Text Summarization using Transformer-based Models

Lavya Midha, Jasmine Alarcon

Abstract

Automatic text summarization is a pivotal domain within natural language processing, aimed at producing concise and accurate summaries of extensive texts while retaining essential information. Machine learning techniques have proven to be effective tools for this task, utilizing sophisticated algorithms such as sequence-to-sequence models with transformer architectures and text embeddings. In this study, we conduct a comparative analysis of two transformer-based machine learning models to ascertain their suitability for summarization tasks on our specific dataset. We evaluate a Bidirectional Encoder Representations from Transformers (BERT) model and a Text-To-Text Transfer Transformer (T5) Model for their summarization performance, evaluating their efficacy based on the Rouge-score metric. Our findings reveal that, following fine-tuning, the T5 Model consistently outperforms the BERT model in terms of summarization quality, as evidenced by higher Rouge scores. This comparative study contributes valuable insights into the strengths and weaknesses of transformer-based models in the context of text summarization, providing guidance for selecting optimal models for similar tasks in the future.

Introduction

Text summarization refers to the technique of shortening long pieces of text. The intention is to create a coherent and fluent summary having only the main points outlined in the document.

Towards Data Science in their article, “A Quick Introduction to Text Summarization in Machine Learning,” on Medium describes the importance of text summarization. They mention how, “ the International Data Corporation (IDC) projects that the total amount of digital data circulating annually around the world would sprout from 4.4 zettabytes in 2013 to hit 180 zettabytes in 2025. With such a big amount of data circulating in the digital space, there is a need to develop machine learning algorithms that can automatically shorten longer texts and deliver accurate summaries that can fluently pass the intended messages.” This highlights the need for good performing machine learning algorithms that can produce summaries of this large sum of data.

In this paper we compare two transformer-based models; a Bidirectional Encoder Representations from Transformers (BERT) model and a Text-To-Text Transfer Transformer (T5) and check their accuracies on the CNN DailyMail dataset by comparing their results using a rouge score metric.

Important Terminologies and Computational Terms

- *BERT*

BERT, short for Bidirectional Encoder Representations from Transformers, is a machine learning framework for natural language processing. BERT makes use of Transformer, an attention mechanism that learns contextual relations between words in a text. The Transformer includes two separate mechanisms; an encoder that reads the text input and a decoder that produces a prediction for the task.

- *T-5*

T5 is an encoder-decoder model pre-trained on a multi-task mixture of unsupervised and supervised tasks and for which each task is converted into a text-to-text format

- *Rouge Score*

The Recall-Oriented Understudy for Gisting Evaluation (ROUGE) scoring algorithm calculates the similarity between a candidate text and a reference text which may be generated from a machine learning algorithm.

- CNN-DailyMail Dataset

This is the dataset we use for training our model. It contains over 300,000 articles and their summaries.

Methodology

- *Data Collection and Research*

Our initial steps were conducting research on existing algorithms and models that have completed summarization tasks. We also focused on finding a dataset which we can use for training and testing our model and decided to focus on the cnn-dailymail dataset

- *Creating a BERT Model*

Since our data was relatively clean, our first focus after data collection was to directly go towards model building. We created our model from a pre-trained bert-base-uncashed Encoder-Decoder Model. We first split our data into train, test and validation sets. We tokenized both our articles and summaries. We also used beam search to generate our summary. Next we trained our model using a Seq2Seq Trainer and Fine-tuned our model based on hyper parameters like number of training steps, eval steps which are all mentioned in our results section. Next we tested our model on our test sets and calculated the rouge scores.

- *Creating the T-5 Model*

Our next step was to create a T-5 Model. Here again, we followed similar steps where we tokenized our data, fine-tuned our trainer and then tested our model using rouge-score.

