CS584 - Project Final Report

David Arias Cuadrado - CWID: A20521638
Miguel Cózar Tramblin - CWID: A20522001
Carlos Muñoz Losa - CWID: A20521562

28th April 2023

1 Introduction

Our project aims to develop a generative model able to autonomously generate concise and informative summaries for news in social media. Specifically, we focus on using machine learning techniques, particularly generative models.

After months of research and development, we present successful results that fulfil our initial objectives. Our generative model is capable of generating readable summaries that capture the essential points of news articles. Moreover, we equip our model to operate on Twitter, where users can easily access news content by mentioning our account and citing a URL with the news. Our model meets the objectives that we set, and we are believe it has enough potential to perform at a high level in real case scenarios, retrieving information, summarizing it and presenting it to the audience. We public all the code, available at [1].

2 Problem description: Theory

News summarization is the task of generating a shorter version of a news article while retaining the most important information. Generative models are a popular approach to this problem, where a model is trained to generate summaries based on input news articles. The state of the art in this area can be characterized by the following:

1. Transformer-based models: Transformer-based models, such as BERT, GPT-2, and T5, have shown great success in natural language processing tasks, including news summarization [2]. These models are pre-trained on large amounts of text data and fine-tuned on specific summarization tasks, resulting in high accuracy and fluency in generating summaries.

2. Encoder-Decoder architectures: Many state-of-the-art models for news summarization use an encoder-decoder architecture, where the input article is encoded into a fixed-length vector and then decoded into a summary. The encoder and decoder can be based on different types of neural networks, such as LSTMs, CNNs, and Transformers [3].

3. Reinforcement Learning: Another approach to generative models for news summarization is using reinforcement learning, where the model learns to optimize a reward function that measures the quality of the generated summary. This approach has shown promising results in generating more concise and coherent summaries [4].

4. Evaluation Metrics: Finally, the state of the art in news summarization also includes the development of evaluation metrics that assess the quality of the generated summaries. Common metrics include ROUGE (Recall-Oriented Understudy for Gisting Evaluation), which measures the overlap between generated and reference summaries, and BLEU (Bilingual Evaluation Understudy), which measures the n-gram similarity between generated and reference summaries.

3 Methodology

3.1 Dataset

The CNN/DailyMail dataset is a large English-language dataset consisting of over 300,000 news articles written by journalists at CNN and the Daily Mail. The dataset is designed to support a variety of natural language processing tasks, including extractive and abstractive summarization, as well as machine reading and comprehension and abstractive question answering. The current version of the dataset, Version 3.0.0, is primarily focused on summarization tasks. Each instance in the dataset contains a string for the article, a string for the highlights, and a string for the id, with a mean token count of 781 for the article and 56 for the highlights.

The dataset is split into three sets: train, validation, and test, with 287,113, 13,368, and 11,490 instances, respectively, in Version 3.0.0. The articles were written between April 2007 and April 2015 for CNN and between June 2010 and April 2015 for the Daily Mail. The dataset is available in both en-US and en-GB BCP-47 codes for English. The original data collection was done using archives of the CNN and Daily Mail websites on the Wayback Machine, and articles were excluded from the dataset if they exceeded 2000 tokens. The dataset also includes nearly 1 million Cloze-style questions to go with the articles.

The dataset was originally curated to support supervised neural methodologies for machine reading and question answering, with the goal of providing a large amount of real natural language training data. Versions 2.0.0 and 3.0.0 changed the structure of the dataset to support summarization tasks, with Version 3.0.0 providing a non-anonymized version of the data.

The code for the original data collection is available at [5], and the dataset is available at [6]. Additionally, we have opted for the Kaggle version of the same, which performs a deeper statistical analysis of the articles/highlights [7].

3.2 Model Proposals

This section covers all the models proposed by the team and their implementations

3.2.1 LSTM Self Trained Model

Our first approach has consisted on our own design of the model. Traditionally, Sequence to sequence models have been used for generative models, for summarization and other types of tasks such as machine translation. These models are composed by two sequential sub models, with their respective inputs.

In our model, the two inputs are the article and the summary. But since models do not treat raw text, we have to transform them to tokens. One typical strategy would have been the employment of sentence transformers. However, and with the aim of having everything self-made, we prepared our own tokenizers fitted on out own vocabulary, extracted from the dataset. We train one summary tokenizer and one article summarizer. We design two different tokenizers because the dimensions of the outputs of the tokenizers are different for summaries and articles, due to the big difference in terms of length.

Before introducing the text to the tokenizers, we apply a preprocess of the text. We leverage previous used preprocesses on other projects to delete non alphanumerical characters, URLs, punctuation at the end of words, multiple spaces and single characters hanging between spaces. We decline removing stopwords and lemmatizing because we need the generated text to be legible. Afterwards, we introduce the text to a NLP Spacy pipeline to add the start and end tokens to the summaries: *sostok* and *eostok*.

