Summarization of Large Text Volumes Using Large Language Models

Adrián David Navarro Betrián [1]
dlearning43@gmail.com
[1] Universitat Pompeu Fabra
Barcelona, Catalonia, Spain

Andreas Kaltenbrunner [2,1]
kaltenbrunner@gmail.com
[2] Universitat Oberta de Catalunya
Barcelona, Catalonia, Spain

ABSTRACT

This article involves the development and evaluation of various pipelines centered on Large Language Models (LLMs). The goal is to make them capable of summarizing extensive texts, such as books or video transcripts, producing complete summaries, despite the context window limitations of these models.

Various models and pipelines are tested using different metrics and an interface to allow volunteers to generate and evaluate summaries from video transcripts. Raw models and more complex pipelines are compared to determine their effectiveness. The pipelines are designed based on three different approaches: a recursive method, a combination method that includes an extractive summarizer, and a hybrid approach that incorporates both the recursive and extractive pipelines.

The baseline models have the capacity to summarize only up to 512 tokens (approximately 400 words), while the pipelines developed here are capable of summarizing video transcriptions of more than one hour, without a specific token limitation.

CCS CONCEPTS

• **Computing methodologies → Information extraction**.

KEYWORDS

AI, LLM, NLP, Text summarization, Pipelines

ACM Reference Format:
Adrián David Navarro Betrián [1] and Andreas Kaltenbrunner [2,1]. 2024. Summarization of Large Text Volumes Using Large Language Models. In *Proceedings of 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD '24).* ACM, New York, NY, USA, 6 pages. https://doi.org/XXXXXXX.XXXXXXX

1 INTRODUCTION

In recent years, the rise of artificial intelligence has enabled significant advances in the area of Natural Language Processing (NLP). This progress has led to great improvements in tasks such as summary generation, text translation, answering text-based questions, and text rewriting, among others. However, Large Language Models (LLMs) face a significant limitation: their context window.

Typically, LLMs had a context window ranging from 512 to 4096 tokens, for example, GPT-3 can process up to 4096 tokens [12]. Until recently, this limit was extended to 128K tokens with models like GPT-4 Turbo [14], approximately 91K words. This means that they can only "remember" and reason about a limited amount of information. For example, if an LLM reads a book, it may not be able to remember the details of the beginning or middle of the book when it reaches the end. This is a major obstacle for tasks such as summary generation and question answering, where all relevant information needs to be taken into account.

Therefore, the objective of this article is to develop a pipeline that enables an LLM to summarize large volumes of text without losing important information. To achieve this, various approaches will be explored and compared:

- **Original Pipeline:** involves using the raw model as is, without any modification.
- **Extractive Pipeline:** combines extractive summarization techniques with the raw model to reduce the original length of the input texts and make the model more capable of summarizing correctly.
- **Recursive Pipeline:** divides the original text into smaller parts and calls the model multiple times on these segments, by doing this we can keep the most important information independent from where it appears.
- **Extractive and Recursive Pipeline:** a combination of the second and third pipelines. It first uses a model that performs extractive summarization reducing the original length and then divides and abstractly summarizes the different segments.

Each of these pipelines will be studied and compared to determine their effectiveness in text summarization. They will be tested with different raw models to see if they generally improve or not the performance of the raw model. To measure the performance, we will use some known metrics like Meteor, Rouge-1, -2, and -L, and cosine similarity. Additionally, some volunteers will be asked to evaluate the generated summaries.

2 STATE OF THE ART OF TEXT SUMMARIZATION

One of the goals of NLP is for machines to understand, interpret, and generate text and words from human language [11]. This field combines computational linguistics with machine learning and deep learning. This article will focus on the area of NLP that seeks to summarize texts.

Text summarization is defined as the process of "Reducing to short and precise terms, or only considering and briefly repeating the essence of a matter or subject" [5].