- Comparison of the two

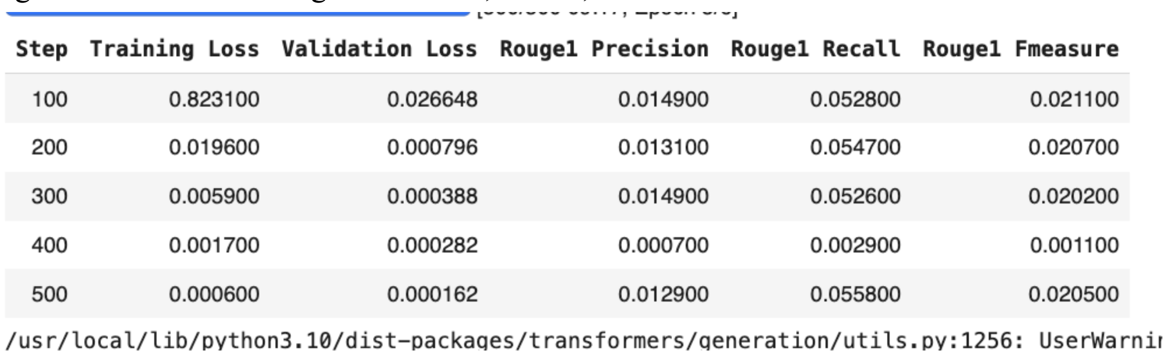
As mentioned before, we calculated the rouge_scores of the articles for both of the models to make our comparison.

Results

When creating our transformer model, we encountered significant limitations due to hardware constraints, including insufficient GPU and RAM resources. Initially, we considered using a BERT model for our summarization tasks, but the complex architecture of BERT, combined with its extensive computational demands, made it impractical for our purposes, particularly as BERT is not traditionally optimized for summarization.

Given these challenges, we shifted our focus to the T5 model, which is better suited for text summarization. We aimed to utilize the "T5-large" model, anticipating that its larger scale would enhance performance for our specific task. However, hardware limitations persisted, preventing us from achieving satisfactory results; even with attempts to augment GPU resources and adjustments to run on a PC, the best accuracy we managed to achieve was just above 0.02 for our BERT Model.

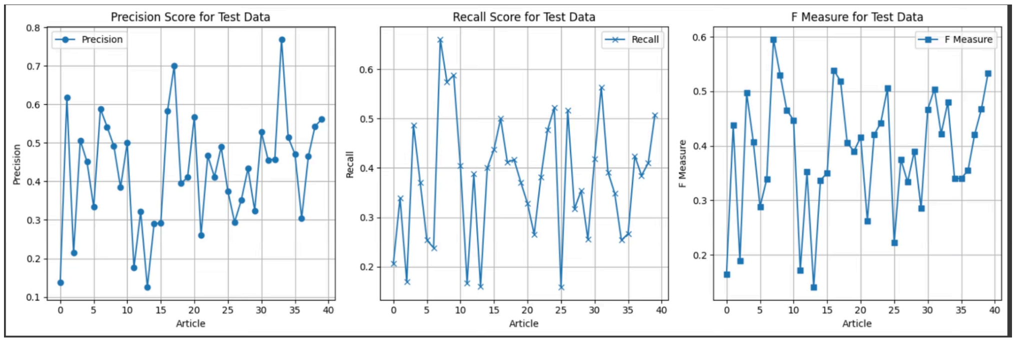
These outcomes were reflected in our initial BERT model trials, where, despite a decrease in training and validation losses as training progressed—indicating some level of model learning—the metrics for Rouge1 Precision, Recall, and F Measure continued to decline.



