take part in the Bupa Great Ireland Run on 9 April in Dublin.O'Sullivan is at present preparing for the London marathon on 17 April."	"The participation of O'Sullivan, currentily training at her base in Australia, would boost the Ireland team who won the bronze three years agio. Athletics Ireland have hinted that the 35-year-old Cobh runner may be included in the official line-up for the event in France on 19-20 March."

Figure 5: One of the generated results for the Word Frequency algorithm

Stage 4: Exploring supervised algorithms

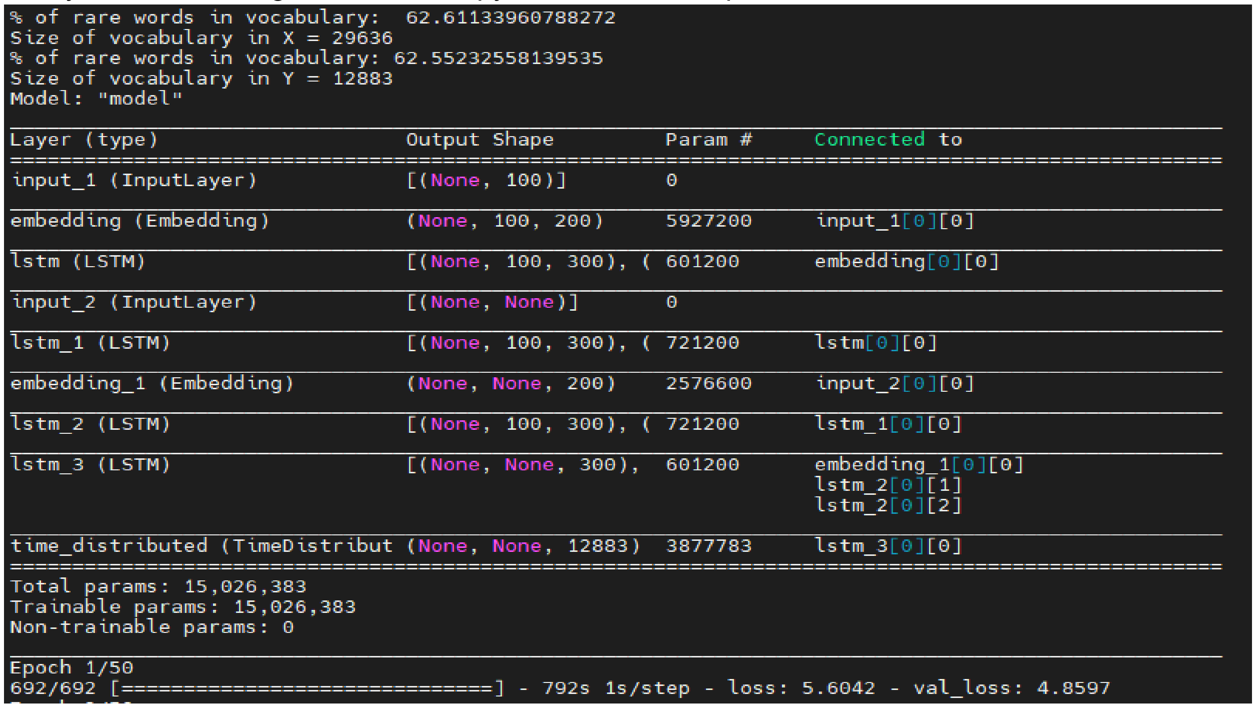
Seq2Seq Model:

The core idea to use Seq2Seq in this task is that if output depends on a sequence of inputs then we could build a neural network which gives importance to sequence information, which somehow retains and leverages the sequence information.In our case, our objective is to build a text summarizer where the input is a long sequence of words(the new story), and the output is a summary (the headline, which is a sequence as well). So, we can model this as a Many-to-Many Seq2Seq problem.

A many to many seq2seq model has two building blocks- Encoder and Decoder. The Encoder-Decoder architecture is mainly used to solve the sequence-to-sequence (Seq2Seq) problems where the input and output sequences are of different lengths.

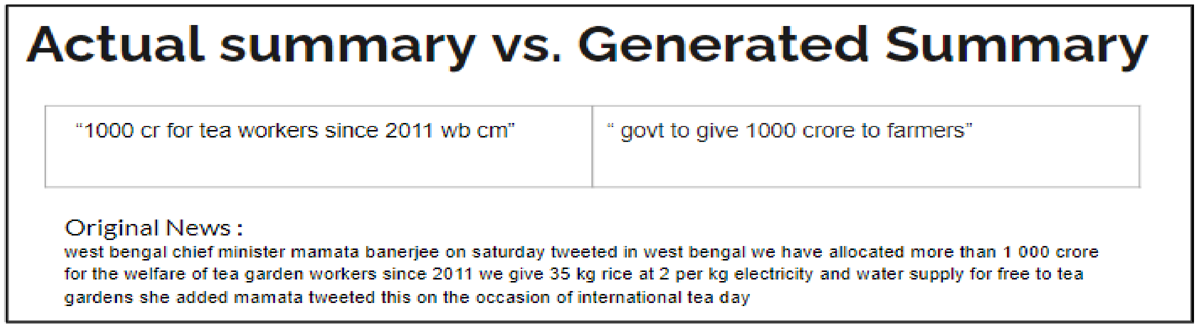
We used the LSTM networks in our encoder-decoder model.The input length that the encoder accepts is equal to the maximum text length,this is then given to an Embedding Layer of dimension (total number of words captured in the text vocabulary) x (number of nodes in an embedding layer).This is followed by three LSTM networks wherein each layer returns the LSTM output, as well as the hidden and cell states observed at the previous time steps.In the decoder, an embedding layer is defined followed by an LSTM network. The initial state of the LSTM network is the last hidden and cell states taken from the encoder. The output of the LSTM is given to a Dense layer wrapped in a TimeDistributed layer with an attached softmax activation function.

