Text Summarization of Online Transcripts

Executive Summary:

To build an application that produces notes based on transcripts generated during online classes across different meeting platforms, we had shortlisted 6 Summarization algorithms to start with.

From simple pretrained architectural summarizers to advanced Transformer-based models that's been trained on huge corpuses.

After experimenting with the algorithms, we analysed the results were very good in transformer-based summarizers like, BERT (Bidirectional Encoder Representations from Transformers) and GPT-2.

The dataset we chose were raw transcripts from online lectures. It had speaker name, time stamp and the textual content. A preprocessing function was written to read the raw .txt file and was converted to a usable, cleaned string.

The cleaning function contains lowering the text, removal of numbers and extra spaces. Stop words are then removed from the sentences. Finally, each word is lemmatized to its origin form and sent as a preprocessed text

Introduction:

Text Summarization is a technique for generating a concise and precise summary of voluminous texts while focusing on the sections that convey useful information, and without losing the overall meaning. It aims to transform lengthy documents into shortened versions, something which could be difficult and costly to undertake if done manually. We train the Machine learning algorithms to comprehend documents and identify the sections that convey important facts and information before producing the required summarized texts.

With the present explosion of data circulating the digital space, which is mostly non-structured textual data, there is a need to develop text summarization tools that allow people to get insights from them easily. The ease of quicker access to enormous amounts of information saves plenty of time. However, most of this information is redundant, insignificant, and may not convey the intended meaning. Therefore, using text summarizers capable of extracting useful information that leaves out inessential and insignificant data becomes vital. Implementing summarization can enhance the readability of documents, reduce the time spent in researching for information, and allow for more information to be fitted in a particular area.

Rationale Statement:

Text Summarization can be done in two ways. They are:

1)Extractive Text Summarization

2)Abstractive Summarization

Extractive Text Summarisation takes out the important sentences from the original text that was comprehended and stitch together the portions of sentences to form a brief version.

While the Abstractive Summarization rephrases the sentences and produces a custom version of the text within the same context.

In this project we will mainly focus on Extractive Text Summarisation as we do not want to change the originality of what was being spoken.

Topic modelling is a type of statistical modelling used to find abstract topics in a set of documents.

Latent Dirichlet Allocation (LDA) is an example of topic model and is used to classify text in a document to a particular topic. It builds a topic per document model and words per topic model, modeled as Dirichlet distribution.

After topic modelling, n-topics are returned with their closely related words which can be used to create visualizations.

Problem:

In situations where it is important to keep track of what is being spoken, such as during an online lecture, taking notes is a popular activity that is used by many.

The art of notetaking does not involve making note of every single word that is spoken, but comprehensive outlines of what is discussed.

The key to good notetaking lies in making concise yet informative summaries. Making notes could be harder to do sometimes during online classes where unexpected circumstances such as, say as drop in network, would result in the student losing some information that was taught during the time they had lost network.

In such cases, obtaining summarised notes post each online class could be beneficial and going through all important points.

Data Requirements:

- A transcription dataset of at least 500 words in the raw text
- S-BERT encoded vector of the raw text column
- Requires the transcripts to include speaker details
- Requires the transcripts to include timestamps
- Requires the transcript with correct punctuation and grammar

- **Assumptions:**
    - Expects the format of transcripts to be as a JSON or .txt file
    - Assumes that the audio transcript contains speech data of one speaker at a time
- **Constraints:**
    - Sometimes the speaker's accent could affect the transcripts.
    - In other cases, if there are multiple speakers, there's a chance that only partial sentences are recorded

Data:

- **Primary data:** Audio transcripts from Zoom / Teams meetings. At the end of every meeting, transcripts autogenerated by Automatic Speech Recognition (ASR) will be available in the chat room. These are raw transcriptions of the meeting's audio containing all the speakers' details and their timestamps.
- **Secondary data:** The Extreme Summarization (XSum) dataset is a dataset for the evaluation of abstractive single-document summarization systems. As our models are generalized, evaluating the accuracy and scores on any textual dataset should work fine. For this purpose, we took the XSum dataset.

Model/ Architecture Approach

Summarization is a process of summing up key concepts or points from a context or paragraph. In our approach, we are using summarization to extract some of the important topics or keywords from transcripts generated from online meetings.

In preprocessing phase, we convert the unstructured data to structured format which includes removal of stop words and lemmatization.

We progress into the next step of representing the lemmatized sentences using a technique called Sentence Bert Encoding (S-Bert). These S-Bert encoded vectors are then passed to a Pytorch (Hugging Face) Model which performs the summarization of sentences.

On an abstract level, the texts are being clustered and the highly similar sentences with the actual contexts are returned.

The collated sentences will have high relevancy score to the cluster's centroid.

The extracted topics along with the summarized text are returned as outputs to a telegram chat by a bot.

Project plan:

Module	Task	Workload
Project Proposal	Statement of Work	10
Procuring Data	Collecting Primary Data	4
	Collecting Secondary Data	10
Preprocess Data	Clean Data	3
	Transform Data	3
	Feature Engineering	3
Creating Data Pipelines	Automating End-to-End Data Pipelines	7
Developing Models	Develop Text Summarization Model	7
	Develop Noise Reduction Model	7
	Perform Topic Modeling	3
Inferring Models	Test and Evaluate Models	3
Visualization	Generating Visualizations	2
Deployment	Build API	3
	Deploy Model	2
Documentation	Final Project Report	3
	Project Presentation	3

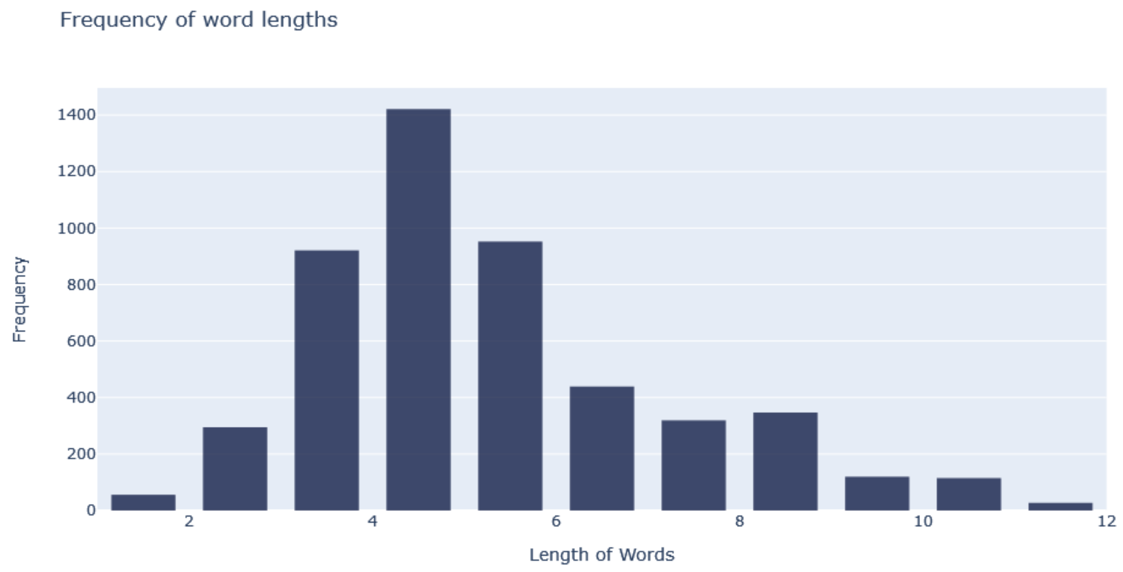
EDA:

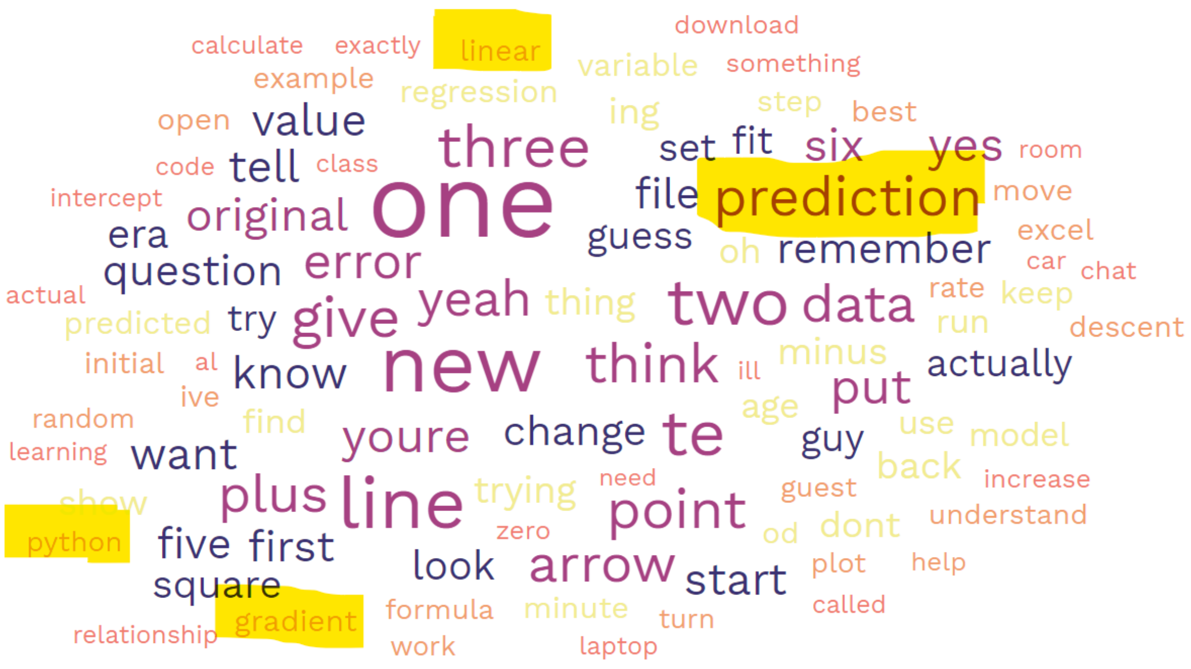
The raw data which we had before preprocessing is as follows:

```
Input Unformatted Text:

    time_stamp                                          sentence
0     17:12:48                                   Okay, perfect.
1     17:12:50                                 Hi, good evening.
2     17:12:51  So I put up a survey on Tuesday, I hope, most ...
3     17:13:11                                  went into the max.
4     17:13:13  And most of you were bored. So, today as try t...
..         ...                                               ...
605   18:56:46                                        All right.
606   18:56:49                                             Okay.
607   18:56:51                                 Thank you everyone.
608   18:56:53                                  Have a good night.
609   18:56:57                                        Thank you.

[610 rows x 2 columns]
```