Once completed the tokenizers training we transform the articles into 1700 dimension embedding vectors and the summaries into 150 dimension vectors (padding shorter sequences). Once ended the process. The vectors are fit with the architecture shown in Figure 1. The article is encoded through the encoder part (the left), whereas the summary goes in the decoder (right). The idea is that, after the non linearity applied by the LSTM layers and the change of shape, we can compare both encodings and minimize the loss in the time distributed layer.

After training, we break the model into decoder and encoder. We use the encoder to generate the vector from the article text and iteratively generate a set of tokens feeding the decoder with the already generated sequence and the encoded vector. The we convert this tokens into words with the tokenizer reverse index dictionary, that contains the numeric translation of all the words in our vocabulary.

3.2.2 Bert extractive summarizer

We have experiment with text summarization using a pre-trained model, specifically the BERT model [8]. To compare our results with a model that has been trained for a similar task, with much more resources and a more complex architecture, so we can appreciate the differences between that model and our own personalized model trained with our own dataset. We have concretely used a bert-extractive-summarizer model [9]. This model has been trained for a lecture summarization task for students. It employs the BERT model and K-Means clustering to identify the most representative sentences for summarization. It is divided in different stages. First, lecture management allows users to create, edit, delete, and retrieve stored items, while the summary generation

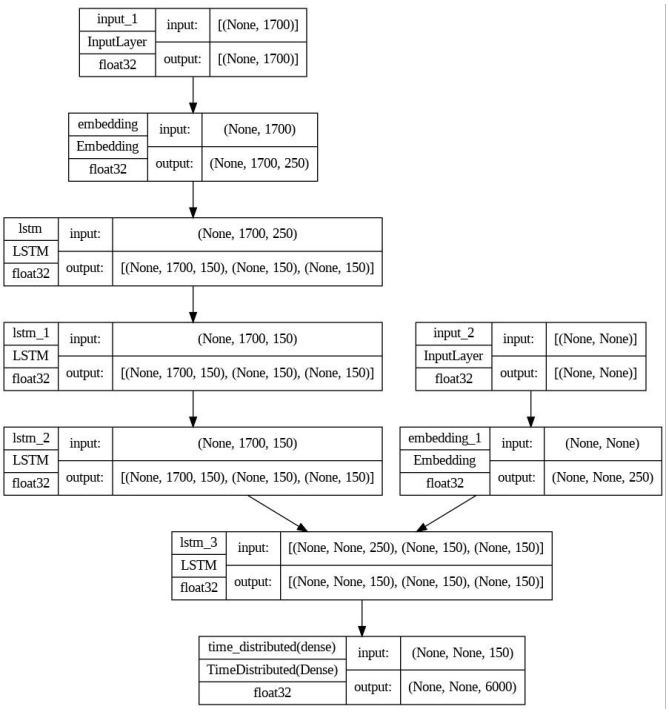

Figure 1: LSTM self-made model architecture

component uses the BERT model to produce embeddings for clustering and creates a summary using K-Means. The model itself is capable of doing text preprocessing (removing stopwords and lemmatization), tokenization, and embedding vector transformation. Process the embedding and finally decode the output of the model to obtain plainly readable text as output.

The BERT model that is uses [8]. BERT (Bidirectional Encoder Representations from Transformers) is a family of masked-language models introduced in 2018 by researchers at Google. BERT is designed to pre-train deep bidirectional representations from unlabeled text by jointly conditioning on both left and right context in all layers. The pre-trained BERT model can be fine-tuned with just one additional output layer to create state-of-the-art models for a wide range of tasks, without substantial task-specific architecture modifications. This model improves the previous GPT unidirectionality constraint by using a "masked language model" (MLM) which allows us to pretrain a deep bidirectional Transformer.

To use of the extractive summarizer we adapt several parameters. This model was initially trained for summarizing large lectures and it was designed for choosing the minimum and maximum number of sentences. News summarization does not require large texts, in fact, the goal is to summarize the texts as much as possible. Reducing the number of maximum sentences at the output we found that the model started to miss very relevant information, the sentences were unconnected and the performance decreased substantially. So finally we have to establish the minimum and maximum length of the output to 20 and 1000 respectively.

3.2.3 DistilBART

DistilBART-CNN-12-6 is a state-of-the-art natural language processing model that has gained popularity in recent years. It is a variant of the BART (Bidirectional and Auto-Regressive Trans-

formers) model that has been distilled to improve its efficiency while retaining its high performance.

DistilBART-CNN-12-6 is a transformer-based language model that uses a combination 12 layer encoders and 6 layer decoders to process text. The model has 12 transformer layers and 6 CNN layers. The transformer layers are used to capture long-range dependencies in the text, while the CNN layers are used to capture local patterns.