The algorithm was implemented using Python, in particular, we used the machine learning library like Keras, Regex, NLTK,Numpy, Tensorflow, Matplotlib,Pandas.



```
% of rare words in vocabulary:  62.61133960788272
Size of vocabulary in X = 29636
% of rare words in vocabulary: 62.55232558139535
Size of vocabulary in Y = 12883
Model: "model"
__________________________________________________________________________________________________
Layer (type)                    Output Shape         Param #     Connected to
==================================================================================================
input_1 (InputLayer)            [(None, 100)]        0
__________________________________________________________________________________________________
embedding (Embedding)           (None, 100, 200)     5927200     input_1[0][0]
__________________________________________________________________________________________________
lstm (LSTM)                     [(None, 100, 300), ( 601200      embedding[0][0]
__________________________________________________________________________________________________
input_2 (InputLayer)            [(None, None)]       0
__________________________________________________________________________________________________
lstm_1 (LSTM)                   [(None, 100, 300), ( 721200      lstm[0][0]
__________________________________________________________________________________________________
embedding_1 (Embedding)         (None, None, 200)    2576600     input_2[0][0]
__________________________________________________________________________________________________
lstm_2 (LSTM)                   [(None, 100, 300), ( 721200      lstm_1[0][0]
__________________________________________________________________________________________________
lstm_3 (LSTM)                   [(None, None, 300),  601200      embedding_1[0][0]
                                                                 lstm_2[0][1]
                                                                 lstm_2[0][2]
__________________________________________________________________________________________________
time_distributed (TimeDistribut (None, None, 12883)  3877783     lstm_3[0][0]
==================================================================================================
Total params: 15,026,383
Trainable params: 15,026,383
Non-trainable params: 0
__________________________________________________________________________________________________
Epoch 1/50
692/692 [==============================] - 792s 1s/step - loss: 5.6042 - val_loss: 4.8597
```

Figure 6: Training process of Seq2Seq Model



Actual summary vs. Generated Summary

"1000 cr for tea workers since 2011 wb cm"	" govt to give 1000 crore to farmers"

Original News :
west bengal chief minister mamata banerjee on saturday tweeted in west bengal we have allocated more than 1 000 crore for the welfare of tea garden workers since 2011 we give 35 kg rice at 2 per kg electricity and water supply for free to tea gardens she added mamata tweeted this on the occasion of international tea day

Figure 7: One of the generated results for the Seq2Seq algorithm

We did face some challenges in obtaining results with the Seq2Seq model. If the vocabulary size was small, the training was a faster rate, but gave unfavourable results. Increasing the vocabulary size even by 100 more words than what we finally tried with, caused our training to exceed the 8 hour limit which the discovery gpu provided.
Hence we decided to explore some pre-trained model which could implement text summarization.

Leveraging BERT as a classification model:

We decided to go ahead with BERTSUM, a simple variant of BERT, for extractive summarization from the paper Text Summarization with Pretrained Encoders (Liu et al., 2019).
The encoder is the pretrained BERT-base encoder from the masked language modeling task (Devlin et at., 2018).

This paper implemented the task of extractive summarization as a binary classification problem at the sentence level.They assigned each sentence a label y in {0,1} indicating whether the sentence should be included in the final summary.
After getting the vector representation of each sentence,they have used a simple feed forward layer as a classifier to return a score for each sentence. In the paper, the author experimented with a simple linear classifier, a Recurrent Neural Network and a small Transformer model with 3 layers. The Transformer classifier yields the best results, showing that inter-sentence interactions through self-attention mechanisms are important in selecting the most important sentences.
So in the encoder, we learn the interactions among tokens in our document while in the summarization classifier, we learn the interactions among sentences. The implementation is done using Pytorch and Transformer packages. This implementation was done simply as an experiment to see how the pre-trained model performed on longer sentences and articles. All code credits belong to.

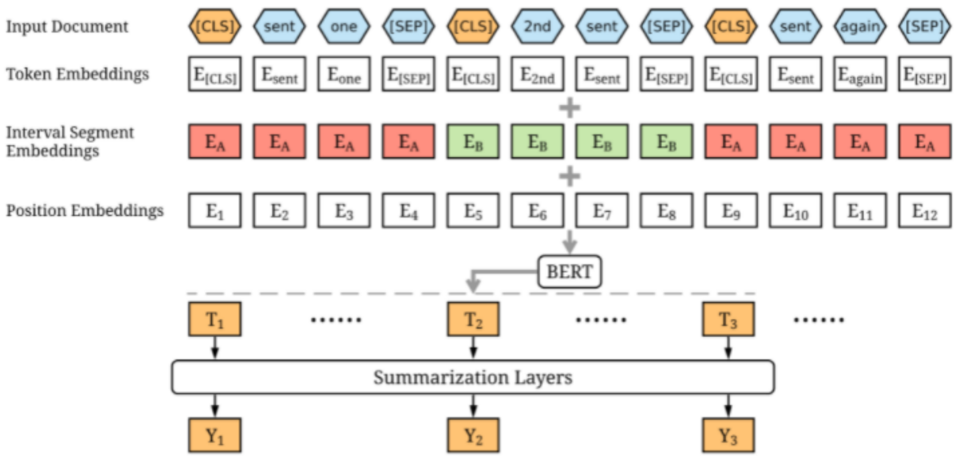
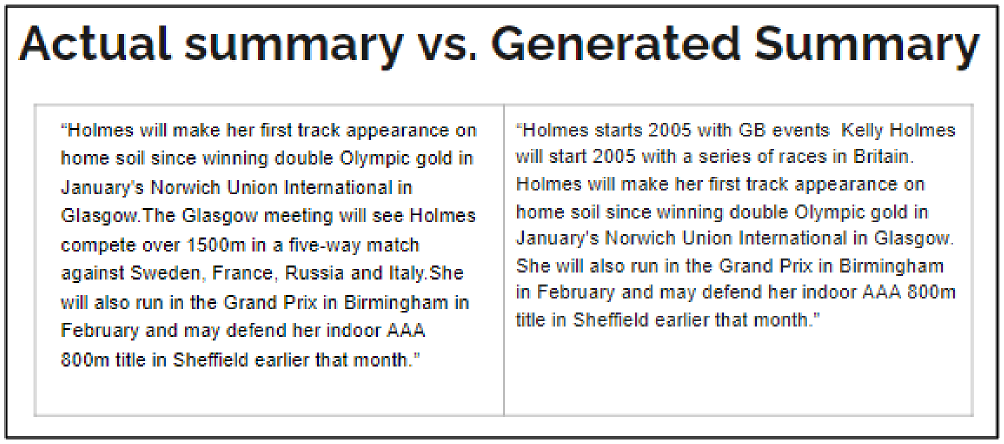