KDD '24, August 25–29, 2024, Barcelona, Spain

ACM ISBN 978-1-4503-XXXX-X/18/06
https://doi.org/XXXXXXX.XXXXXXX

In this article, the aim is to perform summarizations using Automatic Text Summarization (ATS). This technique can offer numerous advantages over summaries generated by humans, such as the ability to generate summaries much faster, obtain more impartial summaries by eliminating human bias, and the possibility of generating higher quality summaries than those generated by humans.

There are two main types of summaries: extractive and abstract. Extractive summaries are created by selecting the most relevant sentences from the original text. This approach is less complex as it does not require a full understanding of the text, simply extracting the most informative parts. Techniques such as TF-IDF, PageRank [1] or clustering algorithms [4] are used to carry out these summaries to determine which words are most relevant in the text.

On the other hand, abstract summaries, where a model can generate new sentences that summarize the text, require a greater understanding of the text. In this case, the model interprets the content and generates a summary that maintains the original meaning. Although this process is more complicated, it can result in more coherent and easy-to-read summaries [18].

In this work, we will focus on the realization of abstract summaries. In Section 3.2 we explain in more detail some of the LLMs capable of performing this task. That models are T5[15] and Pegasus [20] from Google, Bart [9] from Facebook, and PropherNet [19] from Microsoft.

3 METHODOLOGY

This section explains how the project was organized and how it has been planned to obtain and evaluate the results.

3.1 Programming Environment

The programming language utilized is Python. The environment for executing tests and verifying results is Colab, which is equipped with a T4 GPU. Due to the usage limitations of Colab, the High-Performance Computing Cluster of Pompeu Fabra University is used to perform calculations with large volumes of text [17].

3.2 Model Selection

The initial step involves searching for various models and checkpoints from Hugging Face capable of summarizing. Due to resource limitations, the search is centered on lighter models, excluding models with more than a billion parameters, such as GPT-3 [7], Gemma [16], or LLama2[6]. These models, ranging from 200M to 600M parameters, are designed to understand text and generate abstract summaries. They are used because they are more resource efficient than larger LLMs. Additionally, they have a context window of 512 tokens, around 400 words. Consequently, any input exceeding 512 tokens will result in the truncation of subsequent tokens.

Considered models have a similar encoder-decoder structure, based on transformers but with slight variations in their architecture or their pretraining:

- **Bart[9]:** Bart, which stands for Bidirectional and Auto-Regressive Transformers, is an LLM developed by Facebook AI in October of 2019.
- **T5[15]:**T5, also known as "Text-To-Text Transfer Transformer" was an LLM created by Google AI in October of 2019.
- **Pegaus[20]:** Pegasus, which stands for Pre-training with Extracted Gap-sentences for Abstractive Summarization Sequence-to-sequence, is a model developed by Google AI in December 2019.
- **ProphetNet[19]:** ProphetNet was created by Microsoft, the University of Science and Technology of China, and Sichuan University in January of 2020.

3.2.1 *Other auxiliary algorithms.*

- **WhisperX[2]:** This library from Github contains an optimized version of OpenAI's Whisper model, which is useful for transcribing audios quickly and accurately with timestamps. It is used in the pipeline where videos are transcribed to summarize them.
- **bert-extractive-summarizer[4]:** This library uses BERT, Bidirectional Encoder Representations from Transformers. It is trained to perform extractive summaries by grouping the embeddings that represent the sentences of the text by similarity.

3.3 Pipeline Development

We propose several different pipelines to generate summaries of long texts using the models mentioned above. Although we also perform other experiments we illustrate them here using transcriptions of YouTube videos as input which we generated with WhisperX.

3.3.1 Original Pipeline. This pipeline uses the summarization models, explained in Section 3.2 of above, without any modifications to generate the summaries of the input texts.

These are the straightforward approaches that we use as baselines. Their limitation is that they can only process the first X tokens, as the context window of each model restricts it.

