BACHELOR´S THESIS / ENGINEERING 2024

Summarization of Large Text Volumes Using Large Language Models

Adrián David Navarro Betrián

Universitat Pompeu Fabra Barcelona

Summarization of Large Text Volumes Using Large Language Models

BACHELOR'S THESIS

Adrián David Navarro Betrián

Andreas Kaltenbrunner

Bachelor's degree in Mathematical Engineering in Data Science

Curs 2023 - 2024

Universitat Pompeu Fabra Barcelona

Escola d'Enginyeria

"A camino largo, paso corto"

-Refranero Español-

Acknowledgements

First of all, I want to thank all the volunteers who have dedicated their valuable time to participate in the tests to evaluate the summaries generated from videos.

Vull agrair a la meva família, que m'ha donat suport durant aquests quatre anys de carrera i al llarg de tota la meva vida. Han estat d'aquells de qui més he après, ensenyant-me a sempre buscar el costat positiu de les coses, a treballar amb constància i dedicació i a ser la persona que sóc avui.

A mi pareja, Alba, quiero agradecerle por su infinita paciencia, por escucharme hablar de cada una de las asignaturas de la carrera, por su apoyo constante y por compartir conmigo todas las celebraciones y los desafíos de estos años.

A mis amigos, que siempre han estado a mi lado, quiero agradecerles por su compañía, las historias que hemos compartido y los momentos inolvidables que hemos vivido juntos.

Al meu cosí, Albert Navarro, per oferir-me la seva perspectiva i els seus valuosos consells durant aquest TFG.

Abschließend möchte ich meinem Mentor, Andreas Kaltenbrunner, danken. Seine Erfahrung und Ratschläge waren entscheidend für die Gestaltung und den letzten Schliff dieses Projekts.

Abstract

This TFG 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.

Resumen

Este TFG se centra en el desarrollo y evaluación de varias pipelines centradas en Grandes Modelos de Lenguaje (LLMs). El objetivo es hacerlos capaces de resumir textos extensos, como libros o transcripciones de video, generando resúmenes completos, a pesar de las limitaciones de la ventana de contexto de estos modelos.

Varios modelos y pipelines se prueban utilizando diferentes métricas y una interfaz para permitir a voluntarios generar y evaluar resúmenes a partir de transcripciones de videos. Se comparan los modelos originales y pipelines más complejas para determinar su efectividad. Las pipelines están diseñadas basándose en tres enfoques diferentes: un método recursivo, un método combinado que incluye un resumidor extractivo, y un enfoque híbrido que incorpora tanto las pipelines recursivas como extractivas.

Los modelos base tienen la capacidad de resumir solo hasta 512 tokens (aproximadamente 400 palabras) mientras que las pipelines desarrolladas son capaces de resumir transcripciones de video de más de una hora, sin una limitación específica de tokens.

Resum

Aquest TFG implica el desenvolupament i avaluació de diverses pipelines centrades en Grans Models de Llenguatge (LLMs). L'objectiu és fer-los capaços de resumir textos extensos, com llibres o transcripcions de vídeo, produint resums complets, malgrat les limitacions de la finestra de context d'aquests models.

Es testegen diversos models i pipelines utilitzant diferents mètriques i una interfície per permet a voluntaris generar i avaluar resums a partir de transcripcions de vídeo. Es comparen models originals amb pipelines més complexes per determinar la seva eficàcia. Les pipelines estan dissenyades basant-se en tres idees diferents: un mètode recursiu, un mètode combinat que inclou un resumidor extractiu, i un enfocament híbrid que incorpora tant les pipelines recursives com extractives.

Els models base tenen la capacitat de resumir només fins a 512 tokens (aproximadament 400 paraules) mentre que les pipelines desenvolupades aquí són capaces de resumir transcripcions de vídeo de més d'una hora, sense una limitació específica de tokens.

Contents

List of Figures

List of Tables

Introduction

In the last 5 years, the rise of artificial intelligence has enabled significant advances in the area of Natural Language Processing (NLP). This progress has led to large improvements in tasks such as summary generation, text translation, answering text-based questions, and text rewriting, among others. An example is the popular model GPT-3. However, Large Language Models (LLMs) face a significant limitation: their context window.

The context window is a major obstacle for tasks such as summary generation and question answering, where all relevant information needs to be taken into account. Typically, LLMs had a context window ranging from 512 to 4096 tokens, for example, GPT-3 can process up to 4096 tokens[OpenAI, 2023]. Until recently, this limit was extended to 128K tokens with models like GPT-4 Turbo [Pichai and Hassabis, 2024], approximately 91K words[1]. This means that they can only "remember" and reason about a limited amount of information. For instance, if an LLM reads a book, it may not be able to remember the details from the beginning or middle of the book when it reaches the end.

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.

[1]During the time this report was written google released Gemini 1.5 pro that has a context window of 1.5 Million [Pichai and Hassabis, 2024].

- **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 evaluation metrics like Meteor, Rouge-1, -2, and -L, and cosine similarity. Additionally, some volunteers will be asked to evaluate the generated summaries.

Chapter 1

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 [O'Connor and McDermott, 2001]. This field combines computational linguistics with machine learning and deep learning. This TFG will focus on the area of NLP that seeks to summarize texts.

1.1 Text Summarization

Text summarization is defined as the process of "Reducing to brief and precise terms, or only considering and briefly repeating the essence of a matter or subject" [Real Academia Española, 2024].

In this project, the aim is to perform summarizations through 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 [Cajueiro et al., 2023].

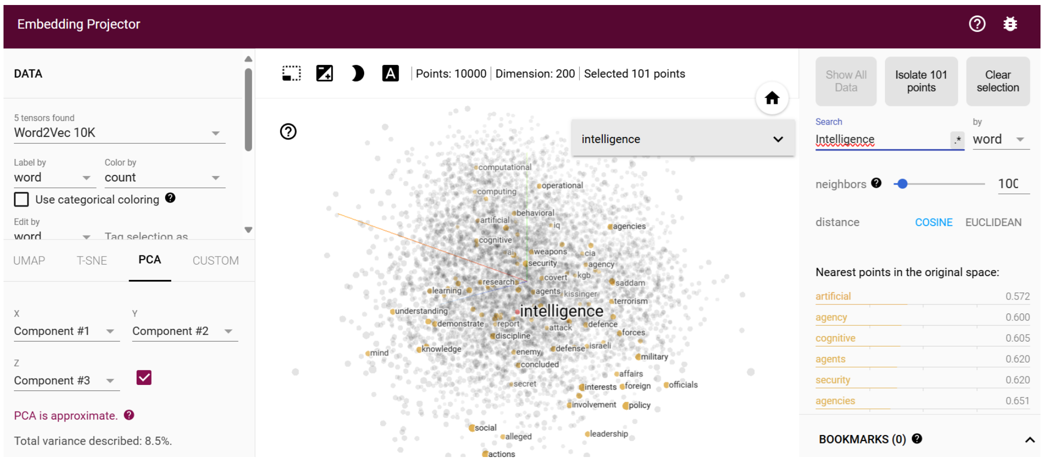
There are two main types of summaries: extractive and abstract. Extractive summaries are created by selecting the most relevant sentences from the original text. This approach is less complex as it does not require a full understanding of the text, simply extracting the most informative parts. Techniques such as TF-IDF, PageRank [Allahyari et al., 2017] or clustering algorithms [dmmiller612, 2022] 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 [Yadav et al., 2022]. In this work, we will focus on the realization of abstract summaries. Next, we will explain how LLMs have evolved to acquire the ability to understand the text and, thanks to this, to summarize it.

1.2 Large Language Models (LLMs)

In recent years there has been a boom in LLMs, which has represented a great advance in the field of NLP in general. To understand how these models work, it is useful to see the evolution of the state of the art of LLMs.