Input Document
[CLS]
sent
one
[SEP]
[CLS]
2nd
sent
[SEP]
[CLS]
sent
again
[SEP]
Token Embeddings
$E_{[CLS]}$ E_{sent} E_{one} $E_{[SEP]}$ $E_{[CLS]}$ E_{2nd} E_{sent} $E_{[SEP]}$ $E_{[CLS]}$ E_{sent} E_{again} $E_{[SEP]}$
Interval Segment Embeddings
E_A E_A E_A E_A E_B E_B E_B E_B E_A E_A E_A E_A
Position Embeddings
E_1 E_2 E_3 E_4 E_5 E_6 E_7 E_8 E_9 E_{10} E_{11} E_{12}
BERT
T_1 T_2 T_3
Summarization Layers
Y_1 Y_2 Y_3

Figure 8: Architecture of BERTSUM model

We added evaluation metrics to the above experiment to check with BBC Dataset to see how the BERT encoder performs.The results will be discussed in the next section.



Actual summary vs. Generated Summary

"Holmes will make her first track appearance on home soil since winning double Olympic gold in January's Norwich Union International in Glasgow.The Glasgow meeting will see Holmes compete over 1500m in a five-way match against Sweden, France, Russia and Italy.She will also run in the Grand Prix in Birmingham in February and may defend her indoor AAA 800m title in Sheffield earlier that month."	"Holmes starts 2005 with GB events Kelly Holmes will start 2005 with a series of races in Britain. Holmes will make her first track appearance on home soil since winning double Olympic gold in January's Norwich Union International in Glasgow. She will also run in the Grand Prix in Birmingham in February and may defend her indoor AAA 800m title in Sheffield earlier that month."

Figure 9: One of the generated results for the BERTSUM model

CNN Model :

Architecture:

The convolution operation in our model is 1-dimensional, convoluted with a filter w of size (mxk) over a vector x (i:i+m−1) representing a window of size m starting from the ith word, we obtain features for these words in the window in the corresponding feature map. The max-pooling operation in our model is to obtain a single feature corresponding to the complete filter. We have used multiple filters (conv kernels) as part of the convolution operation giving us the many feature vectors. In this experimentation we have fixed filter width to 3 and used 500 filters. Max-pooling operation encapsulates the most important feature and leads to a fixed-length feature vector regardless of variable sentence lengths. To avoid overfitting, we have used dropout regularization with a probability of 0.4. The model parameters are fine-tuned via SGD using Adadelta update.

Task:

Motivation behind the experimentation was to generate a brief summary of the News articles using a CNN model. Therefore, the CNN model learns feature representation of each sentence and assigns a salience score to each sentence. By doing so, the CNN model has learnt the important word features and sentence features. Training dataset has documents and their summary. We initially preprocess the text by removing the stopwords, punctuations and segregate them into sentences. We then assign the saliency score to each of these sentences. The function for calculating saliency score is given below.

$$s = \alpha R_1 + (1-\alpha)R_2$$

We have used Rouge-1 (R1) and Rouge-2 (R2) metric to calculate the saliency score. The alpha in the given function is a hyper-parameter (between 0 to 1) and sets the weight of R1 and R2. In out current study, we have experimented alpha from 0 (more weight to R2) to 0.4 and we found alpha at 0.3 to be a right balance. R1 and R2 are obtained by comparing each preprocessed sentence to its corresponding summary. We train the CNN, by minimizing the cross-entropy.

$$CE = -sln(\widehat{s}) - (1-s)ln(1-\widehat{s})$$

Finally, to predict the news articles we used the "newspaper" python library, to scrape the articles from the Huffpost and the BBC websites. We then preprocessed it the same way we preprocessed the training data and then used the summarization CNN model to obtain its salience score. Then, sentences in each article are ranked according to their salience scores and the most informative ones are predicted as the summary.