In the first step of preprocessing, we have removed the special characters such as '[/(){}\[\]\|@,;]' and then we removed the stop words from the list of words we have. The text was then converted to lowercase and finally, the timestamps were removed from the text.

We have generated word-clouds to identify the most frequent words in our data and compare it with our final summaries. This will allow us to check that relevant concepts have been correctly extracted in the summaries.





Data-Preprocessing Pipeline:

Algorithm Evaluation:

(1) SpaCy with TF-IDF Vectorizer:

SpaCy is an open-source advanced natural language processing library, written in the programming languages Python and Cython. SpaCy is mainly used in the development of production software and supports deep learning workflow via statistical models of PyTorch and TensorFlow.

Pros: Maximizing your efficiency by minimizing the time you spend reading can have an impact on productivity. Whether you’re reading textbooks, reports, or academic journals, the power of natural language processing with Python and SpaCy can reduce the time you spend without diluting the quality of information.

It is also very customisable and widely used in most of the NLP research.

Cons: Most of the internal architecture are opaque. Fewer NLP abstractions and They are not optimised for speed

(2) Spacy with TextRank:

TextRank is an extractive and unsupervised text summarization technique. PyTextRank is a python implementation of TextRank as a SpaCy pipeline extension, for graph-based natural language work

Pros: An implementation of TextRank in Python for use in SpaCy pipelines which provides fast, effective phrase extraction from texts, along with extractive summarization. The graph algorithm works independent of a specific natural language and does not require domain knowledge.

Cons: It computationally heavier and slower. Also, Performance is not as effective as other Algorithms.

(3) Genism:

Genism is a free Python library designed to automatically extract semantic topics from documents. The Gensim implementation is based on the popular Textrank algorithm.

Pros: Gensim performs very well on short texts, compared to other algorithms.
As genism's underlying algorithm is a graph-based (Text-ranking) They have a way for deciding the importance of a vertex within a graph, based on global information recursively drawn from the entire graph.

Cons: While gensim performs better with short text scenarios, it's below par in performance when it comes to very long paragraphs.

(4) BERT:

BERT sentence transformer is a Python framework for state-of-the-art sentence, text and image embedding. It is used for sentence-pair regression tasks like semantic textual similarity (STS).

Pros: BERT summarizes the well on semantic textual similarity tasks and transfer learning tasks, where it outperforms other state-of-the-art sentence embeddings methods

Cons: BERT makes it unsuitable for semantic for unsupervised tasks like clustering. It requires that sentence-pairs are fed into the network, which causes a massive computational overhead

(5) BART:

BART is model used developed by Facebook’s. It contains 1024 hidden layers and 406M parameters and has been fine-tuned. Since the model is huge, we require the use of Google Colab to run the code.

Pros: The advantage of BART is that they are advanced they use the combined BERT encoding and GPT2’s decoding to generate text in a free-form manner.

Cons: Like the BERT transformers that require large computation powers for realistic data.

(6) GPT2:

The GPT2 Generative Pre-trained Transformer 2 has around which has around 1 billion parameters) and can only imagine the power of the most recent GPT3 which has 175 billion parameters! It can write from software codes to mind-blowing stories!

Pros: GPT2 adapts to the style and content of the conditioning text. This allows the user to generate realistic and coherent continuations about a topic of their choosing, as seen by the following select samples.

Cons: Though the model is pre-trained on a massive real-time their correctness is often questionable for the human required accuracy.

Candidate Algorithm Selection and Rationale:

GPT2:

GPT-2 is a transformers model pretrained on a very large corpus of English data in a self-supervised fashion. This means it was pretrained on the raw texts only, with no humans labelling them in any way (which is why it can use lots of publicly available data) with an automatic process to generate inputs and labels from those texts. More precisely, it was trained to guess the next word in sentences.

More precisely, inputs are sequences of continuous text of a certain length and the targets are the same sequence, shifted one token (word or piece of word) to the right. The model uses internally a mask-mechanism to make sure the predictions for the token i only uses the inputs from 1 to i but not the future tokens.

This way, the model learns an inner representation of the English language that can then be used to extract features useful for downstream tasks. The model is best at what it was pretrained for however, which is generating texts from a prompt.

We can use the raw model for text generation or fine-tune it to a downstream task. See the model hub to look for fine-tuned versions on a task that interests you. We can use this model directly with a pipeline for text generation as the generation relies on some randomness.

BERT:

BERT, which stands for Bidirectional Encoder Representations from Transformers is a recent developed language representation model. It was 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.

BERT is conceptually simple and empirically powerful. It obtains new state-of-the-art results on eleven natural language processing tasks, including pushing the GLUE score to 80.5% (7.7%-point absolute improvement), MultiNLI accuracy to 86.7% (4.6% absolute improvement), SQuAD v1.1 question answering Test F1 to 93.2 (1.5-point absolute improvement) and SQuAD v2.0 Test F1 to 83.1 (5.1-point absolute improvement).

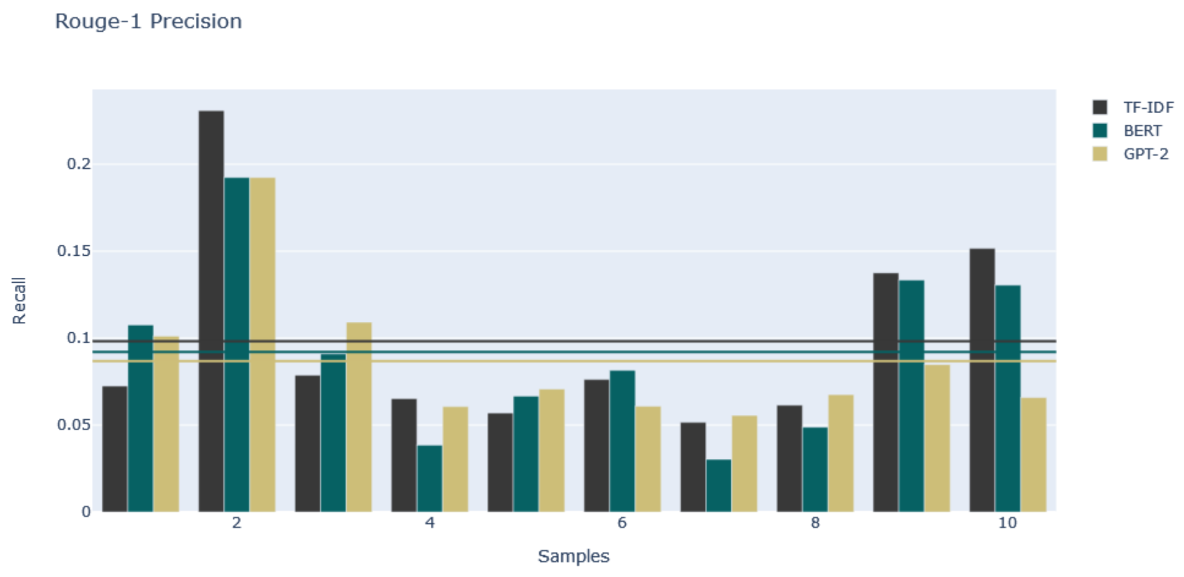
BERT is a model with absolute position embeddings, so it’s usually advised to pad the inputs on the right rather than the left.

