TEXT SUMMARIZER

PROJECT GUIDE PROF. ANISH ABRAHAM

Project idea presented by:
Gokulkrishna Gopakumar
Poornima Nair T
Muhammed Sabik K
Sree Kumar Harith Nair

Introduction

- A summary of an article is a shortened version containing key points in order to condense it into few points and to do so in words used in the article itself.
- With such a big amount of data circulating in the digital space, there is need to develop an efficient tool that can automatically reduce much of this text data to shorter, focused summaries that capture the salient details, both so we can navigate it more effectively as well as check whether the larger documents contain the information that we are looking for.
- There are many reasons and uses for a summary of a larger document, from shortening a lengthy news article to simplifying a complicated study material, a text summarizer is an efficient and useful tool to all target audiences.

Motivation

- Since manual text summarization is a time expensive and generally laborious task, especially when the number of documents is considerably high, the automatization of the task is gaining increasing popularity and therefore constitutes a strong motivation for this project.
- Unlike pre-existing methods, by combining BERT and TextRank algorithm we can achieve faster and efficient results.

Objective

- To summarize effectively, long and complex pieces of articles. NLP algorithms, namely BERT and TextRank, are going to be used in combination, for this project, to obtain better summarising capabilities and an extracting algorithm will be used to extract and display important keywords from the text document.

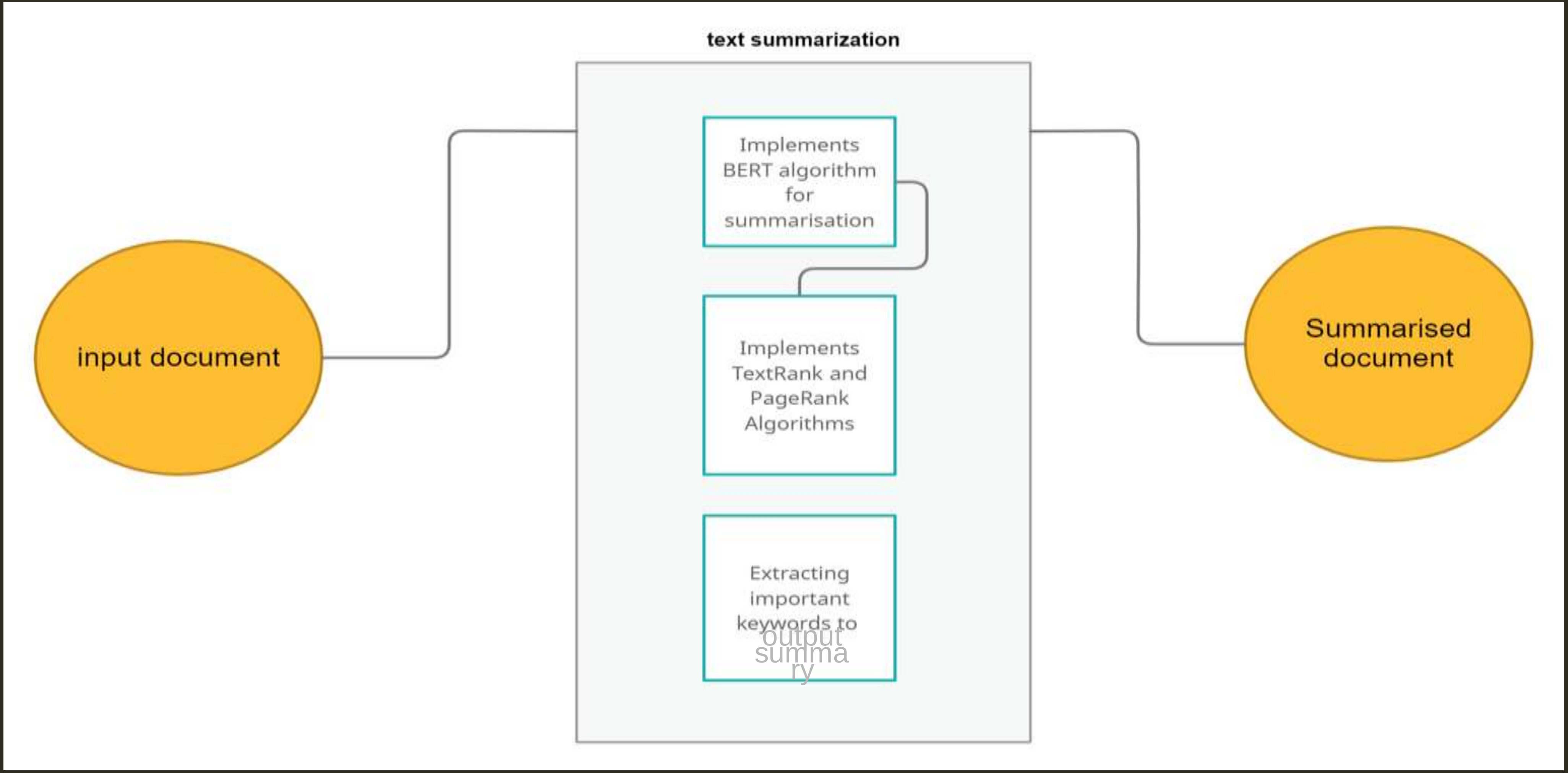
TEXT SUMMARISATION

Text summarization is the concept of employing a machine to condense a document or a set of documents into brief paragraphs or statements using mathematical methods. NLP broadly classifies text summarization into 2 groups.

- Extractive text summarization: here, the model summarizes long documents and represents them in smaller simpler sentences.
- Abstractive text summarization: the model has to produce a summary based on a topic without prior content provided.

We will understand and implement the first category here.

PROPOSED WORK


text summarization
input document
Implements BERT algorithm for summarisation
Implements TextRank and PageRank Algorithms
Extracting important keywords to
output summary
Summarised document

Why BERT?

- designed to pre-train deep bidirectional representations from unlabeled text by jointly conditioning on both left and right context in all layers.

- As a result, the pre-trained BERT model can be fine-tuned with just one additional output layer to create state-of-the-art models for a wide range of tasks, such as question answering and language inference, without substantial task-specific architecture modifications.

UNDERSTANDING BERT

BERT vs OpenAI GPT & ELMo

BERT uses a bidirectional Transformer.

OpenAI GPT uses a left-to-right Transformer.

ELMo uses the concatenation of independently trained left-to-right and right-to-left LSTMs to generate features for downstream tasks.

Among the three, only BERT representations are jointly conditioned on both left and right context in all layers. In addition to the architecture differences, BERT and OpenAI GPT are fine-tuning approaches, while ELMo is a feature-based approach.

