Transformer-based Text Summarization and Comparative Analysis

Jinda Zhang **Juan Yu** **Changyu Liu** **Ye Zhang**

1 Introduction

Our research delves into the capabilities of transformer models in natural language processing, focusing on the fine-tuning and comparative analysis of three prominent Large Language Models (LLMs) — Pegasus, T5, and Bart — tailored for automated text summarization. The fine-tuning process was meticulously carried out on the CNN/DailyMail dataset, a widely recognized benchmark in the domain of text summarization. The primary objective of our experiments was to assess and compare the efficacy of these models in producing summaries that are not only concise but also retain the pivotal information of the source texts.

During our extensive experimentation, it was particularly noteworthy that the T5 model exhibited remarkable performance. It consistently generated summaries that were not only coherent but also accurately reflected the critical content of the original texts. This standout performance of the T5 model, when evaluated against a variety of metrics, distinctly highlighted its superiority in our set of tested models. These findings are pivotal, as they not only demonstrate the effectiveness of the T5 model in the context of automated text summarization but also underscore the transformative potential of transformer-based models in the broader landscape of natural language processing.

2 Problem statement

In the current digital age, the relentless expansion of textual content available both online and in printed media has underscored the critical need for efficient summarization tools. This burgeoning volume of data necessitates automated solutions capable of condensing extensive texts into summaries that are both concise and coherent, while still capturing the essential elements of the original content. Our research is dedicated to meeting this challenge by exploring the realm of automated text summarization, an increasingly vital technology with diverse applications including news aggregation and the summarization of academic research.

Leveraging the transformative advancements in transformer architectures and large language models (LLMs), our project is positioned at the forefront of enabling such automated text summarization. We focus on the fine-tuning and comparative analysis of three pre-trained LLMs, renowned for their text summarization capabilities. The aim is to evaluate and contrast their performance not only against each other but also in relation to three baseline models that we have selected for this purpose. Through this approach, our study seeks to provide valuable insights into the effectiveness of these advanced models in the automated summarization of textual data, thus contributing to the ongoing evolution of natural language processing technologies.

3 Methodology

This project is implemented with three baseline models and three pre-trained LLMs. We use random text summarization, Term Frequency-Inverse Document Frequency (TF-IDF), and TextRank Algorithm as baseline models, and on top of that we fine-tune Pegasus, T5 and Bart based on a CNN / DailyMail dataset.

37th Conference on Neural Information Processing Systems (NeurIPS 2023).

3.1 Baseline Model

3.1.1 Random Summarization

A simple baseline approach is random text summarization. This method randomly selects sentences from the document to create the summary, without considering content relevance or importance.

The random summarization process involves splitting the document into sentences and randomly selecting a subset of sentences to form the summary. The number of sentences selected is capped at a predefined limit, ensuring a concise summary.

For the validation set, random summarization is applied to each document independently. The randomly generated summaries are then evaluated using the ROUGE metric, which measures the similarity between the random summaries and the reference highlights.

3.1.2 TF-IDF Summarization

In this section, we present our baseline approach using TF-IDF for text summarization. TF-IDF is a widely used technique to represent the importance of words in a document corpus. We use this approach to generate extractive summaries for the given documents.

We begin by computing the TF-IDF matrix using the scikit-learn library. The TF-IDF vectorizer is applied to the training set of documents, creating a matrix that represents the importance of each word in the corpus.

We then use the TF-IDF matrix to perform extractive summarization on the validation set. For each document, we calculate the cosine similarity between its TF-IDF representation and those of the entire training set. The sentences with the highest similarity scores are selected to form the summary.

To evaluate the quality of the generated summaries, we use the ROUGE metric, which measures the overlap between system-generated summaries and reference summaries. The ROUGE scores for TF-IDF summarization are computed and presented below.

Finally, we analyze the ROUGE scores obtained from TF-IDF summarization to assess the baseline performance.

3.1.3 TextRank Algorithm Summarization

Another baseline approach we used is TextRank for text summarization. TextRank is an unsupervised extractive summarization technique that assigns importance scores to sentences based on their similarity within the document. We use this approach to generate extractive summaries for the given documents.

