# Abstractive Summarization

## Mandate III

- **Problem Statement:** **Given a set of tweets pertaining to a trending topic, create an abstractive prose summary of the tweets. Do not just string the tweets together to form the summary. The summary will need to paraphrase and/or say more than what is directly said in the tweets. Propose a rubric to evaluate the accuracy of your summarization.**
- **Code:** **https://github.com/rittik9/Abstractive-Summarization-Of-Tweets**
- **Work Done:** **1. Dataset Creation Using Classical NLP**
  **2. Finetuning various pretrained large language models using our custom dataset**

1. **Dataset Creation Using Classical NLP:** Classical NLP, also known as traditional or rule-based NLP, refers to the early approaches and techniques used in the field of natural language processing (NLP) before the advent of deep learning and neural networks. Classical NLP techniques typically rely on rule-based methods, handcrafted features, and statistical algorithms to analyze, understand, and process human language.

As mentioned in the earlier mandate,After collecting the tweets and cleaning & pre processing them I took two approaches to create summaries for the data points in our data set.

1. I used state-of-the art Pegasus model for summary generation of few data points.
2. I used OpenAI's API to automate Chatgpt for generating summaries of our data points .

In this mandate I also tried to explore various classical nlp approaches like - Text Ranking Using Google Page Rank algorithm, POS Tagging using Hidden Markov Models, Data generation using Markov chains to generate summaries for our datapoints.

**Approach:**

**I.** At first given a preprocessed and clean datapoint (A group of sentences) I performed pos tagging on each of them using nltk's pos_tag() function which uses a rule-based approach, which is based on a pre-trained set of hand-crafted rules to assign Part-of-Speech (POS) tags to words in a given text.

**II.** Then I made a list of important pos tags as per my knowledge

```
important_pos_tags = ["NN", "NNS", "VB", "VBD", "VBG", "VBN", "VBP", "VBZ", "JJ", "JJR", "JJS"]
```

And stored those words of our datapoint into a list whose pos tag matches any of our important pos tags.

**III.** Then I made a markov chain model for next word generation (got the transition probabilities of important words using our original datapoints) which randomly chooses important words from the list and generates next word based on transition probabilities.

**IV.** Doing this I generated a group of new sentences from our original datapoint.

**V.** Finally I used google page rank algorithm as mentioned in the previous mandate to rank the newly generated sentences and took the top 50% sentences of my datapoint's length as a summary of my datapoint.

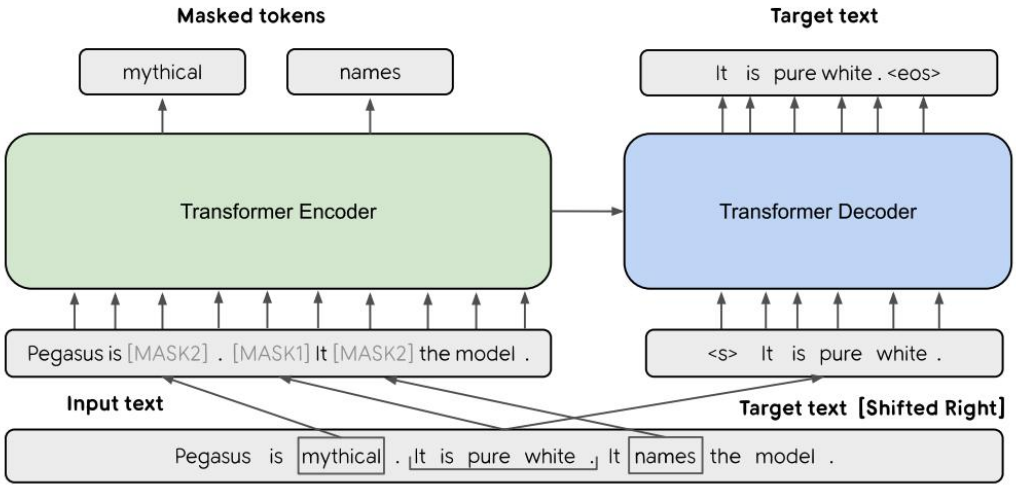
**VI.** I also tried to do POS tagging using spacy which uses Hidden Markov Models.

**VII.** I also tried to do summary generation using POS tagging .At first I did POS tagging on each of the sentences. Then counted how many important POS tags appeared on each of the sentences. Then ranked the sentences in a order that sentence with highest no. of important POS tags gets the top rank and sentences with lowest no. of POS tags gets the low rank.Then picked up the top (50% of the total length of the original datapoint) ranked sentences as summary of the given datapoint.