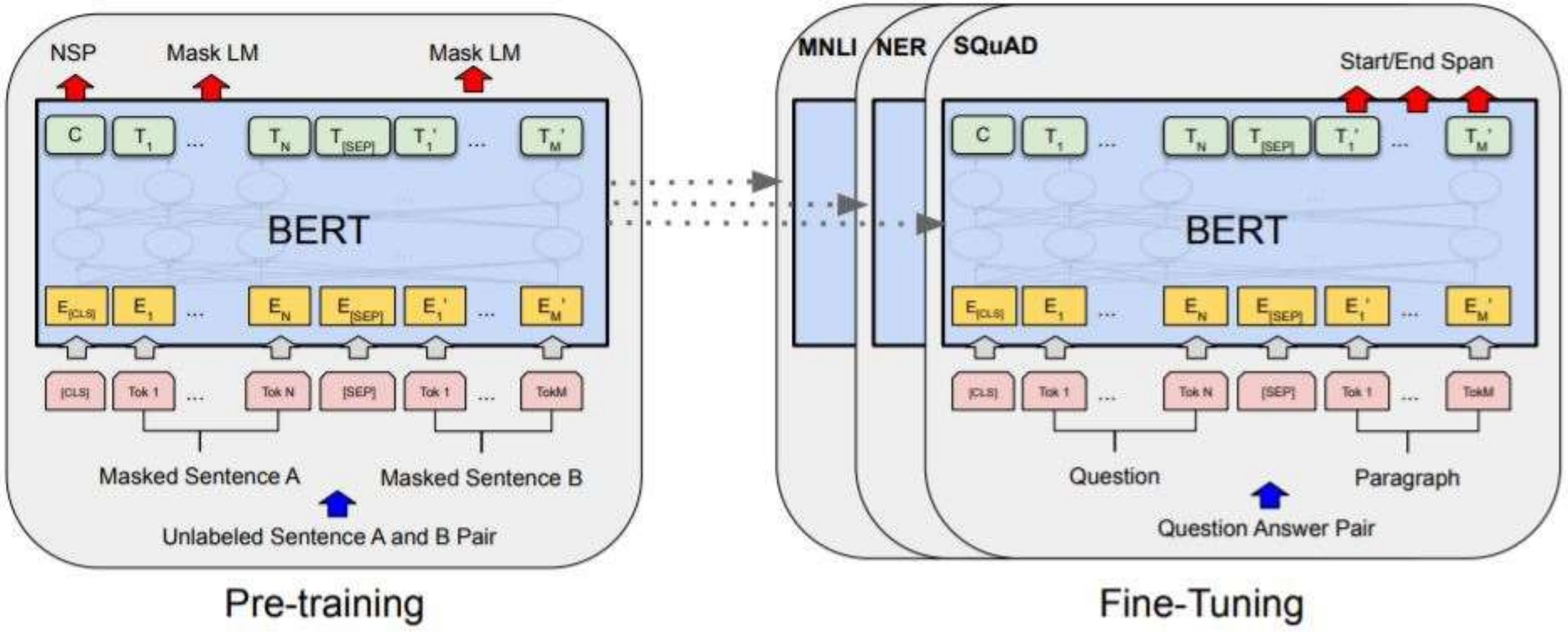


Figure 1: Overall pre-training and fine-tuning procedures for BERT. Apart from output layers, the same architectures are used in both pre-training and fine-tuning. The same pre-trained model parameters are used to initialize models for different down-stream tasks. During fine-tuning, all parameters are fine-tuned. [CLS] is a special symbol added in front of every input example, and [SEP] is a special separator token (e.g. separating questions/answers).

- In order to train a deep bidirectional representation, we simply mask some percentage of the input tokens at random, and then predict those masked tokens. We refer to this procedure as a “masked LM” (MLM)

- In order to train a model that understands sentence relationships, we pre-train for a binarized next sentence prediction task that can be trivially generated from any monolingual corpus.

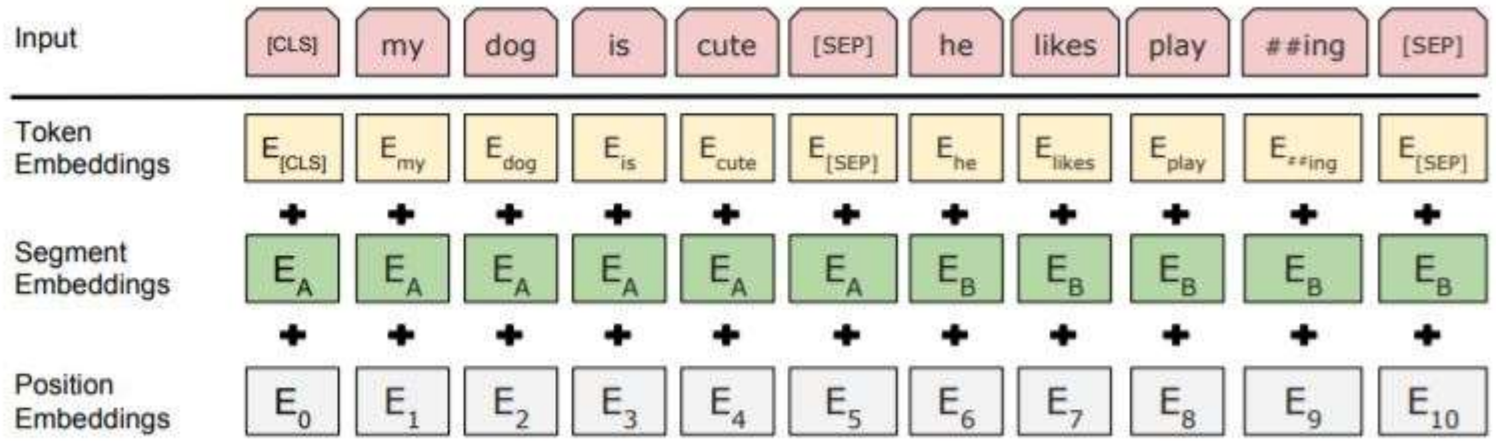

Input
[CLS]
my
dog
is
cute
[SEP]
he
likes
play
##ing
[SEP]
Token Embeddings
$E_{[CLS]}$ E_{my} E_{dog} E_{is} E_{cute} $E_{[SEP]}$ E_{he} E_{likes} E_{play} $E_{\#\#ing}$ $E_{[SEP]}$
Segment Embeddings
E_A E_A E_A E_A E_A E_A E_B E_B E_B E_B E_B
Position Embeddings
E_0 E_1 E_2 E_3 E_4 E_5 E_6 E_7 E_8 E_9 E_{10}

Figure 2: BERT input representation. The input embeddings are the sum of the token embeddings, the segmentation embeddings and the position embeddings.

- The first token of every sequence is always a special classification token ([CLS])
- We differentiate the sentences in two ways. First, we separate them with a special token ([SEP]). Second, we add a learned embedding to every token indicating whether it belongs to sentence A or sentence B.