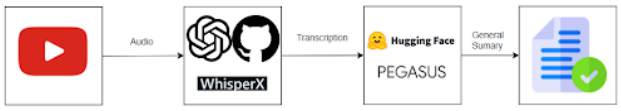

Figure 1: Schema of the original pipeline.

3.3.2 Extractive Pipeline. In this approach, the bert-extractive-summarizer is incorporated, which reduces the original text by first performing an extractive summarisation, selecting its most important parts. Subsequently, the model processes and summarizes this selected subset. This pipeline allows a model with context limitations to process more text, but still has size limitations.

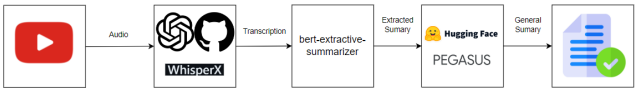

Figure 2: Shema of the extractive pipeline

3.3.3 Recursive Pipeline. Due to the ongoing limitation of the model's context window, a recursive approach has been employed. In this approach, the model progressively summarizes sections of the text, until the text is reduced to less than 500 tokens. By doing this, when the final step is reached, there are no longer problems with having to summarize an excessive amount of text, which can significantly improve the results.

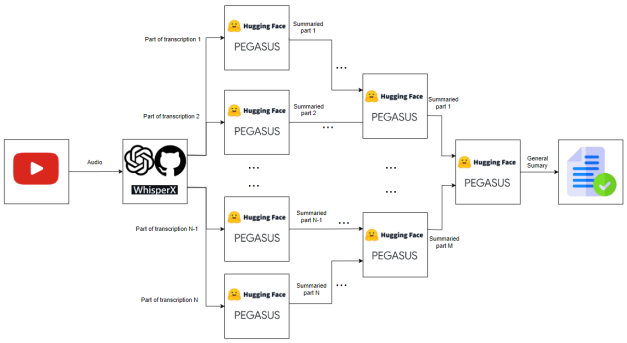

Figure 3: Schema of the recursive pipeline.

3.3.4 Extractive and Recursive Pipeline. This approach combines the two previous pipelines. First, an extractive summary of the text is made to reduce its volume, eliminating the less important sentences. Then, the recursive pipeline is applied and the text is progressively reduced.

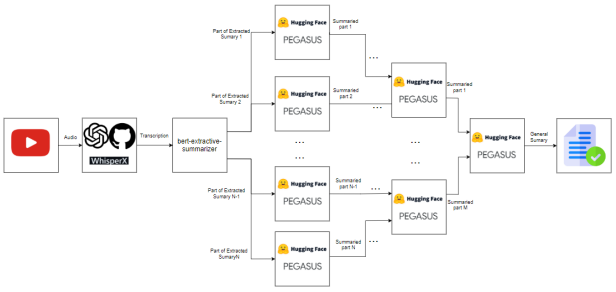

Figure 4: Schema of the extractive and recursive pipeline.

3.4 Model Comparison and Evaluation

Models and pipelines are evaluated to determine which offers the most effective results with two evaluation strategies.

The first (automatized) evaluation strategy (illustrated in Figure 5) uses the BookSum dataset[8], which contains book chapters and their summaries. This dataset, amounting to 379 MB, contains 12,515 samples. These samples provide reference summaries that we compare with the outcomes of our pipelines applying metrics such as ROUGE[10], METEOR[3], and cosine similarity[13].

