Text Summarization using XSum Dataset

Fahim Shahriar
University of Minnesota Duluth
shahr072@d.umn.edu

Abstract

This paper summarizes the process of training llm models on XSUM dataset collected from huggingface and includes evaluation of how the trained models performs along with a baseline comparsion. It aims to see how well llm models can perform in a widely used natural language processing task, summarization. The main focus here was to generate abstractive summary. After initial inspection of the dataset, we decided to create a baseline by training a subset of the data using pre-trained t5-small model, then perfrom our final training using bart-large-xsum model. We evaluated our model performances using ROUGE and BERTscore metrics, with an intention of measuring both lexical similarity and semantic similarity. We were able to achieve a ROUGE-L score of 0.3475 and ROUGE-1 score of 0.4281, for from our final training. This study was conducted using the computational resources at the University of Minnesota. In future, we may explore the use of larger models such as T5-3B or BART-Large, fine-tuning on domain-specific data, and incorporating human evaluation to gain deeper insights into summary quality.

1 Introduction

The world is an ever growing compilation of text data, in forms of news articles, scientific publications, social media, and business reports etc. These enormous amount of texts has necessitated the development of efficient methods for extracting meaningful and concise information. Text summarization has been used by humans since a long time to present in a clear and concise but within limited words, the gist of the actual text. Since the introduction of natural language processing (NLP), it has been employed as a tool to do text summarization. This applicaiton of NLP has also evolved a lot along with a evolution of NLP and introduction of new techniques and pre-trained models. This work aims to explore the potential of modern machine learning techniques for the task of text summarization, leveraging state-of-the-art transformer-based models.

Automatic text summarization has two primary approaches, **extractive summarization** and **abstractive summarization**. While Extractive summarization selects key phrases or sentences from the source text verbatim, abstractive summarization generates new sentences that convey the core meaning of the source text. With the advent of deep learning and transformer-based architectures, abstractive summarization has shown remarkable progress, achieving summaries that are more fluent and contextually accurate, while using phrases and words not present in the actual text, making it more lively and close to what a human would do.

In this work, we perform baselining with the t5-small model and for our final training and evaluation, we utilize the BART (Bidirectional and Auto-Regressive Transformers) model, a pre-trained sequence-to-sequence transformer model designed for text generation tasks, including summarization. BART combines the strengths of bidirectional encoding, like in BERT, and autoregressive decoding, like in GPT, making it particularly effective for abstractive summarization. Specifically, we employ BART-Large and fine-tune it on a dataset of documents and their respective summaries to generate high-quality abstractive summaries.

This work contributes to the growing field of NLP by demonstrating how transformer-based models can be effectively leveraged for summarization tasks. Through detailed analyses and experiments, we aim to provide a robust approach for summarization that can be applied to various domains. The rest of the paper is structured as follows: Section 2 describes the previous works done in this field, Section 3 outlines the methodology followed, describing the dataset and preprocessing steps. Section 4 details our findings. Finally, Section 5 concludes the paper, links to the dataset and our codebase is listed in the appendix section.

2 Related Work

Text summarization, a core task in natural language processing (NLP), has shown considerable progress in recent years, especially due to the emergence of deep learning methodologies and transformer-based architectures. This section offers a summary of the most significant research in the topic, emphasizing essential approaches and their contributions.

Initial research in automatic text summarizing concentrated on extraction techniques, aiming to identify significant sentences or phrases from the source text. Methods such as TextRank (Mihalcea and Tarau, 2004), derived from Google's PageRank algorithm, and latent semantic analysis (LSA) (Gong and Liu, 2001), were frequently utilized. These methods predominantly utilized statistical and graph-based techniques to prioritize phrases by significance; nonetheless, they frequently exhibited insufficient flexibility and fluency for abstractive summarization.

The field underwent a major change when sequence-to-sequence (seq2seq) models, especially those including recurrent neural networks (RNNs) with attention mechanisms were adopted. (Bahdanau, 2014) et al. suggested the attention mechanism to enhance sequential models so that they can concentrate on pertinent portions of the input text during summarizing. Later, (See et al., 2017) et al. added the Pointer-Generator Network, which lets copying of words straight from the input by combining extractive and abstractive powers.