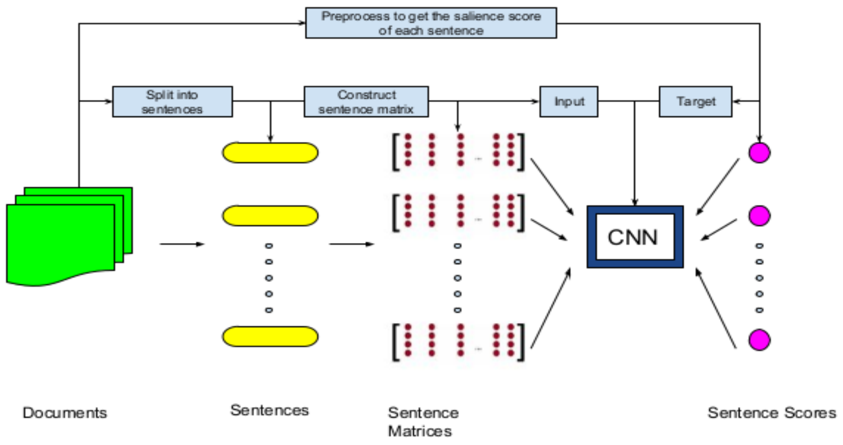
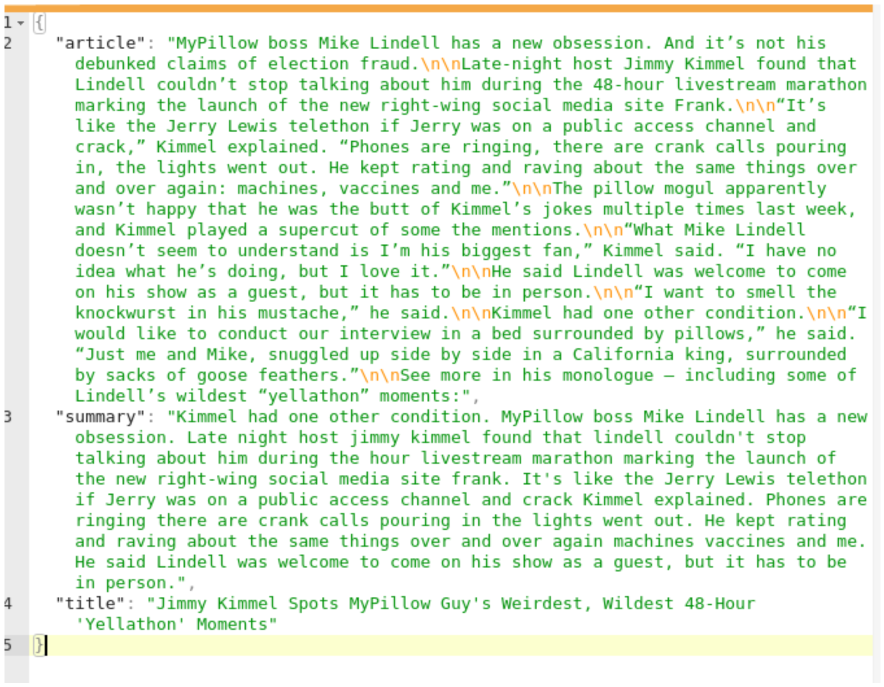


Figure 10: Architecture of CNN model



```
{
  "article": "MyPillow boss Mike Lindell has a new obsession. And it’s not his
    debunked claims of election fraud.\n\nLate-night host Jimmy Kimmel found that
    Lindell couldn’t stop talking about him during the 48-hour livestream marathon
    marking the launch of the new right-wing social media site Frank.\n\n“It’s
    like the Jerry Lewis telethon if Jerry was on a public access channel and
    crack,” Kimmel explained. “Phones are ringing, there are crank calls pouring
    in, the lights went out. He kept rating and raving about the same things over
    and over again: machines, vaccines and me.”\n\nThe pillow mogul apparently
    wasn’t happy that he was the butt of Kimmel’s jokes multiple times last week,
    and Kimmel played a supercut of some the mentions.\n\n“What Mike Lindell
    doesn’t seem to understand is I’m his biggest fan,” Kimmel said. “I have no
    idea what he’s doing, but I love it.”\n\nHe said Lindell was welcome to come
    on his show as a guest, but it has to be in person.\n\n“I want to smell the
    knockwurst in his mustache,” he said.\n\nKimmel had one other condition.\n\n“I
    would like to conduct our interview in a bed surrounded by pillows,” he said.
    “Just me and Mike, snuggled up side by side in a California king, surrounded
    by sacks of goose feathers.”\n\nSee more in his monologue — including some of
    Lindell’s wildest “yellathon” moments:",
  "summary": "Kimmel had one other condition. MyPillow boss Mike Lindell has a new
    obsession. Late night host jimmy kimmel found that lindell couldn't stop
    talking about him during the hour livestream marathon marking the launch of
    the new right-wing social media site frank. It's like the Jerry Lewis telethon
    if Jerry was on a public access channel and crack Kimmel explained. Phones are
    ringing there are crank calls pouring in the lights went out. He kept rating
    and raving about the same things over and over again machines vaccines and me.
    He said Lindell was welcome to come on his show as a guest, but it has to be
    in person.",
  "title": "Jimmy Kimmel Spots MyPillow Guy's Weirdest, Wildest 48-Hour
    'Yellathon' Moments"
}
```

Figure 11 : Rouge Evaluation for CNN Model on Huffpost Articles

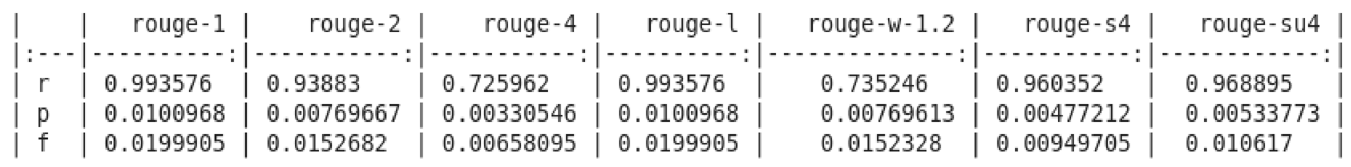
Stage 5: Evaluating performance

The performance of the model was evaluated using :
Rouge metric: ROUGE stands for Recall-Oriented Understudy for Gisting Evaluation. It works by comparing an automatically produced summary or translation against a set of reference summaries (typically human-produced). It is used to calculate the recall, precision and f1-score of the generated summary against the original summary.
Levenshtein distance: It is a text similarity measure that compares two words and returns a numeric value representing the distance between them. The distance reflects the total number of single-character edits required to transform one word into another. We used the fuzzywuzzy library provided in python to measure this value.
The higher the score reported by the fuzzy matching, the more similar the predicted summaries and actual summaries are.