The second (manual) evaluation strategy uses transcriptions of long YouTube videos. Although these are long texts they can be easily evaluated by human volunteers who have watched the

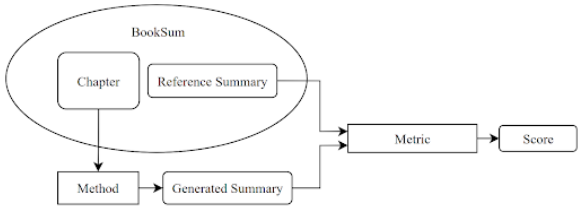

Figure 5: Illustration of the process of comparing pipelines with metrics. BookSum chapters from renowned books and their corresponding summaries are used as references and compared to the generated summaries using various metrics.

videos to identify hallucinations or inconsistencies in the generated summaries. We refer to this process as User Evaluation and Feedback

3.5 User Evaluation and Feedback Framework

A framework for human evaluation is designed to compare various summarization pipelines. Four different summaries, generated from video transcriptions through software with an interactive interface, are provided to volunteers. Instead of reading extensive texts, volunteers are suggested to select a video they have already viewed, after which they evaluate the accuracy of the generated summaries. The objective is to collect feedback from the volunteers on the summarization pipeline they perceive as most effective. This feedback is collected through a survey where participants indicate on a Likert scale from 1 to 5 if the summary does not reflect the content of the video they viewed (1), or on the contrary, represents the video totally (5). The summaries are anonymous, meaning volunteers are unaware of which summary is generated by each method. The collected data is then used to evaluate and compare the effectiveness of the various summarization pipelines.

3.6 Challenges Faced

During the development of this project, some challenges appeared and modified the initially foreseen path of experiments.

Initially, we planned to use the GPUs provided by Colab. However, Colab has limitations in terms of time and computation. We primarily used it for human evaluation, running Streamlit with the Colab backend. A challenge we faced was that a new link was generated each time Colab was executed. Furthermore, computational limitations led us to seek alternatives for calculating the metrics of the samples.

To address this, we utilized the High-Performance Computing (HPC) infrastructure of the UPF, which facilitated working with high volumes of samples. However, a policy change restricted access to the GPUs during the execution of the project, preventing us from calculating our experiments for larger sample sizes. As a result, we had to proceed with only 10 samples of the BookSum data set for the metrics evaluation.

Additionally, it was observed that different models produced summaries of varying lengths. To ensure fairness in the metric evaluation and the human evaluation process, the models were adjusted to output summaries of approximately 75 words. This adjustment was important as most metrics assign scores based on the