Vaswani et al. created the transformer architecture (Vaswani, 2017), which revolutionized NLP by enabling parallel processing and greater capture of long-range dependencies. This design is the foundation of modern summarization models. BERTSUM (Liu and Lapata, 2019), an extension of BERT for summarizing, enhanced extractive summarization by fine-tuning BERT for sentence-level tasks. Similarly, PEGASUS (Zhang et al., 2020) established pre-training objectives tailored exclusively for summarization tasks, demonstrating cutting-edge performance on many benchmarks.

Among the most influential transformer-based models for abstractive summarization are BART (Bidirectional and Auto-Regressive Transformers) (Lewis, 2019) and T5 (Text-to-Text Transfer Transformer) (Raffel et al., 2020). With pre-training on a denoising aim, BART integrates a bidirectional encoder and an autoregressive decoder to efficiently handle noisy inputs. On summarizing datasets like XSum and CNN/DailyMail, it has performed exceptionally well.

Datasets such as CNN/DailyMail (Hermann et al., 2015) and XSum (Narayan et al., 1808) have become standard benchmarks for summarization tasks, providing diverse and challenging examples for both extractive and abstractive methods. Evaluation metrics like ROUGE (Recall-Oriented Understudy for Gisting Evaluation) (Lin, 2004) are widely used to measure the overlap between generated and reference summaries. However, researchers have highlighted the limitations of ROUGE in capturing fluency and semantic correctness, prompting exploration of metrics such as BERTScore (Zhang et al., 2019).

Recent work has demonstrated the importance of transfer learning and task-specific training techniques to achieve state-of-the-art performance. For instance, Liu and Lapata (Liu and Lapata, 2019) demonstrated the effectiveness of pre-trained language models like BERT for summarization tasks, while (Lewis, 2019) et al. emphasized the versatility of BART for both abstractive and extractive summarization.

Despite the remarkable progress, challenges remain in generating summaries that are both factually accurate and linguistically fluent. Faithfulness, in particular, is a significant concern, with models sometimes generating "hallucinated" content that deviates from the input. Addressing this issue has been the focus of recent research, such as the work by Maynez et al. (Maynez et al., 2020), who proposed novel metrics for factual consistency.

3 Methodology

For our exploration, We used the,Xsum dataset. Key attributes of the dataset include Input, which are entries like News articles (referred to as the "document" field). Outputs are one-sentence summary (referred to as the "summary" field). There are 204,045 examples

3.1 Exploratory data analysis

The dataset contains 204,045 entries across three columns: document, summary, and id.
Each entry consists of:
Document: The full text of an article or content.
Summary: A shorter version or abstract of the doc-

ument.
ID: A unique identifier for each entry.
Dataset Characteristics: All columns are non-null, indicating no missing values. Unique Counts:
document: 203,846 unique entries.
summary: 203,107 unique entries.
id: All 204,045 entries are unique.
Most Frequent Entry: The most frequent document occurs 28 times and is "A chronology of key events". The most frequent summary occurs 108 times. Below is a table giving a closer look at document and summary lengths.

Metric	Document Length	Summary Length
Count	204,045	204,045
Mean	2,202.12	125.46
Std. Deviation	1,795.62	30.24
Minimum	0	1
25th Percentile	1,040	106
Median (50%)	1,742	126
75th Percentile	2,898	144
Maximum	174,045	399

Table 1: Statistics for Document and Summary Lengths

From Table 1, we can see that document lengths are highly variable, with a maximum of 174,045 characters. On the other hand, Summary lengths are more consistent, with a maximum of 399 characters and a median of 126 characters. A small number of documents have length 0, likely indicating missing or empty text content.
We have also included two figures showing the length distribution of the document text (Figure 1) and summary text (Figure 2).

