**Project Name:** Text Summarization

**Github Link:** https://github.com/utkrisht2000/Text-Summarization.git

**Why was this project created?**

In the modern world, there is a wealth of knowledge that may be used to explore many new options. One of the key uses for natural language processing is text summarization. One of the popular techniques for processing a text corpus to produce a precise text that catches the complete context and maintains the key information being communicated through the text is text summarization. The summary has been improved by the addition of additional words and phrases.

**What problem is it solving?**

The challenge of text summarization is to condense a document's meaning into fewer phrases and words. The methods used to extract data from unstructured text data and utilize it for a summarization model may be broadly divided into Extractive and Abstractive methods. The result summary is only a subset of the entire text since extractive algorithms choose the most significant phrases within a text. Contrarily, Abstractive models make use of sophisticated Natural Language Processing techniques, such as word embeddings, to comprehend the semantics of the text and produce an insightful summary. As a result, abstractive approaches require a large amount of parameters and data, making them far more difficult to train from scratch. So we are going to use the Seq2seq Model.

**Entire explanation of project**

* **PROPOSED APPROACH**

In order to keep the dataset light, just load the first N items from the collection of 30 000 articles. The details stated in the summary were manually annotated in red. Sports stories are difficult for computers to read since there isn't much room for interpretation of what is significant and what is not, and the headline must state the key outcomes. After breaking up the text into sentences, the algorithm creates a graph with the sentences as the nodes and the links as the overlapping words. The most significant sentences in this sentence network are identified by PageRank. The majority of the information from the original summary is included in the forecast. The projected summary is entirely contained in the text, as would be anticipated from an extractive algorithm.

Using neural networks, sequence-to-sequence models produce a new sequence in a different domain from an input sequence from a certain domain. Using the same code as previously, we can now generate the feature matrix with the summaries. You ought to be able to see the special tokens at the top if you print the y vocabulary. The predicted text will start when the start token arrives in a later prediction, and it will terminate when the end token appears. Moving on to the Word Embeddings now. Either utilise a pre-trained model or train our word embedding model from scratch.

Once the context of the input text has been understood by the encoder, you can next show the decoder how the summary begins so that the model can learn to anticipate how it will conclude. In order for the network to learn to anticipate the word that will come after the start token, the next word, and so on, a training technique known as "teacher forcing" is used. Even when the training is completed, the work is not yet done! As the final stage, we must construct the Inference Model to produce predictions in order to test the Seq2Seq model. A fresh sequence (X test) is fed into the prediction encoder, which outputs the final LSTM layer's output along with its states.

Algorithm for creating next word prediction model :

**Step 1:** Dataset is imported

**Step 2:** Data Exploration

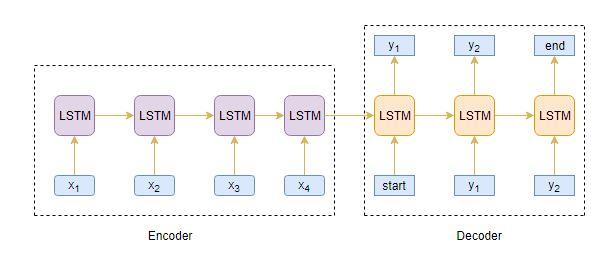
**Step 3:** Data Preparation

**Step 4:** Tokenization

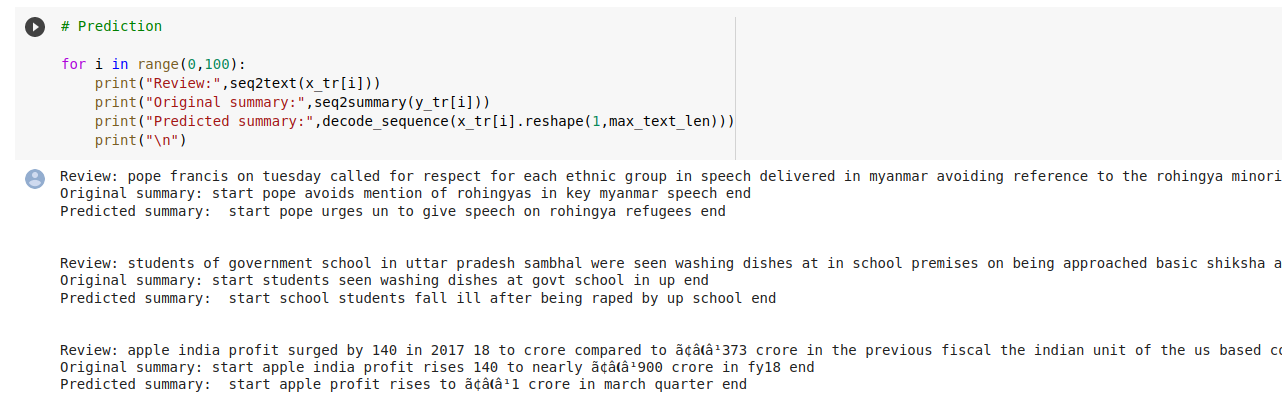
**Step 5:** Model Fitting

**Step 6:** Prediction

* **DATA FLOW DIAGRAM**



* **RESULT**



* **CONCLUSION**

In the area of abstractive summarization, deep learning-based methods have shown encouraging results. To achieve successful text summarization results, the encoder-decoder paradigm has been successfully combined with an attention mechanism. The model is adjusted appropriately to create summaries that resemble those written by humans by computing the attention vector.