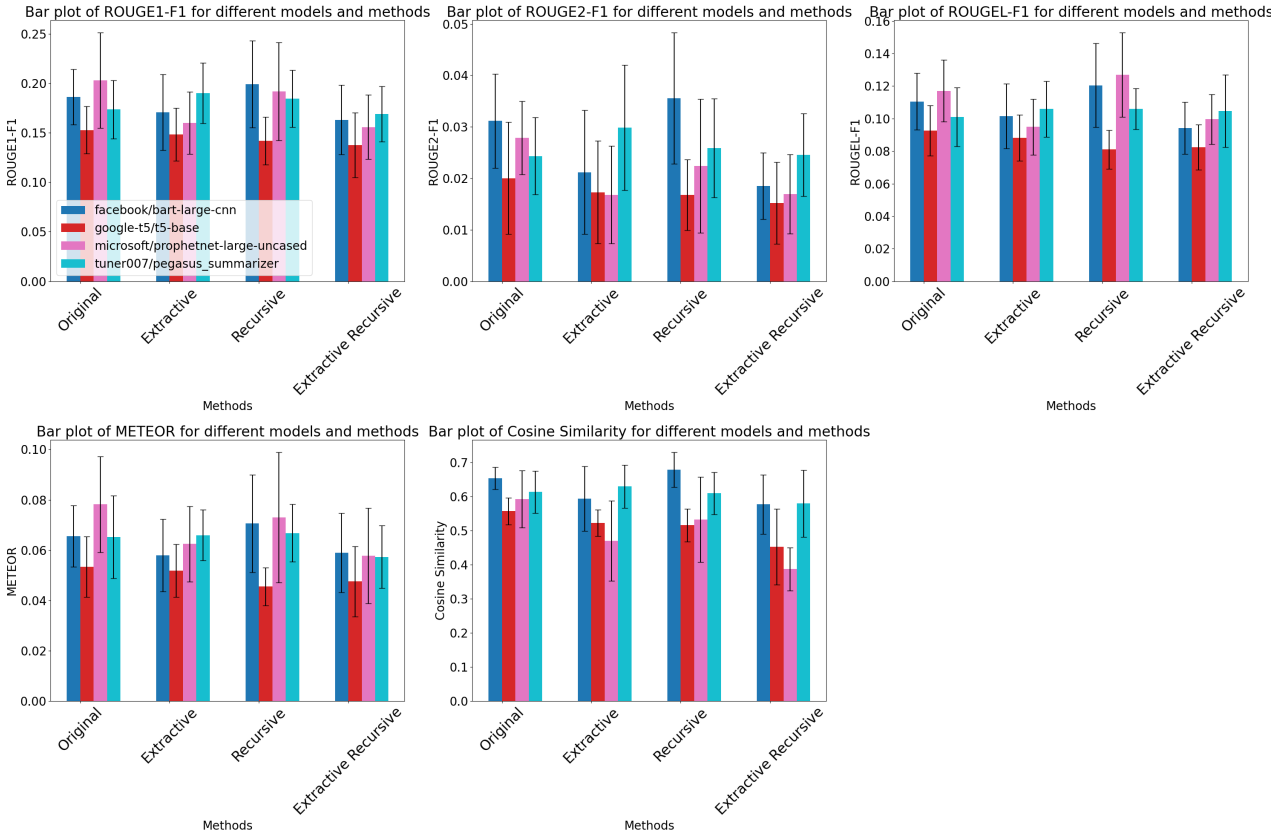

Figure 6: Evaluation metrics seen in a Barplot for different raw models (colors) and pipelines (x-axis) applying bootstrap.

coincidence of n-grams in the reference and generated summaries. Therefore, longer generations typically result in better scores. Moreover, in human evaluation, summaries of different lengths can be perceived as better than others.

4 RESULTS

We now proceed to present which pipeline is best suited for the models under consideration.

4.1 Automatic evaluation

Initially, the focus will be on the automatized evaluation metrics calculated for 10 samples from the BookSum dataset.

Different models with different pipelines have been used and their performance has been compared using the various automatized metrics. Figure 6 shows that in most cases, the recursive pipeline achieves the highest values in the different metrics. Due to resource limitations, although the original idea was to make the plots with 1000 samples, only 10 samples are available for each case and bootstrapping has been performed to calculate the error bars (mean ± standard deviation of 1000 random resamplings)

In the results, two trends can be observed. One trend involves the Pegasus and T5 models, and the other involves Bart and ProphetNet. Both trends exhibit similar variations across most metrics, with Bart outperforming ProphetNet and Pegasus surpassing T5.

For Pegasus and T5, the best results are observed for the original pipeline and the recursive. However, the performance can vary depending on the metric. The extractive and recursive pipelines show lower performance.

The other two LLMs, Bart and ProphetNet, follow a different trend that can vary depending on the metric. It appears that with more complex pipelines, the performance worsens. This suggests that this pipeline does not meet the expected performance and does not improve the results for the metrics used.

It should be noted that the metrics used in this study are not perfect and have some disadvantages. For instance, they depend on a "reference summary", and the length of the generated summary. Also, we noted a strong negative correlation of repeated n-grams with the evaluation metrics. This is because, in the dataset, the reference summaries were very long, longer than the summaries generated by the models. Thus, if a model generates larger summaries and/or has fewer repetitions, it performs better.

4.2 Human Evaluation

To compare the different models and methods, a human evaluation has been conducted with the help of 20 volunteers. An app has been implemented with Streamlit where people can choose a YouTube video and generate four summaries with it, one with each pipeline. Without knowing how each summary has been generated, users

Table 1: Average score of Human Manual evaluation of the different models and pipelines on 5-point Likert scale.

