KTH – Royal Institute of Technology
Language Engineering – DD2417
Project report: Abstractive Text Summarization

Zineb Senane
Reza Qorbani

May 2023

.

1 Introduction

Efficient text summarization systems are becoming increasingly necessary as digital content grows at an exponential rate. These systems can revolutionize various fields such as academic research and news by compressing lengthy texts into small briefs. The advent of advanced machine learning models in Natural Language Processing (NLP) and the transformers facilitates the development of such systems and makes them more feasible than ever.

In this project, we delve into the potential of abstractive text summarization, which aims to generate summaries similar to how a human would - by understanding the essence of the text and producing a concise, coherent summary in new words. Compared to extractive summarization, which chooses only the primary sentences from original text, abstractive summarization provides contextually rich summaries that are much more meaningful and contextually relevant, though imposing additional complexity. By leveraging recent advancements in pre-trained transformer models, more precisely T5[5] (Text-To-Text Transfer Transformer) and GPT-2 [4](Generative Pretrained Transformer 2), our approach consists of using self-attention mechanisms to comprehend the context of the sequence of words in a text. Unlike traditional architectures such as Recurrent Neural Networks (RNNs) and Long-Short Term Memory (LSTMs) networks, the self-attention mechanism makes them exceptionally suited for abstractive summarization.

Throughout the scope of this work, we harnessed the general language understanding abilities of pre-trained models of textual data, and their potential to accomplish an abstractive summarization. To adapt these models to our task, we fine-tuned T5 and GPT2 transformers on a subset of the XSUM dataset [3], which is an extensive compilation of BBC articles along with their short summaries specifically crafted for summarization tasks.

To assess the performance of our models, we used established evaluation metrics like ROUGE [2] (Recall-Oriented Understudy for Gisting Evaluation) for a quantitative measure of their quality and Bert Scores [6] leveraging contextualized embeddings from BERT (Bidirectional Encoder Representations from Transformers) to measure the closeness of generated summaries. Furthermore, We also supplement this with a subjective evaluation of the generated summaries to compare the two models' performance.

2 Background

The field of text summarization has seen significant progress in the last few decades, fuelled by the growth of digital content and the need for quick, concise interpretations of these texts. Traditional approaches to text summarization were extractive in nature, meaning they involved selecting and collating relevant sentences from the original text to form a summary. While this approach had its merits, it often resulted in summaries that lacked cohesiveness and often missed the main points of the text.

As computational power increased and machine learning algorithms evolved, a new approach to text summarization became possible. This approach, known as abstractive summarization, involves generating a summary from scratch, instead of merely extracting parts of the original text. It's more analogous to how a human would summarize a text - by understanding the content, distilling the key ideas, and expressing these ideas in a concise and coherent manner. This method presents more complexities than the extractive approach, as it requires a deeper understanding of the text and the capability to generate coherent new text.

The advent of neural network-based methods, particularly Recurrent Neural Networks (RNNs) with attention mechanisms and Transformer models, made abstractive summarization a realistic possibility. RNNs have the ability to process sequences of data, making them well-suited to text processing tasks. Attention mechanisms, on the other hand, allow models to 'focus' on the most relevant parts of the text when generating a summary.

More recently, pre-trained Transformer models have taken centre stage, offering unprecedented performance across a wide range of Natural Language Processing (NLP) tasks. Transformer models, such as T5 and GPT-2, use self-attention mechanisms that allow them to understand the context of words in a sentence, making them highly effective for tasks such as abstractive summarization.

These models can be further fine-tuned on specific tasks or datasets to improve their performance. This project builds upon these advancements by fine-tuning T5 and GPT-2, two pre-trained transformer models, on the XSUM dataset, a collection of BBC articles used for summarization tasks. This dataset, along with others like WikiHow and Amazon Fine Food Reviews, provide a rich source of text data for training and evaluating text summarization models.