An initial challenge of using text in neural networks (NN) is that they cannot receive a word as input; they must be vectorized to be processed. Several approaches have been proposed to solve this problem, such as one-hot encoding or assigning a number to each word. However, these methods can be inefficient or not yield optimal results. Therefore, the idea of creating embeddings emerged, assigning words a position in a vector space, but with reduced dimensionality. To assign these positions, a neural network is used that compacts the dimensionality of the words and organizes them by their meaning [CSV, 2020]. Google, for example, created Word2Vec in 2013, a widely used embedding (Figure 1.1) [Mikolov et al., 2013]



5 tensors found
Word2Vec 10K

Label by: word
Color by: count

☐ Use categorical coloring

Edit by: word
Tag selection as

X: Component #1
Y: Component #2
Z: Component #3 ✓

PCA is approximate.

Total variance described: 8.5%.

Points: 10000 | Dimension: 200 | Selected 101 points

intelligence

Search: Intelligence
by: word

neighbors: 100

distance: COSINE EUCLIDEAN

Nearest points in the original space:

artificial	0.572
agency	0.600
cognitive	0.605
agents	0.620
security	0.620
agencies	0.651

Figure 1.1: A screenshot from https://projector.tensorflow.org/ displays the representation of 10,000 words in the Word2Vec embedding. Selecting the word ‘Intelligence’, the nearest words identified are ‘artificial’, ‘agency’, ‘cognitive’, ‘agents’, and ‘security’.

Once the words have been vectorized, the first approach that was adopted was the use of Recurrent Neural Networks (RNN), developed to process sequential data such as texts. However, RNNs have difficulties handling long-term dependencies, which means that if a sequence is very long, the impact of the first elements decreases compared to the last ones. This is known as the vanishing gradient problem, which causes the network to "forget" these first words [CSV, 2021].

To solve this problem, Long Short-Term Memory (LSTM) was introduced, a version of RNNs that is not affected by the vanishing gradient. This is achieved thanks to a structure called a memory cell, which regulates long-term memory through gates [Staudemeyer and Morris, 2019]. Other types of simpler cells were also developed, the Gated Recurrent Units (GRU), which help to avoid the problem of vanishing gradient. However, the process of the LSTM still had to be performed sequentially, which prevented taking full advantage of the NN [IBM, 2022].

Another way to avoid the vanishing gradient was to add an attention mechanism to the RNNs. This mechanism looks for the relationships between each word and the rest of the words in the sentence. To automate this search, two neural networks are trained to generate two vectors per word: the "key" vector which identifies interesting properties, and the "query" vector, which represents the characteristics that a word needs to relate to others. If two words are compatible, the "key" vector of one will be similar to the "query" vector of the other. The vector resulting from the dot product of the "query" vector of a word we process with the "key" vector of the rest of the words is called the attention vector. This vector allows us to see which words are considered relevant to give context to the word we process. In addition, a neural network is used to process the words and obtain an output, similar to an RNN, creating the "value" vector. Thanks to this, we can multiply the value of the output by the attention of each word, obtaining the output vector. This allows any word in the sentence to be contextualized with the rest [CSV, 2021].

In 2017, Google presented a new architecture called Transformers in their article "Attention is All You Need" [Vaswani et al., 2023]. This architecture, which stops using RNNs and uses attention mechanisms along with another type of NN, allows processing all the elements of a sequence at once, which facilitates its parallelization and takes advantage of the power of GPUs to perform calculations. This makes Transformers more efficient in training and allows the use of much more data.

Despite having the capacity to train models with a lot of data, these data are missing, supervised learning is expensive, slow, and costly. Therefore, two main ideas emerged: self-supervised training and Fine-Tuning.

Self-supervised training is a way to train a model without the need to use labeled data. Certain data are masked that the model will try to predict (Figura 1.2) [IBM, 2024]. For example, in BART, words are masked and the model tries to guess the hidden word based on the context [Lewis et al., 2019a]. In GPT, the model is trained with a sequence of words, where the final word is masked. This type of training is easy, economical, and quick to prepare and scale, so the trend has been to move toward larger and more powerful models.

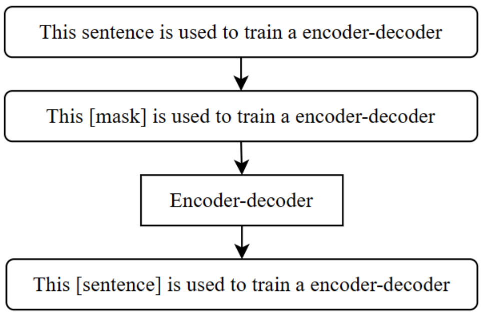

This sentence is used to train a encoder-decoder
This [mask] is used to train a encoder-decoder
Encoder-decoder
This [sentence] is used to train a encoder-decoder

Figure 1.2: This scheme illustrates the process of data masking used in training models. A token within a sentence is masked and the remaining sentence is provided as context. The model then attempts to predict the masked word.

As for fine-tuning, it has been discovered that using a pre-trained algorithm on a generic task and training it on a specific task significantly improves its performance, and requires fewer examples than using a model without pre-training [Zhang et al., 2020].

It has also been observed that larger models can develop other skills. For example, GPT-3, due to its large size and training to complete sentences, can generate translations or summaries simply by adding a text and the words “translation to English:” or “summary” [Raffel et al., 2023].

Similarly, Google’s Pegasus acquired a basic ability to count during its training, in addition to achieving its goal of understanding text to then summarize it [Zhang et al., 2020].

In this work, we will focus on the realization of abstract summaries. In Section 2.2 we explain in more detail some of the LLMs capable of performing this task. That models are T5[Raffel et al., 2020] and Pegasus[Zhang et al., 2019] from Google, Bart[Lewis et al., 2019b] from Facebook, and PropherNet [Yan et al., 2020] from Microsoft.

1.3 Research gaps

In summary, the field of Natural Language Processing (NLP) and more specifically summarization has experienced significant advances since 2019, especially with the development of Large Scale Language Models (LLMs).

However, a problem posed by these models is their context window, as they are unable to understand large amounts of text. Therefore, the objective of this TFG is to create a pipeline that allows generating summaries of texts that exceed the context window of conventional models.

Chapter 2

METHODOLOGY

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

2.1 Programming Environment

The code is written in Python 3.7. We use Colab for running tests and verifying results, which provides a T4 GPU for computation. The main libraries we use include transformers and a dataset from Hugging Face, along with torch, pandas, and nltk.

Due to Colab's limitations, which only allow for 12GB of RAM usage for 12 hours per day, we use the High-Performance Computing Cluster at Pompeu Fabra University for the automatic evaluation [Pompeu Fabra University, 2022]. Changes in the policy of the Computing Cluster led to restrictions on the GPUs during the process of obtaining results, causing a reduction in the number of samples used in the evaluation of metrics.

2.2 Model Selection

The initial step involves searching for various models and checkpoints from Hugging Face capable of summarizing. Because of our 12GB RAM limit, the search is centered on lighter models, excluding models with more than a billion parameters, such as GPT-3 [Brown et al., 2020], Gemma [Team et al., 2024] or LLama2 [Touvron et al., 2023]. These models, ranging from 200M to 600M parameters, are designed to understand text and generate abstract summaries. They are used as 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:

2.2.1 Bart [Lewis et al., 2019a]

Bart, which stands for Bidirectional and Auto-Regressive Transformers, is an LLM developed by Facebook AI in October 2019.

That LLM consists of 406M parameters and is based on transformers. Bart uses an encoder-decoder structure. The bidirectional encoder is based on BERT and is designed to understand the context of a given input. The autoregressive decoder, based on GPT, has the capacity to generate text.

Bart is pre-trained by trying to reconstruct texts to which noise has been added, which can be by masking, deleting tokens, permuting phrases, or rotating documents (Figure 2.1). That pre-training involves different datasets. Thanks to that pre-training, Bart can understand the text by the encoder part and can generate more coherent and contextual text. This makes BART effective for generating text and also for summarizing it.