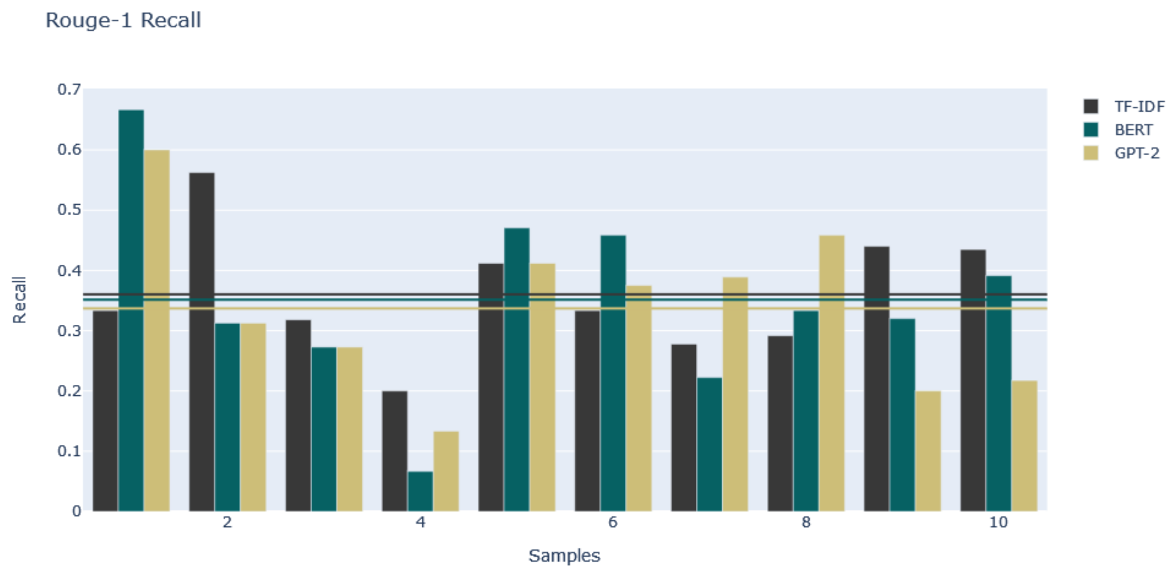
BERT was trained with the masked language modeling (MLM) and next sentence prediction (NSP) objectives. It is efficient at predicting masked tokens and at NLU in general but is not optimal for text generation.

Model Evaluation:

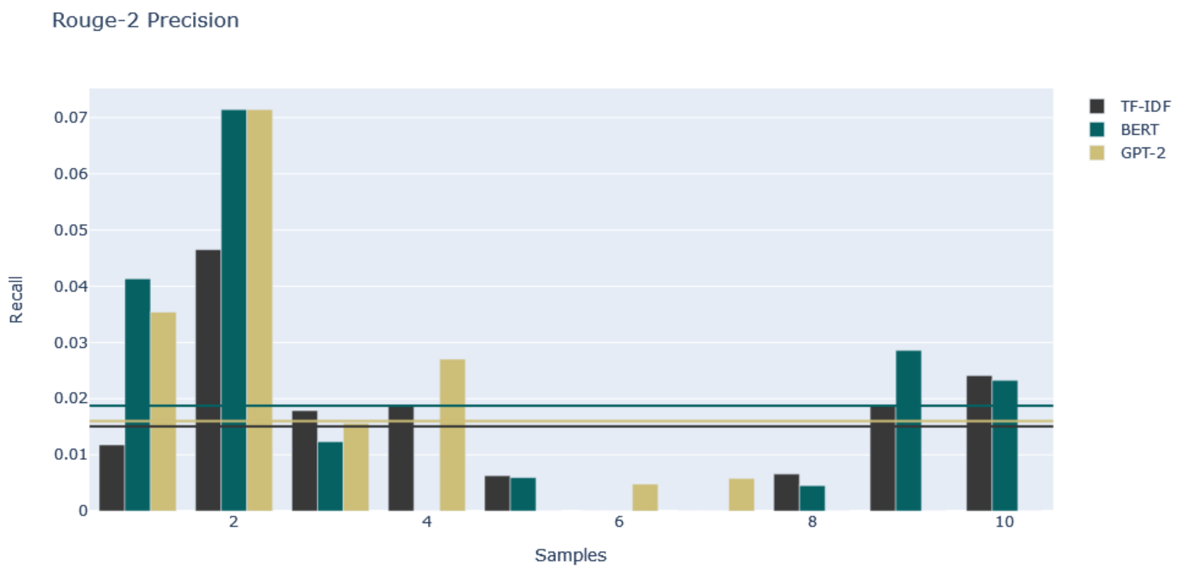
ROUGE, or Recall-Oriented Understudy for Gisting Evaluation, is a set of metrics and a software package used for evaluating automatic summarization and machine translation software in natural language processing. The metrics compare an automatically produced summary or translation against a reference or a set of references (human-produced) summary or translation.

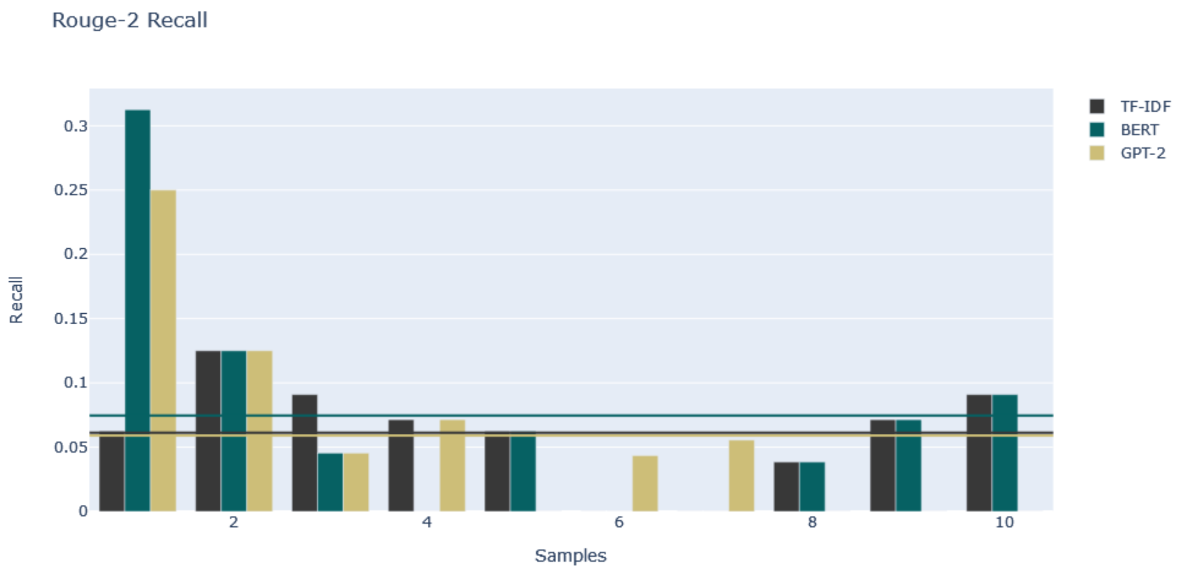
ROUGE-1 refers to the overlap of unigram (each word) between the system and reference summaries.

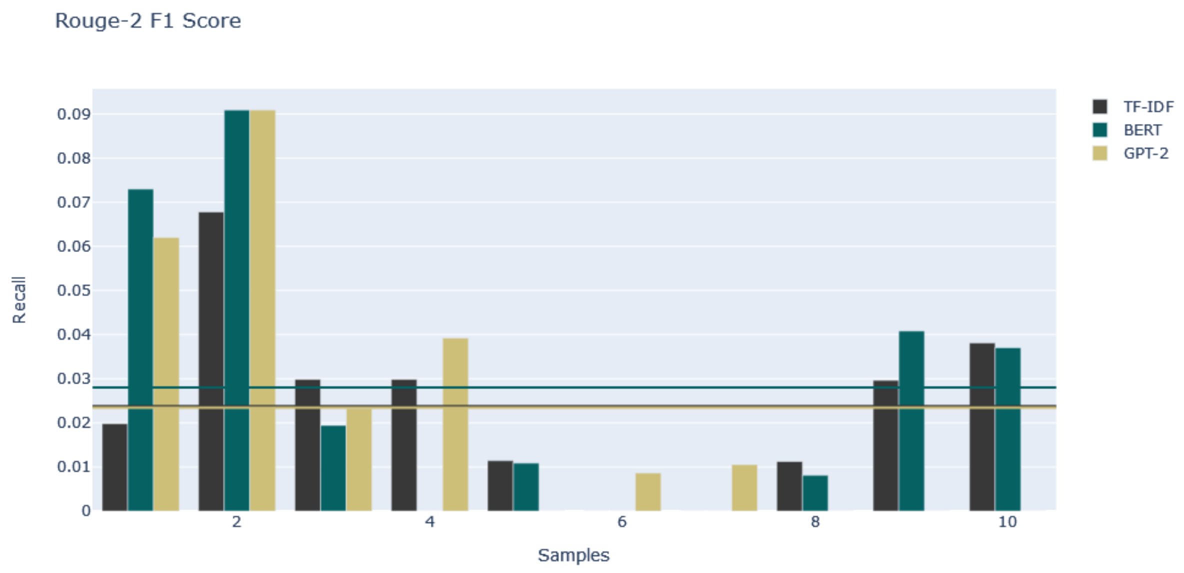




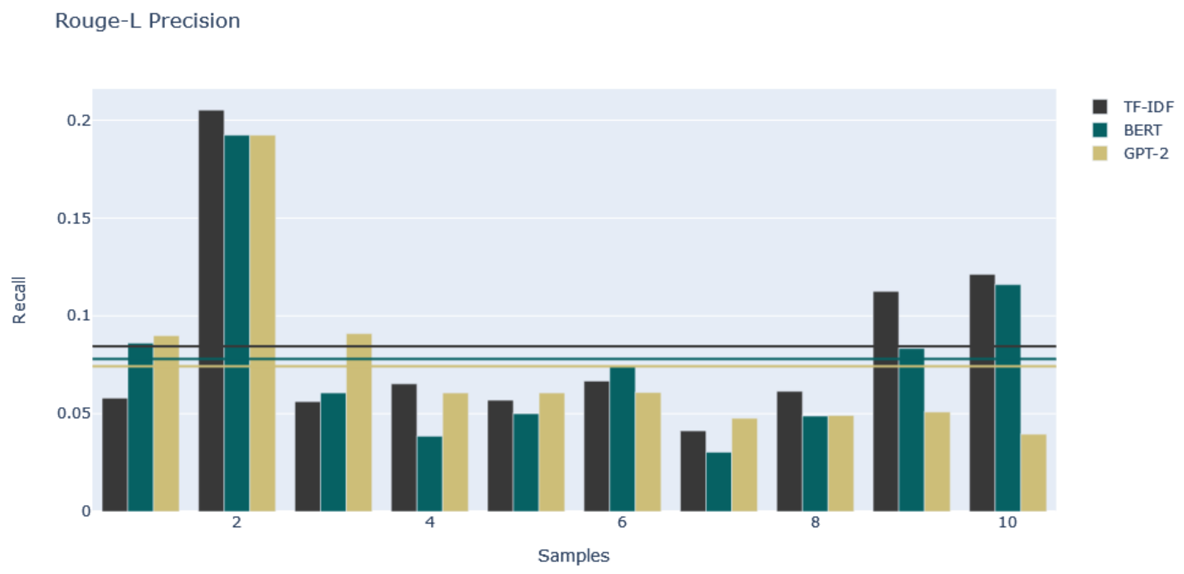
ROUGE-2 refers to the overlap of bigrams between the system and reference summaries.