Evaluating the effectiveness of text summarization models is a critical aspect of this work. Established metrics like ROUGE (Recall-Oriented Understudy for Gisting Evaluation) are often used for this purpose, providing a quantitative measure of the quality of the generated summaries. In addition to these automatic evaluation metrics, manual evaluation plays an important role in judging the quality and coherence of the summaries.

3 Method

3.1 Data

3.1.1 Data Exploration

The XSUM dataset is a popular benchmark dataset for text summarization tasks. It consists of a collection of news articles and their corresponding single-sentence summaries. The articles cover various domains, including politics, entertainment, technology, sports, and more. The dataset was specifically curated to emphasize document compression and encourage abstractive summarization techniques.

The XSUM dataset used in our abstractive text summarization project comprises several columns, including "Document" and "Summary." The "Document" column contains the full text of the news articles, covering diverse domains such as politics, entertainment, technology, sports, and more, with structured content spanning multiple paragraphs. The "Summary" column consists of single-sentence summaries that capture the main points and essence of each article concisely. Our analysis revealed that the training dataset consists of approximately 204045 news articles, each accompanied by its

corresponding summary. This large-scale dataset provided us with an effective training resource, however, to mitigate the time-consuming process of fine-tuning transformer models on large-scale datasets and to cope with lack of adequate compute, we employed a strategy of utilizing only a subset of the XSUM dataset.

The validation split was used for model evaluation during the training and early stopping, while the test split served as a final evaluation set to assess the performance of our models on unseen data.

3.1.2 Preprocessing

Before inputting the data into the models, we conducted distinct preprocessing steps for each model to prepare the text for training and ensure optimal performance.

The preprocessing procedures for the abstractive text summarization task include:

Tokenization: Tokenization is the process of breaking down the text into smaller units, usually words or subwords, to facilitate further analysis. We tokenize both the input document and the target summary into individual tokens.

Text-to-Integer Conversion: Once the vocabulary is derived from the various tokens, we transform the tokenized text into integer sequences. Each token is replaced with its corresponding integer ID from the vocabulary. This conversion allows the model to manipulate numerical representations of the text.

Truncation: We set a limit on the maximum sequence length for both the input document and the target summary. Sequences exceeding the maximum length are truncated. This step helps us accommodate longer texts while preserving computational feasibility.

Padding: Since neural networks typically require inputs of a fixed length, we pad the sequences with a special padding token to reach the desired length. Padding ensures that all input sequences are the same length, thereby enabling efficient batch processing during training.

For the `T5 transformer`, we began by adding a prefix 'summarize: ' to the input texts, which helps to specify the task for the model. Next, we applied the aforementioned preprocessing steps to both the input text and summaries so that they can be fed to the T5 as encoder_input_ids and decoder_input_ids.

`GPT2` is not an encoder-decoder architecture-based model, so it required different formatting. To prepare data for GPT-2 for text summarization as a downstream task, we added three special tokens: <bos>to mark the start of the input text, a separator token used to separate the text from its summary after concatenation, and <eos>to mark the end of the text. Afterward, we applied the same preprocessing steps to this new textual input using the GPT-2 tokenizer.

3.2 Models

3.2.1 GPT2 Model

The GPT-2 (Generative Pretrained Transformer 2) model developed by OpenAI is a transformer-based language model renowned for its high performance across various natural language processing tasks. It boasts an impressive 1.5 billion parameters, offering an expansive architecture that leverages the capabilities of the transformer structure. The model we used in this project is a much smaller version of GPT2, with only 124 million parameters.

Model architecture The GPT-2 model is founded on the principles of the transformer architecture, dispensing with the need for sequential data processing intrinsic to recurrent or convolutional neural network structures. This characteristic of the transformer-based architecture allows the model to capture dependencies between words and phrases in the text, irrespective of their respective positions or distances from each other, making it particularly effective for tasks that require comprehension of long-range semantic dependencies.