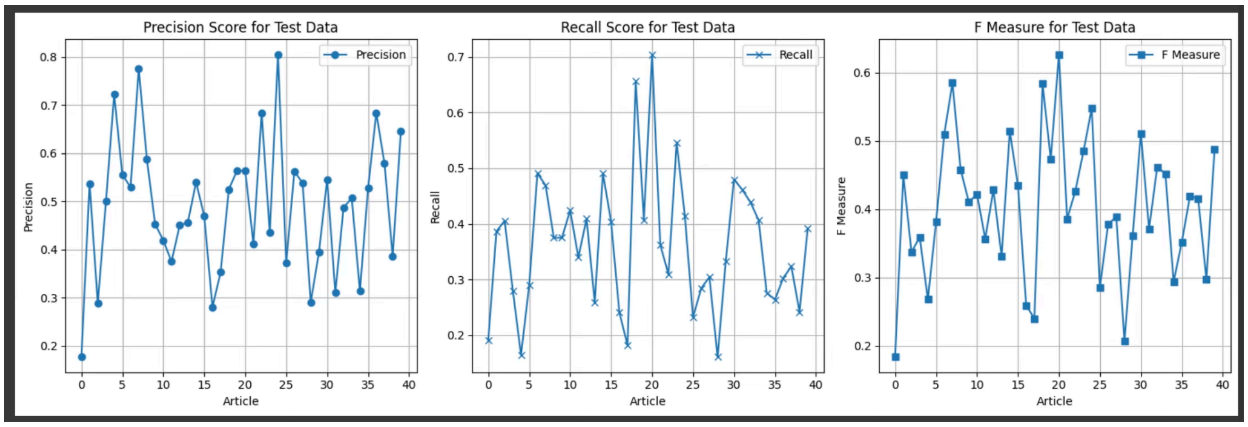
Step	Training Loss	Validation Loss	Rouge1 Precision	Rouge1 Recall	Rouge1 Fmeasure
100	0.823100	0.026648	0.014900	0.052800	0.021100
200	0.019600	0.000796	0.013100	0.054700	0.020700
300	0.005900	0.000388	0.014900	0.052600	0.020200
400	0.001700	0.000282	0.000700	0.002900	0.001100
500	0.000600	0.000162	0.012900	0.055800	0.020500

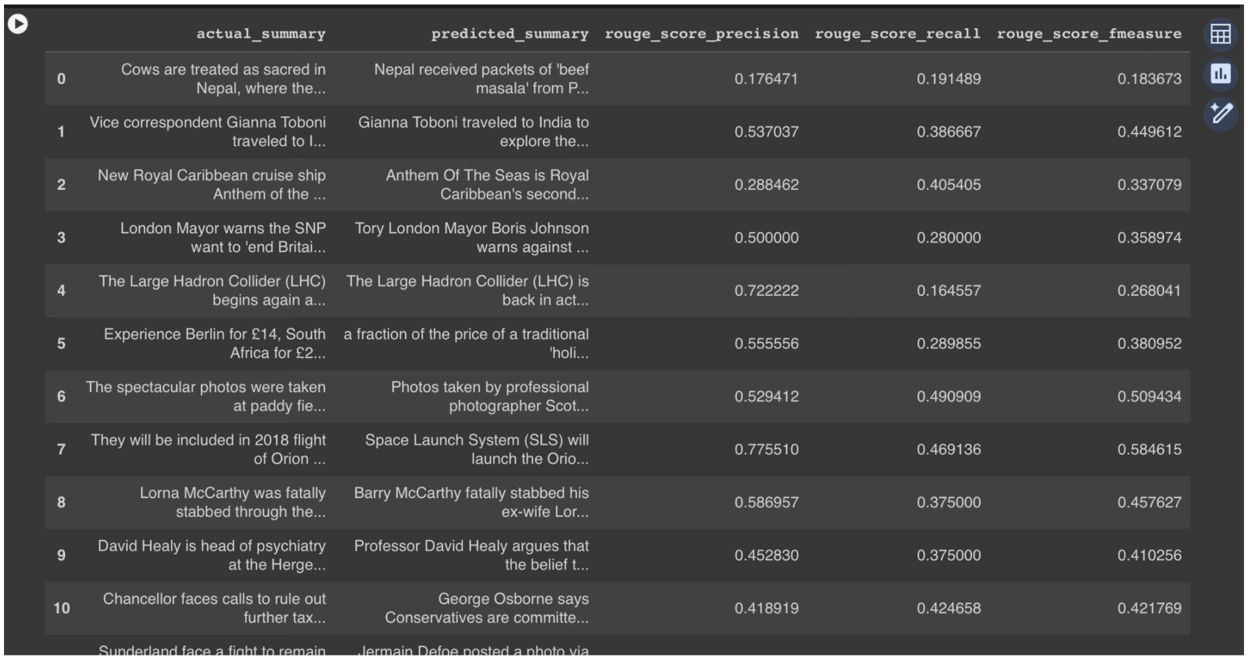
```
/usr/local/lib/python3.10/dist-packages/transformers/generation/utils.py:1256: UserWarnir
```

The switch from a BERT-based model to a T5 architecture was a strategic decision influenced by research indicating T5's superior performance in summarization tasks. Initially, our model, configured with a batch size of 4, a top-p sampling probability of 0.92, a maximum of 10 epochs, and stochastic sampling enabled (do_sample set to true), yielded a precision below 0.6. However, it demonstrated a more favorable recall of 0.5 and an F-measure slightly above 0.5.