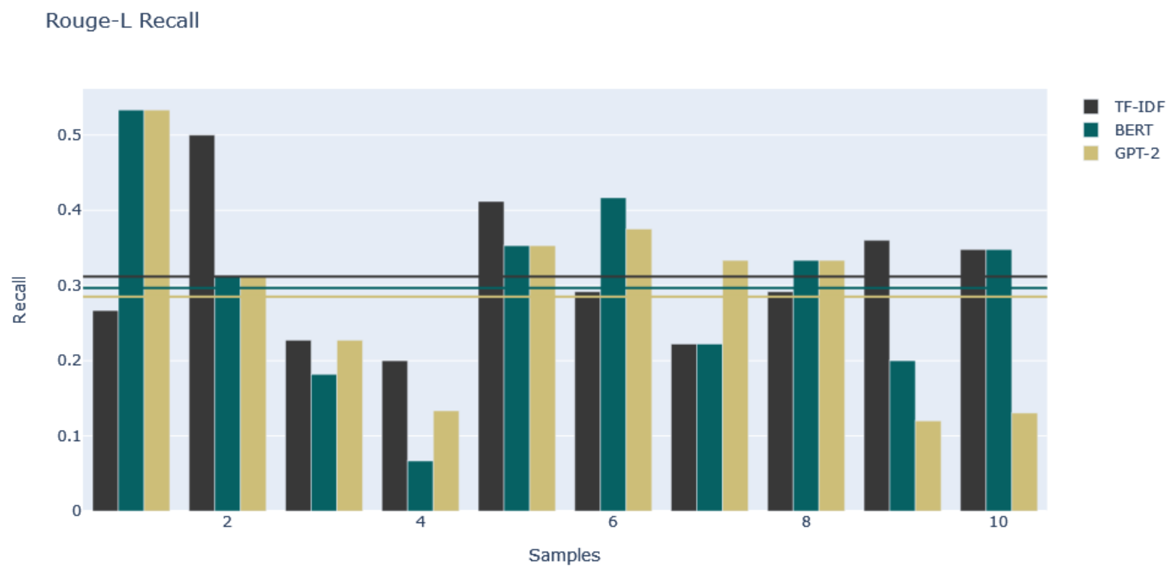


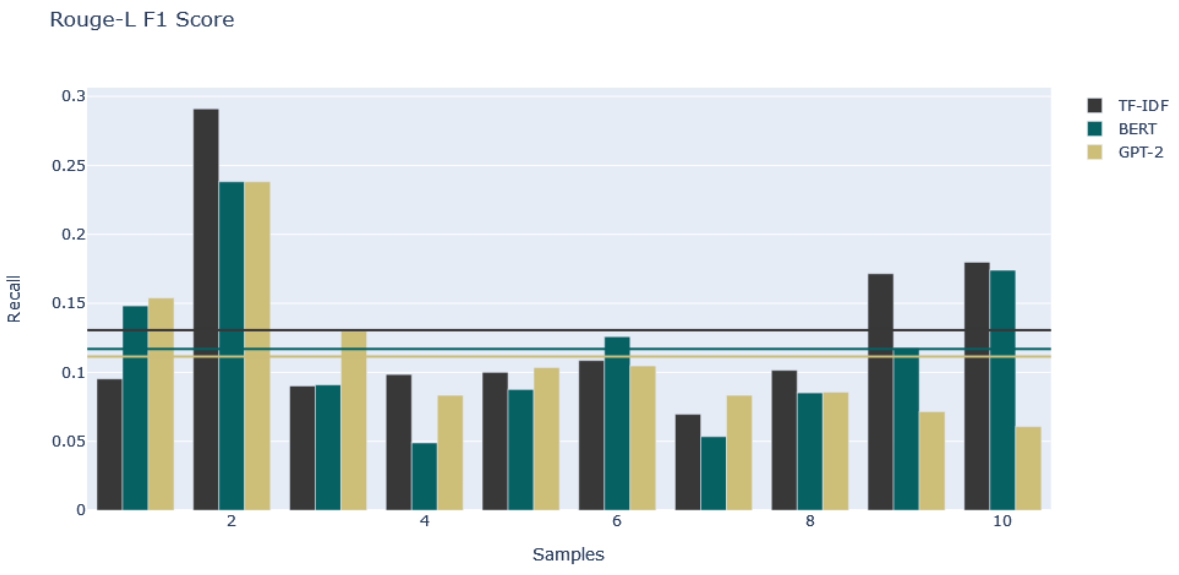




ROUGE-L: Longest Common Subsequence (LCS) based statistics. The longest common subsequence problem considers sentence-level structure similarity naturally and identifies the longest co-occurring in sequence n-grams automatically.





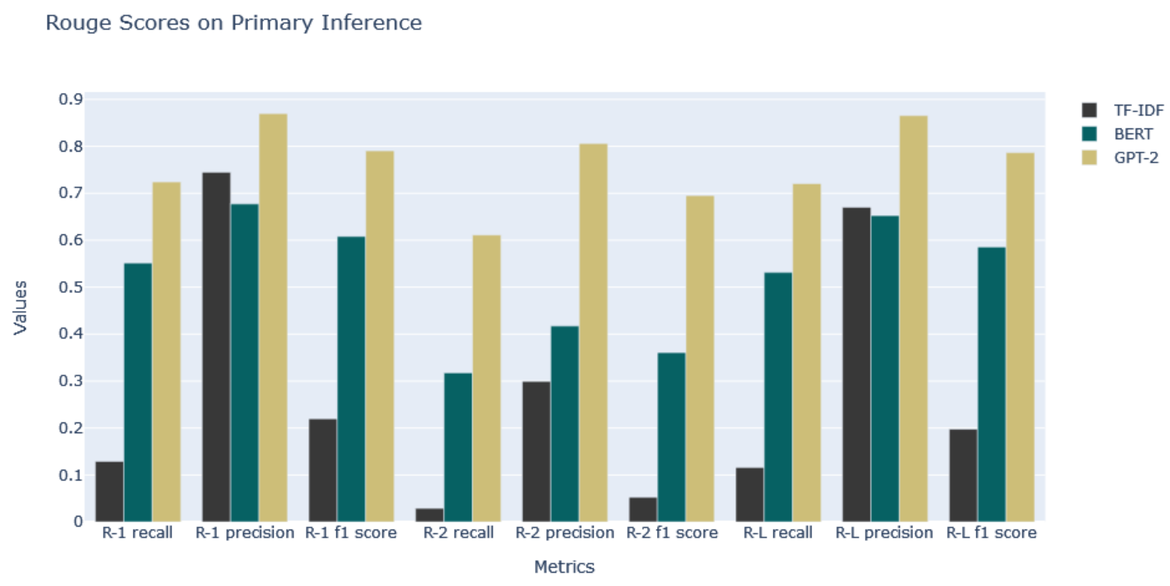


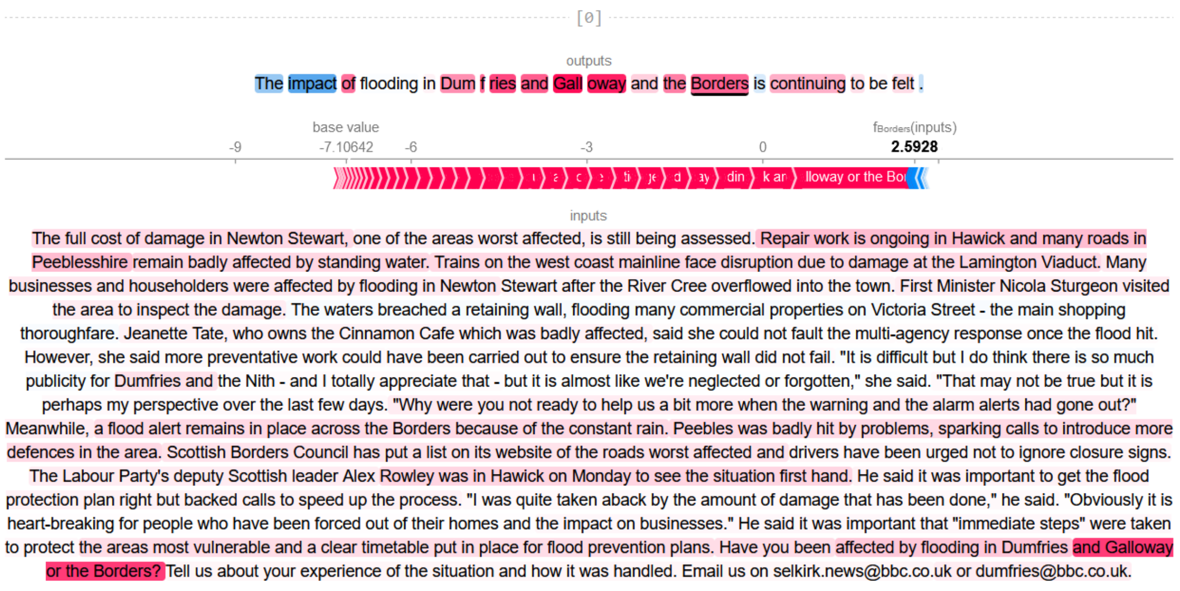
Final Inference:

For the inference, we have written some test cases to catch the implementation errors early on. This helped us in avoiding the extra time and effort spent in fixing the compatibility issues later. Some of the test cases include the checks such as

- The model predicted output shape is proper or not
- The input parameters passed to the model are in proper shape
- How the model behaves with varying optional parameters
- If the text data is being loaded in the right format
- If the text still contains the stop words after preprocessing
- If the timestamps were removed properly during preprocessing
- If the words in corpus are lemmatized properly
- If the text is in lower case after preprocessing
- If the final summaries generated by model are meaningful.

Some of these tests are automatic and very few of them would need human supervision. Our aim is to reduce the little human interference we have and build a robust system which can identify all the possible failures in our model



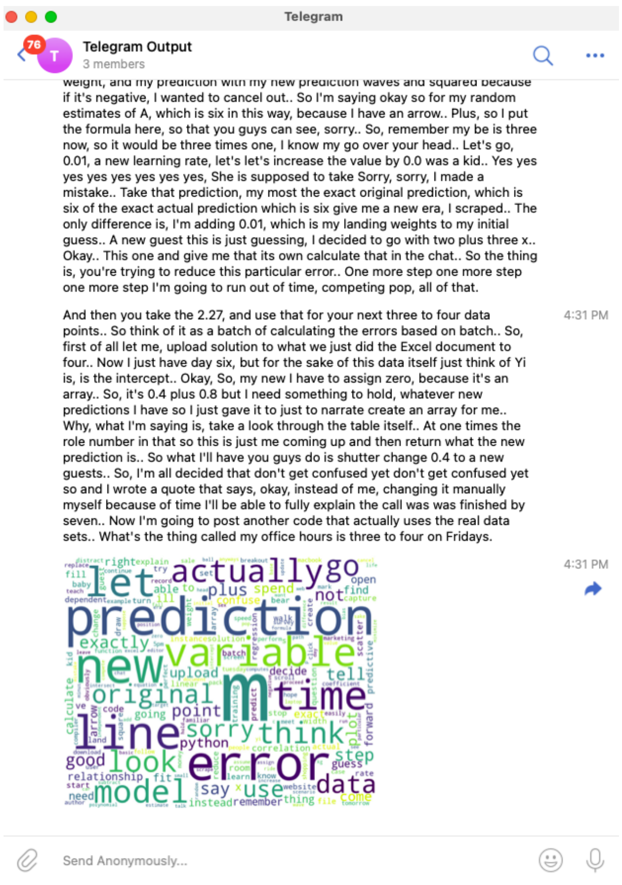


The impact of flooding in Dum f ries and Gall oway and the Borders is continuing to be felt .

The full cost of damage in Newton Stewart, one of the areas worst affected, is still being assessed. Repair work is ongoing in Hawick and many roads in Peeblesshire remain badly affected by standing water. Trains on the west coast mainline face disruption due to damage at the Lamington Viaduct. Many businesses and householders were affected by flooding in Newton Stewart after the River Cree overflowed into the town. First Minister Nicola Sturgeon visited the area to inspect the damage. The waters breached a retaining wall, flooding many commercial properties on Victoria Street - the main shopping thoroughfare. Jeanette Tate, who owns the Cinnamon Cafe which was badly affected, said she could not fault the multi-agency response once the flood hit. However, she said more preventative work could have been carried out to ensure the retaining wall did not fail. "It is difficult but I do think there is so much publicity for Dumfries and the Nith - and I totally appreciate that - but it is almost like we're neglected or forgotten," she said. "That may not be true but it is perhaps my perspective over the last few days. "Why were you not ready to help us a bit more when the warning and the alarm alerts had gone out?" Meanwhile, a flood alert remains in place across the Borders because of the constant rain. Peebles was badly hit by problems, sparking calls to introduce more defences in the area. Scottish Borders Council has put a list on its website of the roads worst affected and drivers have been urged not to ignore closure signs. The Labour Party's deputy Scottish leader Alex Rowley was in Hawick on Monday to see the situation first hand. He said it was important to get the flood protection plan right but backed calls to speed up the process. "I was quite taken aback by the amount of damage that has been done," he said. "Obviously it is heart-breaking for people who have been forced out of their homes and the impact on businesses." He said it was important that "immediate steps" were taken to protect the areas most vulnerable and a clear timetable put in place for flood prevention plans. Have you been affected by flooding in Dumfries and Galloway or the Borders? Tell us about your experience of the situation and how it was handled. Email us on selkirk.news@bbc.co.uk or dumfries@bbc.co.uk.

Output:

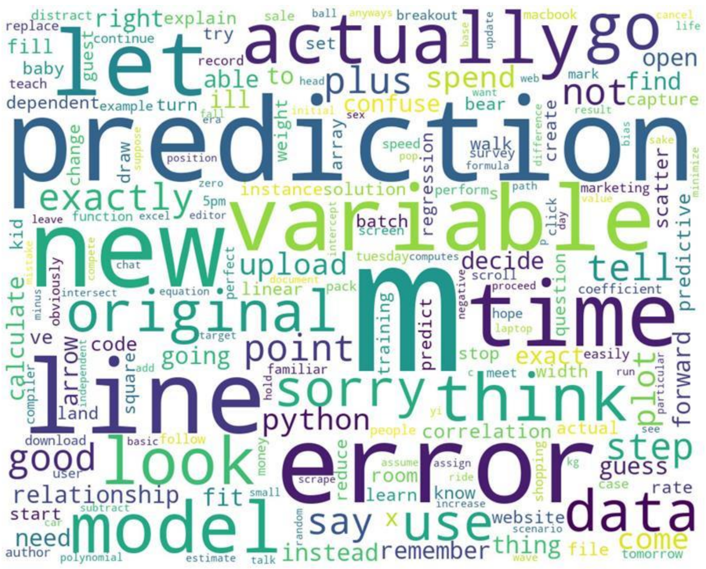
The Telegram bot, after summarization, will send the summarized text and the word cloud to a telegram group for the end-user to consume.



weight, and my prediction with my new prediction waves and squared because if it's negative, I wanted to cancel out.. So I'm saying okay so for my random estimates of A, which is six in this way, because I have an arrow.. Plus, so I put the formula here, so that you guys can see, sorry.. So, remember my be is three now, so it would be three times one, I know my go over your head.. Let's go, 0.01, a new learning rate, let's let's increase the value by 0.0 was a kid.. Yes yes yes yes yes yes yes yes, She is supposed to take Sorry, sorry, I made a mistake.. Take that prediction, my most the exact original prediction, which is six of the exact actual prediction which is six give me a new era, I scraped.. The only difference is, I'm adding 0.01, which is my landing weights to my initial guess.. A new guest this is just guessing, I decided to go with two plus three x.. Okay.. This one and give me that its own calculate that in the chat.. So the thing is, you're trying to reduce this particular error.. One more step one more step one more step I'm going to run out of time, competing pop, all of that.

And then you take the 2.27, and use that for your next three to four data points.. So think of it as a batch of calculating the errors based on batch.. So, first of all let me, upload solution to what we just did the Excel document to four.. Now I just have day six, but for the sake of this data itself just think of Yi is, is the intercept.. Okay, So, my new I have to assign zero, because it's an array.. So, it's 0.4 plus 0.8 but I need something to hold, whatever new predictions I have so I just gave it to just to narrate create an array for me.. Why, what I'm saying is, take a look through the table itself.. At one times the role number in that so this is just me coming up and then return what the new prediction is.. So what I'll have you guys do is shutter change 0.4 to a new guests.. So, I'm all decided that don't get confused yet don't get confused yet so and I wrote a quote that says, okay, instead of me, changing it manually myself because of time I'll be able to fully explain the call was was finished by seven.. Now I'm going to post another code that actually uses the real data sets.. What's the thing called my office hours is three to four on Fridays. 4:31 PM