Model	Pipeline	No. of samples '	Mean	Std	Min	Max
bart-large-cnn	Original	8	2,50	1,13	1	4
bart-large-cnn	Extractive	8	2,25	0,93	1	3
bart-large-cnn	Recursive	8	2,88	1,13	1	4
bart-large-cnn	Extractive and Recursive	8	2,00	1,41	1	4
pegasus_summarizer	Original	8	3,13	2,12	1	4
pegasus_summarizer	Extractive	8	2,75	1,41	1	3
pegasus_summarizer	Recursive	8	3,38	2,83	1	5
pegasus_summarizer	Extractive and Recursive	8	2,13	0,71	2	3
prophetnet-large-uncased	Original	2	2,50	0,71	1	2
prophetnet-large-uncased	Extractive	2	2,00	1,41	1	3
prophetnet-large-uncased	Recursive	2	1,50	1,41	1	3
prophetnet-large-uncased	Extractive and Recursive	2	2,00	2,12	1	4
t5-base	Original	2	2,50	1,41	1	5
t5-base	Extractive	2	3,00	0,99	1	4
t5-base	Recursive	2	2,50	0,89	2	4
t5-base	Extractive and Recursive	2	2,00	1,25	1	5

score the four summaries individually with a score between 1, if the summary does not reflect the content of the video they viewed, to 5 where the summary represents the video very well.

The results of the scores of 20 volunteers, each of whom has evaluated 1 different video, are shown in Table 1. The volunteers performed fewer evaluations with the T5 and ProphetNet models because we decided to focus our resources on the most promising avenues. In this case, after starting testing, it was observed that the T5 and ProphetNet models were not performing as well as the other models. To ensure the quality of the summaries and the user experience of the volunteers, the decision was made to concentrate on the models that were providing the best results. This approach is not only efficient but also respects the time and effort of the volunteers by providing them with the highest quality summaries for evaluation.

To determine whether we can confidently reject the hypothesis that the results of the recursive pipeline are equal to those of the original model, we have performed a statistical test. This involved comparing the results of the original pipeline with those of the recursive pipeline using a two-sided t-test. With a confidence level of 90% (t-statistic: 1.843, p-value: 0.0787), we could reject the hypothesis that the recursive pipeline yields the same results as the original pipeline.

We have also compared the recursive pipeline's results with those of other pipelines for Bart and Pegasus. The results (t-statistic: 2.074, p-value: 0.0501) suggest that, with a 90% confidence level, we could reject the hypothesis that the other pipelines perform the same as the recursive pipeline. These results provide a statistical basis for asserting the recursive pipeline's best performance over the original pipeline and the other ones.

Nevertheless, several insights can be extracted from the results: The extractive and recursive approach performed the worst across all models. Pegasus is the best-evaluated model, with a score greater than 3 for both the original and recursive pipeline. This contrasts with the metric results, which indicated that Bart had better performance. The performance of Bart followed the distribution of the metrics, but, interestingly, Pegasus mirrored the trend of Bart and ProphetNet in the metrics, indicating that the original and recurrent pipelines achieved better performance. ProphetNet and T5 were not considered in this analysis due to the small number of volunteers who voted for these models.

4.3 Discussion of the results

From the previous tests, we can extract several conclusions: Firstly, it is important to note that raw models are capable of reading only the first 400 words. This limitation contrasts with the finding that these models perform well in both automatic and human evaluations. There are two reasons for this. With the metrics, the reference summary was significantly larger than the 400-word context window of the model. As a result, a brief summary of the beginning and a brief summary of the entire text would have a similar percentage of n-grams appearing, leading to similar metric scores. In human evaluations, another factor comes into play, which we can label as a 'introduction summary' bias. Many YouTube videos begin with an introduction that provides general information about the content of the video. Summaries of this type are well-received because they encapsulate what the video will discuss. However, if the video author includes an in-person advertisement during the initial part of a video it tends to be part of the summary with the original pipeline and receive low evaluation scores. The recursive approach, on the other hand, takes all the information into account, generating a summary that can discuss important points that appear later in the video and is not affected by the context window limitation.