- To use BERT for extractive summarization, we require it to output the representation for each sentence. However, since BERT is trained as a masked-language model, the output vectors are grounded to tokens instead of sentences.

- In order to represent individual sentences, we insert external [CLS] tokens at the start of each sentence, and each [CLS] symbol collects features for the sentence preceding it.

- Although BERT has segmentation embeddings for indicating different sentences, it only has two labels (sentence A or sentence B), instead of multiple sentences as in extractive summarization. Therefore, we modify the input sequence and embeddings of BERT to make it possible for extracting summaries.

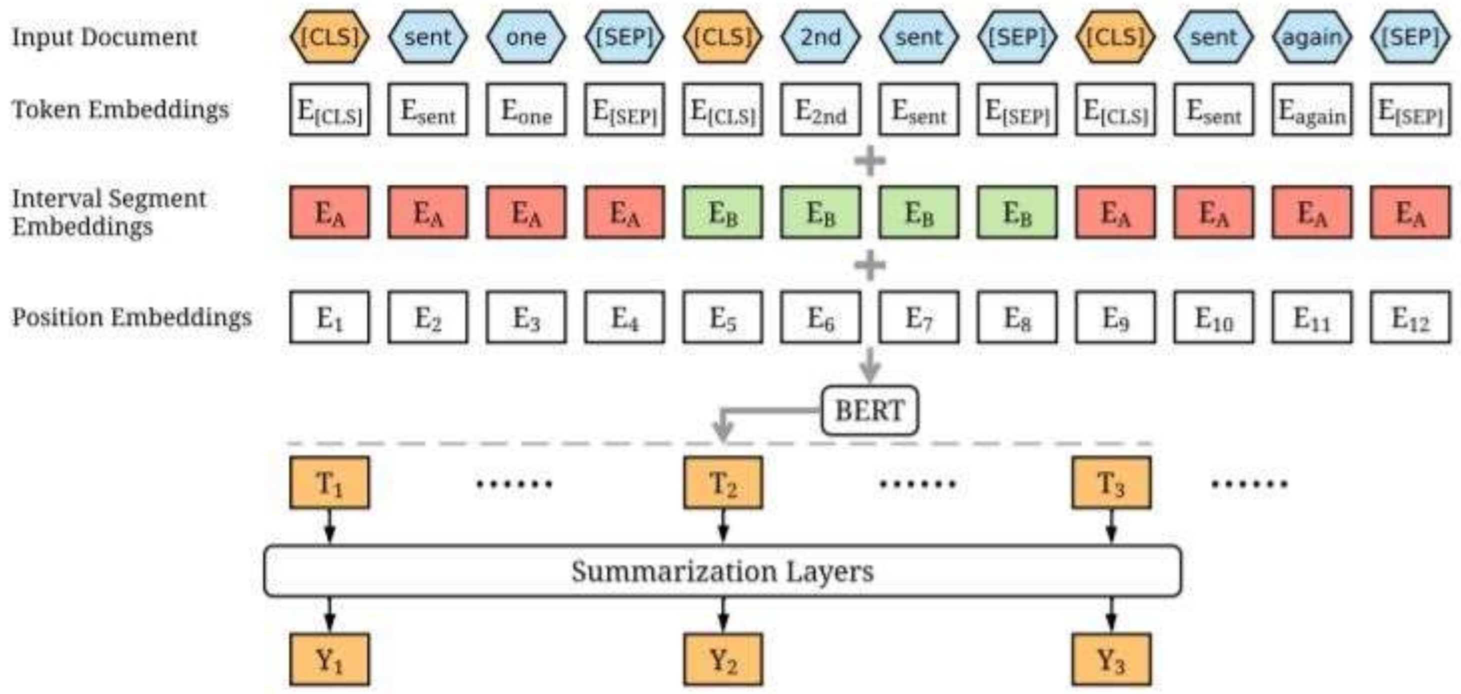


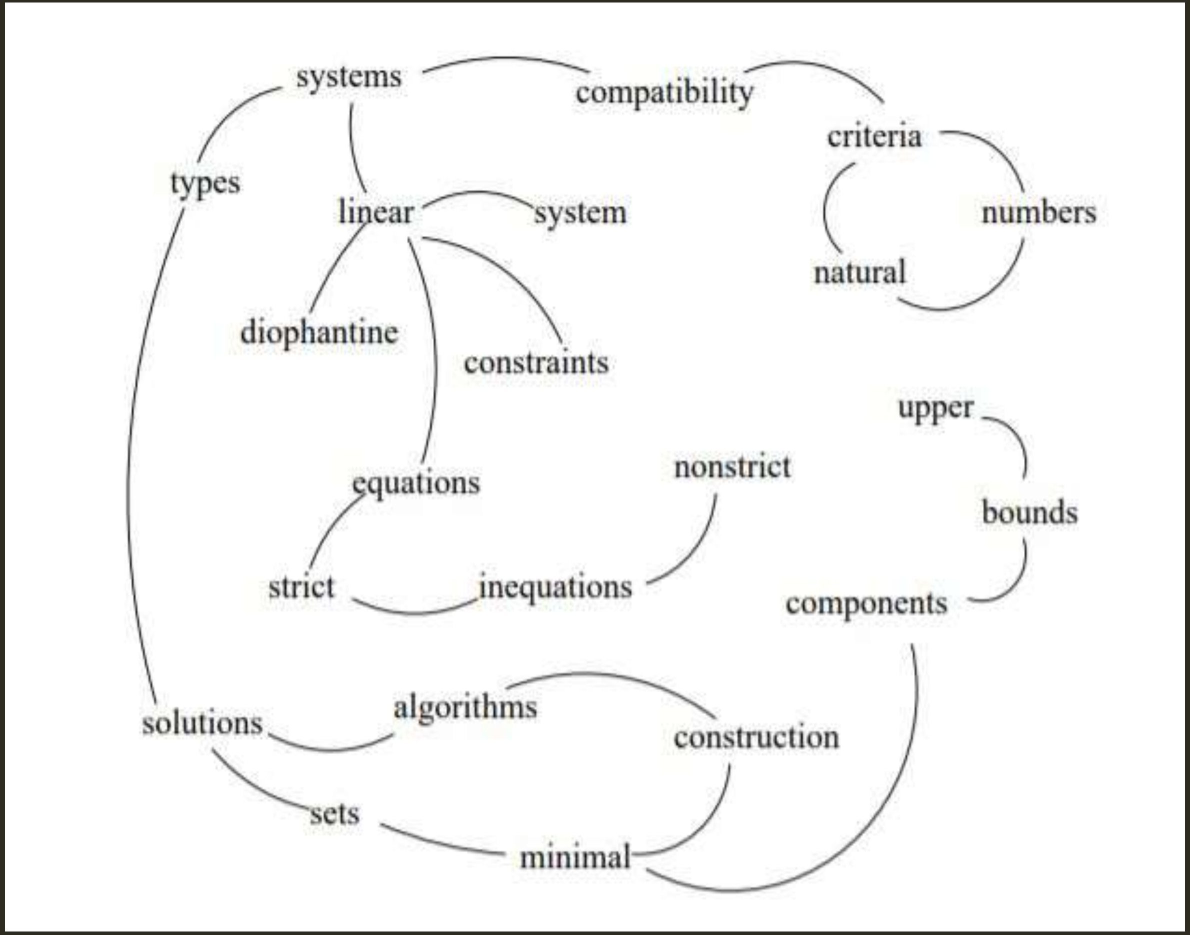
Figure 1: The overview architecture of the BERTSUM model.