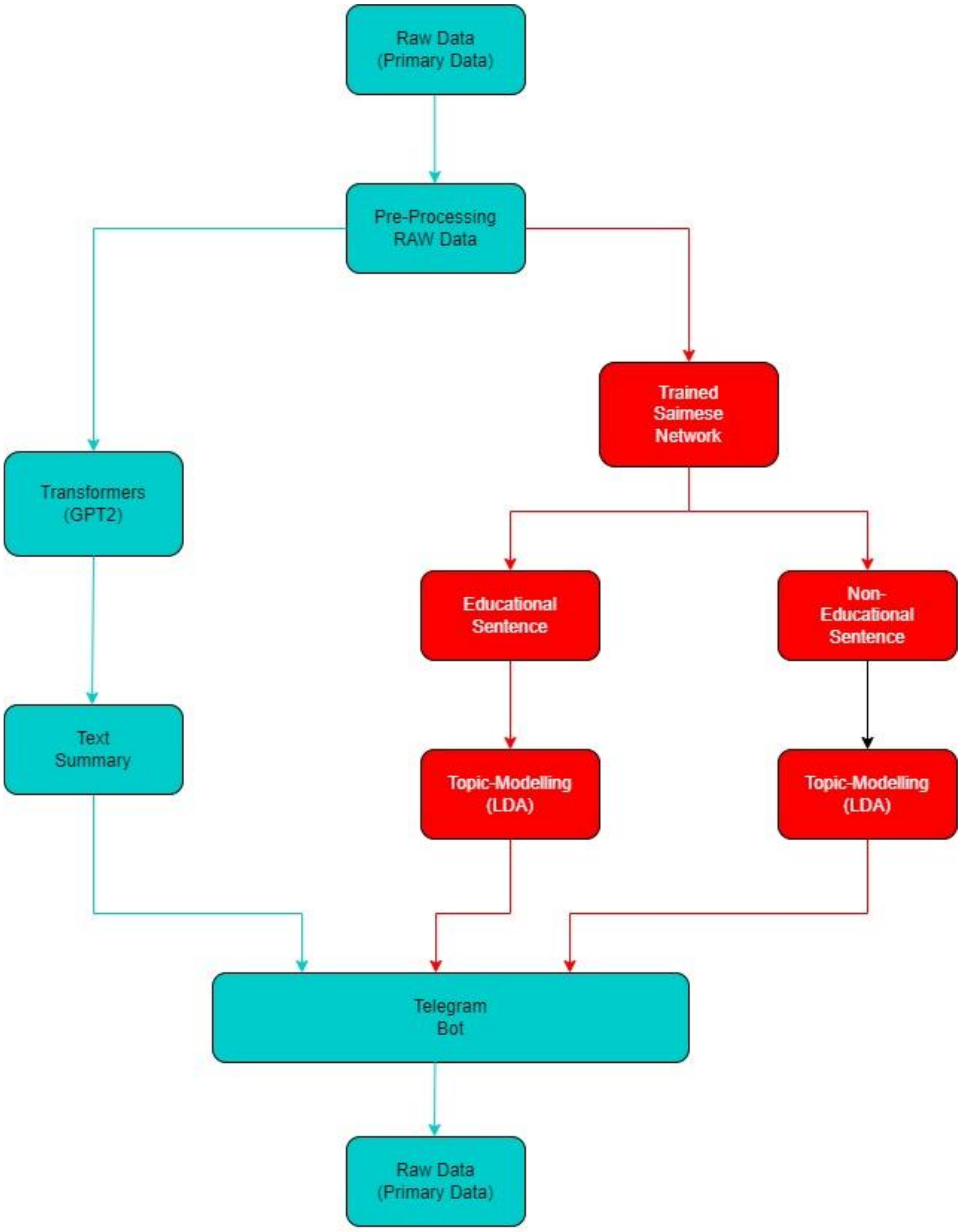
4:31 PM



Appendix A:

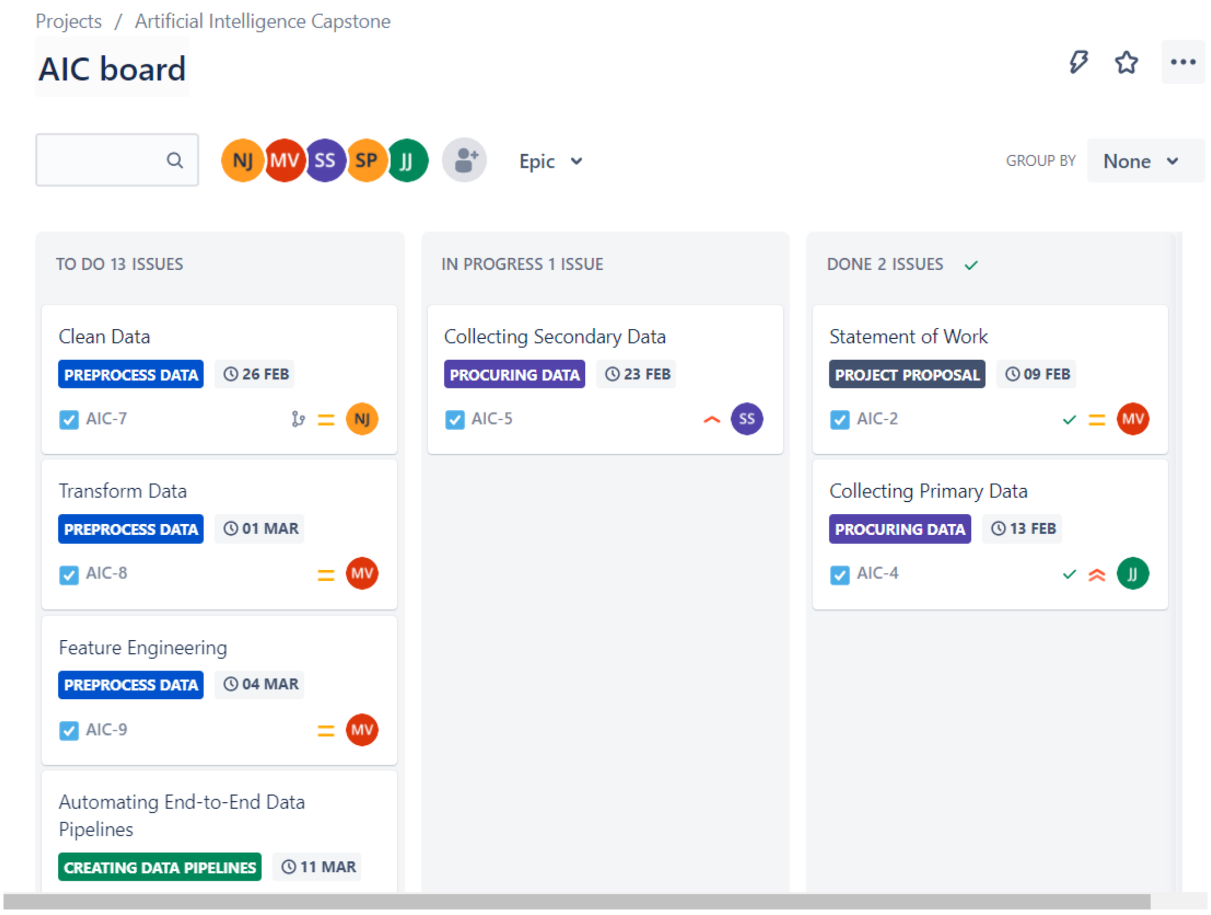
1. Project Flowchart

The flowchart depicted below shows the project's objective.



2. Project Board

The Project Board was created on the JIRA’s framework for tracking the project flow and synchronization.



Projects / Artificial Intelligence Capstone

AIC board

TO DO 13 ISSUES

Clean Data
PREPROCESS DATA · 26 FEB
AIC-7 · NJ

Transform Data
PREPROCESS DATA · 01 MAR
AIC-8 · MV

Feature Engineering
PREPROCESS DATA · 04 MAR
AIC-9 · MV

Automating End-to-End Data Pipelines
CREATING DATA PIPELINES · 11 MAR

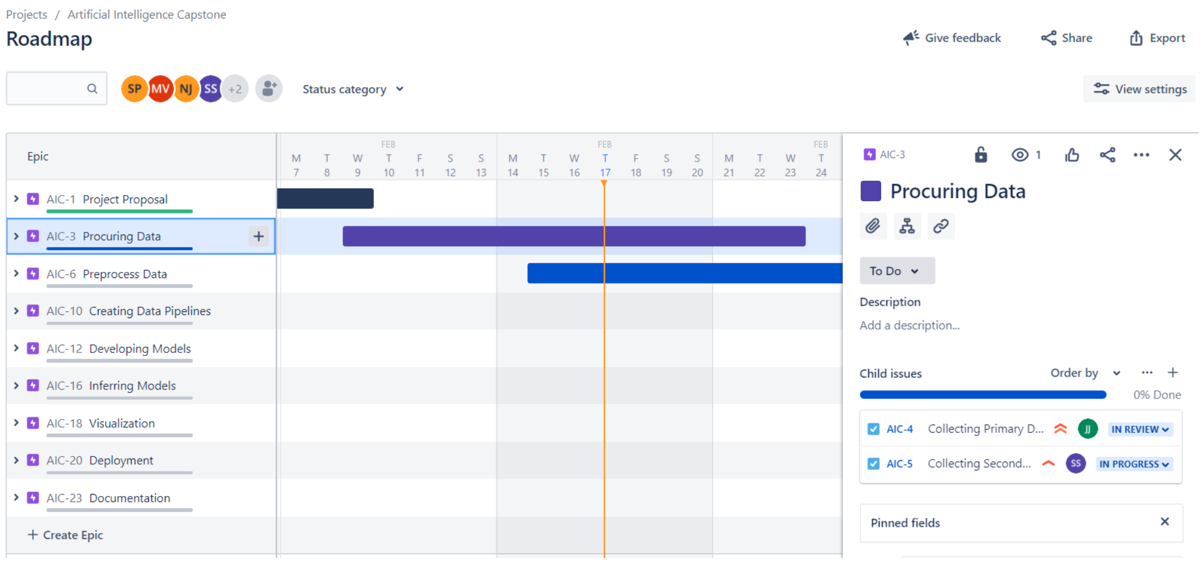
IN PROGRESS 1 ISSUE

Collecting Secondary Data
PROCURING DATA · 23 FEB
AIC-5 · SS

DONE 2 ISSUES ✓

Statement of Work
PROJECT PROPOSAL · 09 FEB
AIC-2 ✓ · MV

Collecting Primary Data
PROCURING DATA · 13 FEB
AIC-4 ✓ · JJ



Projects / Artificial Intelligence Capstone

Roadmap

Epic

- AIC-1 Project Proposal
- AIC-3 Procuring Data
- AIC-6 Preprocess Data
- AIC-10 Creating Data Pipelines
- AIC-12 Developing Models
- AIC-16 Inferring Models
- AIC-18 Visualization
- AIC-20 Deployment
- AIC-23 Documentation
- + Create Epic

AIC-3

Procuring Data

To Do

Description

Add a description...