At its core, the architecture of GPT-2 consists of a series of stacked transformer decoders. Each of these decoders is composed of a self-attention layer and a position-wise feed-forward neural network.

The self-attention layer facilitates the understanding of the context of each word in relation to all other words in the input sequence. This allows GPT-2 to generate rich and context-aware representations of the text.

The input to the model is an array of tokens, derived from the input text. Each token is then embedded into a continuous vector representation through an embedding layer. The positional encoding is subsequently added to retain the sequence order of the words. These vectors serve as the input to the stack of transformer decoders.

The decoders process the input data, with each decoder layer contributing to the refinement of the representations. The self-attention mechanism within each decoder allows the model to weigh the impact of each word on the others, resulting in a contextual representation of the entire sequence. The feed-forward network further refines this output, which is then passed onto the next layer.

The final layer of the model is a linear layer followed by a softmax function, which produces the probability distribution over the vocabulary for the next word in the sequence. The word with the highest probability is selected as the next word, and this process is repeated until the end-of-sequence token is generated.

This unique architecture allows GPT-2 to generate highly coherent and contextually relevant sequences of text, making it an excellent choice for tasks such as text summarization.

3.2.2 T5

The T5 (Text-to-Text Transfer Transformer) model, developed by Google Research, is a versatile transformer-based model designed to handle any text-to-text task. It is distinguished by its unified text-to-text format where all tasks are cast into a text generation problem, allowing the model to leverage pre-training and fine-tuning for diverse tasks.

Model architecture T5 uses a variant of the transformer model architecture called the "Transformer encoder-decoder", which combines the capabilities of transformer-based encoders and decoders in a single model. With 11 billion parameters in its largest variant, it offers substantial capacity to learn complex mappings from inputs to outputs. The version we use in this project is a smaller version which only has 60 million parameters.

The T5 model processes input text by first tokenizing the text into subword units using SentencePiece tokenization. Each token is then associated with a unique integer ID. The tokenized input is processed by an encoder that produces a sequence of continuous vector representations, capturing the context and semantic information of the input text.

The encoder is composed of a stack of identical layers, with each layer containing two sub-layers: a multi-head self-attention mechanism and a position-wise feed-forward neural network. The self-attention mechanism allows each token to attend to all other tokens in the input sequence, making it possible for the model to handle long-range dependencies. The position-wise feed-forward network applies a non-linear transformation to each token independently, allowing the model to learn complex intra-token interactions.

After the encoder processes the input text, a decoder, which has a similar structure to the encoder, generates the output text. The decoder also uses self-attention and position-wise feed-forward networks. However, in addition to the self-attention mechanism, the decoder includes an encoder-decoder attention layer. This layer allows each token in the output sequence to attend to all tokens in the input sequence, thereby enabling the model to utilize information from the entire input while generating each token in the output.

The output of the decoder is a sequence of vectors, each corresponding to a token in the output text. A final linear transformation followed by a softmax function is applied to each vector to produce a probability distribution over the vocabulary. The token associated with the highest probability is chosen as the output for each position.

This comprehensive architecture allows T5 to perform a wide range of text-to-text tasks with high accuracy, making it an effective choice for tasks such as text summarization.

3.3 Metrics

To assess and evaluate the performance of our abstractive text summarization models quantitatively, we employed several established metrics. These metrics provide objective measures of the quality and effectiveness of the generated summaries compared to the reference summaries. In our evaluation, we focused on four commonly used metrics: Rouge-N (1 and 2-grams), Rouge-L, and BERTScore.

`Rouge-N` (Recall-Oriented Understudy for Gisting Evaluation) is a metric that quantifies the overlap of N-grams between the generated summaries (candidate texts) and the reference summaries (ground truth). We utilized both Rouge-1 and Rouge-2 scores to measure the overlaps of unigrams and bigrams, respectively. By evaluating the shared N-grams, Rouge-N assesses the quality of the generated summaries based on their similarity to the reference summaries.