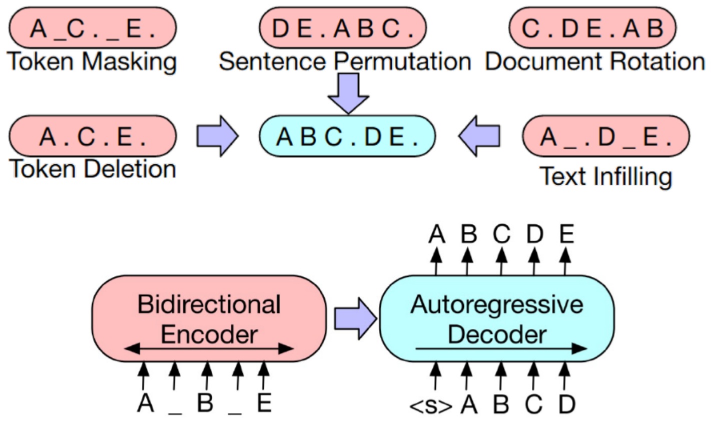


Figure 2.1: This diagram from [Lewis et al., 2019a] shows the pretraining process of the BART model. This process involves adding corruption into the text, which the model tries to rectify.

The version used has been [Lewis et al., 2019b] from Hugging Face, which has been fine-tuned with the CNN Daily Mail dataset, that checkpoint is used for summarization. The model re-trained is more effective in text summarization.

2.2.2 T5 [Raffel et al., 2023]

T5, also known as “Text-To-Text Transfer Transformer” is a LLM created by Google AI in October of 2019. This model has 223M parameters, making it the smallest model being used. It uses an encoder-decoder architecture based on transformers. The main difference with Bart is that it solves all the NLP problems with a text-to-text approach, which means that all the tasks are solved by giving the instruction in the input. For example saying “summarize:”, or “translate:“, since it is a multifunction model and is able to solve different tasks (Figure 2.2) .

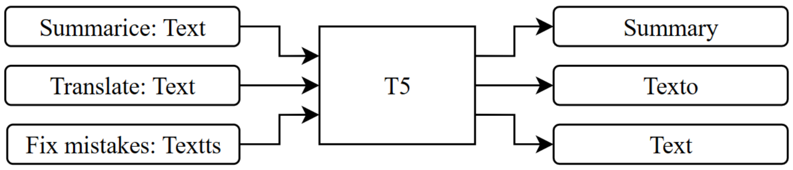


Figure 2.2: This diagram shows the multifaceted functionality of the T5 model. It highlights how tasks are directly given in the input.

Also, T5 is pre-trained on a high-quality massive corpus made by Google called c4 (Colossal Clean Crawled Corpus). This corpus allows the model to learn language patterns and structures of different languages. The checkpoint used is [Raffel et al., 2020] in Hugging Face.

2.2.3 Pegasus [Zhang et al., 2020]

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. The model has 568M parameters, being the largest model being used. The architecture is based on transformers with an encoder-decoder structure.

The key difference of Pegasus is the double pre-training objectives: doing GSG and MLM (Figure 2.3) . In GSG, keywords in sentences are masked and the encoder tries to predict them. In MLM, complete sentences are masked and the decoder will try to predict them (Figure 2.4). This model has been pre-trained with very massive datasets (c4 and HugeNews). The results indicate that a Pegasus pre-trained and trained with very few examples outperforms fully trained models without pre-training.

The version of Pegasus that has been used is [tuner007, 2021], which is on Hugging Face. This version is a fine-tuned model, using cnn_dailymail dataset [See et al., 2017].

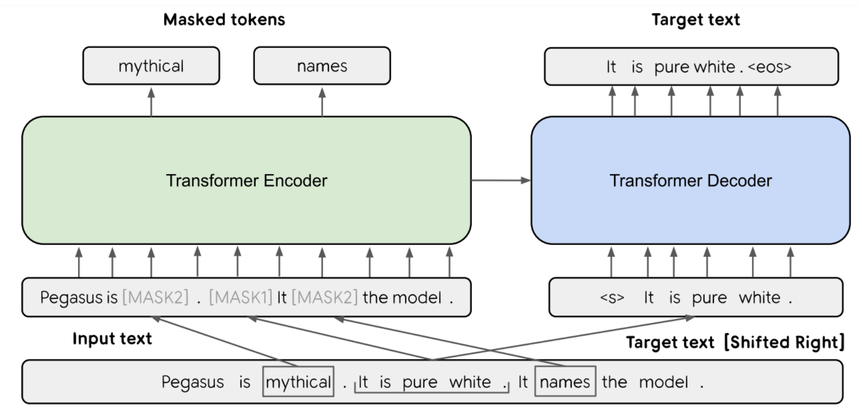

Masked tokens
mythical
names
Target text
It is pure white . <eos>
Transformer Encoder
Transformer Decoder
Pegasus is [MASK2] . [MASK1] It [MASK2] the model .
<s> It is pure white .
Input text
Target text [Shifted Right]
Pegasus is mythical . It is pure white . It names the model .

Figure 2.3: The schema, from [Zhang et al., 2020], presents the encoder-decoder structure of Pegasus. Showing how Gap Sentence Generation (GSG) and Masked Language Modeling (MLM) are employed as pre-training objectives, the main difference of that model.

2.2.4 ProphetNet [Qi et al., 2020]

ProphetNet was created by Microsoft, the University of Science and Technology of China and Sichuan University in January of 2020.

In terms of architecture, ProphetNet consists of 391M parameters, with an encoder and a decoder structure based on transformers, similar to other sequence-to-sequence models.

ProphetNet was pre-trained with the self-supervised objective of predicting future n-grams. With a mechanism of n-stream self-attention in its decoder. This mechanism allows the model to learn to predict multiple future tokens, so the n-tokens predicted will be based on the tokens that have already appeared (Figure 2.5). This prediction of n-grams serves as a guide to the model, as having to plan n-tokens in the future makes its word sequences more coherent and precise and avoids overfitting.

The specific version of ProphetNet that has been used is [Yan et al., 2020], which is available on Hugging Face.

2.2.5 Other auxiliary algorithms

whisperX [Bain et al., 2023]

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. **Whisper** is an Automatic Speech Recognition system and is a very accurate model

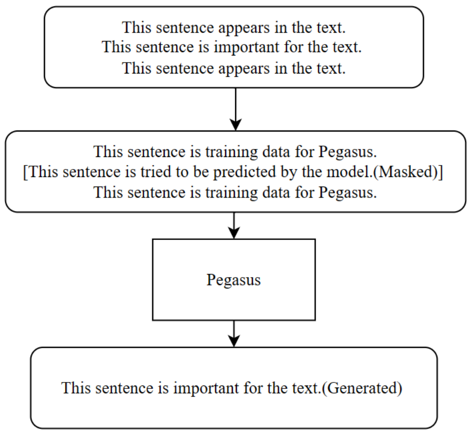

This sentence appears in the text.
This sentence is important for the text.
This sentence appears in the text.
This sentence is training data for Pegasus.
[This sentence is tried to be predicted by the model.(Masked)]
This sentence is training data for Pegasus.
Pegasus
This sentence is important for the text.(Generated)

Figure 2.4: This diagram illustrates the summarization process employed by the Pegasus model. It captures the context of multiple sentences and generates a significant sentence that explains the overall context, summarizing effectively all the text.

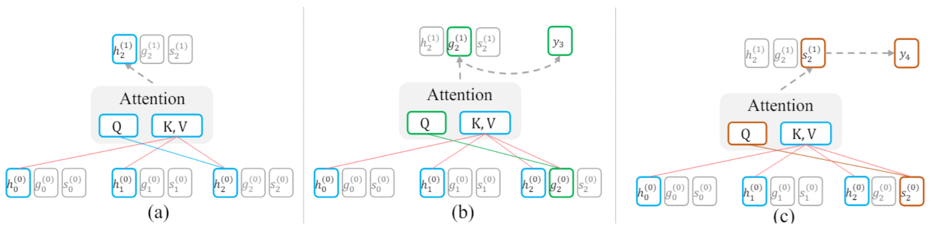


Figure 2.5: The diagram from [Qi et al., 2020] illustrates the mechanism of future 2-gram prediction in ProphetNet. Figure (a) explains the attention process, which is understanding the context. Following this, Figure (b) and Figure (c) show the generation of the first and second tokens respectively.

for transcribing voice to text. It is composed of an encoder-decoder architecture based on transformers. The part of the encoder tries to understand the spoken language. The training has used supervised data from various languages collected on the web. It has been trained with 680,000 hours of audio in various languages and it is very resistant to background noise and technical language. It is used to transcript YouTube videos from the Human Evaluation tests. The checkpoint used is openai/whisper-large-v2 from Hugging Face.

bert-extractive-summarizer [dmmiller612, 2022]

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. For each group, the most representative sentences of the original text are chosen, then they are ordered coherently. The model uses the PageRank algorithm to summarize. It is used for the extractive part of the pipelines.