We employ the TextRank algorithm to summarize the documents in the validation set. For each document, TextRank assigns importance scores to sentences and selects the top sentences to form the summary. The ratio parameter, set to 0.2 in our implementation, controls the length of the generated summary. We analyze the ROUGE scores obtained from TextRank summarization to evaluate the baseline performance.

3.2 Pre-trained LLMs

The three LLMs we fine-tuned are all based on the Transformer architecture, as show in 1. It was first introduced by Vaswani et al. in the paper "Attention is All You Need," revolutionized natural language processing tasks with self-attention mechanisms (1). The self-attention mechanism allows the model to weigh different parts of the input sequence differently and capture complex relationships within the data. Transformer is the achiture

3.2.1 Pegasus

Pegasus, introduced by Zhang et al., is a transformer-based model designed specifically for abstractive text summarization (2) as shown in 2. Pegasus uses a pre-training approach where the model is trained on a diverse range of document-level denoising objectives, enabling it to generate summaries. The use of extractive summarization during pre-training allows Pegasus to learn a robust representation of document structures and content.

Figure 1: Attention is All you Need

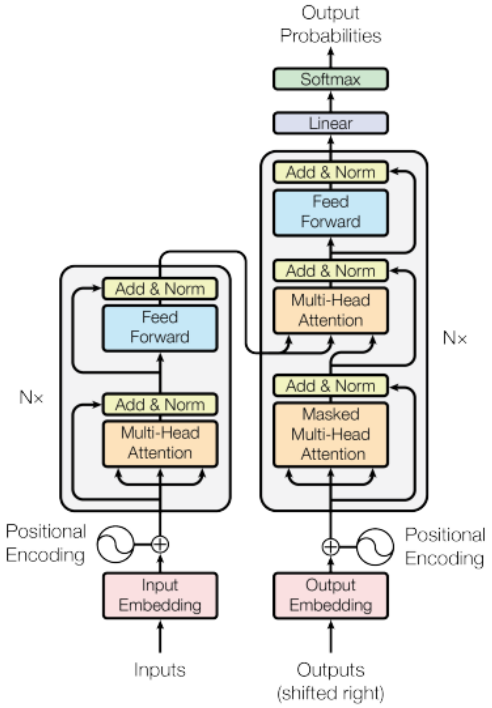
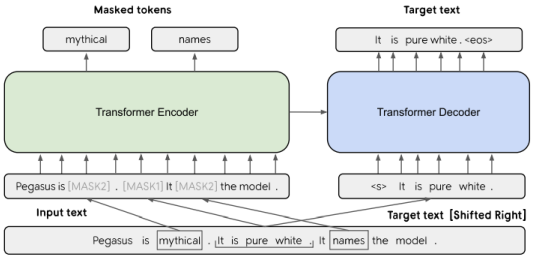