**2. <u>Finetuning various pretrained large language models using our custom dataset:</u>**

As I mentioned in the previous mandate I explored various pretrained attention based seq2seq models from Huggingface's Transformer Library for abstractive text summarization.In this mandate I used some of them for finetuning purpose on our custom dataset.

**I.<u>Pegasus(Pre-training with Extracted Gap-sentences for Abstractive Summarization</u>)** - It is developed by Google AI, Pegasus is a transformer-based model that was specifically designed for abstractive text summarization.Instead of the standard masked language modeling or next sentence prediction tasks, PEGASUS is trained using a gap-sentence generation task, where it is tasked with generating missing sentences.

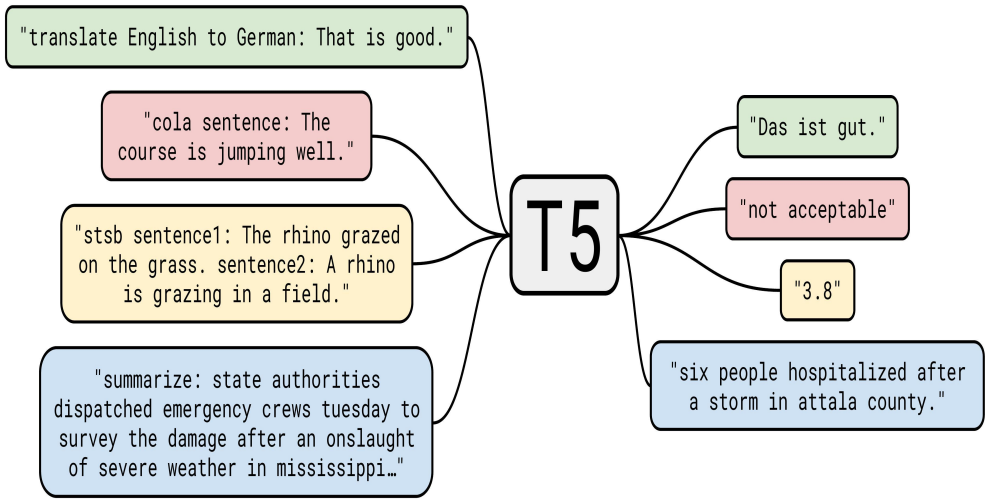



The base architecture of PEGASUS is a standard Transformer encoder-decoder. Both GSG and MLM are applied simultaneously to this example as pre-training objectives. Originally there are three sentences. One sentence is masked with

[MASK1] and used as target generation text (GSG). The other two sentences remain in the input, but some tokens are randomly masked by [MASK2] (MLM).

I fine tuned pre-trained "google/pegasus-cnn_dailymail" model on my custom dataset for 10 epochs and evaluated on test data using Rouge score and Bert score metrics.

**II. T5 (Text-to-Text Transfer Transformer) -** It is developed by Google AI.It is a variant of the Transformer model architecture that has been trained on a large corpus of text data to perform a wide range of natural language processing (NLP) tasks, including abstractive summarization.T5 is well-suited for abstractive summarization due to its ability to understand the context and meaning of text at a deep level.T5 uses positional encoding to provide information about the position of words in the input text, which helps the model to understand the sequential order of words in the text..T5 can be used for abstractive summarization by fine-tuning the pre-trained model on a specific dataset of paired source texts and their corresponding summaries. During fine-tuning, the model learns to generate summaries by predicting the most probable words or phrases based on the input text.




The T5 model architecture consists of the following components:

**a.Encoder:** T5 uses a multi-layer bidirectional Transformer encoder that takes in the input text and encodes it into a series of hidden representations. The encoder consists of multiple layers of self-attention and feed-forward neural networks, which allow the model to capture contextual information from the input text.

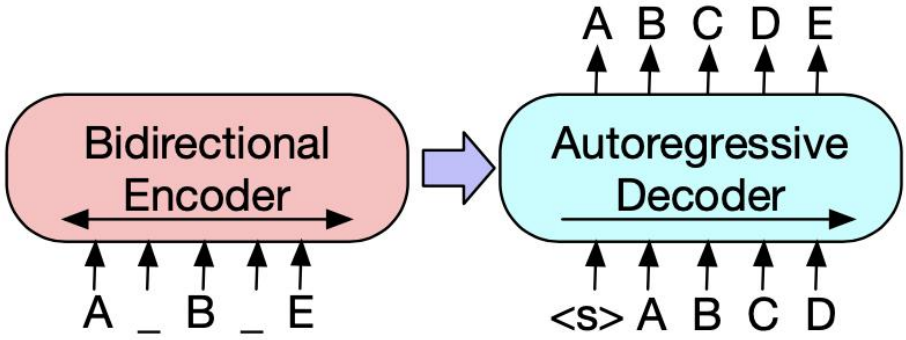
**b.Decoder**: T5 also includes a decoder, which is used during text generation tasks.The decoder is similar to the encoder in terms of architecture, with multiple layers of self-attention and feed-forward neural networks. The decoder takes in the encoded representations from the encoder and generates output text sequentially, one token at a time.