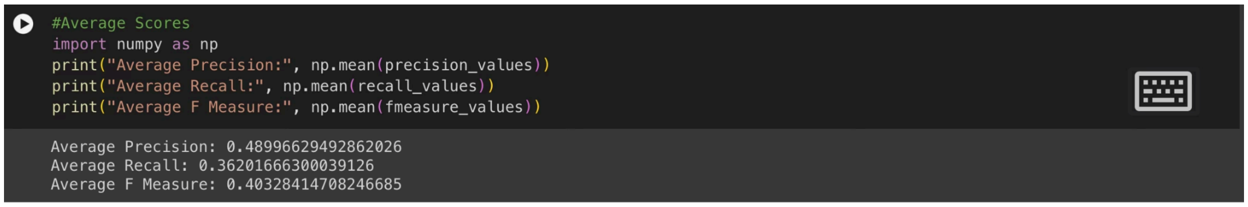


To refine this, we adjusted our model parameters to a batch size of 5, reduced the top-p to 0.9, increased the maximum epochs to 12, and switched to deterministic sampling by setting do_sample to false. These changes led to an improvement in precision but not in recall. Subsequent tests with a batch size of 7 did not further enhance the model's performance, leading us to conclude and report on the configuration with a batch size of 5, where precision scores peaked at 0.8.





	actual_summary	predicted_summary	rouge_score_precision	rouge_score_recall	rouge_score_fmeasure
0	Cows are treated as sacred in Nepal, where the...	Nepal received packets of 'beef masala' from P...	0.176471	0.191489	0.183673
1	Vice correspondent Gianna Toboni traveled to I...	Gianna Toboni traveled to India to explore the...	0.537037	0.386667	0.449612
2	New Royal Caribbean cruise ship Anthem of the ...	Anthem Of The Seas is Royal Caribbean's second...	0.288462	0.405405	0.337079
3	London Mayor warns the SNP want to 'end Britai...	Tory London Mayor Boris Johnson warns against ...	0.500000	0.280000	0.358974
4	The Large Hadron Collider (LHC) begins again a...	The Large Hadron Collider (LHC) is back in act...	0.722222	0.164557	0.268041
5	Experience Berlin for £14, South Africa for £2...	a fraction of the price of a traditional 'holi...	0.555556	0.289855	0.380952
6	The spectacular photos were taken at paddy fie...	Photos taken by professional photographer Scot...	0.529412	0.490909	0.509434
7	They will be included in 2018 flight of Orion ...	Space Launch System (SLS) will launch the Orio...	0.775510	0.469136	0.584615
8	Lorna McCarthy was fatally stabbed through the...	Barry McCarthy fatally stabbed his ex-wife Lor...	0.586957	0.375000	0.457627
9	David Healy is head of psychiatry at the Herge...	Professor David Healy argues that the belief t...	0.452830	0.375000	0.410256
10	Chancellor faces calls to rule out further tax...	George Osborne says Conservatives are committe...	0.418919	0.424658	0.421769
	Sunderland face a fight to remain	Jermain Defoe posted a photo via			