`Rouge-L` is another evaluation metric that considers the sentence-level structure similarity between the candidate and reference summaries. It calculates the longest common subsequence between the two summaries, identifying the longest co-occurring in-sequence n-grams automatically. Rouge-L captures the structural similarity in terms of sentence order and content, providing valuable insights into the performance of abstractive text summarization models.

In addition to Rouge-N and Rouge-L, we also employed `BERTScore` to evaluate the quality of the generated summaries. BERTScore leverages contextual embeddings provided by BERT (Bidirectional Encoder Representations from Transformers) to measure the similarity between the candidate and reference summaries. Unlike traditional n-gram-based metrics, BERTScore considers the contextual information and semantic similarity of the text, resulting in more accurate evaluations. BERTScore has been shown to correlate well with human judgments and provides a robust evaluation metric for abstractive text summarization.

By utilizing these established evaluation metrics, including Rouge-N, Rouge-L, and BERTScore, we obtained a comprehensive assessment of the performance of our abstractive text summarization models. These metrics allowed us to quantitatively measure the effectiveness of the models in generating summaries that capture the key information from the source texts while maintaining coherence and semantic similarity to the reference summaries. Furthermore, we conducted a subjective analysis of some of the generated summaries to compare the performance of the two models.

4 Experiments

In this section we are going to present the experiments we performed on the models, their implementations, and the results we obtained.

We performed the same exact experiments for both of the models in order to compare the results for each model. The hyper-parameters used for training can be found in Table 1. We used Pytroch Lightning [1] in our implementation for effective logging and configuration of model training.

4.1 T5

We started by fine-tuning the T5 model, which we imported from the Hugging Face Transformers library. Because of lack of resources, we used only a portion of the original dataset. More specifically, we used 15% of the original train dataset and 30% of validation and test dataset respectively. The proportions of our splits can be found in Table1. We employed a sequence-to-sequence setup for fine-tuning the T5 model. With our setup, each epoch took approximately 13 minutes to complete. To prevent over-fitting and improve training efficiency, we incorporated early stopping, where we only save the epoch checkpoint that gave the lowest loss on validation set (not neccessarily the last epoch). We used early stopping with patience of 2 as the number of epochs used in our setup is low.

After the fine-tuning process, we saved the trained T5 model, and the corresponding tokenizer, in a directory for future use. This enabled us to easily retrieve and load the model for inference tasks such as text summarization.

4.2 GPT2

In addition to the T5 model, we also fine-tuned GPT-2, another transformer-based architecture widely used in natural language processing tasks. Similar to the T5 model, we implemented a PyTorch Lightning model and initialized it using the Hugging Face GPT-2 transformer with Language Model Head. We used a smaller portion of the XSum training subset, specifically 3% of the original dataset and 6% of validation and test dataset respectively, refer to Table1 for relative split proportions. The fine-tuning process was conducted over multiple 5 epochs, with each epoch taking approximately 30 minutes to complete. We didn't incorporate early stopping for GPT2 as it is a large and complex model and it easily overfits and was trained only for a few epochs. After the fine-tuning process, the trained model and its tokenizer were saved in a directory for future inference.

Furthermore, the saved models and tokenizers were pushed to Hugging Face hub where they are used as the back-end of our demo.

Model	Train/Val/Test	Epochs	Train BS	Val BS	lr
T5	82/9/9	10	8	4	5e-5
GPT2	82/9/9	5	1	2	1e-4

Table 1: Training Configuration of Models

Similar to the T5 fine-tuning process, we implemented checkpointing and the trained GPT-2 model was saved in a directory for later use during inference.