- After obtaining the sentence vectors from BERT, we build several summarization-specific layers stacked on top of the BERT outputs, to capture document-level features for extracting summaries. For each sentence senti , we will calculate the final predicted score Yˆ i, either by simply using a sigmoid function or a coupling of an “inter-sentence Transformer” and a Sigmoid activation fucntion . The loss of the whole model is the Binary Classification Entropy of Yˆ i against gold label Yi

TextRank Algorithm

- a graph-based ranking model for text processing

- The basic idea implemented by a graph-based ranking model is that of “voting” or “recommendation”. When one vertex links to another one, it is basically casting a vote for that other vertex. The higher the number of votes that are cast for a vertex, the higher the importance of the vertex



TextRank Algorithm

- Formally, let G=(V,E) be a directed graph with the set of vertices V and set of edges E, where E is a subset of V x V . For a given vertex Vi , let *in(Vi)* be the set of vertices that point to it predecessors, and let *out(Vi)* be the set of vertices that vertex Vi points to successors. The score of a vertex is defined as follows (Brin and Page, 1998):

$$S(V_i) = (1 - d) + d * \sum_{j \in In(V_i)} \frac{1}{|Out(V_j)|} S(V_j)$$

where d is a damping factor that can be set between 0 and 1.

TextRank Algorithm

- The units to be ranked are sequences of one or more lexical units extracted from text, and these represent the vertices that are added to the text graph
- two vertices are connected if their corresponding lexical units co-occur within a window of maximum words, where can be set anywhere from 2 to 10 words. Co-occurrence links express relations between syntactic elements.
- The vertices added to the graph can be restricted with syntactic filters, which select only lexical units of a certain part of speech; best results observed for nouns and adjectives only.
- Once a final score is obtained for each vertex in the graph, vertices are sorted in reversed order of their score, and the top vertices in the ranking are retained for post-processing

TextRank Algorithm

- After the ranking algorithm is run on the graph, components are sorted in reversed order of their score, and the top ranked sentences are selected for inclusion in the summary.

- Output will be a text document file that includes the summary.

- Will use gensim python library functions to implement this algorithm

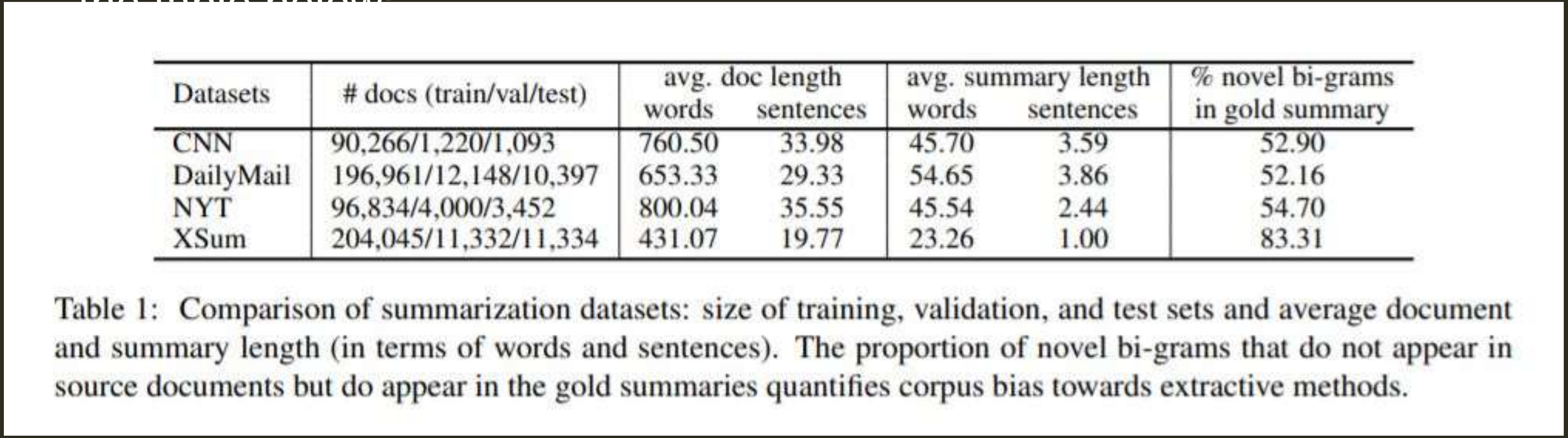
Keyword Extraction

- As an additional feature to our summariser model, we will extract a few high priority keywords from the text document and highlight them at the top or bottom of the summary.

- Will use spaCy/gensim python library functions to extract important keywords

Dataset details

- CNN/DailyMail dataset, which we will use, contains news articles and associated highlights, i.e., a few bullet points giving a brief overview of the article.
- The paper we based this project on, uses all these datasets mentioned in the table below:



Datasets	# docs (train/val/test)	avg. doc length		avg. summary length		% novel bi-grams in gold summary
		words	sentences	words	sentences	
CNN	90,266/1,220/1,093	760.50	33.98	45.70	3.59	52.90
DailyMail	196,961/12,148/10,397	653.33	29.33	54.65	3.86	52.16
NYT	96,834/4,000/3,452	800.04	35.55	45.54	2.44	54.70
XSum	204,045/11,332/11,334	431.07	19.77	23.26	1.00	83.31

Table 1: Comparison of summarization datasets: size of training, validation, and test sets and average document and summary length (in terms of words and sentences). The proportion of novel bi-grams that do not appear in source documents but do appear in the gold summaries quantifies corpus bias towards extractive methods.