Another interesting observation is the difference in the performance of Pegasus in metrics and human evaluations. As mentioned, Pegasus tends to create shorter summaries we instructed it to make them longer to ensure fairness across all pipelines. However, creating longer summaries introduced "n", as a filler character, which reduces performance in metrics because it counts as an n-gram that does not appear in the reference summary.

A significant issue is the limitation of the metrics. The primary question is, how can one evaluate whether a summary is good?

The solution is complex and relies on a reference summary for comparison. However, this transforms then into a question of whether the summaries are similar, which may be seen as a limitation since alternative good summaries may exist but the metrics would not detect it if they are very different from the reference summary.

Lastly, it is necessary to understand why ProphetNet does not perform as well as the other models. Despite being the newest model we used, the paper of ProphetNet [19] reveals that it has received less pretraining than the other models we are using. It does not lead to good results because of the frequent repetition of n-grams. The recursive pipeline gives the worst result, since for that pipeline we add "summarize:" in each model call. This seems to cause difficulty for ProphetNet in understanding this instruction.

5 CONCLUSIONS

This work has been motivated by the significant evolution of artificial intelligence in the generation and processing of natural language in recent years. It has served as an initial approach to the area by investigating simpler models than the state-of-the-art LLMs, aiming to solve the issue of short context windows for massive data in the task of text summarization in the context of resource limitations.

Learning to build pipelines on large language models and studying whether this improves their performance has been a key focus. Several language generation models have been studied, along with various datasets for creating and comparing summaries. The functionality of several metrics has been examined to evaluate the quality of the summaries. Additionally, an interface has been implemented for volunteers to use the tools and generate their own summaries from YouTube videos.

Due to resource constraints, the focus was on lightweight models, specifically those with less than a billion parameters. Google Colab's GPUs were used, which have time and computational limitations. Other resources like the High Performance Computing (HPC) of UPF were employed. This facilitated the testing of pipelines and the obtaining of results, allowing longer execution times.

Our study has a few limitations. One is the small sample sizes in experiments, which were caused by the only very limited access to HPC resources.

Another limitation appeared in the datasets; the difference in length between the reference summaries and the generated summaries led to a smaller precision but a better recall. By ensuring that the summaries were of similar length, this could be partially mitigated. Additionally, the requirement for a GPU limited the number of human evaluations.

In summary, pipelines that use recursion are more complete as they receive the entire text to be summarized as input, which is not the case with the original and the extractive pipelines. The recursive pipeline is not suitable for models like Prophetnet, but it works very well with models like T5, Pegasus, or Bart. Recursive pipelines generate summaries that accurately represent the original text and are perceived as better by the volunteers. Finally, statistical tests confirm that the recursive pipeline is better than the original pipeline, especially with T5, Pegasus, and Bart.

This work has clear applications in the real world. Since lightweight models are used to generate summaries, this could serve as a method to summarize large volumes of text in a resource-efficient program, making it suitable for devices with limited memory.

In future work, fine-tuning could be performed with datasets that are more specialized in long summaries. For example, the issue with the "n" filler character of Pegasus could be solved with re-training, allowing to uncover the full potential of these models. It would also be interesting to compare our results with the performance of larger language models such as GPT-4 or Gemma in these tasks for which we have enough resources or permissions.

REFERENCES

[1] Mehdi Allahyari, Seyedamin Pouriyeh, Mehdi Assefi, Saeid Safaei, Elizabeth D. Trippe, Juan B. Gutierrez, and Krys Kochut. 2017. Text Summarization Techniques: A Brief Survey. arXiv:1707.02268 [cs.CL]

[2] Max Bain, Jaesung Huh, Tengda Han, and Andrew Zisserman. 2023. WhisperX: Time-Accurate Speech Transcription of Long-Form Audio.