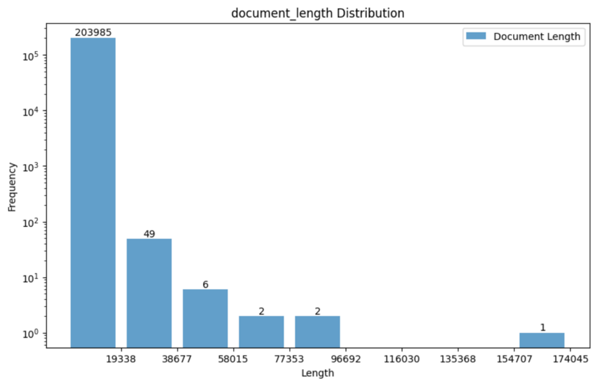

Figure 1: Document Length Distribution

Common Words in Documents:
The dataset's most frequent words were analyzed after tokenization using NLTK. The 20 most common words are listed in the Table 2
The top words consist mostly of stopwords (the, to, of, etc.). Punctuation marks (., ,) also appear frequently, reflecting the raw nature of the text.

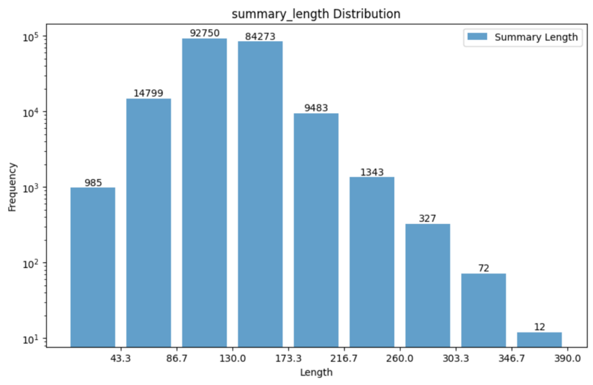

Figure 2: Summary Length Distribution

Word	Frequency
the	4,743,678
.	3,665,232
,	3,352,460
to	2,097,468
of	1,796,086
and	1,772,944
a	1,740,210
in	1,688,656
“	1,143,349
”	944,286

Table 2: Most common words and their occurances

3.2 Models

We trained our data using t5-Small and BART-large-Xsum model. You can find a comparison of these two models along with the advanced t5-Base model in table 3.

3.3 Evaluation Metric

To evaluate the performance of the summarization models, we utilized two widely-used evaluation metrics: **ROUGE** and **BERTScore**.

ROUGE (Recall-Oriented Understudy for Gisting Evaluation) is a set of metrics primarily used for evaluating automatic summarization by comparing n-grams (unigrams, bigrams, etc.) in the generated summary against reference summaries. Specifically, we employed the following ROUGE metrics:

ROUGE-1: Measures the overlap of unigrams between the generated and reference summaries. ROUGE-2: Measures the overlap of bigrams between the generated and reference summaries. ROUGE-L: Measures the longest common subsequence (LCS) between the generated and reference summaries, which accounts for the sequence of

Feature	T5-Small	T5-Base	BART-Large-XSum
Model Size	60M parameters	220M parameters	406M parameters
Pretraining Task	Text-to-Text (denoising)	Text-to-Text (denoising)	Denoising autoencoder
Dataset for Pretraining	C4 dataset (Colossal Clean Crawled Corpus)	C4 dataset	Books, Wikipedia, and Web-Data
Fine-Tuned Dataset	None (Generic Pretraining)	None (Generic Pretraining)	XSum (Extreme Summarization)
Max Input Length	512 tokens	512 tokens	1024 tokens
Max Output Length	512 tokens	512 tokens	1024 tokens
Performance	Lightweight, Faster Training	Balanced Performance	High Quality, Resource Intensive
Best Use Cases	Quick prototyping, small-scale tasks	Moderate-scale tasks, general-purpose text generation	High-quality abstractive summarization
Fine-Tuning Speed	Fast	Moderate	Slower
Inference Speed	Fast	Moderate	Slower
Computational Resources	Low	Moderate	High
ROUGE-L Score on XSum	Lower (e.g., 0.18-0.22)	Moderate (e.g., 0.23-0.27)	Higher (e.g., 0.29-0.31)

Table 3: Comparison of T5-Small, T5-Base, and BART-Large-XSum Pre-trained Models

words and their order. ROUGE is widely used in summarization tasks as it exhibits better correlations for coherence and fluency. But it evaluates poorer correlations for consistency and relevance, which is a common problem fro lexical similarity based-metrics.