Implementation

Intuitively understand what BERT is

Preprocess text data for BERT and build PyTorch Dataset (tokenization, attention masks, and padding)

Build Text Summarizer using the Transformers library by Hugging Face

Evaluate the model on test data

Produce summaries for raw text

Implement Pretrained BERT Models and obtain summaries

Evaluate the ROUGUE values for obtained summaries

Data Preprocessing

- ❑ Download the Dataset(CNN/DailyMail Articles)
- ❑ Tokenization and Sentence Splitting
- ❑ Format to Simpler Json Files
- ❑ Format to PyTorch Files

TOKENIZATION

The transformer Library has prebuild tokenizers to tokenize the data.

```
sample_txt = 'When was I last outside? I am stuck at home for 2 weeks.'
```

```
tokens = tokenizer.tokenize(sample_txt)
```

```
token_ids = tokenizer.convert_tokens_to_ids(tokens)
```

```
Tokens: ['When', 'was', 'I', 'last', 'outside', '?', 'I', 'am', 'stuck', 'at', 'home', 'for', '2', 'weeks', '.']
```

```
Token IDs: [1332, 1108, 146, 1314, 1796, 136, 146, 1821, 5342, 1120, 1313, 1111, 123, 2277, 119]
```

Special Tokens

Add special tokens to separate sentences.

[SEP] - marker for ending of a sentence

tokenizer.sep_token, tokenizer.sep_token_id

[CLS] – Start of each Sentence

tokenizer.cls_token, tokenizer.cls_token_id

[PAD] special token for padding:

tokenizer.pad_token, tokenizer.pad_token_id

BERT understands tokens that were in the training set.

Everything else can be encoded using the [UNK] (unknown) token:

tokenizer.unk_token, tokenizer.unk_token_id

Pytorch

An open source machine learning framework that allows you to work state-of-the-art models and deploy them to production.

Imperative Programming

Dynamic computation graphing

Completely Pythonic

Huggingface

Huggingface is a transformers library, which runs on top of PyTorch, it allows us to implement Transformer models, including Distilbert-base-uncased and BART(BERT+GPT), and use them for a variety of language tasks.

Bert Models

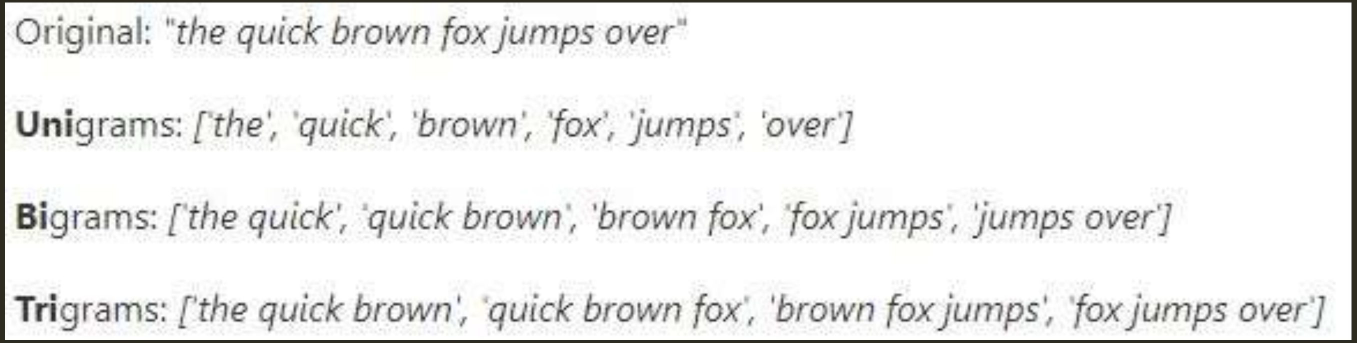
- Distilbert-base-uncased(Pretrained)
- BertAbs(Pretrained)
- BART(Bert+GPT)
- BertRank (Project model using BertSum+TextRank)

Evaluation measures(ROUGE)

- We will evaluate summarization quality automatically using ROUGE (Recall-Oriented Understudy for Gisting Evaluation). We shall report unigram and bigram overlap (ROUGE-1 and ROUGE-2) as a means of assessing informativeness and the longest common subsequence (ROUGE-L) as a means of assessing fluency.

ROUGE-N measures the number of matching ‘n-grams’ between our model-generated text and a ‘reference’.

An n-gram is simply a grouping of tokens/words. A unigram (1-gram) would consist of a single word. A bigram (2-gram) consists of two consecutive words



Original: *"the quick brown fox jumps over"*

Unigrams: *['the', 'quick', 'brown', 'fox', 'jumps', 'over']*

Bigrams: *['the quick', 'quick brown', 'brown fox', 'fox jumps', 'jumps over']*

Trigrams: *['the quick brown', 'quick brown fox', 'brown fox jumps', 'fox jumps over']*

ROUGE-L measures the longest common subsequence (LCS) between our model output and reference.

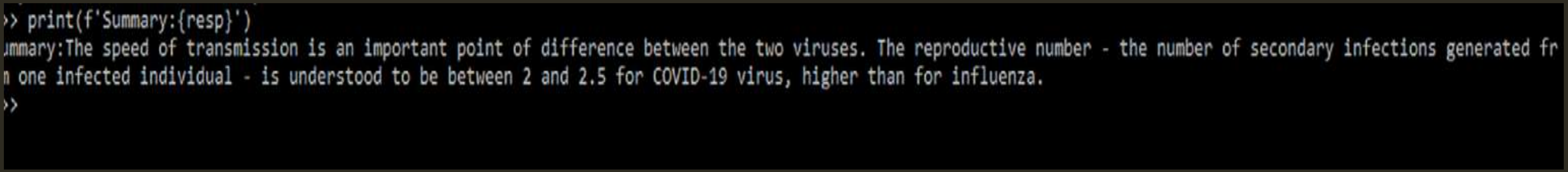
Sample summaries and Rouge Scores

Sample_text= The speed of transmission is an important point of difference between the two viruses. Influenza has a shorter median incubation period (the time from infection to appearance of symptoms) and a shorter serial interval (the time between successive cases) than COVID-19 virus. The serial interval for COVID-19 virus is estimated to be 5-6 days, while for influenza virus, the serial interval is 3 days. This means that influenza can spread faster than COVID-19. Further, transmission in the first 3-5 days of illness, or potentially pre-symptomatic transmission –transmission of the virus before the appearance of symptoms – is a major driver of transmission for influenza. In contrast, while we are learning that there are people who can shed COVID-19 virus 24-48 hours prior to symptom onset, at present, this does not appear to be a major driver of transmission. The reproductive number – the number of secondary infections generated from one infected individual – is understood to be between 2 and 2.5 for COVID-19 virus, higher than for influenza. However, estimates for both COVID-19 and influenza viruses are very context and time-specific, making direct comparisons more difficult.