2.3 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.

2.3.1 Original Pipeline

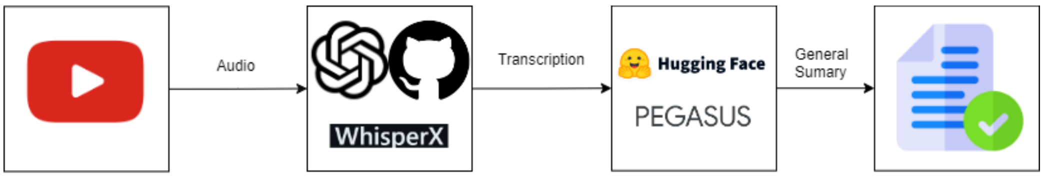


Figure 2.6: This schema shows how an original pipeline works.

This pipeline uses the summarization models, explained in Section 2.2, 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.

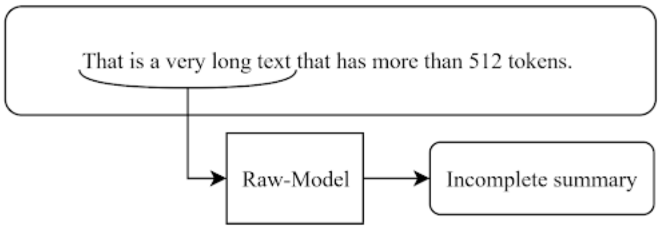


Figure 2.7: This diagram demonstrates the behavior of a raw model when processing large volumes of text. The model takes into account the entire context window but does not consider the remaining text, resulting in an incomplete summary.

2.3.2 Extracted Pipeline

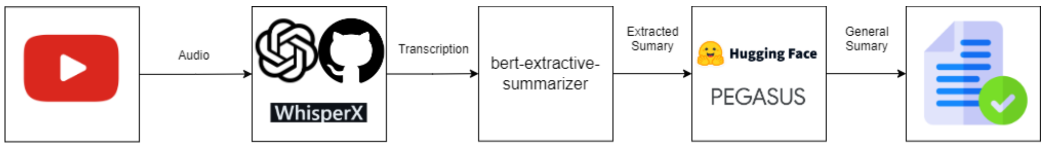


Figure 2.8: This schema shows how an extracted pipeline works.

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.

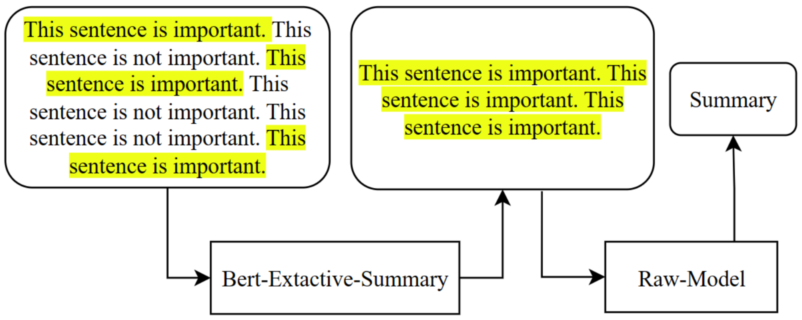

This sentence is important. This sentence is not important. This sentence is important. This sentence is not important. This sentence is not important. This sentence is important.
This sentence is important. This sentence is important. This sentence is important.
Summary
Bert-Extactive-Summary
Raw-Model

Figure 2.9: This diagram illustrates the workings of an extractive summary. It can be interpreted as highlighting the most significant sentences and taking them literally. This condensed text is then fed into the raw model to generate a more comprehensive summary.

2.3.3 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(Figure 2.11). 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.

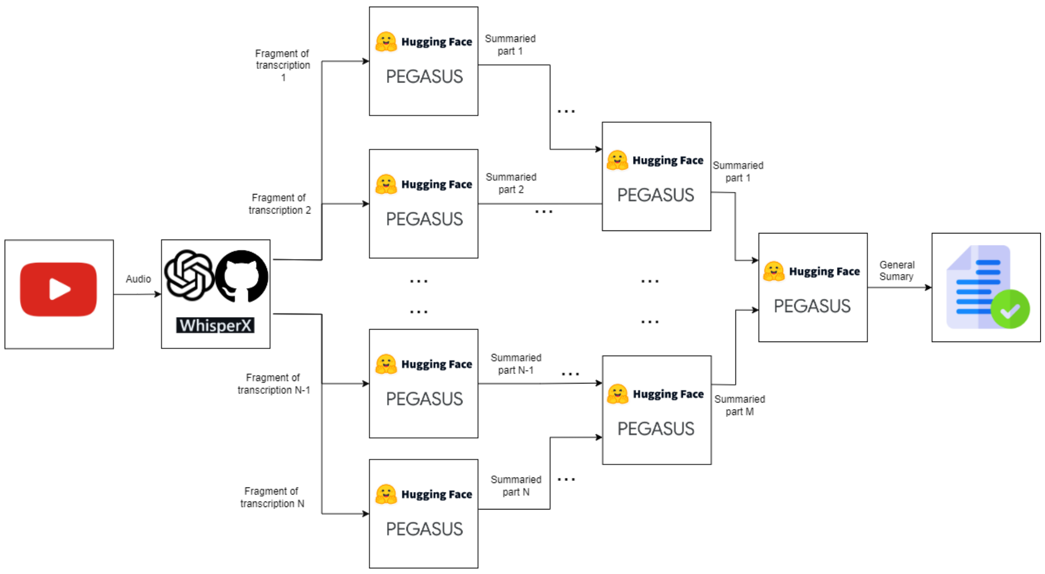


Figure 2.10: This schema shows how a recursive pipeline works.

Figure 2.11: The diagram illustrates the process of recursive summarization. In this process, the text is segmented into fragments which are then summarized by the model. This summarization procedure is iteratively performed until the resulting text is less than 512 tokens in length. Finally, the model performs an additional summarization to give coherence and consistency to the final summary. This approach gives a complete view of the text in the summary, regardless of the initial text's length.

2.3.4 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.

2.4 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 2.13) uses the BookSum dataset[Kryściński et al., 2022], which contains book chapters and their summaries. The chapters of the novels provide reference summaries which we compare with the outcomes of our pipelines applying metrics like ROUGE[Lin, 2004],

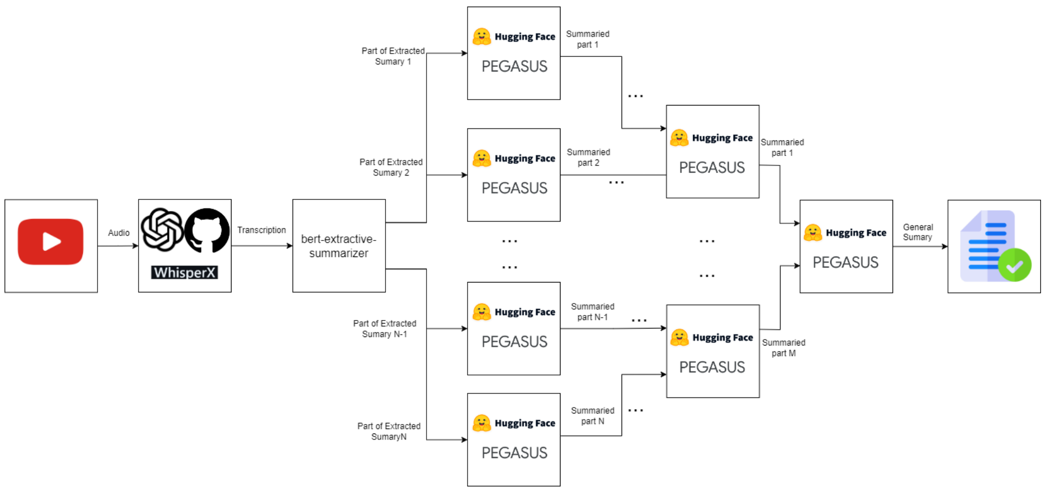


Figure 2.12: This schema shows how an extractive and recursive pipeline works.

