MANDATE 4
ABSTRACTIVE SUMMARIZATION OF TWEETS

[Colab Notebook]
[Dataset]

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.

DATASET:

Dataset contains tweets of multiple topics along with the relevant hashtags. The dataset is created by collecting the tweets and corresponding summaries are generated using external sources. The dataset is in the form of Comma-separated values containing input tweets and their summaries. The dataset contains a total of 1351 tweets along with their summaries. The dataset is split into train, validation and test set in the ratio of 80:10:10.

```
DatasetDict({
    train: Dataset({
        features: ['inputs', 'summaries'],
        num_rows: 1080
    })
    valid: Dataset({
        features: ['inputs', 'summaries'],
        num_rows: 135
    })
    test: Dataset({
        features: ['inputs', 'summaries'],
        num_rows: 136
    })
})
```

PREPROCESSING :

Tweets usually contain hashtags, emojis, links etc. So, before applying our model, we have to clean the dataset. To remove these unwanted characters, I have used the tweet-preprocessor library which is twitter specific preprocessing library.

Below are few examples of text cleaning using tweet-preprocessor

```
Nothing beats a morning run to start the day off right! 🏃‍♂️☀️ #morningmotivation
Attended a yoga class for the first time today and it was such a calming experience 🧘‍♀️👩‍🏫 #yogalife
Just finished reading a book on mindfulness and it has changed my perspective on life 📚🙏 #mindfulness
Started a new workout routine and already feeling the burn 🔥💪 #workoutmotivation
Took a dance class for the first time in years and it was so much fun! 💃🎉 #dancelife
Just finished a challenging hike and the view at the top was worth it! 🏞️🥾 #hikingadventures
```

Before preprocessing

```
Nothing beats a morning run to start the day off right!
Attended a yoga class for the first time today and it was such a calming experience
Just finished reading a book on mindfulness and it has changed my perspective on life
Started a new workout routine and already feeling the burn
Took a dance class for the first time in years and it was so much fun!
Just finished a challenging hike and the view at the top was worth it!
```

After preprocessing

SOUNDEX

I have used soundex to handle lexical variations in tweets. Often, the words we use while writing a tweet/message vary from person to person. Following can be the lexical variations of the word "tomorrow": ‘2marrow’, ‘2morow’, ‘2morrow’, ‘tmmrw’, , ‘tmorrow’ ‘tmrow’, ‘tmrrow’, ‘tmrrw’, ‘tmrw’, ‘tmrww’, ‘tmw’, Thus, we need to normalize the words. I tried soundex to normalize them.

NOTE: THIS APPROACH DIDN’T WORK.

This didn’t work because the final encoding produced by the soundex function didn’t match the original words’ encoding. For example,
Encoding of tomorrow - T560
Encoding of 2mrw - 2560

We can try resolving this by defining our own dictionary which maps the digits to their corresponding character. But again, that is not a concrete solution.

A solution to this problem is proposed in this paper.

PLM : Pegasus(Xsum)

The model that I have used is trained on the Xsum dataset which is a document summarization dataset.

TOKENIZATION:

Before we can feed those texts to our model, we need to preprocess them. This is done by a Pegasus Tokenizer which will (as the name indicates) tokenize the inputs (including converting the tokens to their corresponding IDs in the pretrained vocabulary) and put it in a format the model expects, as well as generate the other inputs that the model requires.

To do all of this, we instantiate our tokenizer with the PegasusTokenizer.from_pretrained method, which will ensure:

- We get a tokenizer that corresponds to the model architecture we want to use,
- We download the vocabulary used when pretraining this specific checkpoint.

That vocabulary will be cached, so it's not downloaded again the next time we run it.

FINE TUNING:

1) Freezing the embeddings - To save the computation time in the Transformer package, embeddings are pre-computed up to the length of 512 and stored as a variable that serves as a cache which should not change during training.

The reason for not training the position embeddings is that the embeddings for later positions would be undertrained, but with cleverly analytically defined position embeddings, the network can learn the regularity behind the equations and generalize for longer sequences more easily.

2) Optimizers - Using optimizers can accelerate the performance of a PLM. I have used the AdaFactor optimizer which is used for adaptive learning rates.

3) LR Schedulers : These schedulers are often employed to adjust the learning rates as the training epoch progresses. The input layers or bottom layers usually prefer a bigger learning rate than the output layer. This decaying or annealing of learning rate is controlled by LR schedulers.

4) Regularization - Regularization techniques such as dropout and weight decay can help prevent overfitting to the training data. Here, I am using the weight decay technique.