Results

We will present the activities done and results obtained for each of the stages described in the previous section.

Rouge Metric and Levenshtein distance for TextRank Model:



	rouge-1	rouge-2	rouge-4	rouge-l	rouge-w-1.2	rouge-s4	rouge-su4
r	0.993576	0.93883	0.725962	0.993576	0.735246	0.960352	0.968895
p	0.0100968	0.00769667	0.00330546	0.0100968	0.00769613	0.00477212	0.00533773
f	0.0199905	0.0152682	0.00658095	0.0199905	0.0152328	0.00949705	0.010617

Figure 12: Rouge Evaluation for TextRank Model

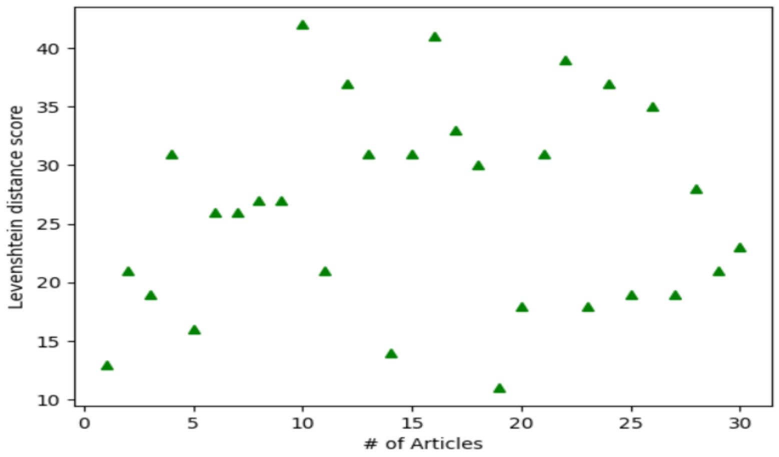
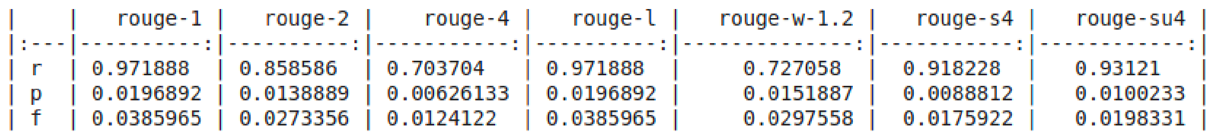


Figure 13: Levenshtein score of 30 BBC articles under politics category.

Rouge Metric and Levenshtein distance for word frequency model



```
|     |   rouge-1 |   rouge-2 |    rouge-4 |   rouge-l |    rouge-w-1.2 |    rouge-s4 |    rouge-su4 |
|:----|----------:|----------:|-----------:|----------:|---------------:|------------:|-------------:|
| r   | 0.971888  | 0.858586  | 0.703704   | 0.971888  |      0.727058  |  0.918228   |   0.93121    |
| p   | 0.0196892 | 0.0138889 | 0.00626133 | 0.0196892 |      0.0151887 |  0.0088812  |   0.0100233  |
| f   | 0.0385965 | 0.0273356 | 0.0124122  | 0.0385965 |      0.0297558 |  0.0175922  |   0.0198331  |
```

Figure 14: Rouge Evaluation of Word Freq Model

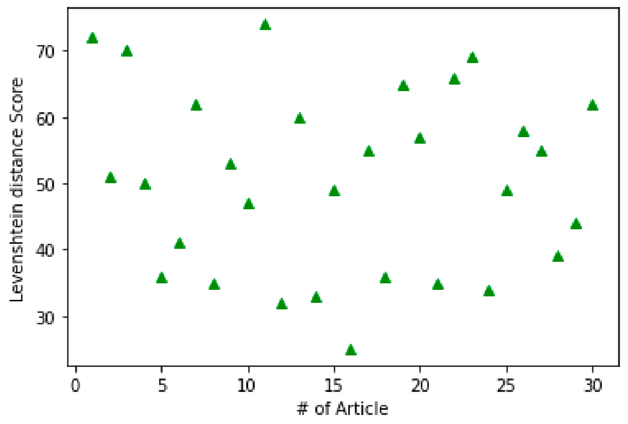
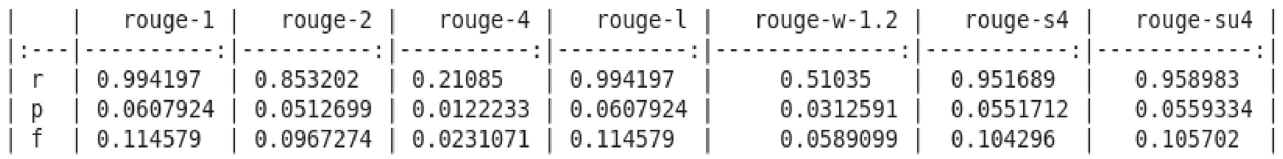