Figure 2: Pegasus



3.2.2 T5

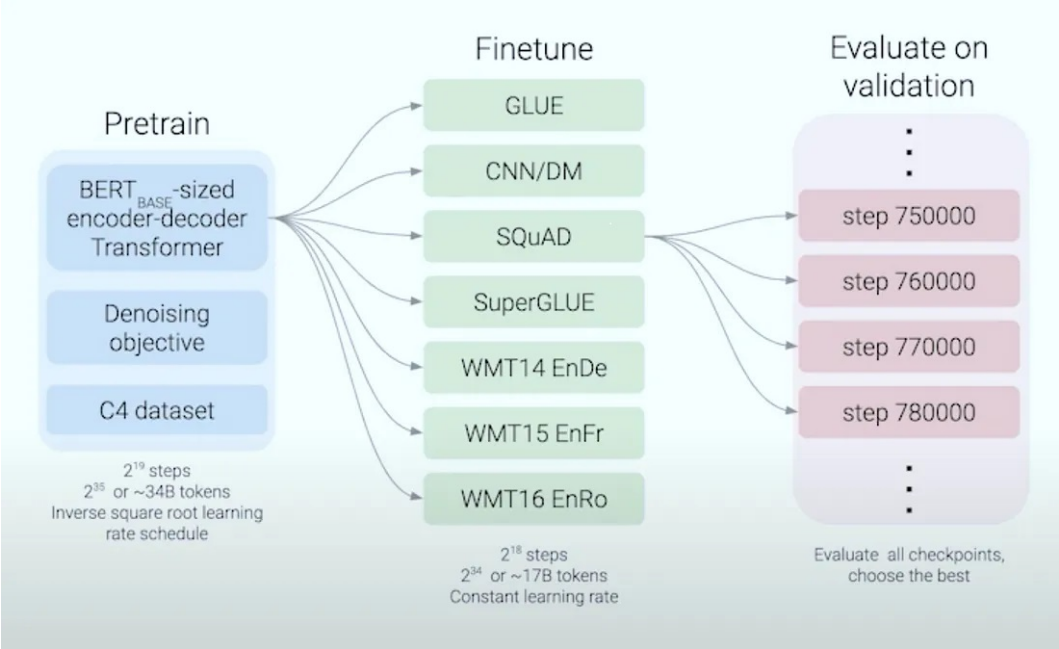
T5 (7), which stands for Text-to-Text Transfer Transformer, was introduced by Colin Raffel et al. It coverts all the NLP tasks into text-to-text formats. It is actually a standard vanilla encoder-decoder model with unsupervised pre-training and supervised fine-tuning.

3.2.3 Bart

The BART (3) (Bidirectional and Auto-Regressive Transformers) model is a powerful and flexible sequence-to-sequence architecture. BART is constructed from a bi-directional encoder like in BERT and an autoregressive decoder like GPT. So it incorporates the strengths of bidirectional pre-training model, enabling it to understand more bidirectional contextual information than GPT. It also absorbs the respective characteristics of GPT's left-to-right decoder, making it more suitable to generate text than BERT. Figure 4 shows the comparision of BART with BERT (4) and GPT (5) .

BART is pre-trained by corrupting documents and then optimizing a reconstruction loss—the cross-entropy between the decoder's output and the original document. The transformation is summarized below:
Token Masking: Following BERT, random tokens are sampled and replaced with MASK elements.

Figure 3: T5 procedure from Colin Raffel's presentation



(3) Token Deletion: Random tokens are deleted from the input. In contrast to token masking, the model must predict which positions are missing inputs. (3) Text Infilling: Several text spans are sampled, with span lengths drawn from a Poisson distribution ($\lambda = 3$). Each span is replaced with a single MASK token. Text infilling teaches the model to predict how many tokens are missing from a span. (3) Sentence Permutation: A document is divided into sentences based on full stops, and these sentences are shuffled in random order. (3) Document Rotation: A token is chosen uniformly at random, and the document is rotated to begin with that token. This task trains the model to identify the start of the document. (3) We can finetune pre-trained bart for our summarization task.

4 Experimental Results and Discussion

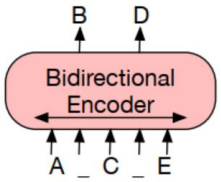
4.1 Experiment setup

4.1.1 Data description

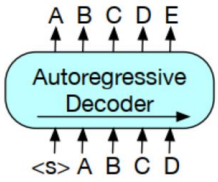
We used the CNN / DailyMail Dataset, which contains 311, 971 unique news articles from CNN and the Daily Mail. Each sample consists of data for three fields, "article", "highlights", and "id". "id" contains the heximal formated SHA1 hash of the url where the story was retrieved from, "article" contains the body of the news article, while "highlights" contains the highlight of the article as written by the article author. We used the data of "highlights" as the labels during train and as "ground-truth" for evaluation.