4.3 Evolution of loss function

The GPT2 loss evolution reveals a clear pattern: the training loss steadily decreases, while the validation loss shows a contrasting trend of increasing over time. This suggests that the model is likely over-fitting to the training data and memorizing its specific patterns rather than generalizing the text summarization task.
Considering the GPT2 architecture's complexity and the limited size of the fine-tuning dataset, overfitting is unsurprising. The extensive parameter count (124M) allows the model to capture details from the training articles. To mitigate this overfitting, augmenting the training dataset by incorporating additional diverse articles and more samples can improve the model's ability to generalize across different new texts.

Figure 1: GPT2: Training and validation loss evolution

4.4 Subjective analysis

Here we present some of the generated summaries from our models and compare it to the corresponding ground truth summaries, for both of the models. The generated summaries can be found in Figure 3. Generally, what we observed was that T5 model gave much better summaries compared to GPT2, as expected, because T5 is specifically made for summarizing tasks. If we look at the second row in Figure 3, we can even see that the generated summary by T5 is able to pack roughly the same information in

In contrast, the graph for T5 is much different. We see that the training loss is never lower than the validation loss. We have some ideas for why this is happening. T5 model is specifically made for summarization, which means it should perform better that GPT2 even with fewer parameters. This could mean that even in this case the model has overfitted, something that is maybe suggested by Figure 2. Furthermore, the size of the validation/test might be too small (as we did not use the entire dataset), so it might be the case that are model happens to be very good at summarizing the subset we are using. Because the number of parameters of the T5 model is only half of that of GPT2 model, we were able to use more training data (without depleting the resources).

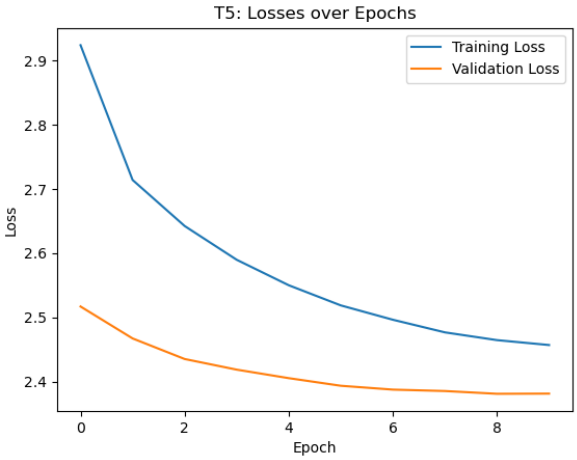

Figure 2: T5, Training and validation loss evolution

fewer words, which is not the case for GPT2, and if the generation by T5 was larger, it was usually to summarize the document more clearly. Despite GPT2 generating bad summaries, the samples show that the model comprehends quite well the sequence of words and gets the context.

```
Ground Truth: There is a "chronic" need for more housing for prison leavers in Wales,
according to a charity.
T5: A charity has said it is "desperate" for people to find accommodation after their
release from prison.
GPT2: Almost two-thirds of people found a permanent new home in Cymru have received
the help, according to a Welsh investigation.
--------------------------------------------------------------------------------
Ground Truth: A man has appeared in court after firearms, ammunition and cash were
seized by police in Edinburgh.
T5:A man has appeared in court charged with firearms offences in Edinburgh.
GPT2: Police seized four handguns found in west Scotland on 24-year-old
Ronnie O'Brien's murder victim's murder has come into force.

--------------------------------------------------------------------------------
Ground Truth: Four people accused of kidnapping and torturing a mentally disabled man
in a "racially motivated" attack streamed on Facebook have been denied bail.
T5: two men have appeared in court charged with hate crimes and aggravated
kidnapping of a white man who was found dead in a van.
GPT2: The four men featured in a Facebook Live video have been dropped on
Tuesday, activists say.
```

Figure 3: Generated summaries

4.5 Evaluation metrics

Finally, in Table **??**, we can see the result of evaluations. We can clearly see that T5 model is performing much better than the GPT2, proving once again that T5 is the better model for summarization in our experiments. We also included the results for pre-trained T5 model (not fine-tuned) as a baseline to compare to our experiments. The fine tuned T5 outperforms also the pre-trained T5, this suggests that

further fine-tuning of T5 improves its ability for abstractive text summarization. The ROUGE scores support this, indicating that T5 excels at capturing important words and phrases from the reference summaries. Moreover, the BERT scores further reinforce this, highest scores showcase its enhanced ability to capture relevant information and generate accurate summaries.