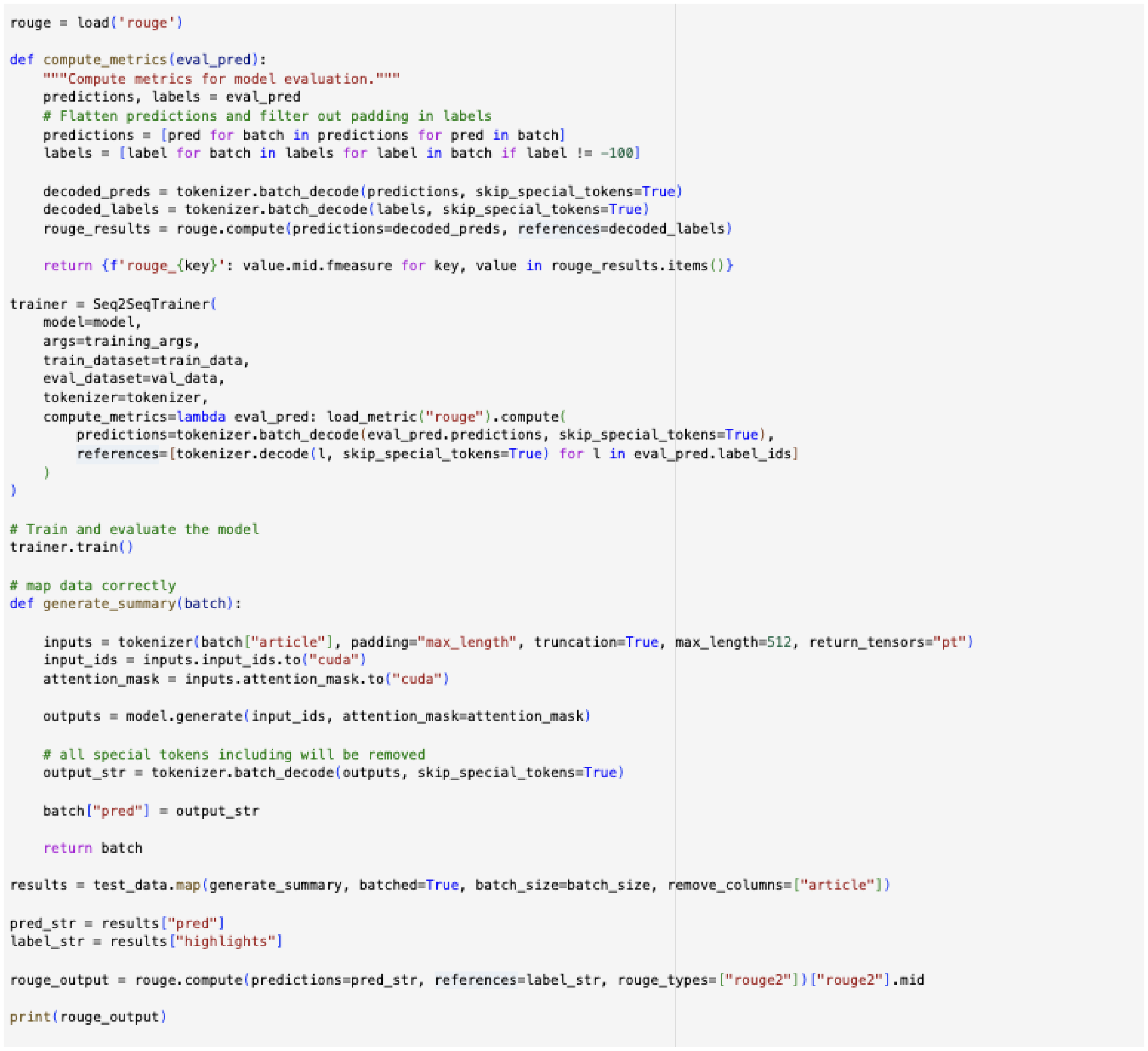
```
#Average Scores
import numpy as np
print("Average Precision:", np.mean(precision_values))
print("Average Recall:", np.mean(recall_values))
print("Average F Measure:", np.mean(fmeasure_values))
```

```
Average Precision: 0.48996629492862026
Average Recall: 0.36201666300039126
Average F Measure: 0.40328414708246685
```