BERTScore is another evaluation metric that uses contextual embeddings from pre-trained language models like BERT to compute a similarity score between the generated and reference summaries. BERTScore is based on the cosine similarity between the embeddings of the words in the generated summary and the reference summary. The key benefits of BERTScore are: It leverages contextualized word representations, making it more sensitive to semantic similarity, unlike traditional n-gram-based metrics.
Both ROUGE and BERTScore offer complementary views of the summary quality, with ROUGE focusing on surface-level overlap and BERTScore assessing deeper semantic similarity. The combination of these two metrics provides a comprehensive evaluation framework for comparing different summarization models.

4 Result

4.1 Baselining

Initially, we decided to get a baseline on a subset of our actual dataset. We saved our training, test and validation data into hugging face dataset instead of pandas dataframe, to ensure the use of gpu power. Then we used shuffle function to trim the training dataset to 10000, validation and test dataset to 2000 each.

For baselining, We decided to train using t5-small pre-trained model. So we tokenized our training and validation set using the AutoTokenizer from t5-small model.
We used learning_rate=5e-5, which is standart for nlp tasks, both train and evaluation batch size was set to 8, we also set fp16=True reducing the floating point precision for faster computations.Then, running for only one epoch, we got Training Loss (0.705100) and Validation Loss (0.643028), indicating good generalisation on unseen data.

For 2nd run, we made the following changes, increased the token length to 1024, instead of 512 and trained it for 5 epoches instead of 1. The changes are evident from Figure 3, indicating very small improvement on losses after 3rd epoch.

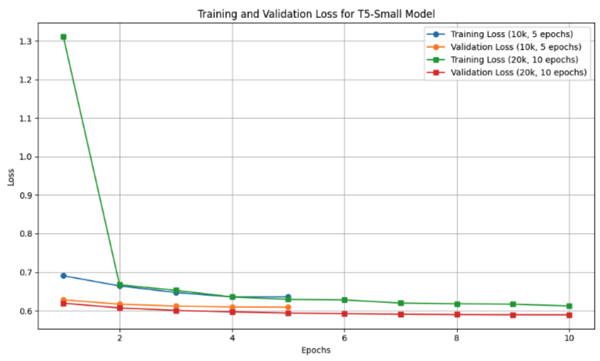

Figure 3: Line plot for Training vs Validation Loss

The Rogue score we got after changes are listed in Table 4. The contrast between Rouge1 and Rouge2 scores indicated that our model struggled to maintain contextual word pairs, potentially affecting readability and context.

At this point, we were able to obtain access to our university's remote computational resource containing 2 NVIDIA A100-PCIE-40GB. Then, we increased our dataset to 20k and ran it for 10 epochs, losses for this iteration is plotted in Figure 3:
This time, all the rouge scores increased a bit, as you can see in Figure 3. Rouge-1 and Rouge-L scores increased by almost 5%, while Rouge-2 score increased only 1%. In the following chapter, we will discuss how we trained our final model and the results we obtained.

4.2 Final Training

For our final model, we decided to use facebook/bart-large-xsum pre-trained model, which has 406M parameters. As this can take a long time to load and training using this can go on for very long time, we trained on 20k documents to get an initial estimation of the computational power we have. We also put significant effort to optimize the training parameters with a target to put the powerful gpu and cpu to maximum use. after this, with just 3 epochs, we got better result than t5-small. Training loss came down to 0.32 and validation loss came down to 0.44. The rouge scores also improved, as shown in Figure 3.
After that, we ran the facebook/bart-large-xsum pre-trained model with the whole dataset, it took around 17 hours to train on "akka" machine having 2 A100 40gb nvidia gpu and 128 core cpu. We set the per device train and evaluation batch size to 16, increased number of dataloader workers to 32 to maximize cpu usage and set gradient accumulation steps to 2, effectively doubling the batch size. We summarized our findings based on rogue score in table 4.