Figure 15: Levenshtein score of 30 BBC articles

Rouge Metric and Levenshtein distance for Seq2Seq Model:



```
|     |   rouge-1 |   rouge-2 |   rouge-4 |   rouge-l |    rouge-w-1.2 |    rouge-s4 |    rouge-su4 |
|:----|----------:|----------:|----------:|----------:|---------------:|------------:|-------------:|
| r   | 0.994197  | 0.853202  | 0.21085   | 0.994197  |      0.51035   |  0.951689   |   0.958983   |
| p   | 0.0607924 | 0.0512699 | 0.0122233 | 0.0607924 |      0.0312591 |  0.0551712  |   0.0559334  |
| f   | 0.114579  | 0.0967274 | 0.0231071 | 0.114579  |      0.0589099 |  0.104296   |   0.105702   |
```

Figure 16: Rouge Evaluation for Seq2SeqModel

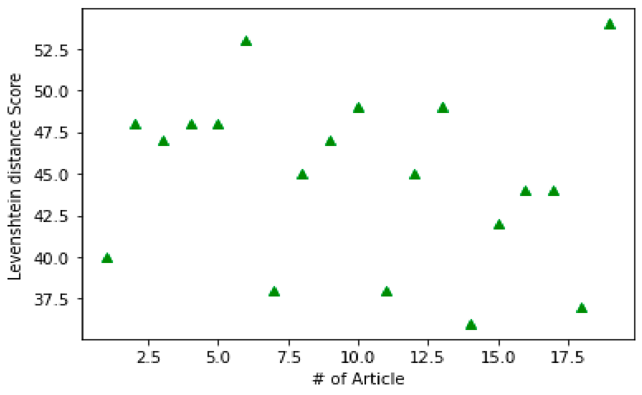


Figure 17: Levenshtein score of 20 articles using Seq2Seq

Rouge Metric and Levenshtein distance for BERTSUM Model:

```
|    |   rouge-1 |   rouge-2 |    rouge-4 |   rouge-l |   rouge-w-1.2 |   rouge-s4 |   rouge-su4 |
|:---|----------:|----------:|-----------:|----------:|--------------:|-----------:|------------:|
| r  | 0.981873  | 0.891386  | 0.738255   | 0.981873  |      0.732385 | 0.910354   |  0.929178   |
| p  | 0.016224  | 0.0119191 | 0.00554435 | 0.016224  |     0.0123749 | 0.00726815 |  0.00825725 |
| f  | 0.0319206 | 0.0235236 | 0.0110061  | 0.0319206 |     0.0243386 | 0.0144212  |  0.016369   |
```

Figure 18: Rouge Evaluation for BERTSUM model

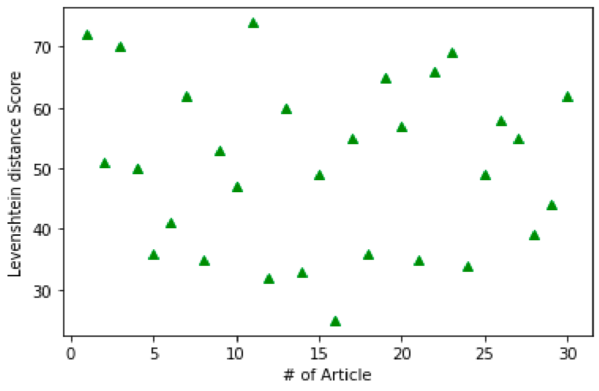


Figure 19: Levenshtein score of 30 BBC articles using BERTSUM

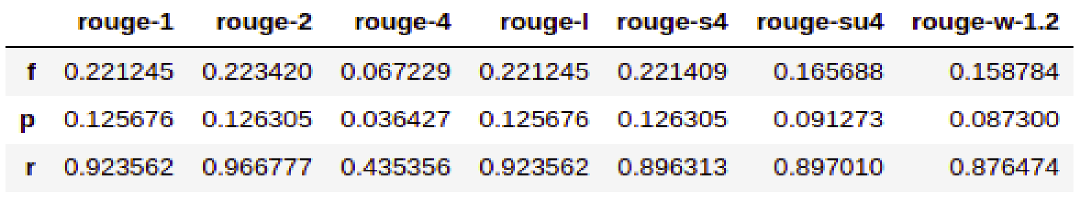
Rouge Metric for CNN Model:

1) Rouge Metric for the DUC2002 articles

	rouge-1	rouge-2	rouge-4	rouge-l	rouge-s4	rouge-su4	rouge-w-1.2
f	0.266910	0.232356	0.112442	0.266910	0.239447	0.243708	0.124315
p	0.154743	0.134554	0.064963	0.154743	0.138339	0.140855	0.072150
r	0.970100	0.850671	0.417808	0.970100	0.889726	0.903299	0.448827

Figure 20: Rouge Evaluation for CNN Model on Huffpost Articles

2) Rouge Metric for the BBC articles



	rouge-1	rouge-2	rouge-4	rouge-l	rouge-s4	rouge-su4	rouge-w-1.2
f	0.221245	0.223420	0.067229	0.221245	0.221409	0.165688	0.158784
p	0.125676	0.126305	0.036427	0.125676	0.126305	0.091273	0.087300
r	0.923562	0.966777	0.435356	0.923562	0.896313	0.897010	0.876474

Figure 21: Rouge Evaluation for CNN Model on BBC Dataset

User Interface (UI)

We created a UI for our service.