Future

In future considerations, to further improve the BERT model a BERT model with n_grams preprocessing could be used, drawing inspiration from research that underscored the benefits of such preprocessing for improving model performance in text summarization tasks. This approach was informed by the paper "An N-gram-Based BERT model for Sentiment Classification Using Movie Reviews", which detailed how integrating advanced preprocessing techniques could significantly refine the model's output. The code below could be tried with improved hardware and GPU in order to see if further accuracy can be obtained by BERT for summarization.

```python
sys.path.append("/content/drive/MyDrive/colab_env/lib/python3.10/site-packages")
from evaluate import load


# Function to clean text
def clean_and_ngrams(text, n=2):
    text = text.lower().translate(str.maketrans('', '', string.punctuation))
    tokens = word_tokenize(text)
    tokens = [token for token in tokens if token not in stopwords.words('english')]
    n_grams = list(ngrams(tokens, n))
    return " ".join(["_".join(gram) for gram in n_grams])

# Function to generate n-grams
def generate_ngrams(tokens, n=2):
    return list(ngrams(tokens, n))

# Load the dataset
df = pd.read_csv('/content/drive/MyDrive/340 final project/test.csv')

# Preprocess text and generate n-grams
df['cleaned_text'] = df['article'].apply(lambda x: clean_and_ngrams(x, 1))
df['bigrams'] = df['cleaned_text'].apply(lambda x: clean_and_ngrams(x, 2))
df['trigrams'] = df['cleaned_text'].apply(lambda x: clean_and_ngrams(x, 3))


# Convert n-grams to a string format that can be processed by BERT
df['n_grams'] = df['bigrams'].astype(str) + " " + df['trigrams'].astype(str)
df['processed_text'] = df['article'] + " " + df['n_grams']

# Define the dataset splits
train_ratio = 0.8
val_ratio = 0.1
test_ratio = 0.1


num_samples = len(df)
num_train = int(num_samples * train_ratio)
num_val = int(num_samples * val_ratio)
num_test = num_samples - num_train - num_val

# Split the dataset
train_df = df.iloc[:num_train]
val_df = df.iloc[num_train:num_train + num_val]
test_df = df.iloc[num_train + num_val:]

# Initialize the tokenizer
tokenizer = BertTokenizerFast.from_pretrained("bert-base-uncased")
model = EncoderDecoderModel.from_encoder_decoder_pretrained('bert-base-uncased', 'bert-base-uncased',
                                                            encoder_decoder_tie=True)
model.config.decoder.add_cross_attention = True
model.config.decoder_start_token_id = tokenizer.cls_token_id
model.config.encoder_start_token_id = tokenizer.cls_token_id


# Prepare datasets
train_data = Dataset.from_pandas(train_df)
val_data = Dataset.from_pandas(val_df)
test_data = Dataset.from_pandas(test_df[50:120])

# Function to convert batch data for the model
def process_data_to_model_inputs(batch):
    # Preparing the inputs for the encoder
    inputs = tokenizer(batch['processed_text'], padding="max_length", truncation=True, max_length=512, return_tensors='pt')
    # Preparing the targets for the decoder, where the labels are the target text
    outputs = tokenizer(batch['highlights'], padding="max_length", truncation=True, max_length=128, return_tensors='pt')

    batch['input_ids'] = inputs.input_ids
    batch['attention_mask'] = inputs.attention_mask
    batch['decoder_input_ids'] = outputs.input_ids[:, :-1]  # Exclude the last token for decoder input
    batch['decoder_attention_mask'] = outputs.attention_mask[:, :-1]
    batch['labels'] = outputs.input_ids[:, 1:].clone()  # Exclude the first token for labels
    batch['labels'][batch['labels'] == tokenizer.pad_token_id] = -100  # Mask padding

    return batch


# Apply preprocessing to data
batch_size = 4
train_data = train_data.map(process_data_to_model_inputs, batched=True, batch_size=batch_size, remove_columns=train_data.column_names)
val_data = val_data.map(process_data_to_model_inputs, batched=True, batch_size=batch_size, remove_columns=val_data.column_names)

# Set format for PyTorch tensors
train_data.set_format(type="torch", columns=["input_ids", "attention_mask", "decoder_input_ids", "decoder_attention_mask", "labels"])
val_data.set_format(type="torch", columns=["input_ids", "attention_mask", "decoder_input_ids", "decoder_attention_mask", "labels"])

model = EncoderDecoderModel.from_encoder_decoder_pretrained('bert-base-uncased', 'bert-base-uncased', tie_encoder_decoder=True)
model.config.decoder.add_cross_attention = True
model.save_pretrained('./bert2bert')
model.to("cuda")
model.config.decoder_start_token_id = tokenizer.cls_token_id
training_args = Seq2SeqTrainingArguments(
    output_dir="./",
    evaluation_strategy="steps",
    per_device_train_batch_size=8,
    per_device_eval_batch_size=8,
    predict_with_generate=True,
    logging_steps=50,
    save_steps=100,
    eval_steps=100,
    warmup_steps=500,
```



```python
rouge = load('rouge')

def compute_metrics(eval_pred):
    """Compute metrics for model evaluation."""
    predictions, labels = eval_pred
    # Flatten predictions and filter out padding in labels
    predictions = [pred for batch in predictions for pred in batch]
    labels = [label for batch in labels for label in batch if label != -100]

    decoded_preds = tokenizer.batch_decode(predictions, skip_special_tokens=True)
    decoded_labels = tokenizer.batch_decode(labels, skip_special_tokens=True)
    rouge_results = rouge.compute(predictions=decoded_preds, references=decoded_labels)

    return {f'rouge_{key}': value.mid.fmeasure for key, value in rouge_results.items()}

trainer = Seq2SeqTrainer(
    model=model,
    args=training_args,
    train_dataset=train_data,
    eval_dataset=val_data,
    tokenizer=tokenizer,
    compute_metrics=lambda eval_pred: load_metric("rouge").compute(
        predictions=tokenizer.batch_decode(eval_pred.predictions, skip_special_tokens=True),
        references=[tokenizer.decode(l, skip_special_tokens=True) for l in eval_pred.label_ids]
    )
)

# Train and evaluate the model
trainer.train()

# map data correctly
def generate_summary(batch):

    inputs = tokenizer(batch["article"], padding="max_length", truncation=True, max_length=512, return_tensors="pt")
    input_ids = inputs.input_ids.to("cuda")
    attention_mask = inputs.attention_mask.to("cuda")

    outputs = model.generate(input_ids, attention_mask=attention_mask)

    # all special tokens including will be removed
    output_str = tokenizer.batch_decode(outputs, skip_special_tokens=True)

    batch["pred"] = output_str

    return batch

results = test_data.map(generate_summary, batched=True, batch_size=batch_size, remove_columns=["article"])

pred_str = results["pred"]
label_str = results["highlights"]

rouge_output = rouge.compute(predictions=pred_str, references=label_str, rouge_types=["rouge2"])["rouge2"].mid

print(rouge_output)
```