Distilbert-base-uncased

Summary obtained using Distilbert-base-uncased

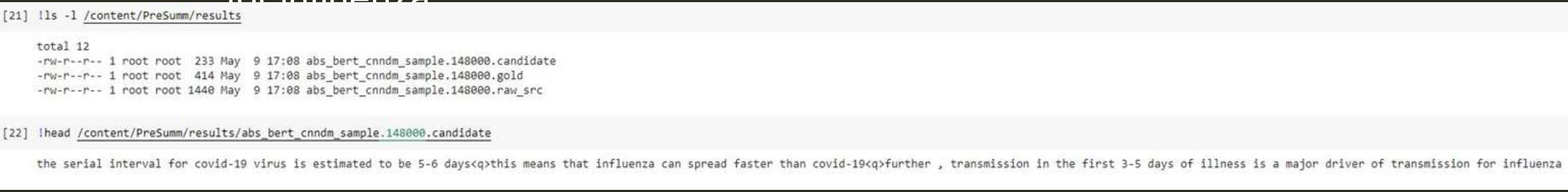
The speed of transmission is an important point of difference between the two viruses. The reproductive number - the number of secondary infections generated from one infected individual - is understood to be between 2 and 2.5 for COVID-19 virus, higher than for influenza.



```
>> print(f'Summary:{resp}')
ummary:The speed of transmission is an important point of difference between the two viruses. The reproductive number - the number of secondary infections generated fr
n one infected individual - is understood to be between 2 and 2.5 for COVID-19 virus, higher than for influenza.
>>
```

BertAbs

The serial interval for covid-19 virus is estimated to be 5-6 days. This means that influenza can spread faster than covid-19. Further transmission in the first 3-5 days of illness is a major driver of transmission for influenza

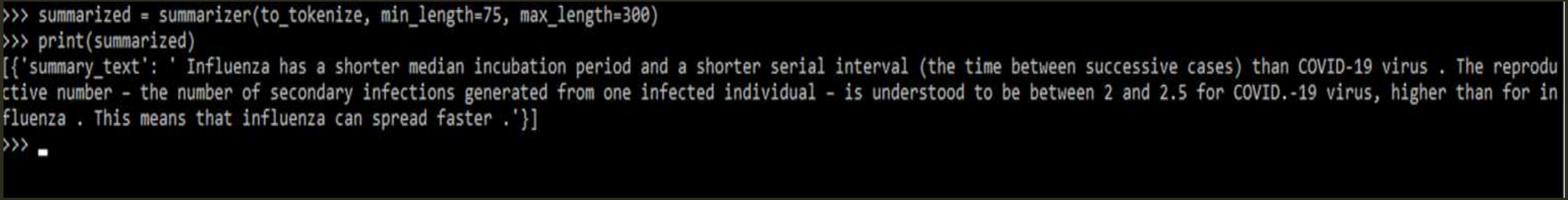


```
[21] !ls -l /content/PreSumm/results

     total 12
     -rw-r--r-- 1 root root  233 May  9 17:08 abs_bert_cnndm_sample.148000.candidate
     -rw-r--r-- 1 root root  414 May  9 17:08 abs_bert_cnndm_sample.148000.gold
     -rw-r--r-- 1 root root 1440 May  9 17:08 abs_bert_cnndm_sample.148000.raw_src

[22] !head /content/PreSumm/results/abs_bert_cnndm_sample.148000.candidate

     the serial interval for covid-19 virus is estimated to be 5-6 days<q>this means that influenza can spread faster than covid-19<q>further , transmission in the first 3-5 days of illness is a major driver of transmission for influenza
```

BART (BERT+GPT)

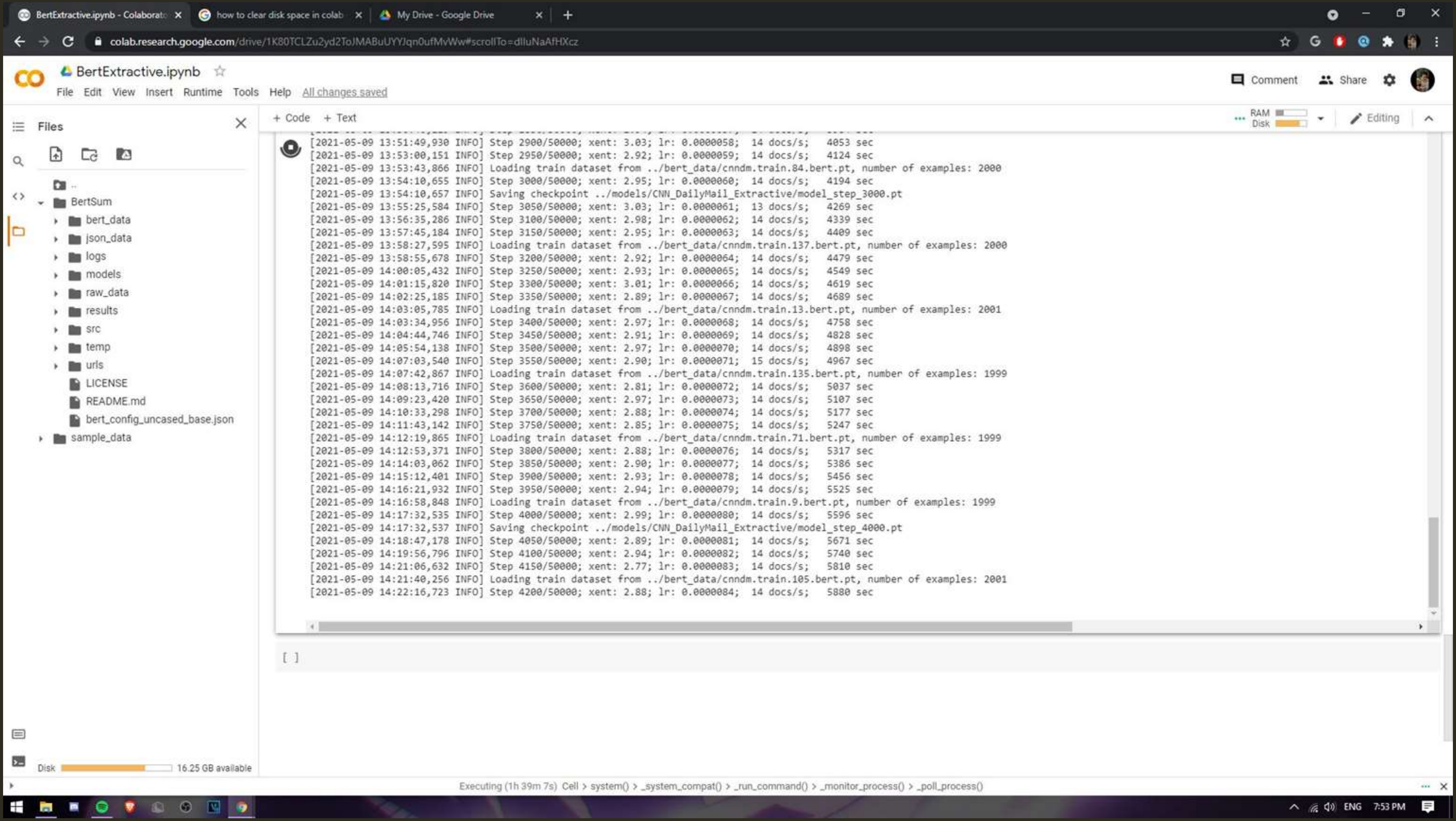
Summary obtained using BART (Bert+GPT)