[3] Satanjeev Banerjee and Alon Lavie. 2005. METEOR: An Automatic Metric for MT Evaluation with Improved Correlation with Human Judgments. In *Proceedings of the ACL Workshop on Intrinsic and Extrinsic Evaluation Measures for Machine Translation and/or Summarization.* Association for Computational Linguistics, Ann Arbor, Michigan, 65–72. https://www.aclweb.org/anthology/W05-0909

[4] dmmiller612. 2022. bert-extractive-summarizer. https://pypi.org/project/bert-extractive-summarizer/.

[5] Real Academia Española. [n. d.]. resumir | Diccionario de la lengua española. https://dle.rae.es/resumir?m=form.

[6] Hugo Touvron et al. 2023. Llama 2: Open Foundation and Fine-Tuned Chat Models. arXiv:2307.09288 [cs.CL]

[7] Tom B. Brown et al. 2020. Language Models are Few-Shot Learners. arXiv:2005.14165 [cs.CL]

[8] Wojciech Kryściński, Nazneen Rajani, Divyansh Agarwal, Caiming Xiong, and Dragomir Radev. 2022. BookSum: A Collection of Datasets for Long-form Narrative Summarization. arXiv:2105.08209 [cs.CL]

[9] Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Veselin Stoyanov, and Luke Zettlemoyer. 2019. BART: Denoising Sequence-to-Sequence Pre-training for Natural Language Generation, Translation, and Comprehension. *CoRR* abs/1910.13461 (2019). arXiv:1910.13461 http://arxiv.org/abs/1910.13461

[10] Chin-Yew Lin. 2004. ROUGE: A Package for Automatic Evaluation of Summaries. In *Text Summarization Branches Out.* Association for Computational Linguistics, Barcelona, Spain, 74–81. https://aclanthology.org/W04-1013

[11] J. O'Connor and I. McDermott. 2001. *NLP.* Thorsons, J O'Connor, I McDermott - 2001 - academia.edu.

[12] OpenAI. 2023. GPT-3.5 Turbo fine-tuning and API updates. https://openai.com/index/gpt-3-5-turbo-fine-tuning-and-api-updates.

[13] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. 2011. Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research* 12 (2011), 2825–2830.

[14] Sundar Pichai and Demis Hassabis. 2024. Our next-generation model: Gemini 1.5. https://blog.google/technology/ai/google-gemini-next-generation-model-february-2024/#build-experiment. Accessed: 2024-05-20.

[15] 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 (2020), 1–67. http://jmlr.org/papers/v21/20-074.html

[16] Gemma Team. 2024. Gemma: Open Models Based on Gemini Research and Technology. arXiv:2403.08295 [cs.CL]

[17] Pompeu Fabra University. 2022. HPC High Performance Computing: Home. https://guiesbibtic.upf.edu/recerca/hpc.

[18] Divakar Yadav, Jalpa Desai, and Arun Kumar Yadav. 2022. Automatic Text Summarization Methods: A Comprehensive Review. arXiv:2204.01849 [cs.CL]

[19] Yu Yan, Weizhen Qi, Yeyun Gong, Dayiheng Liu, Nan Duan, Jiusheng Chen, Ruofei Zhang, and Ming Zhou. 2020. Prophetnet: Predicting future n-gram for sequence-to-sequence pre-training. *arXiv preprint arXiv:2001.04063* (2020), 2401–2410.

[20] Jingqing Zhang, Yao Zhao, Mohammad Saleh, and Peter J. Liu. 2019. PEGASUS: Pre-training with Extracted Gap-sentences for Abstractive Summarization. arXiv:1912.08777 [cs.CL]

All the project code can be found at https://github.com/Adri-Data/TFG_Summarization_Large_Texts.

Received July 31, 2024