Conclusion

In conclusion the T5 model significantly outperformed the BERT model for the task of text summarization. Despite the inherent challenges associated with transformer-based models, including their computational demands and the need for substantial training data, the T5 model's architecture proved to be more effective, particularly more effective in optimizing precision in summarization tasks. The improved performance metrics, such as Rouge scores clearly demonstrate the T5's ability to generate concise and relevant summaries of the original texts.

Further research could focus on refining the T5 models by incorporating additional pre-training stages to improve the adaptability of various texts, such as with n_grams. Additionally, further refinement of BERT alongside this preprocessing of n_grams could improve the complex architecture. Hybrid models between both the strengths of BERT and T5 could yield more robust solutions for text summarizations. However, additional GPU and RAM than hardware we had would need to be utilized.

Important Links
Data Set:
https://www.kaggle.com/datasets/gowrishankarp/newspaper-text-summarization-cnn-dailymail
Github: https://github.com/lavyam/Transformer-based-summarization

Work Cited

Medium article:
https://towardsdatascience.com/a-quick-introduction-to-text-summarization-in-machine-learning-3d27ccf18a9f

Model for BERT based on:
https://huggingface.co/patrickvonplaten/bert2bert_cnn_daily_mail

Trueman, Tina. *An N-Gram-Based Bert Model for Sentiment Classification ...*, sentic.net/sentiment-classification-using-movie-reviews.pdf. Accessed 30 Apr. 2024.

Model for T-5 based on:
Azizamirsaidova. "Abstractive_text_summarization/Notebooks/Fine_tune_simplet5_absractive_text_summarization.Ipynb at Main · Azizamirsaidova/Abstractive_text_summarization." *GitHub*, github.com/azizamirsaidova/abstractive_text_summarization/blob/main/notebooks/Fine_tune_simplet5_absractive_text_summarization.ipynb. Accessed 29 Apr. 2024.