Influenza has a shorter median incubation period and a shorter serial interval (the time between successive cases) than COVID-19 virus . The reproductive number – the number of secondary infections generated from one infected individual – is understood to be between 2 and 2.5 for COVID.-19 virus, higher than for influenza . This means that influenza can spread faster .



```
>>> summarized = summarizer(to_tokenize, min_length=75, max_length=300)
>>> print(summarized)
[{'summary_text': ' Influenza has a shorter median incubation period and a shorter serial interval (the time between successive cases) than COVID-19 virus . The reprodu
ctive number - the number of secondary infections generated from one infected individual - is understood to be between 2 and 2.5 for COVID.-19 virus, higher than for in
fluenza . This means that influenza can spread faster .'}]
>>> 
```

MODELS	ROUGE-1	ROUGE-2	ROUGE-L
Distilbert	24	23	32
BERTAbs	28	24	37
BART	30	28	43


BertExtractive.ipynb - Colaborato
how to clear disk space in colab
My Drive - Google Drive
colab.research.google.com/drive/1K80TCLZu2yd2ToJMABuUYYJqn0ufMvWw#scrollTo=dIluNaAfHXcz
BertExtractive.ipynb
File Edit View Insert Runtime Tools Help All changes saved
Comment
Share
Files
+ Code + Text
RAM
Disk
Editing
BertSum
bert_data
json_data
logs
models
raw_data
results
src
temp
urls
LICENSE
README.md
bert_config_uncased_base.json
sample_data
Disk 16.25 GB available
Executing (1h 39m 7s) Cell > system() > _system_compat() > _run_command() > _monitor_process() > _poll_process()
ENG 7:53 PM

```
[2021-05-09 13:51:49,930 INFO] Step 2900/50000; xent: 3.03; lr: 0.0000058;  14 docs/s;   4053 sec
[2021-05-09 13:53:00,151 INFO] Step 2950/50000; xent: 2.92; lr: 0.0000059;  14 docs/s;   4124 sec
[2021-05-09 13:53:43,866 INFO] Loading train dataset from ../bert_data/cnndm.train.84.bert.pt, number of examples: 2000
[2021-05-09 13:54:10,655 INFO] Step 3000/50000; xent: 2.95; lr: 0.0000060;  14 docs/s;   4194 sec
[2021-05-09 13:54:10,657 INFO] Saving checkpoint ../models/CNN_DailyMail_Extractive/model_step_3000.pt
[2021-05-09 13:55:25,584 INFO] Step 3050/50000; xent: 3.03; lr: 0.0000061;  13 docs/s;   4269 sec
[2021-05-09 13:56:35,286 INFO] Step 3100/50000; xent: 2.98; lr: 0.0000062;  14 docs/s;   4339 sec
[2021-05-09 13:57:45,184 INFO] Step 3150/50000; xent: 2.95; lr: 0.0000063;  14 docs/s;   4409 sec
[2021-05-09 13:58:27,595 INFO] Loading train dataset from ../bert_data/cnndm.train.137.bert.pt, number of examples: 2000
[2021-05-09 13:58:55,678 INFO] Step 3200/50000; xent: 2.92; lr: 0.0000064;  14 docs/s;   4479 sec
[2021-05-09 14:00:05,432 INFO] Step 3250/50000; xent: 2.93; lr: 0.0000065;  14 docs/s;   4549 sec
[2021-05-09 14:01:15,820 INFO] Step 3300/50000; xent: 3.01; lr: 0.0000066;  14 docs/s;   4619 sec
[2021-05-09 14:02:25,185 INFO] Step 3350/50000; xent: 2.89; lr: 0.0000067;  14 docs/s;   4689 sec
[2021-05-09 14:03:05,785 INFO] Loading train dataset from ../bert_data/cnndm.train.13.bert.pt, number of examples: 2001
[2021-05-09 14:03:34,956 INFO] Step 3400/50000; xent: 2.97; lr: 0.0000068;  14 docs/s;   4758 sec
[2021-05-09 14:04:44,746 INFO] Step 3450/50000; xent: 2.91; lr: 0.0000069;  14 docs/s;   4828 sec
[2021-05-09 14:05:54,138 INFO] Step 3500/50000; xent: 2.97; lr: 0.0000070;  14 docs/s;   4898 sec
[2021-05-09 14:07:03,540 INFO] Step 3550/50000; xent: 2.90; lr: 0.0000071;  15 docs/s;   4967 sec
[2021-05-09 14:07:42,867 INFO] Loading train dataset from ../bert_data/cnndm.train.135.bert.pt, number of examples: 1999
[2021-05-09 14:08:13,716 INFO] Step 3600/50000; xent: 2.81; lr: 0.0000072;  14 docs/s;   5037 sec
[2021-05-09 14:09:23,420 INFO] Step 3650/50000; xent: 2.97; lr: 0.0000073;  14 docs/s;   5107 sec
[2021-05-09 14:10:33,298 INFO] Step 3700/50000; xent: 2.88; lr: 0.0000074;  14 docs/s;   5177 sec
[2021-05-09 14:11:43,142 INFO] Step 3750/50000; xent: 2.85; lr: 0.0000075;  14 docs/s;   5247 sec
[2021-05-09 14:12:19,865 INFO] Loading train dataset from ../bert_data/cnndm.train.71.bert.pt, number of examples: 1999
[2021-05-09 14:12:53,371 INFO] Step 3800/50000; xent: 2.88; lr: 0.0000076;  14 docs/s;   5317 sec
[2021-05-09 14:14:03,062 INFO] Step 3850/50000; xent: 2.90; lr: 0.0000077;  14 docs/s;   5386 sec
[2021-05-09 14:15:12,401 INFO] Step 3900/50000; xent: 2.93; lr: 0.0000078;  14 docs/s;   5456 sec
[2021-05-09 14:16:21,932 INFO] Step 3950/50000; xent: 2.94; lr: 0.0000079;  14 docs/s;   5525 sec
[2021-05-09 14:16:58,848 INFO] Loading train dataset from ../bert_data/cnndm.train.9.bert.pt, number of examples: 1999
[2021-05-09 14:17:32,535 INFO] Step 4000/50000; xent: 2.99; lr: 0.0000080;  14 docs/s;   5596 sec
[2021-05-09 14:17:32,537 INFO] Saving checkpoint ../models/CNN_DailyMail_Extractive/model_step_4000.pt
[2021-05-09 14:18:47,178 INFO] Step 4050/50000; xent: 2.89; lr: 0.0000081;  14 docs/s;   5671 sec
[2021-05-09 14:19:56,796 INFO] Step 4100/50000; xent: 2.94; lr: 0.0000082;  14 docs/s;   5740 sec
[2021-05-09 14:21:06,632 INFO] Step 4150/50000; xent: 2.77; lr: 0.0000083;  14 docs/s;   5810 sec
[2021-05-09 14:21:40,256 INFO] Loading train dataset from ../bert_data/cnndm.train.105.bert.pt, number of examples: 2001
[2021-05-09 14:22:16,723 INFO] Step 4200/50000; xent: 2.88; lr: 0.0000084;  14 docs/s;   5880 sec
```