Swagger

First, we created a *Swagger* API interface. This can be found in the following files:

```
news_api.py
news_api.yaml
```

In order to run the *Swagger* API, please run the following commands from the APIs folder in your terminal:

```
python news_api.py
```

Then, in your browser, navigate to:

```
http://localhost:8080/ui/
```

`/summarize` takes two parameters:

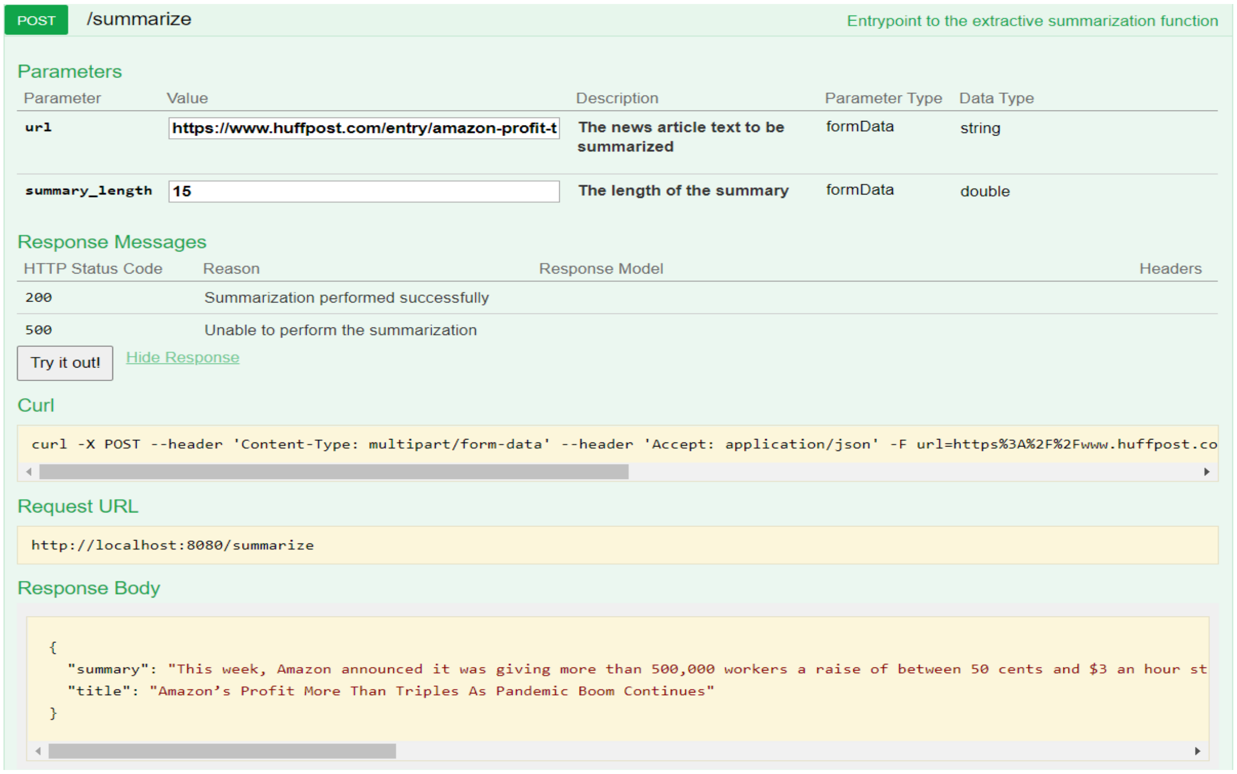
1. **Article URL:** Only Huffington Post news article URLs
2. **Summary_length:** Integer value within range 0-100

The article url is used to scrape the article text behind the scenes and the summary length value is used as a percentage. For instance, when summary_length is 10 then only the top 10% ranked sentences of the total sentences present in the article are returned as a summary, along with the title of the article.

`/wf_summarize` takes two parameters:

1. **Article URL:** Only Huffington Post news article URLs
2. **Summary_length:** Integer value within total length of the article sentences

The article url is used to scrape the article text behind the scenes and the summary length value is used as the number of sentences we want in the summary. For instance: when summary_length is 10, then only the top 10 ranked sentences of the total sentences present in the article are returned as a summary along with the title of the article.



POST /summarize — Entrypoint to the extractive summarization function

Parameters

Parameter	Value	Description	Parameter Type	Data Type
url	https://www.huffpost.com/entry/amazon-profit-t	The news article text to be summarized	formData	string
summary_length	15	The length of the summary	formData	double

Response Messages

HTTP Status Code	Reason	Response Model	Headers
200	Summarization performed successfully		
500	Unable to perform the summarization		

Try it out! Hide Response

Curl

```
curl -X POST --header 'Content-Type: multipart/form-data' --header 'Accept: application/json' -F url=https%3A%2F%2Fwww.huffpost.co
```