For an easier visualization, we also plotted the rouge scores into a grouped bar chart in Fig 4, depicting both score changes for each model we trained on and at the same time, how rogue scores increased along with the complexity of our models.

We also calcualated BERTscore using both t5-small and bart-large xsum model that we trained and saved in our directory. The BERTScore evaluation provided valuable insights into the semantic quality of the summaries generated by the two trained models, T5-Small and BART-Large-XSum. The BART-Large-XSum model achieved higher scores across all metrics, with an average precision of 0.9024, recall of 0.9148, and F1 score of 0.9085

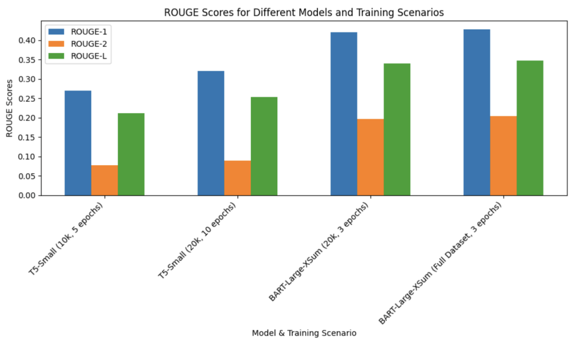

Figure 4: Grouped Bar chart showing change in Rouge scores by model

which are very close to state-of-the-art values. In comparison, the T5-Small model yielded slightly lower results, with an average precision of 0.8763, recall of 0.8831, and F1 score of 0.8795. These results highlight the superior semantic understanding and contextual alignment of the BART-Large-XSum model, attributed to its larger capacity and better optimization for summarization tasks. Also worth mentioning here is that the t5-small model was trained on a subset of data, rather than the whole dataset. The relatively high BERTScores for both models demonstrate their effectiveness in capturing semantic similarity, but the substantial improvement with BART-Large-XSum underscores the advantages of leveraging larger, more advanced pre-trained models for abstractive summarization.

4.3 Qualitative analysis

We saved all our models after we trained them. So we were able to generate summaries from each of the models and look at them. To get an understanding, we analyzed summaries generated for first ten test cases, both by t5-small and BART-large trained on the whole dataset. We discussed some interesting findings below.

Firstly, in one summary, BART-large model identified the victim as a black man, while in reality he was white, correctly summarized by t5-small. You can find each of those summaries below:
Actual reference summary: "Four people accused of kidnapping and torturing a mentally disabled man in a "racially motivated" attack streamed on Facebook have been denied bail".
Summary by t5-small: "Four men have appeared in court charged with aggravated kidnapping and aggravated kidnapping of a white man who was

Model	Training Type	ROUGE-1	ROUGE-2	ROUGE-L
T5-Small	Baselining (10k, 5 epochs)	0.2702	0.0772	0.2110
T5-Small	Expanded Dataset (20k, 10 epochs)	0.3200	0.0895	0.2534
BART-Large-XSum	Baselining (20k, 3 epochs)	0.4208	0.1957	0.3402
BART-Large-XSum	Full Dataset , 3 epochs, 17 hours	0.4281	0.2042	0.3475

Table 4: ROUGE Scores for Different Models and Training Scenarios

beaten in a van".
Summary by BART-large: "A judge has ordered four people accused of beating a **disabled black man** and posting video of it on Facebook, to be held without bail pending further court action".