Model	ROUGE-1	ROUGE-2	ROUGE-L	BERT		
				Precision	Recall	F1 Score
`Pre-trained T5`	0.175	0.025	0.127	0.848	0.859	0.853
`Fine-Tuned GPT2`	0.164	0.027	0.126	0.762	0.765	0.763
`Fine-Tuned T5`	0.298	0.088	0.233	0.893	0.881	0.887

Table 2: Evaluation metric scores

4.6 Resource problems

Due to the lack of enough compute this project was very hard to complete. We used a wide range of services such as Amazon sagemaker, GCP, kaggle, and Colab. On all of these platforms we errors related to not enough GPU memory (or RAM) and also when managed to get a stable training, the session would close randomly (we did not have enough permission to use tools such as Screen and tmux). Our implementations are definitely capable of handling entire dataset, but sadly, we realized that the scale of our project was too big for freely available GPU too late, after we already had an implementation and could not afford to pivot. We wanted to stress that it was hard for us to know about these issues before-hand as we are not experienced with working with large-scale dataset before this course.

5 Conclusion

In conclusion, our experiments comparing the T5 and GPT-2 models for abstractive text summarization tasks highlight the superior performance of the T5 model. With its specifically designed architecture and fine-tuning, the T5 model consistently outperforms the GPT-2 model in terms of evaluation metrics, including Rouge-N, Rouge-L, and BERTScore. The T5 model demonstrates its ability to generate summaries with a higher overlap of grams, and better structural similarity compared to both the GPT-2 model and the pre-trained T5 model. These results emphasize the importance of utilizing task-specific models and the effectiveness of fine-tuning in achieving high-quality abstractive text summarization.

References

[1] William Falcon and The PyTorch Lightning team. *PyTorch Lightning*. Version 1.4. Mar. 2019. DOI: `10.5281/zenodo.3828935`. URL: `https://github.com/Lightning-AI/lightning`.

[2] Chin-Yew Lin. "ROUGE: A Package for Automatic Evaluation of Summaries". In: *Text Summarization Branches Out*. Barcelona, Spain: Association for Computational Linguistics, July 2004, pp. 74–81. URL: `https://www.aclweb.org/anthology/W04-1013`.

[3] Shashi Narayan, Shay B. Cohen, and Mirella Lapata. "Don't Give Me the Details, Just the Summary! Topic-Aware Convolutional Neural Networks for Extreme Summarization". In: *ArXiv* abs/1808.08745 (2018).

[4] Alec Radford et al. "Language Models are Unsupervised Multitask Learners". In: (2019).

[5] Colin Raffel et al. "Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer". In: *Journal of Machine Learning Research* 21.140 (2020), pp. 1–67. URL: http://jmlr.org/papers/v21/20-074.html.

[6] Tianyi Zhang* et al. "BERTScore: Evaluating Text Generation with BERT". In: *International Conference on Learning Representations*. 2020. URL: https://openreview.net/forum?id=SkeHuCVFDr.

Appendix

1 User Interface

In addition to fine-tuning the models, we have created a user interface that offers an interactive and user-friendly experience. The interface allows users to input their texts, and through it, they can get summaries and play with the capabilities of our fine-tuned models. To access the user interface, please visit Text Presso Machine.

2 Fine-Tuned GPT2 for Abstractive Text Summarization

Please visit GPT2 Summarizer for the fine-tuned GPT2 model and configuration with its tokenizer.

3 Fine-Tuned T5 for Abstractive Text Summarization

Please visit T5 Summarizer for the fine-tuned T5 model and configuration with its tokenizer.