METEOR[Banerjee and Lavie, 2005], and cosine similarity[Pedregosa et al., 2011].

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

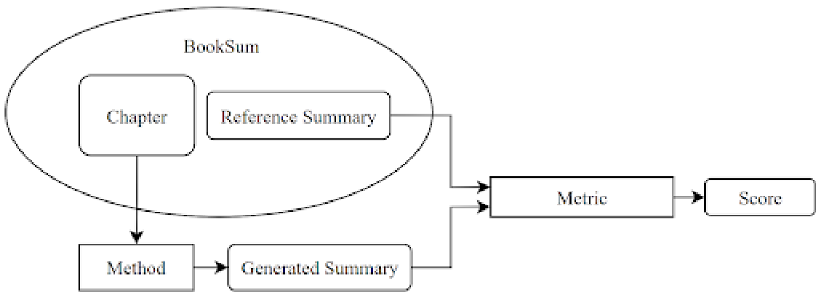


Figure 2.13: Illustration of the process of comparing pipelines with metrics. BookSum chapters from renowned books and their corresponding summaries are used as references and compared to the generated summaries using various metrics.

2.4.1 Dataset: BookSum [Kryściński et al., 2022]

The evaluation metrics for the pipelines need a reference summary for comparison with the generated summary. The reference summaries are taken from the

BookSum dataset from Hugging Face. That dataset, which consists of 379 MB, and contains 12,515 samples of chapters of well-known novels and their reference summaries. The length of those chapters makes it a robust way of testing the pipelines.

2.4.2 Metrics

Rouge-1, -2, and -L, Meteor and cosine similarity will be used to evaluate the generated summaries in comparison with reference summaries:

- **Rouge [Lin, 2004]**
  Rouge is a metric for evaluating the quality of a summary by comparing it with a "reference" summary. In Rouge-N, it counts the N-grams that match between both summaries and divides them by the number of N-grams in the reference summary. Rouge-1 is used for comparing the unigrams and Rouge-2 for the bigrams. Rouge-L is based on the longest common sequence of the two summaries, which doesn't need to be a consecutive match. Precision is calculated by dividing the matches of the two summaries by the generated summary, while recall is divided by the reference summary. The F1 score combines precision and recall.

$$Rouge - N = \frac{\sum_{S \in \text{References}} \sum_{gram_n \in S} \text{Count}_{\text{match}}(gram_n)}{\sum_{S \in \text{References}} \sum_{gram_n \in S} \text{Count}(gram_n)} \tag{2.1}$$

- **Meteor [Banerjee and Lavie, 2005]**
  METEOR(Metric for Evaluation of Translation with Explicit ORdering), is a metric used in the field of machine translation to evaluate the quality of translated text. However, it has been proven that it also works well for comparing generated summaries with a reference summary.
  It is similar to Rouge-1 F1, taking the score of precision and recall of the unigrams, but it gives more weight to recall. The main difference is that it also uses stemming, synonym matching, and paraphrasing when evaluating the summaries.

- **Cosine Similarity [Pedregosa et al., 2011]**
  The words of the summaries are tokenized and then vectorized. By comparing these vectors, the cosine similarity is obtained.

$$\text{cosine similarity} = \frac{\mathbf{A} \cdot \mathbf{B}}{||\mathbf{A}||_2 \times ||\mathbf{B}||_2} \tag{2.2}$$

2.5 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.

2.6 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 backend of Colab. A challenge we faced was that a new link was generated each time Colab was executed. Additionally, the computational limitations prompted 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 bigger sample sizes. As a result, we had to proceed with only 10 samples of the BookSum dataset for the evaluation with metrics.

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 is a length at which all four models can perform correctly. This adjustment was important as most metrics assign scores based on the 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.

In summary, these challenges and the implemented solutions provided valuable lessons for future projects. I would recommend exploring alternatives to Colab from the start and seeking resources that are accessible, such as those from their institutions, companies, or any other means of obtaining GPUs. Additionally, consider experimenting with various models, including those that may appear in the future, that are more capable or optimized. Adjust these models to produce summaries of a better quality. If faced with resource constraints, searching for smaller models is a practical approach. Lastly, it is beneficial to explore different evaluation methods, like other human evaluations, or comparisons between generated summaries, and consider the development of new pipelines capable of summarizing large volumes of text.

Chapter 3

RESULTS

In this section, we will present the results from three distinct tests to determine which pipeline is best suited for the models under consideration. Initially, the focus will be on the automatic evaluation, where we will compare the metric results of the models and pipelines using 10 samples from the BookSum dataset. These results will be visually represented with a bar plot. This will be followed by the outcomes of a human evaluation with volunteers. Lastly, examples of summaries, generated from the same video by different models and pipelines, will be presented to provide a comparative view of the generated summaries.

3.1 Automatic evaluation

Evaluation metrics in the Barplot 3.1 show the performance for different models (colors) and pipelines (x-axis) applying bootstrap. Their performance has been compared using the various automatized metrics. The figure 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 $\pm$ standard deviation of 1000 random resamplings).

The four pipelines tested were Original, Extractive, Recursive, and Extractive_Recursive, and they were evaluated using four models: Bart, T5, Prophetnet, and Pegasus. Focusing on the Rouge1-F1 metric, it can be observed that approximately one out of every five words in the generated summary also appears in the reference summary. The highest value achieved was 0.199 by the Bart model using the Recursive pipeline. The lowest value, 0.137, was obtained by the T5 model using the Extractive_Recursive pipeline.

For the Rouge2-F1 metric, the values are less than 0.04 due to the larger number of 2-grams. Looking at the METEOR metric, the best performance was by the Prophetnet model in the Recursive pipeline with a score of 0.07, while the worst was by the T5 model in the Recursive pipeline. The METEOR metric is similar to Rouge1-F1, but it places more weight on recall, resulting in smaller values since the recall of Rouge is much lower than precision. This is due to the fact that reference summaries are longer than generated ones.

It is also important that the cosine similarity between the two summaries is high, meaning that they are very similar in terms of content. The Recursive pipeline from Bart shows the highest cosine similarity with 0.678, indicating a strong similarity between the generated and reference summaries. The Extractive_Recursive pipeline with the Prophetnet model has the lowest cosine similarity with 0.386. These metrics provide valuable insights into the performance and effectiveness of the different models and pipelines.

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 recursive pipeline shows lower performance, meaning that it differs more from the reference summaries.

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 being used.

It should be noted that the metrics used in this study 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 results of the metrics. This is because, in the dataset, the reference summaries were very long, longer than the summaries generated by the models. This implies that if a model generates larger summaries or has fewer repetitions, it performs better.

3.2 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 score the four summaries individually with a score between 1, if the