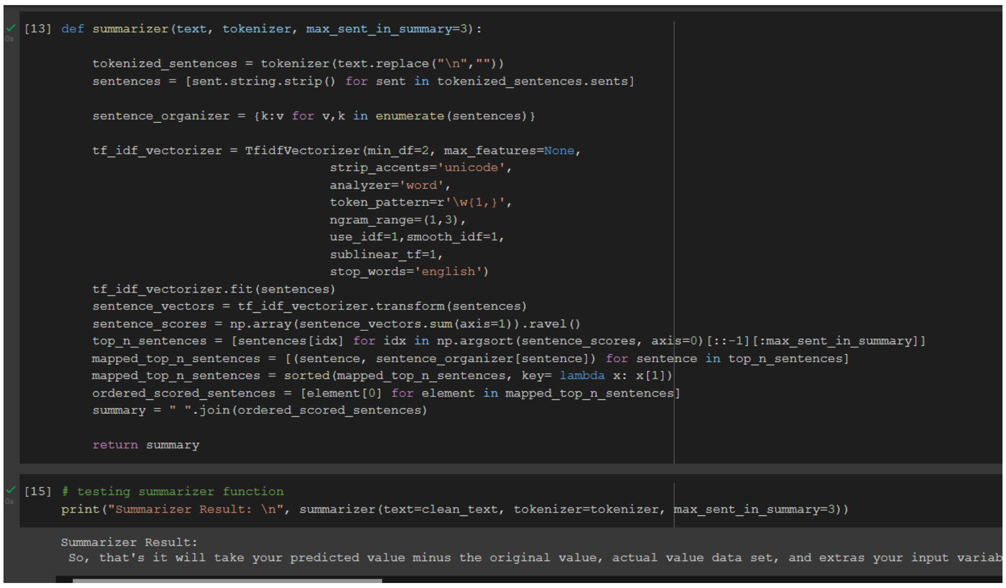
Child issues · Order by · 0% Done

AIC-4	Collecting Primary D...	JJ	IN REVIEW
AIC-5	Collecting Second...	SS	IN PROGRESS

Pinned fields

Appendix B:

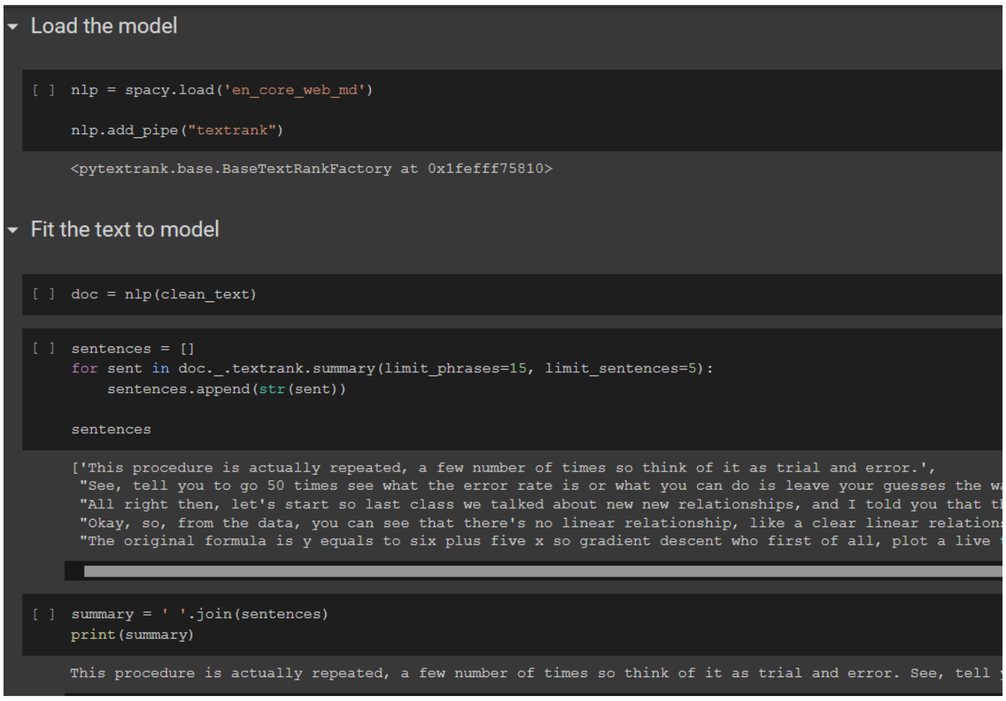
1. Implementation Code using TF-IDF Vectorizer



```
[13] def summarizer(text, tokenizer, max_sent_in_summary=3):

         tokenized_sentences = tokenizer(text.replace("\n",""))
         sentences = [sent.string.strip() for sent in tokenized_sentences.sents]

         sentence_organizer = {k:v for v,k in enumerate(sentences)}

         tf_idf_vectorizer = TfidfVectorizer(min_df=2, max_features=None,
                                             strip_accents='unicode',
                                             analyzer='word',
                                             token_pattern=r'\w{1,}',
                                             ngram_range=(1,3),
                                             use_idf=1,smooth_idf=1,
                                             sublinear_tf=1,
                                             stop_words='english')
         tf_idf_vectorizer.fit(sentences)
         sentence_vectors = tf_idf_vectorizer.transform(sentences)
         sentence_scores = np.array(sentence_vectors.sum(axis=1)).ravel()
         top_n_sentences = [sentences[idx] for idx in np.argsort(sentence_scores, axis=0)[::-1][:max_sent_in_summary]]
         mapped_top_n_sentences = [(sentence, sentence_organizer[sentence]) for sentence in top_n_sentences]
         mapped_top_n_sentences = sorted(mapped_top_n_sentences, key= lambda x: x[1])
         ordered_scored_sentences = [element[0] for element in mapped_top_n_sentences]
         summary = " ".join(ordered_scored_sentences)

         return summary
```

```
[15] # testing summarizer function
     print("Summarizer Result: \n", summarizer(text=clean_text, tokenizer=tokenizer, max_sent_in_summary=3))

     Summarizer Result:
      So, that's it will take your predicted value minus the original value, actual value data set, and extras your input variab
```

2. Implementation Code using Spacy's PyTextRank



Load the model

```
[ ] nlp = spacy.load('en_core_web_md')

    nlp.add_pipe("textrank")

    <pytextrank.base.BaseTextRankFactory at 0x1fefff75810>
```

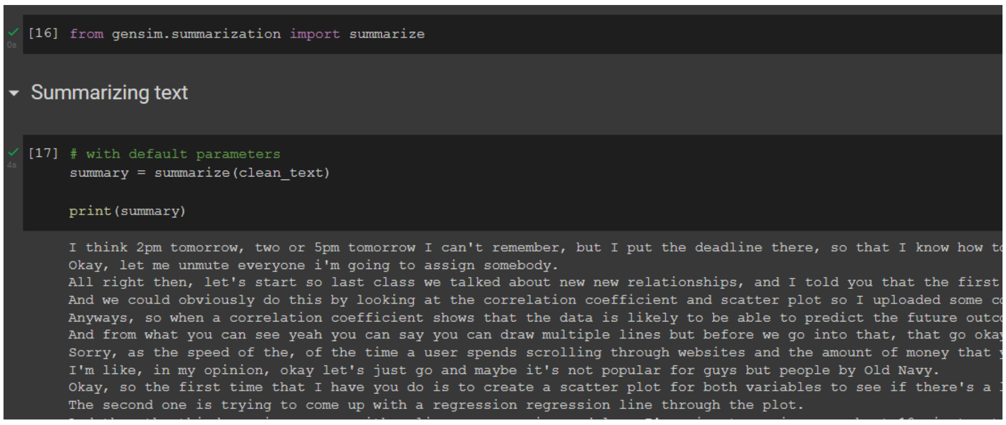
Fit the text to model

```
[ ] doc = nlp(clean_text)
```

```
[ ] sentences = []
    for sent in doc._.textrank.summary(limit_phrases=15, limit_sentences=5):
        sentences.append(str(sent))

    sentences

    ['This procedure is actually repeated, a few number of times so think of it as trial and error.',
     "See, tell you to go 50 times see what the error rate is or what you can do is leave your guesses the wa
     "All right then, let's start so last class we talked about new new relationships, and I told you that th
     "Okay, so, from the data, you can see that there's no linear relationship, like a clear linear relation
     "The original formula is y equals to six plus five x so gradient descent who first of all, plot a live
```

```
[ ] summary = ' '.join(sentences)
    print(summary)

    This procedure is actually repeated, a few number of times so think of it as trial and error. See, tell y
```