All these fine-tuning steps can be done using a Trainer class. Trainer is a simple but feature-complete training and eval loop for PyTorch, optimized for Transformers. Since these transformers are used to develop the sequence-to-sequence paradigm, I have used a Seq2SeqTrainer class.

OUTPUT :

```
['India Against Corruption is a grassroots movement aimed at combating corruption in Indian society and promoting tr
ansparency and accountability in government.',
 'Twitter is constantly evolving and changing, with new features and updates being rolled out regularly. Let us stay
informed and adapt to these changes to make the most of this platform.',
 'Apple has always been at the forefront of innovation, and with the recent launch of the iPhone , they have once ag
ain proven why they are the leading technology company in the world. The new device boasts improved cameras, a faste
r processor, and a longer battery life, making it a must-have for anyone looking for the latest and greatest smartph
one.',
 "Tesla's Model S Plaid is the fastest production car ever made, with mind-boggling acceleration, top speed, and per
formance that puts it ahead of the competition.",
 "Microsoft's cloud computing platform, Azure, is one of the fastest-growing cloud services in the world. It offers
a wide range of services, including virtual machines, data storage, and machine learning."]
```

Test tweets

```
['This tweet introduces the India Against Corruption movement, highlighting its goal of eradicating corruption in In
dian society and promoting government transparency and accountability.',
 "A recognition of Twitter's constantly evolving nature and a call to stay informed and adapt to changes.",
 "Apple's iPhone launch shows their commitment to innovation with improved cameras, faster processors and a longer b
attery life.",
 "Tesla's Model S Plaid is the fastest production car ever made, with unmatched acceleration, top speed, and perform
ance.",
 'Azure is a rapidly-growing cloud computing platform offered by Microsoft, with a variety of services including mac
hine learning and data storage.']
```

Given summaries

```
["This tweet notes the grassroots movement's aim to combat corruption in Indian society and promote transparency and
accountability in government.",
 'A call to stay informed and adapt to the latest features and updates on Twitter.',
 "Apple's iPhone offers improved cameras, faster processor, and longer battery life.",
 "Tesla's Model S Plaid is the fastest production car ever, with incredible acceleration, top speed, and performanc
e.",
 "Microsoft's cloud computing platform, Azure, offers a wide range of services, including virtual machines, data sto
rage, and machine learning."]
```

Generated summaries

Epoch	Training Loss	Validation Loss	Rouge1	Rouge2	Rougel	Rougelsum	Gen Len
1	1.563500	1.074569	64.337232	46.307959	58.528214	58.535784	26.540741
2	1.012000	1.091732	59.396322	41.513633	54.007965	54.045972	21.829630
3	0.692200	1.226390	62.350464	43.535124	56.860943	57.119873	24.992593
4	0.575200	1.413349	62.216377	43.183369	56.427634	56.550777	24.822222

Metrics for training data

```
output.metrics
```

```
{'test_loss': 1.3670787811279297,
 'test_rouge1': 61.87638352369107,
 'test_rouge2': 43.52033446338909,
 'test_rougeL': 56.54791548153695,
 'test_rougeLsum': 56.614247582772805,
 'test_gen_len': 23.095588235294116,
 'test_runtime': 89.6402,
 'test_samples_per_second': 1.517,
 'test_steps_per_second': 0.759}
```

Metrics for test data

EVALUATION - MEASURES :

Recall Oriented Understudy for Gisting Evaluation(ROGUE) is essentially a set of metrics for evaluating automatic summarization of texts as well as machine translations.

The reason to choose rogue as the evaluation measure is because rogue scores capture the fluency of the summary. The intuition behind it is that if the output summary follows the word ordering of the reference summary more closely, it will be more fluent.

REFERENCES :

Model

https://huggingface.co/docs/transformers/model_doc/pegasus

Fine-tuning

https://towardsdatascience.com/advanced-techniques-for-fine-tuning-transformers-82e4e61e16e#6196

https://huggingface.co/learn/nlp-course/chapter7/5?fw=tf

https://sandipanweb.wordpress.com/2020/08/30/named-entity-recognition-ner-on-twitter-with-bi-directional-lstm-with-tensorflow-in-python/#:~:text=NER%20is%20a%20common%20task,named%20 entities%20from%20Twitter%20texts.

Word normalization
http://www.cs.uccs.edu/~jkalita/work/reu/REUFinalPapers2010/Kaufmann.pdf

Pegasus research paper
https://arxiv.org/pdf/1912.08777.pdf