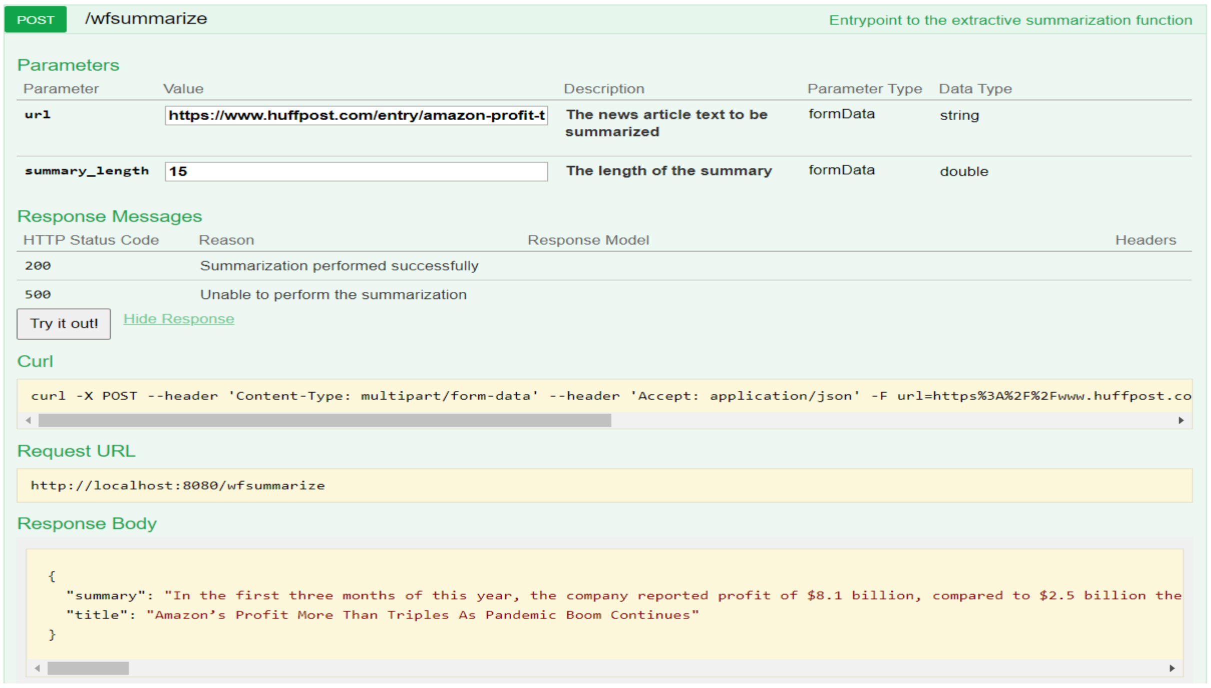
Request URL

```
http://localhost:8080/summarize
```

Response Body

```
{
  "summary": "This week, Amazon announced it was giving more than 500,000 workers a raise of between 50 cents and $3 an hour st
  "title": "Amazon’s Profit More Than Triples As Pandemic Boom Continues"
}
```

Fig 22: Swagger endpoint for the TextRank summarization



POST /wfsummarize — Entrypoint to the extractive summarization function

Parameters

Parameter	Value	Description	Parameter Type	Data Type
url	https://www.huffpost.com/entry/amazon-profit-t	The news article text to be summarized	formData	string
summary_length	15	The length of the summary	formData	double

Response Messages

HTTP Status Code	Reason	Response Model	Headers
200	Summarization performed successfully		
500	Unable to perform the summarization		

Try it out! Hide Response

Curl

```
curl -X POST --header 'Content-Type: multipart/form-data' --header 'Accept: application/json' -F url=https%3A%2F%2Fwww.huffpost.co
```

Request URL

```
http://localhost:8080/wfsummarize
```

Response Body

```
{
  "summary": "In the first three months of this year, the company reported profit of $8.1 billion, compared to $2.5 billion the
  "title": "Amazon’s Profit More Than Triples As Pandemic Boom Continues"
}
```

Fig 23: Swagger endpoint for word frequency summarization

Conclusion and Future work

Strategies for summarizing have been a subject for study for a long time and it was interesting to explore how unsupervised algorithms can be used for extractive summarization giving comparable results to high computational neural network algorithms like CNN and Seq2Seq with cheap computational low cost fast-speed algorithm like TextRank and word-frequency for decent outputs.
Of course, with better dataset CNN and RNN based algorithms may outperform with better datasets and better GPU, and that is something that we want to explore as our future work.
Other than these, we can also explore into the abstractive approach in order to print out the summaries for expanding the scope of the project.

References

1. C. Prasad, J. S. Kallimani, D. Harekal and N. Sharma, "Automatic Text Summarization Model using Seq2Seq Technique," 2020 Fourth International Conference on I-SMAC (IoT in Social, Mobile, Analytics and Cloud) (I-SMAC), 2020, pp. 599-604, doi: 10.1109/I-SMAC49090.2020.9243572.
2. Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding.
3. M Zhong, P Liu, Y Chen, D Wang, X Qiu, X Huang - arXiv preprint arXiv:2004.08795, 2020 : Extractive summarization as text matching
4. Stepwise Extractive Summarization and Planning with Structured Transformers : S Narayan, J Maynez, J Adamek, D Pighin, B Bratanič… - arXiv preprint arXiv:2010.02744, 2020
5. Y. Zhang, J. E. Meng and M. Pratama, "Extractive document summarization based on convolutional neural networks," IECON 2016 - 42nd Annual Conference of the IEEE Industrial Electronics Society, 2016, pp. 918-922, doi: 10.1109/IECON.2016.7793761.
6. https://towardsdatascience.com/how-to-create-simple-news-summarization-from-scratch-using-python-83adc33a409c/
7. https://blog.paperspace.com/introduction-to-seq2seq-models/
8. https://blog.paperspace.com/implement-seq2seq-for-text-summarization-keras/
9. https://blog.floydhub.com/gentle-introduction-to-text-summarization-in-machine-learning/
10. https://newspaper.readthedocs.io/en/latest/
11. https://github.com/je-suis-tm/web-scraping
12. https://redditsearch.io/
13. https://praw.readthedocs.io/en/latest/
14. https://www.analyticsvidhya.com/blog/2017/06/word-embeddings-count-word2veec/
15. https://www.researchgate.net/profile/Apurba-Sarkar-4/publication/327136473_Graph-Based_Text_Summarization_Using_Modified_TextRank/links/5bceacf392851c1816ba50cb/Graph-Based-Text-Summarization-Using-Modified-TextRank.pdf
16. https://www.aclweb.org/anthology/W04-1013.pdf
17. https://pypi.org/project/fuzzywuzzy/
18. https://www.kaggle.com/rmisra/news-category-datase

Link to codebase