3. Implementation Code using Gensim’s Model

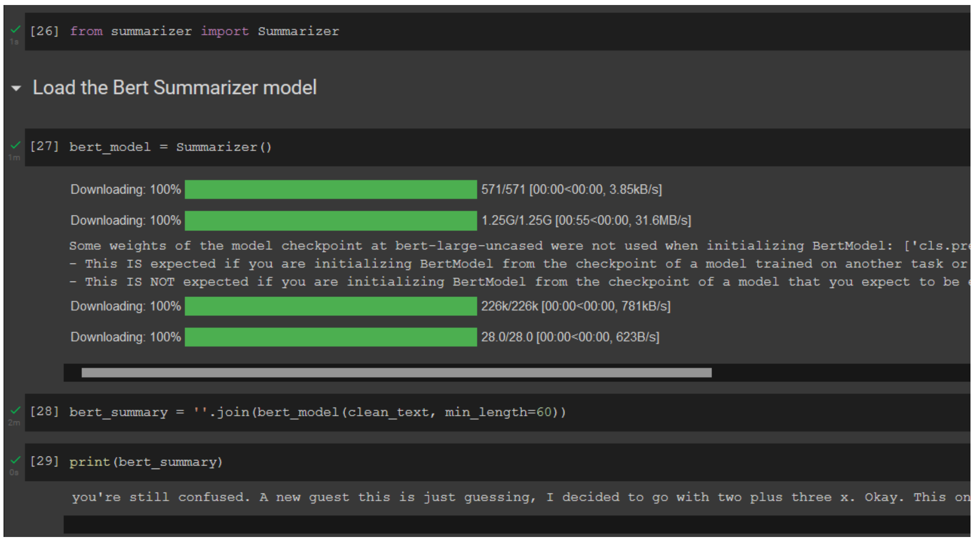


```
[16] from gensim.summarization import summarize
```

Summarizing text

```
[17] # with default parameters
     summary = summarize(clean_text)

     print(summary)

     I think 2pm tomorrow, two or 5pm tomorrow I can't remember, but I put the deadline there, so that I know how to
     Okay, let me unmute everyone i'm going to assign somebody.
     All right then, let's start so last class we talked about new new relationships, and I told you that the first
     And we could obviously do this by looking at the correlation coefficient and scatter plot so I uploaded some co
     Anyways, so when a correlation coefficient shows that the data is likely to be able to predict the future outco
     And from what you can see yeah you can say you can draw multiple lines but before we go into that, that go okay
     Sorry, as the speed of the, of the time a user spends scrolling through websites and the amount of money that y
     I'm like, in my opinion, okay let's just go and maybe it's not popular for guys but people by Old Navy.
     Okay, so the first time that I have you do is to create a scatter plot for both variables to see if there's a l
     The second one is trying to come up with a regression regression line through the plot.
```

4. Implementation Code using BERT Transformer



```
[26] from summarizer import Summarizer
```

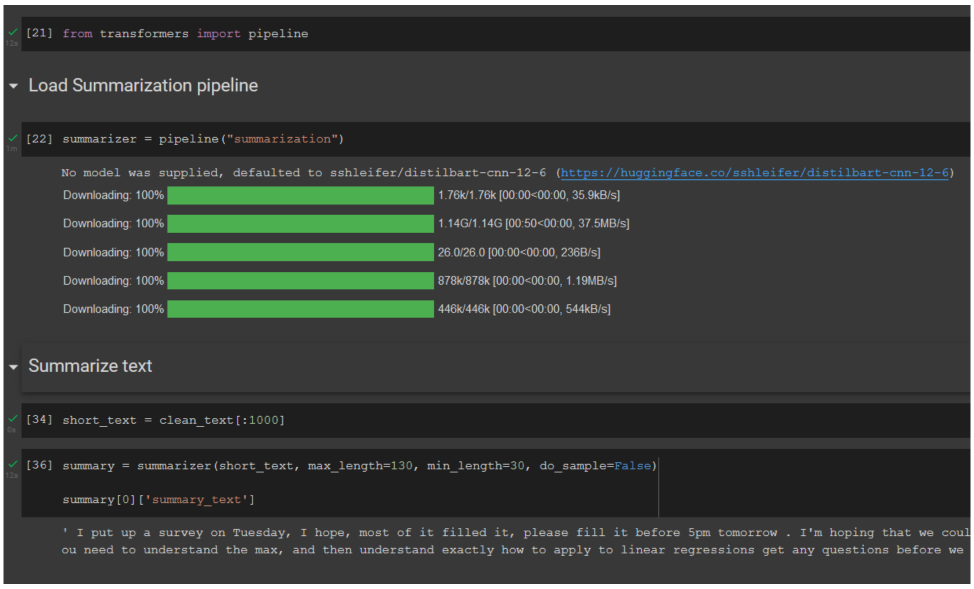
Load the Bert Summarizer model

```
[27] bert_model = Summarizer()

     Downloading: 100% 571/571 [00:00<00:00, 3.85kB/s]
     Downloading: 100% 1.25G/1.25G [00:55<00:00, 31.6MB/s]
     Some weights of the model checkpoint at bert-large-uncased were not used when initializing BertModel: ['cls.pre
     - This IS expected if you are initializing BertModel from the checkpoint of a model trained on another task or
     - This IS NOT expected if you are initializing BertModel from the checkpoint of a model that you expect to be e
     Downloading: 100% 226k/226k [00:00<00:00, 781kB/s]
     Downloading: 100% 28.0/28.0 [00:00<00:00, 623B/s]
```

```
[28] bert_summary = ''.join(bert_model(clean_text, min_length=60))
```

```
[29] print(bert_summary)

     you're still confused. A new guest this is just guessing, I decided to go with two plus three x. Okay. This on
```

5. Implementation Code using BART Transformer



```
[21] from transformers import pipeline
```

Load Summarization pipeline

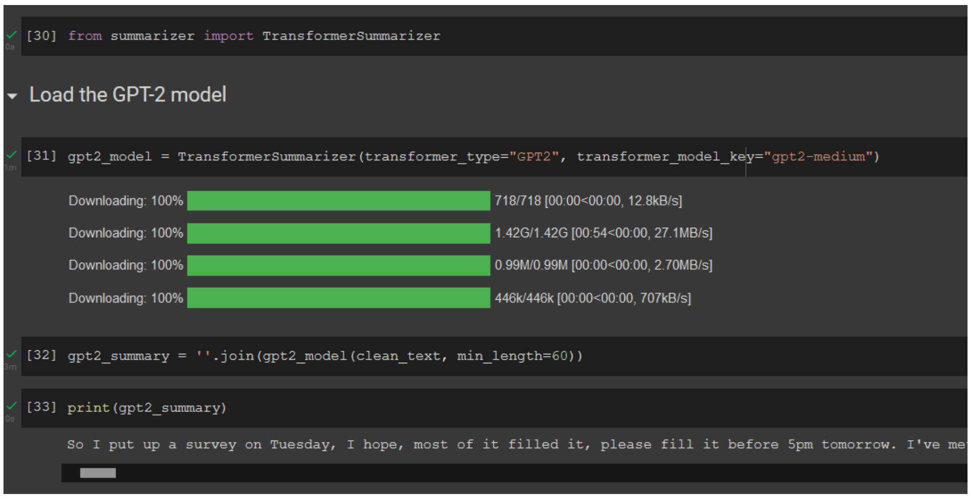
```
[22] summarizer = pipeline("summarization")

     No model was supplied, defaulted to sshleifer/distilbart-cnn-12-6 (https://huggingface.co/sshleifer/distilbart-cnn-12-6)
     Downloading: 100% 1.76k/1.76k [00:00<00:00, 35.9kB/s]
     Downloading: 100% 1.14G/1.14G [00:50<00:00, 37.5MB/s]
     Downloading: 100% 26.0/26.0 [00:00<00:00, 236B/s]
     Downloading: 100% 878k/878k [00:00<00:00, 1.19MB/s]
     Downloading: 100% 446k/446k [00:00<00:00, 544kB/s]
```

Summarize text

```
[34] short_text = clean_text[:1000]
```

```
[36] summary = summarizer(short_text, max_length=130, min_length=30, do_sample=False)

     summary[0]['summary_text']

     ' I put up a survey on Tuesday, I hope, most of it filled it, please fill it before 5pm tomorrow . I'm hoping that we coul
     ou need to understand the max, and then understand exactly how to apply to linear regressions get any questions before we
```

6. Implementation Code using GPT2 Transformer



```
[30] from summarizer import TransformerSummarizer
```

Load the GPT-2 model

```
[31] gpt2_model = TransformerSummarizer(transformer_type="GPT2", transformer_model_key="gpt2-medium")

Downloading: 100%  718/718 [00:00<00:00, 12.8kB/s]
Downloading: 100%  1.42G/1.42G [00:54<00:00, 27.1MB/s]
Downloading: 100%  0.99M/0.99M [00:00<00:00, 2.70MB/s]
Downloading: 100%  446k/446k [00:00<00:00, 707kB/s]
```

```
[32] gpt2_summary = ''.join(gpt2_model(clean_text, min_length=60))
```

```
[33] print(gpt2_summary)

So I put up a survey on Tuesday, I hope, most of it filled it, please fill it before 5pm tomorrow. I've me
```