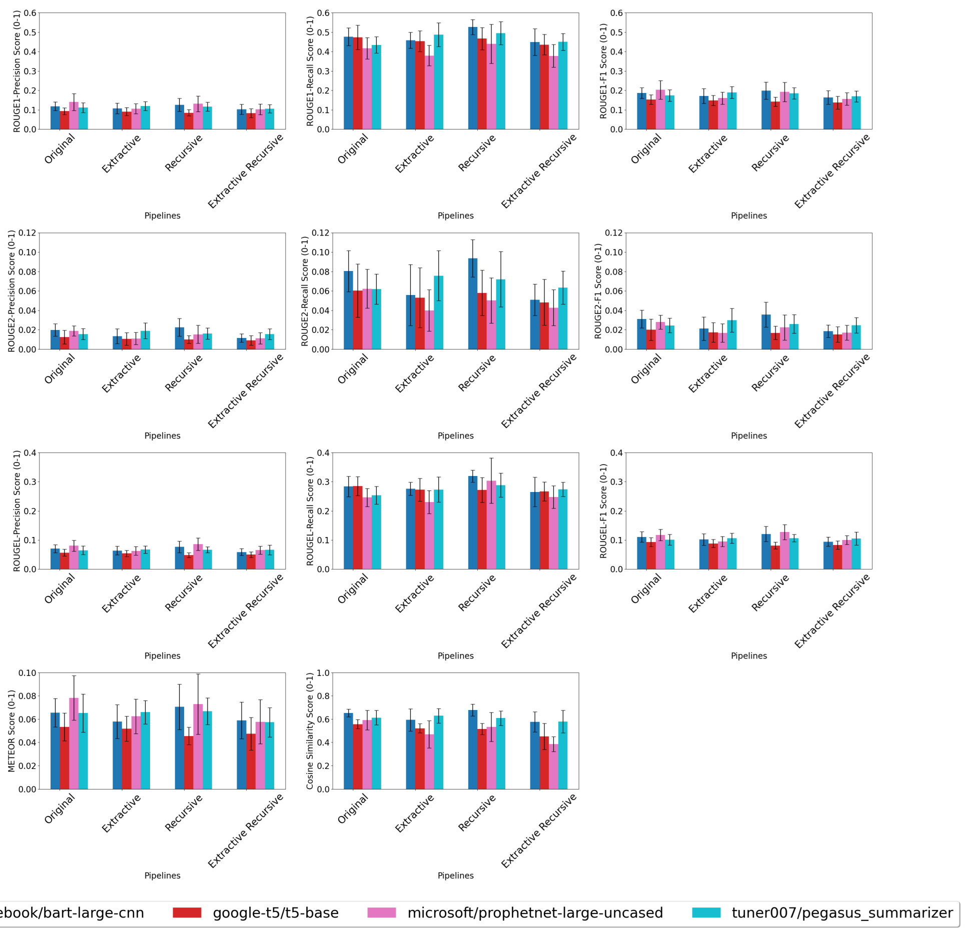


Figure 3.1: Evaluation metrics seen in a Barplot for different raw models (colors) and pipelines (x-axis) applying bootstrap.

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, where each one has evaluated 1 different video are shown in Table 3.1:

The volunteers performed fewer evaluations with the T5 and ProphetNet models because we decided to focus 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.

Table 3.1: Average score of Human Manual evaluation of the different models and pipelines on a 5-point Likert scale.

Modelo	Método	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

To determine whether we can confidently reject the hypothesis that the recursive pipeline's results are equal to those of the original model, we have conducted a statistical test. This involved comparing the results of the original pipeline with those of the recursive pipeline using a two-sided t-test.

These results provide a statistical basis for asserting the recursive pipeline's best performance over the original pipeline and the other ones. 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: 1.903, p-value: 0.0667) suggest that, with a 90% confidence level, we could reject the hypothesis that the other pipelines perform the same as the recursive pipeline.

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.

Table 3.2: Summaries generated by different pipelines with the model T5

google-t5/t5-base	
Pipeline	**Generated Summary**
Original	the arrival of accelerated computing and AI is timely as industries tackle powerful dynamics. without Moore's law, data center power is skyrocketing, and companies struggle to achieve net zero. we will share new advances in NVIDIA's full stack, data center scale, accelerated computing platform. we'll learn from leaders like Demas Hassabis of Deepmine, Valerie Taylor of Argonne Labs, and Paul Debevik of Netflix.
Extractive	invidia invented accelerated computing to solve problems that normal computers can't. it requires full stack invention from chips, systems, networking, acceleration libraries. each optimized stack accelerates an application domain, from graphics, imaging, particle or fluid dynamics, quantum physics. NVIDIA has established the accelerated computing virtual cycle, or ccv, with libraries in almost every domain of science and industry, says dr. nicolaus mills.
Recursive	today, we're announcing Parabricks 4.1, and we'll run on NVIDIA Accelerated Genomics Instruments. computational lithography is the largest computation workload in chip design and manufacturing. invidia announces cloud service for custom large language models and generative AI. the world's largest industries make physical things, but they want to build them digitally. by 2030, auto manufacturers will build 300 factories to make 200 electric vehicles.
Extractive and Recursive	invidia has invented accelerated computing to solve problems that normal computers can't. by 2030, auto manufacturers will build 300 factories to make 200 electric vehicles. the DGX H100 AI supercomputer is in production and available soon from an expanding network of OEM and cloud partners. it requires full stack invention from chips, systems, networking, acceleration libraries, and a full stack of invidia's chipsets, systems, networks and libraries.

3.3 Use Case

A use case is provided which involves a video of a keynote from Nvidia's CEO, Jensen Huang, lasting 1 hour and 20 minutes. Summaries were generated from the transcription of this video using four different models and four different pipelines. It should be noted that the raw models have a context window of 500 tokens. This implies that they can approximately retain the content from the first 3 minutes of the video [NVIDIA, 2023][1].

From the use case of T5 (shown in Table 3.2) we can analyse the 4 pipelines. In the original pipeline, a summary is generated that focuses on the initial part of the video, where the contestants are announced. However, this summary does not provide relevant information about the main themes of the video.

[1]https://www.youtube.com/watch?v=DiGB5uAYKAg

Table 3.3: Summaries generated by different pipelines with the model Pegasus

tuner007/pegasus_summarizer	
Pipeline	**Generated Summary**
Original	NVIDIA CEO Alex Kershevsky, while addressing the GPU Technology Conference (GTC) in San Francisco, said, "The world's most important companies are here." Kershevsky added, "We have 650 amazing talks from the minds in academia and the world's largest industries. There are more than 70 talks on generative AI alone. Other great talks, like pre-trained, models for robotics. I'm AI, I'm a healer, and I'm translator."
Extractive	NVIDIA CEO Jen-Hsun Huang said, "I invented accelerated computing to solve problems that normal computers can't." He added, "It requires full stack invention from chips, systems, networking, acceleration libraries to optimised applications. Each stack accelerates an application domain, from graphics, imaging, particle or fluid dynamics, quantum physics, to data processing and machine learning." Huang further said, "Acceleration is the best way to reclaim power and achieve sustainability and net zero." (nnn)
Recursive	NVIDIA CEO Jen-Hsun Huang said, "The arrival of accelerated computing and AI is timely, as industries tackle powerful dynamics, sustainability, generative AI, and digitalization." He added, "Without Moore's law, as computing surges, data centre power is skyrocketing, and companies struggle to achieve net zero." Intel has announced a new generation of workstations powered by I, Ada, RTXs and Intel's newest CPUs. The workstations will be available from Dell, HP, and Lenovo.
Extractive and Recursive	NVIDIA CEO Jen-Hsun Huang has said that he invented accelerated computing to solve problems that normal computers can't. "Each stack accelerates an application domain, from graphics, imaging, particle or fluid dynamics, quantum physics, to data processing and machine learning," Huang added. He further said NVIDIA has established accelerated computing virtual cycle. NVIDIA has announced a new generation of workstations powered by NVIDIA RTXs and Intel's newest CPUs.nnnnnnnnnnnnnnnnnnnnnnnnn

The extractive approach produces a high-quality summary, but it omits several important themes that are present in the video.
The recursive pipeline begins with a significant announcement from the keynote, the Parabricks 4.1. While this approach provides a good summary, it includes some details, such as the mention of electric vehicle factories, which are not important to the video's content. Despite this, it is the best among the four pipelines. Lastly, the extractive and recursive pipeline produces a good summary. However, towards the end, there is a repetition where terms like 'full stack', 'chipsets', 'systems networks', and 'libraries' are mentioned twice.

From the use case of Pegasus (shown in Table 3.3) we can analyse its be-

haviour across the 4 pipelines. The model in question has a limitation: it has been trained to generate shorter summaries than the other models. To ensure fairness in metric evaluation, it is compelled to produce longer summaries. This often results in the end of most generations being filled with the character “n”. The model’s results tend to cite the most important sentences. Let’s examine the pipelines one by one:
The original pipeline summary incorrectly identifies the Nvidia CEO and provides a very superficial overview of the video’s introduction.
The extractive pipeline summary, while good, remains superficial and omits important aspects of the keynote. The recursive pipeline adds a mention of the “new generation of workstations”.
The extractive and recursive pipeline is quite similar to the recursive but is slightly more difficult to comprehend. The recursive pipeline provides the clearest and most comprehensive summary.