The average length of "article" is 781 and "highlights" is 56. The following is a sample of the dataset: {'id': '0054d6d30dbcad772e20b22771153a2a9cbeaf62', 'article': '(CNN) – An American woman died aboard a cruise ship that docked at Rio de Janeiro on Tuesday, the same ship on which 86 passengers previously fell ill, according to the state-run Brazilian news agency, Agencia Brasil. The American tourist died aboard the MS Veendam, owned by cruise operator Holland America. Federal Police told Agencia Brasil that forensic doctors were investigating her death. The ship's doctors told police that the woman was elderly and suffered from diabetes and hypertension, according the agency. The other passengers came down with diarrhea prior to her death during an earlier part of the trip, the ship's doctors said. The Veendam left New York 36 days ago for a South America tour.' 'highlights': 'The elderly woman suffered from diabetes and hypertension, ship's doctors say . nPreviously, 86 passengers had fallen ill on the ship, Agencia Brasil says .'}

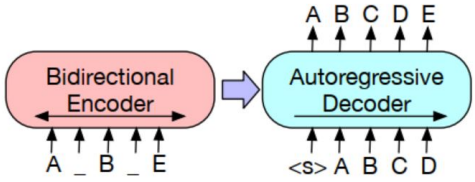
Figure 4: A schematic comparison of BART with BERT(4) and GPT(5)



(a) BERT: Random tokens are replaced with masks, and the document is encoded bidirectionally. Missing tokens are predicted independently, so BERT cannot easily be used for generation.



(b) GPT: Tokens are predicted auto-regressively, meaning GPT can be used for generation. However words can only condition on leftward context, so it cannot learn bidirectional interactions.



(c) BART: Inputs to the encoder need not be aligned with decoder outputs, allowing arbitary noise transformations. Here, a document has been corrupted by replacing spans of text with mask symbols. The corrupted document (left) is encoded with a bidirectional model, and then the likelihood of the original document (right) is calculated with an autoregressive decoder. For fine-tuning, an uncorrupted document is input to both the encoder and decoder, and we use representations from the final hidden state of the decoder.

4.1.2 Data preprocessing

The CNN/DailyMail dataset has 3 splits: training, validation, and test. There are 287,113 instances in the training dataset, 13,368 in the validation dataset, and 11,490 in the test dataset. In order to feed the textual data to the pre-trained LLMs, we processed the data into specific format.

First, we tokenize the textual data into single tokens, and then map the tokens to dictionary indices. Then, in order to train different data instances in a batch, we need to make sure that each instance in a batch has the same length. If an instance is longer than the model requires, we truncate each instance to the maximum length the model can accept. If an instance is shorter than the model requires, we need to pad it to the maximum length the model can accept. We also return an attention_mask pointing out which tokens the model should pay attention to and which ones it should not.In our case, we don't pay attention to padding tokens.

Here is an example after preprocessing:
{'input_ids': [0, 1640, ... 5601, 19, 5, 244, 9, 1416, 31, 5, 21807, 3930, 4, 2, 1, 1, 1, 1], 'attention_mask': [1, 1, ... 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 0], 'labels': [0, 133, 493, 6,..... 1, 1]}.
'input_ids' represents the article we need to summarize, and each number represents a token in the article. We use ones to pad the article to meet the model requirement. In 'attention mask', zeros indicate which tokens are padding. Similar to input_ids, 'labels' represent the tokens in the highlights.

4.1.3 Experiment environment

We run our experiments on Ubuntu 16.04.3(Driver Version: 510.68.0, CUDA Version: 11.6, CuDNN Version: 8.1.0) with 4 GPUs(NVIDIA GeForce GTX 1080 Ti).

The python environment is:
python = 3.9
torch = 2.1.1 tensorflow = 2.14.0
torchvision = 0.16.1
transformers = 4.35.2
numpy = 1.24.1

Table 1: Results from different models

Model	Rouge 1	Rouge 2	Rouge L
Random	0.15	0.08	0.10
TF-IDF	0.24	0.01	0.13
TextRank	0.31	0.13	0.20
Pegasus	0.12	0.02	0.07
T5	0.38	0.12	0.23
Bart	0.25	0.12	0.21

nltk = 3.8.1
tokenizers = 0.15.0

4.1.4 Fine tuning of pre-trained LLMs

Pegasus We download the pre-trained Pegasus model from Hugging Face. There are 568M parameters in the model. We fine tune it on CNN / DailyMail Dataset for 3 epochs, which takes around 7 hours. Some important fine tuning parameters are: learning_rate=2e-5, weight_decay=0.01, per_device_train_batch_size=16, and num_train_epochs=3.