One of the unique aspects of T5 is its "text-to-text" framework, where all tasks are framed as text-to-text problems. This means that both the input and output of the model are treated as text, regardless of the type of NLP task being performed. This allows T5 to handle a wide range of tasks, including text classification, text summarization, machine translation, question answering, and text generation, in a unified manner.

I fine tuned pre-trained 't5-base' on my custom dataset for 10 epochs and evaluated it on test dataset using Rouge and Bert Score Metrics.

**III.BART (Bidirectional and Auto-Regressive Transformer) -** developed by Facebook AI Research (FAIR).It is a sequence-to-sequence (Seq2Seq) model for abstractive text summarization which builds upon the transformer architecture and incorporates both autoencoding and autoregressive components to generate summaries from input text.

BART has several unique features that distinguish it from other models used for abstractive text summarization:




**Bidirectional encoding**: Unlike many other text summarization models that rely solely on unidirectional encoding, BART incorporates bidirectional encoding using a transformer-based encoder. This allows BART to capture contextual representations of the input text from both the left and right contexts, which helps in generating more coherent and contextually relevant summaries.

**Autoencoding and autoregressive components**: BART combines the strengths of autoencoding and autoregressive approaches. During pre-training, it uses a masked language model (MLM) objective to learn bidirectional representations of text, similar to BERT. It also includes a denoising autoencoder (DAE) objective to learn to generate abstractions while reconstructing the input text. This combination of autoencoding and autoregressive components makes BART versatile for both encoding and decoding, and enables it to generate summaries that are both fluent and abstractive.

BART is pre-trained on a large corpus of text data using the MLM and DAE objectives, which helps it learn general language representations.

I fine tuned pre-trained 'facebook/bart-large' on my custom dataset for 3 epochs and evaluated it on test dataset using Rouge and Bert Score Metrics.

## ● Results On Unseen Test Data:

| Model | #Epochs | Rouge1 | Rouge2 | RougeL | RougeLsum | Bert Score | | |
|---|---|---|---|---|---|---|---|---|
| | | | | | | Precision | Recall | F1 |
| Pretrained t5 | | 0.2101 | 0.082602 | 0.1554 | 0.1555 | 0.8376 | 0.8339 | 0.8351 |
| Finetuned t5 | 10 | 0.4195 | 0.2052 | 0.3173 | 0.3176 | 0.8887 | 0.8787 | 0.8833 |
| Finetuned BART | 3 | .2609 | .1600 | .2323 | .2476 | 0.9151 | 0.8415 | 0.8765 |
| Pretrained Pegasus | | .0175 | .0001 | 0.01706 | .01708 | 0.8389 | 0.8437 | 0.8409 |
| Finetuned Pegaus | 10 | .0129 | .0002 | .01294 | .01293 | 0.9415 | 0.9350 | 0.9381 |

## ● Observations:

**1.** I observed the summaries generated using classical NLP approaches, even though they are syntactically and grammatically correct they are not always semantically meaningful.

**2.**ROGUE measures the similarity between the generated summary and the reference summary in terms of overlapping n-grams (contiguous sequences of n words) or word sequences.It calculates the percentage of overlapping n-grams (typically, 1-gram, 2-gram, or 3-gram) between the generated summary and the reference summaries. Higher overlap indicates greater similarity.So when the no. of tokens increases in the reference and generated summaries the rogue score between them tends to decrease even though both the texts have semantically similar meaning.In my datapoints summary has 300 tokens. That's why we tend to observe pretty low rogue score almost in all the models.