Secondly, both models seems to perform bad while summarizing a short passage, with t5-small making a wrong interpretation and BART-large taking a wrong name not even mentioned in the passage.
The main passage: "The 48-year-old former Arsenal goalkeeper played for the Royals for four years. He was appointed youth academy director in 2000 and has been director of football since 2003. A West Brom statement said: "He played a key role in the Championship club twice winning promotion to the Premier League in 2006 and 2012.""
Summary by t5-small: "West Brom have appointed West Brom's former youth academy director of football after he was appointed a youth academy director."
Summary by BART-large: "West Brom have appointed **Steve Round** as their new head of academy and first team coach after parting company with Reading on a two-year deal."
Additionally, t5-small model also gave incorrect information while summarizing a passage about a global merger. While both models demonstrate strong summarization capabilities, BART-Large outperforms T5-Small in generating detailed summaries. However, its tendency to introduce factual errors highlights the importance of manual review or hybrid models that combine extractive and abstractive approaches. T5-Small, while less informative, is more reliable in avoiding fabricated information, making it better suited for contexts where accuracy is paramount over detail.

5 Discussion

The primary goal of this study was to evaluate the performance of large language models (LLMs) in abstractive summarization tasks, using the XSUM dataset. Our final model, BART-Large-XSum, achieved a ROUGE-L score of 0.3475 and a ROUGE-1 score of 0.4281, demonstrating competitive performance in the summarization task. The relatively high ROUGE-1 score suggests that BART-Large-XSum is successful in capturing key phrases from the reference summaries. The ROUGE-L score also indicates that the model is capable of maintaining the overall structure and coherence of the original documents. Previous studies using BART for summarization on similar datasets have reported higher ROUGE scores, but our results are in line with the typical performance range for models of this scale on the XSUM dataset. These findings suggest that BART-Large-XSum is a strong candidate for abstractive summarization tasks, with the potential for integration into automated content generation applications, such as news summarization or summarizing large volumes of scientific literature. Despite some limitations, the results provide a solid foundation for future research and applications in text summarization.

References

Dzmitry Bahdanau. 2014. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*.

Yihong Gong and Xin Liu. 2001. Generic text summarization using relevance measure and latent semantic analysis. In *Proceedings of the 24th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 19–25.

Karl Moritz Hermann, Tomas Kocisky, Edward Grefenstette, Lasse Espeholt, Will Kay, Mustafa Suleyman, and Phil Blunsom. 2015. Teaching machines to read and comprehend. *Advances in neural information processing systems*, 28.

M Lewis. 2019. Bart: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. *arXiv preprint arXiv:1910.13461*.

Chin-Yew Lin. 2004. Rouge: A package for automatic evaluation of summaries. In *Text summarization branches out*, pages 74–81.

Yang Liu and Mirella Lapata. 2019. Text summarization with pretrained encoders. *arXiv preprint arXiv:1908.08345*.

Joshua Maynez, Shashi Narayan, Bernd Bohnet, and Ryan McDonald. 2020. On faithfulness and factuality in abstractive summarization. *arXiv preprint arXiv:2005.00661*.

Rada Mihalcea and Paul Tarau. 2004. Textrank: Bringing order into text. In *Proceedings of the 2004 conference on empirical methods in natural language processing*, pages 404–411.

Shashi Narayan, Shay B Cohen, and Mirella Lapata. 1808. Don't give me the details, just the summary. *Topic-Aware Convolutional Neural Networks for Extreme Summarization ArXiv, abs*.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J Liu. 2020. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of machine learning research*, 21(140):1–67.

Abigail See, Peter J Liu, and Christopher D Manning. 2017. Get to the point: Summarization with pointer-generator networks. *arXiv preprint arXiv:1704.04368*.

A Vaswani. 2017. Attention is all you need. *Advances in Neural Information Processing Systems*.

Jingqing Zhang, Yao Zhao, Mohammad Saleh, and Peter Liu. 2020. Pegasus: Pre-training with extracted gap-sentences for abstractive summarization. In *International conference on machine learning*, pages 11328–11339. PMLR.

Tianyi Zhang, Varsha Kishore, Felix Wu, Kilian Q Weinberger, and Yoav Artzi. 2019. Bertscore: Evaluating text generation with bert. *arXiv preprint arXiv:1904.09675*.

A Appendix

A.1 Dataset

Here is the link to the dataset we used:
Xsum
It's a publicly available dataset from Hugging Face.

A.2 Codebase

All the codes we used for our training and calulation (.py and .ipynb) are available in this github repository.