We can study the behavior of Bart in the 4 pipelines based on the use case (shown in Table 3.4). The original summary looks like an introductory overview of the video. The extractive summary performs satisfactorily, mentioning the applications of NVIDIA in the field of genomics. The recursive summary provides a comprehensive synopsis, mentioning data and valuable information about the keynote. The extractive and recursive pipeline produces effective summaries, with a slight loss in coherence. For these reasons, the recursive summary is the best.

The use case (shown in Table 3.5) allows us to see how Prophetnet behaves across the 4 pipelines.
ProphetNet generates numerous repetitions, resulting in sentences that lack coherence and are difficult to understand. The original summary summarizes the introduction, but frequent repetitions appear.
The extractive pipeline appears to be the most effective for this model, although it fails to convey any relevant information from the video, it avoids the incoherence observed in other pipelines.
The recursive pipeline summary does not function well with this model, resulting in a non-understandable text.
The extractive and recursive pipeline produces less nonsensical content than the pipeline alone, but it still contains many repetitions and fails to produce a summary that accurately represents the video.
The best summary, although not really good, is better than the others and is the extractive pipeline.

Table 3.4: Summaries generated by different pipelines with the model Bart

facebook/bart-large-cnn	
Pipeline	**Generated Summary**
Original	The arrival of accelerated computing and AI is timely, as industries tackle powerful dynamics, sustainability, generative AI, and digitalization. Without Moore's law, as computing surges, data center power is skyrocketing, and companies struggle to achieve net zero. At GTC, 2023, we'll learn from leaders like Demas Hassabis of Deepmine, Valerie Taylor of Argonne Labs, Scott Belski of Adobe, Paul Debevik of Netflix, Thomas Sheltas of E.T.H. Zurich.
Extractive	For nearly four decades, Moore's law has been the governing dynamics of the computer industry. Each optimized stack accelerates an application domain, from graphics, imaging, particle or fluid dynamics, quantum physics, to data processing and machine learning. Acceleration libraries are at the core of accelerated computing with libraries in almost every domain of science and industry. NVIDIA accelerated computing helped achieve a genomics milestone. AT&T is tapping into Invidia Accelerated Computing and AI for sustainability, cost savings, and new services.
Recursive	Data processing is the leading workload of the world's $500 billion cloud computing spent. The entire 144 core Gray Superchip module with 1 terabyte of memory is only 5 by 8 inches. Snapchat will use L4 for AV1 video processing, generative AI and augmented reality. Proteus is Amazon's first fully autonomous warehouse robot. Omniverse is the digital to physical operating system to realize industrialization. New workstations are ideal for doing ray tracing, physics simulation, neural graphics, and generative AI.
Extractive and Recursive	NVIDIA Coopt has broken the world record and discovered the best solution for Lee and Limbs' challenge. Grace Hopper is our new superchip that connects Grace CPU and Hopper GPU over a high-speed 900 gigabytes per second coherent chip-to-chip interface. BMW, Lotus and Lucent Motors use Omniverse to build digital stores from actual design data that faithfully represent their cars. The DGX H100 AI supercomputer is in production and available soon from an expanding network of OEM and cloud partners worldwide.

Table 3.5: Summaries generated by different pipelines with the model Prophetnet

microsoft/prophetnet-large-uncased	
Pipeline	**Generated Summary**
Original	for nearly four decades, moore's law has been the governing dynamics of the computer industry. but now, moore has changed. and so has ai. and ai has gone to light speed. and the energy source is ai. and the speed of light. and the power source is energy. and that energy source is net zero. net zero would be impossible, as companies struggle to achieve net zero. without net zero, companies struggle to reach net zero, where without net zero there would be no net zero.
Extractive	moore's law has gone to light speed. created a sense of urgency for companies to innovate. changed the world. changed the game. changed everything. created a new paradigm. created a paradigm shift. changed the rules of the game. created an urgency for companies. created the urgency for change. created the sense of urgency by invidia research on robotics. and av safety. and other important works by the invidia team. important works by nvidia research, like
Recursive	summarize : and now it is : and so it is with ai. and : and : and so is the future of ai. and so is : and the future is : and this is the future. and this is : and that is the future : and that energy source is the law of the land. and so we're going to show you : and so we are : and so : and so, here we are. why not? so we'll show you what happens when we do that : and then : and then we'll see what happens. so... so. so. here we are : here we go. why not : so. and so. so here we go : so here we
Extractive and Recursive	a. q. a. a. c. a : changed the rules of the game for everyone. changed the rules for everyone. created an era for computing innovation and decoherence. created new rules for the game. created a paradigm shift. created a new era of computing innovation and created a new paradigm for computing innovation. we are available in every instrument, including instruments like terra and terra are available in all instruments, including instruments such as terra and terra were available in instruments like terra.

3.4 Discussion of the results

From the previous tests, we can extract several insights:

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 automatics 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 an '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. Despite this, if the author realizes an ad in the first moments it tends to be the summary of the original pipeline, that type of summaries are scored low.

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.

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 [Yan et al., 2020] reveals that it has received less pretraining than the other models we are using. It does not lead to good results because of 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.

Certain limitations were observed, one is the small sample sizes in the results. Where the metrics test used 10 samples that were bootstrapped and the human evaluations were based on the summaries of the videos from 20 volunteers. Additionally, the requirement for a GPU limited the use of human evaluation. If CPU compatibility were possible, it could be uploaded to the Streamlit web, making it accessible to anyone.

A significant issue is the limitation of the metrics. The primary question is, how can one evaluate whether a summary is good or not? The solution is complex and relies on a reference summary for comparison. However, this transforms the question into whether the summaries are similar or not. That is a limitation since maybe there are good generated summaries but the metrics do not show it because they are very different from the reference.

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 in the metrics. By ensuring that the summaries were of similar length, this limitation could be avoided.

The principal advantage of using multimodel pipelines is mitigating limitations, and improving the results. For example, individual models are restricted to summarizing 3 minutes of audio, and multimodel pipelines enable the summarization of videos of any length, removing any context windows.

This flexibility allows these ideas to be applied to various models and tasks. In summary, multimodel pipelines offer a versatile approach to improving task performance. They provide the flexibility to use one or multiple models and the capability to handle different tasks without being limited by specific constraints.

Chapter 4

CONCLUSIONS

This work has been motivated by the significant evolution of artificial intelligence in generating and processing 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.

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 unsuitable for models like Prophetnet but 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 real-world applications. Since lightweight models are used to generate summaries, this could serve as a method for summarizing 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. However, we currently lack the necessary resources or permissions to do so.

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

Bibliography

[Allahyari et al., 2017] Allahyari, M., Pouriyeh, S., Assefi, M., Safaei, S., Trippe, E. D., Gutierrez, J. B., and Kochut, K. (2017). Text summarization techniques: A brief survey.

[Bain et al., 2023] Bain, M., Huh, J., Han, T., and Zisserman, A. (2023). Whisperx: Time-accurate speech transcription of long-form audio.

[Banerjee and Lavie, 2005] Banerjee, S. and Lavie, A. (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*, pages 65–72, Ann Arbor, Michigan. Association for Computational Linguistics.

[Brown et al., 2020] Brown, T. B., Mann, B., Ryder, N., Subbiah, M., Kaplan, J., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., Agarwal, S., Herbert-Voss, A., Krueger, G., Henighan, T., Child, R., Ramesh, A., Ziegler, D. M., Wu, J., Winter, C., Hesse, C., Chen, M., Sigler, E., Litwin, M., Gray, S., Chess, B., Clark, J., Berner, C., McCandlish, S., Radford, A., Sutskever, I., and Amodei, D. (2020). Language models are few-shot learners.

[Cajueiro et al., 2023] Cajueiro, D. O., Nery, A. G., Tavares, I., Melo, M. K. D., dos Reis, S. A., Weigang, L., and Celestino, V. R. R. (2023). A comprehensive review of automatic text summarization techniques: method, data, evaluation and coding.

[CSV, 2020] CSV, D. (2020). Intro al natural language processing (nlp) #2 - ¿qué es un embedding? `https://www.youtube.com/watch?v=RkYuH\_K7Fx4`.