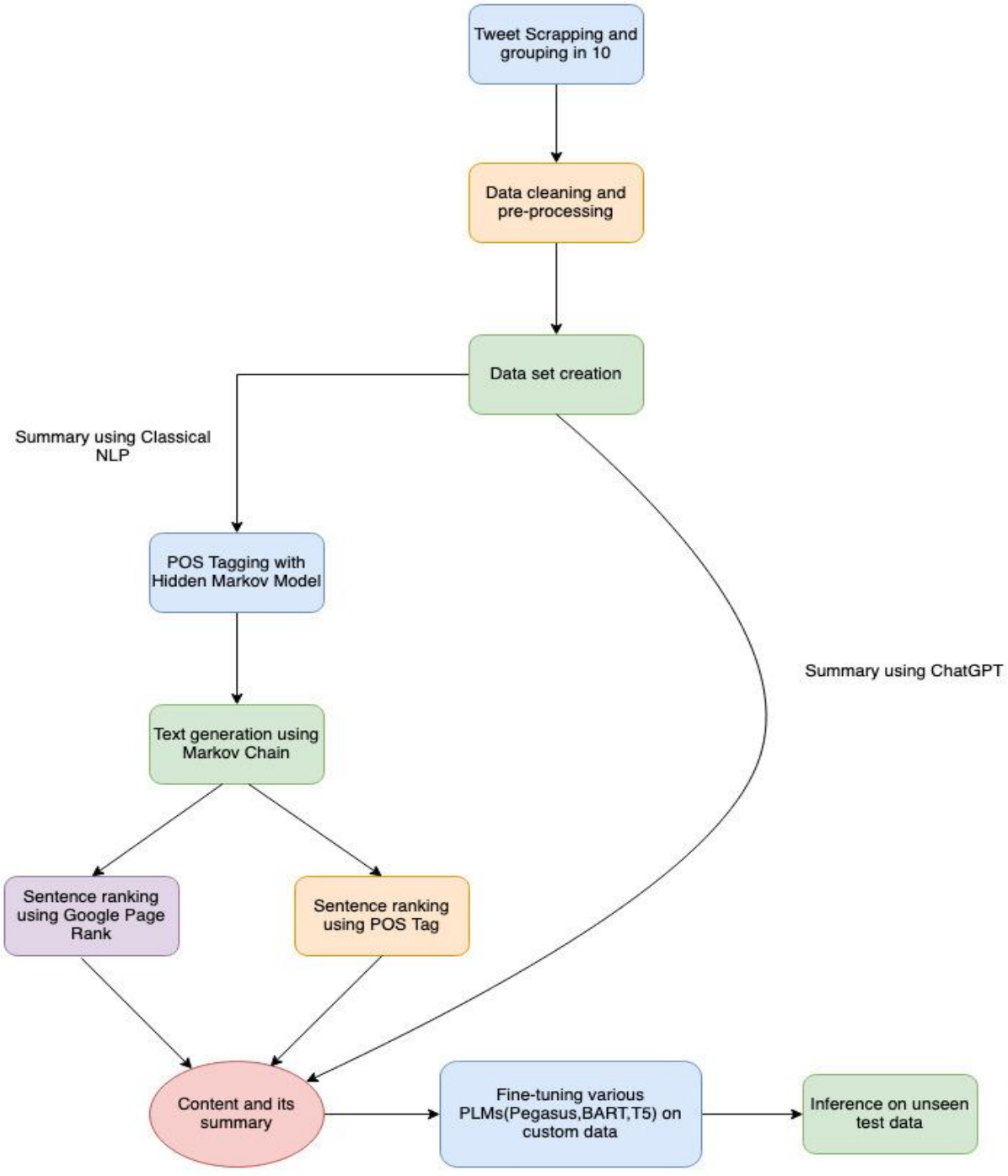
**3.**BERT scores sentences based on their semantic similarity to the summary generated so far, their saliency or importance in the source text, or other relevant criteria. These scoring criteria can be determined based on the specific requirements of the summarization task and the design of the summarization model.BERT scoring can be integrated into the overall architecture of the abstractive summarization model. BERT's relevance scores can be used to guide the selection or generation of sentences or phrases in the summary, helping to ensure that the generated summary is coherent, relevant, and representative of the source text which can not be done by rouge score. High Bert scores by our fine tuned models indicate good quality summary generation given a set of tweets & in case of fine-tuned pegasus rogue score decreases(rouge2 increases because it takes contextual meaning in to account by using bigram) after fine tuning while bert scores increases significantly.

**Comparison Between Rouge and Bert Score :1.The book is written by me 2.I am the author of that book . We can see that both the sentences carry same meaning.But Rouge score indicates low similarity**

| | Rouge1 | Rouge2 | RougeL | RougeLsum |
|---|---|---|---|---|
| scores | O.3076 | 0.0 | 0.3076 | 0.3076 |

**While Bert Score indicates high similarity:Precision = 0.9171 ,Recall=0.9120,F1-Score=0.9146**

- ## Full Project Flowchart:




---

- ## Future Work:

1. I will try more PLM models like BERT and GPT2.

2. Then I will push fine-tuned model on Huggingface model hub and will generate api for the model.
3. At last,I will create an api for the full project using Flask.