T5 We download the pre-trained T5-small model from Hugging Face. There are 60M parameters in the model. We fine tune it on CNN / DailyMail Dataset for 2 epochs, which takes 7 hours. Some important fine tuning parameters are: learning_rate=2e-5, per_device_train_batch_size=8, and num_train_epochs=2.

Bart We download the pre-trained bart-base model from Hugging Face. There are 139M parameters in the model. We fine tune it on CNN / DailyMail Dataset for 10 epochs, which takes 3 days. Some important fine tuning parameters are: learning_rate=2e-5, weight_decay=0.01, per_device_train_batch_size=8, gradient_accumulation_steps=3, and num_train_epochs=10.

4.2 Model evaluation

We use ROUGE (6), a commonly used evaluation metric for text summarization to compare the performance across different models. Specifically, we calculate ROUGE 1, ROUGE 2 and ROUGE L based F1-score, those three metrics are based on uni-grams, bi-grams and the Largest Common Sequence (LCS) respectively.

We use the six models to generate summaries for the articles in the test dataset, and compare those generated summaries with ground-truth, which is the highlights in the test dataset. The average values of the evaluation metrics for different models are as follows: Note: the results for T5 are based on the first 1,000 instances in the test dataset due to limited resources.

As we can see from the table, both T5 and Bart has decent performance. Especially, T5 outperforms not only the baseline models, but also the other two pre-trained LLMs.

5 Conclusion

Being fine-tuned based on the target datasets, the existing pre-trained LLMs are capable of efficiently condensing extensive textual data into coherent and concise summaries. Those summaries are not just shorter versions of the original text but also maintain the essence and context of the content.

This project is an excellent learning experience for us. The application of different summarization techniques, from simple baselines like random summarization and TF-IDF based summarization to more complex ones like TextRank and transformer-based models, has allowed us to explore a broad spectrum of methodologies. Especially, given the immense popularity of LLMs, it's exciting to fine-tune those models using the data and for the tasks that we're interested in.

Encouraged by the results of this project, we will focus on tuning the parameters and increasing the number of epoches to further improve the performance in the future, and exploring the integration of the summarization models to real-world platforms and systems.

6 Contribution

Jinda(Justin) Zhang: Coding: 3 baseline models, Pegasus, Evaluation, Report, Presentation, Dataset Visualization, Slides.
Juan Yu: Coding: T5, Evaluation, Report, Presentation, Slides.
Changyu Liu: Coding: Bart, Evaluation, Report, Presentation, Slides.
Ye Zhang: Report, Slides.

References

[1] A. Vaswani et al., “Attention Is All You Need.” arXiv, Aug. 01, 2023. Accessed: Nov. 09, 2023. [Online]. Available: http://arxiv.org/abs/1706.03762

[2] J. Zhang, Y. Zhao, M. Saleh, and P. J. Liu, “PEGASUS: Pre-training with Extracted Gap-sentences for Abstractive Summarization,” arXiv.org. Accessed: Dec. 01, 2023. [Online]. Available: https://arxiv.org/abs/1912.08777v3

[3] Lewis, Mike, et al. "Bart: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension." arXiv preprint arXiv:1910.13461 (2019)

[4] Devlin, Jacob, et al. "Bert: Pre-training of deep bidirectional transformers for language understanding." arXiv preprint arXiv:1810.04805 (2018)

[5] Radford, Alec, et al. "Improving language understanding by generative pre-training." (2018)

[6] Lin, Chin-Yew. "Rouge: A package for automatic evaluation of summaries." Text summarization branches out. 2004.

[7] Raffel, Colin, et al. "Exploring the limits of transfer learning with a unified text-to-text transformer." The Journal of Machine Learning Research 21.1 (2020): 5485-5551.