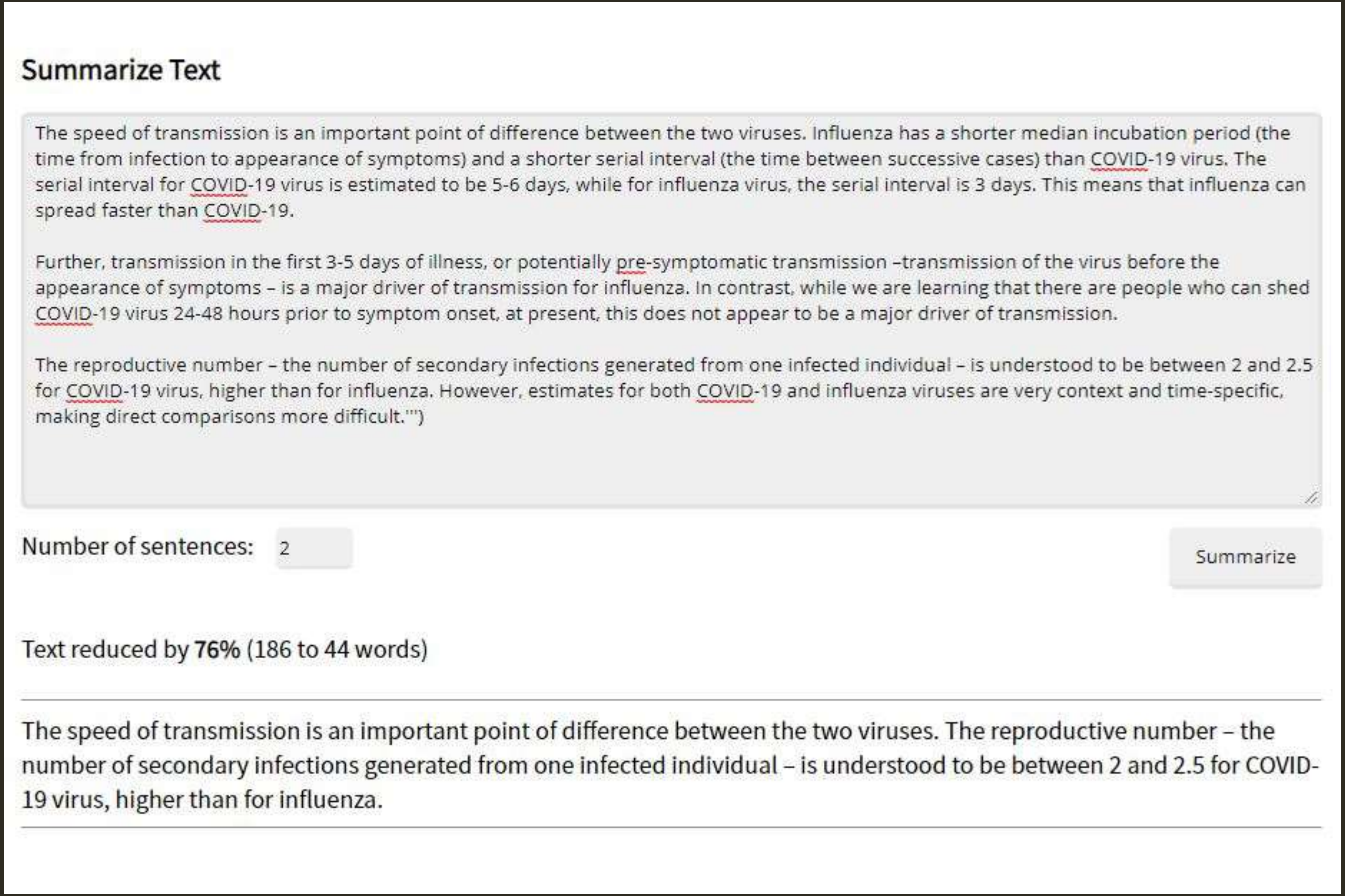
Work to be done

Incorporate Bert with TextRank

Extract Important Keywords

Compare Rouge Values (BERT vs Textrank vs BERTRank)

Produce a Web based Application(Post Processing)



Summarize Text

The speed of transmission is an important point of difference between the two viruses. Influenza has a shorter median incubation period (the time from infection to appearance of symptoms) and a shorter serial interval (the time between successive cases) than COVID-19 virus. The serial interval for COVID-19 virus is estimated to be 5-6 days, while for influenza virus, the serial interval is 3 days. This means that influenza can spread faster than COVID-19.

Further, transmission in the first 3-5 days of illness, or potentially pre-symptomatic transmission –transmission of the virus before the appearance of symptoms – is a major driver of transmission for influenza. In contrast, while we are learning that there are people who can shed COVID-19 virus 24-48 hours prior to symptom onset, at present, this does not appear to be a major driver of transmission.

The reproductive number – the number of secondary infections generated from one infected individual – is understood to be between 2 and 2.5 for COVID-19 virus, higher than for influenza. However, estimates for both COVID-19 and influenza viruses are very context and time-specific, making direct comparisons more difficult.''')

Number of sentences: 2

Text reduced by **76%** (186 to 44 words)

The speed of transmission is an important point of difference between the two viruses. The reproductive number – the number of secondary infections generated from one infected individual – is understood to be between 2 and 2.5 for COVID-19 virus, higher than for influenza.

CONCLUSION

- An efficient and user friendly text summarisation tool is developed.
- BERT combined with TextRank will help to improve summarisation capabilities and obtain sensible summaries.
- Large documents can be simplified and shortened in reduced time.
- Important points and Key sentences in lengthy paragraphs are highlighted.

WORK PROGRESS

Work	Expected Date Of Completion	Status
Problem Identification	10/11/2020	Completed
High Level Design	8/01/2021	Completed
Dataset Collection	08/01/2021	Completed
Testing pre-trained models for reference and comparison	25/04/2021	Completed
Training of BERT for our model	15/05/2021	Ongoing
Incorporating TextRank into our model	17/05/2021	Yet to start
Testing and validating BertRank and calculating Rouge scores	22/05/2021	Yet to start
Adding extraction feature for keywords and post-processing	31/05/2021	Yet to start
Final stages of noting observations	2/06/2021	Yet to start

THANK YOU!