[CSV, 2021] CSV, D. (2021). ¿qué es un transformer? la red neuronal que lo cambió todo! `https://www.youtube.com/watch?v=aL-EmKuB078`.

[dmmiller612, 2022] dmmiller612 (2022). bert-extractive-summarizer. `https://pypi.org/project/bert-extractive-summarizer/`.

[IBM, 2022] IBM (2022). ¿qué son las redes neuronales recurrentes? `https://www.ibm.com/es-es/topics/recurrent-neural-networks`.

[IBM, 2024] IBM (2024). What is self-supervised learning? `https://www.ibm.com/topics/self-supervised-learning`.

[Kryściński et al., 2022] Kryściński, W., Rajani, N., Agarwal, D., Xiong, C., and Radev, D. (2022). Booksum: A collection of datasets for long-form narrative summarization.

[Lewis et al., 2019a] Lewis, M., Liu, Y., Goyal, N., Ghazvininejad, M., Mohamed, A., Levy, O., Stoyanov, V., and Zettlemoyer, L. (2019a). Bart: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension.

[Lewis et al., 2019b] Lewis, M., Liu, Y., Goyal, N., Ghazvininejad, M., Mohamed, A., Levy, O., Stoyanov, V., and Zettlemoyer, L. (2019b). BART: denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. *CoRR*, abs/1910.13461.

[Lin, 2004] Lin, C.-Y. (2004). ROUGE: A package for automatic evaluation of summaries. In *Text Summarization Branches Out*, pages 74–81, Barcelona, Spain. Association for Computational Linguistics.

[Mikolov et al., 2013] Mikolov, T., Chen, K., Corrado, G., and Dean, J. (2013). Efficient estimation of word representations in vector space.

[NVIDIA, 2023] NVIDIA (2023). Gtc 2023 keynote with nvidia ceo jensen huang. `https://www.youtube.com/watch?v=DiGB5uAYKAg`.

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

[OpenAI, 2023] OpenAI (2023). Gpt-3.5 turbo fine-tuning and api updates. `https://openai.com/index/gpt-3-5-turbo-fine-tuning-and-api-updates`.

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

[Pichai and Hassabis, 2024] Pichai, S. and Hassabis, D. (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.

[Pompeu Fabra University, 2022] Pompeu Fabra University (2022). Hpc high performance computing: Home. `https://guiesbibtic.upf.edu/recerca/hpc`.

[Qi et al., 2020] Qi, W., Yan, Y., Gong, Y., Liu, D., Duan, N., Chen, J., Zhang, R., and Zhou, M. (2020). Prophetnet: Predicting future n-gram for sequence-to-sequence pre-training.

[Raffel et al., 2020] Raffel, C., Shazeer, N., Roberts, A., Lee, K., Narang, S., Matena, M., Zhou, Y., Li, W., and Liu, P. J. (2020). Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of Machine Learning Research*, 21(140):1–67.

[Raffel et al., 2023] Raffel, C., Shazeer, N., Roberts, A., Lee, K., Narang, S., Matena, M., Zhou, Y., Li, W., and Liu, P. J. (2023). Exploring the limits of transfer learning with a unified text-to-text transformer.

[Real Academia Española, 2024] Real Academia Española (2024). resumir — diccionario de la lengua española. `https://dle.rae.es/resumir?m=form`.

[See et al., 2017] See, A., Liu, P. J., and Manning, C. D. (2017). Get to the point: Summarization with pointer-generator networks. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1073–1083, Vancouver, Canada. Association for Computational Linguistics.

[Staudemeyer and Morris, 2019] Staudemeyer, R. C. and Morris, E. R. (2019). Understanding lstm – a tutorial into long short-term memory recurrent neural networks.

[Team et al., 2024] Team, G., Mesnard, T., Hardin, C., Dadashi, R., Bhupatiraju, S., Pathak, S., Sifre, L., Rivière, M., Kale, M. S., Love, J., Tafti, P., Hussenot, L., Sessa, P. G., Chowdhery, A., Roberts, A., Barua, A., Botev, A., Castro-Ros, A., Slone, A., Héliou, A., Tacchetti, A., Bulanova, A., Paterson, A., Tsai, B., Shahriari, B., Lan, C. L., Choquette-Choo, C. A., Crepy, C., Cer, D., Ippolito, D., Reid, D., Buchatskaya, E., Ni, E., Noland, E., Yan, G., Tucker, G., Muraru, G.-C., Rozhdestvenskiy, G., Michalewski, H., Tenney, I., Grishchenko, I., Austin, J., Keeling, J., Labanowski, J., Lespiau, J.-B., Stanway, J., Brennan, J., Chen, J., Ferret, J., Chiu, J., Mao-Jones, J., Lee, K., Yu, K., Millican, K., Sjoesund, L. L., Lee, L., Dixon, L., Reid, M., Mikuła, M., Wirth, M., Sharman, M., Chinaev, N., Thain, N., Bachem, O., Chang, O., Wahltinez, O., Bailey, P., Michel, P., Yotov, P., Chaabouni, R., Comanescu, R., Jana, R., Anil, R., McIlroy, R., Liu, R., Mullins, R., Smith, S. L., Borgeaud, S., Girgin, S., Douglas, S., Pandya, S., Shakeri, S., De, S., Klimenko, T., Hennigan, T., Feinberg, V., Stokowiec, W., hui Chen, Y., Ahmed, Z., Gong, Z., Warkentin, T., Peran, L., Giang, M., Farabet, C., Vinyals, O., Dean, J., Kavukcuoglu, K., Hassabis, D., Ghahramani, Z., Eck, D., Barral, J., Pereira, F., Collins, E., Joulin, A., Fiedel, N., Senter, E., Andreev, A., and Kenealy, K. (2024). Gemma: Open models based on gemini research and technology.

[Touvron et al., 2023] Touvron, H., Martin, L., Stone, K., Albert, P., Almahairi, A., Babaei, Y., Bashlykov, N., Batra, S., Bhargava, P., Bhosale, S., Bikel, D., Blecher, L., Ferrer, C. C., Chen, M., Cucurull, G., Esiobu, D., Fernandes, J., Fu, J., Fu, W., Fuller, B., Gao, C., Goswami, V., Goyal, N., Hartshorn, A., Hosseini, S., Hou, R., Inan, H., Kardas, M., Kerkez, V., Khabsa, M., Kloumann, I., Korenev, A., Koura, P. S., Lachaux, M.-A., Lavril, T., Lee, J., Liskovich, D., Lu, Y., Mao, Y., Martinet, X., Mihaylov, T., Mishra, P., Molybog, I., Nie, Y., Poulton, A., Reizenstein, J., Rungta, R., Saladi, K., Schelten, A., Silva, R., Smith, E. M., Subramanian, R., Tan, X. E., Tang, B., Taylor, R., Williams, A., Kuan, J. X., Xu, P., Yan, Z., Zarov, I., Zhang, Y., Fan, A., Kambadur, M., Narang, S., Rodriguez, A., Stojnic, R., Edunov, S., and Scialom, T. (2023). Llama 2: Open foundation and fine-tuned chat models.

[tuner007, 2021] tuner007 (2021). pegasus_summarizer. `https://huggingface.co/tuner007/pegasus\_summarizer`.

[Vaswani et al., 2023] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L., and Polosukhin, I. (2023). Attention is all you need.

[Yadav et al., 2022] Yadav, D., Desai, J., and Yadav, A. K. (2022). Automatic text summarization methods: A comprehensive review.

[Yan et al., 2020] Yan, Y., Qi, W., Gong, Y., Liu, D., Duan, N., Chen, J., Zhang, R., and Zhou, M. (2020). Prophetnet: Predicting future n-gram for sequence-to-sequence pre-training. *arXiv preprint arXiv:2001.04063*, pages 2401–2410.

[Zhang et al., 2019] Zhang, J., Zhao, Y., Saleh, M., and Liu, P. J. (2019). Pegasus: Pre-training with extracted gap-sentences for abstractive summarization.

[Zhang et al., 2020] Zhang, J., Zhao, Y., Saleh, M., and Liu, P. J. (2020). Pegasus: Pre-training with extracted gap-sentences for abstractive summarization.