The model is pre-trained on a large corpus of text data using a self-supervised learning technique called masked language modeling (MLM). During training, a random subset of the input tokens are replaced with a special token [MASK], and the model is trained to predict the original token given the context. This enables the model to learn the underlying patterns in the language and make predictions on unseen text data.

After pre-training, the model is fine-tuned on a specific task using supervised learning, in this case the CNN/DailyMail dataset for text summarization. The fine-tuning process involves training the model on a labeled dataset to learn task-specific patterns in the data. The model can be fine-tuned for various natural language processing tasks, such as text classification, question answering and language generation (text summarization for our task).

The DistilBART model, like the original BART, has a maximum input length of 1024 tokens. This means that it can only process text inputs that are less than or equal to 1024 tokens in length. To address this constraint, we split the input text into segments of up to 4096 characters (which is an approximation of 1024 tokens). Note we have used the character approximation instead of the tokens to improve the performance of the model and the coherence of the texts. The code uses regular expressions to split the text into sentences and then iteratively adds sentences to a current segment until the length of the current segment exceeds 4096 characters, at which point the current segment is added to the list of segments, and a new segment is started.

Each segment is then summarized using the DistilBART model with a maximum summary length of 1024 tokens, and the resulting summaries are combined into a final summary by joining them with a space character.

To further reduce the length of the final summary, the code also summarizes the combined summaries again, this time with a maximum length of 512 tokens, and prints the resulting summary text.

3.2.4 Performance metrics

To assess the models' performance, we researched various evaluation metrics [10] and settled on BLEU, Rouge1, and RougeL.

BLEU, Rouge1, and RougeL are evaluation metrics commonly used in the field of text summarization. BLEU measures the similarity between the generated summary and the reference summaries by counting the number of overlapping n-grams (contiguous sequences of n words) between them.

Rouge-1 and RougeL, on the other hand, are based on the recall of n-grams in the generated summary and reference summaries, where Rouge-1 only considers unigrams (single words) and RougeL uses the longest common subsequence of words between the generated summary and the reference summaries.

These metrics are suitable for text summarization because they provide a quantitative measure of how well a summary captures the important information from the original text. They are also language-agnostic, meaning that they can be used to evaluate summaries in any language. Additionally, they have been widely adopted by the research community and have become the de facto standard for evaluating text summarization systems. [11]

4 Results

4.1 LSTM Sequence to Sequence model

We tested different settings for trying to make this model work. However, we have experienced a complicated lack of computational resources. We achieved the training of the network over a small sample of the dataset (25K articles), but the training set was apparently not enough to make it work satisfactorily.

We trained over 20 epochs, minimizing the loss (crossentropy) at less that 1. The results seemed promising, since the initial loss was of more than 10. However, when we splitted the model into encoder-decoder and tested some sequences, we observed that the model was outputting all the time the same sequence. Even though it was the model in which we invested the most time, we were not able to make a good training. Also because we were far away from the current state of the art, that we later cover with the rest of the results.

4.2 BERT and BART

After compiling a test dataset with more than eleven thousand articles and highlights, BERT-generated summaries, DistilBART summaries and the aforementioned metrics for each summary, we were ready to analyze the results.

Table1 summarizes the metrics we found. Overall, the reported metrics suggest that the BERT model's performance on the test dataset for text summarization is modest, with low scores for BLEU and moderate scores for Rouge1 and RougeL. However, it is important to consider these metrics in the context of the specific use case and the nature of the text being summarized.

	BLEU	**Rouge1**	**RougeL**
count	11490	11490	11490
mean	0.0122	0.239	0.155
std	0.0318	0.0983	0.0746
50%	9.033e-4	0.226	0.141
max	0.658	0.862	0.775

Table 1: BERT metrics on test dataset.

Table 2 presents performance metrics for a model trained on the DistilBART architecture and evaluated on the test dataset.

Compared to the BERT Table 1 metrics, the DistilBART model achieves higher scores on all three metrics, suggesting that it performs better than the BERT model in generating high-quality summaries. Specifically, the BLEU scores with a mean of 0.0875 and a median of 0.0578, indicating that the DistilBART model has a higher level of overlap with the reference translations than the BERT model. The Rouge1 scores with a mean of 0.403 and a median of 0.396, while the RougeL scores with a mean of 0.273 and a median of 0.252. These scores indicate that the DistilBART model achieves higher overlap with the reference summaries than the BERT model.

The reported standard deviations in Table 2 are also lower than in Table 1, indicating that the results obtained from the DistilBART model are more consistent than those obtained from the BERT model.

In summary, the DistilBART model outperforms the BERT model on all three metrics, suggesting that it generates higher-quality summaries than the BERT model. However, it is important to note that the specific use case and the nature of the text being summarized may affect the relative performance of the two models, and further evaluation may be necessary to determine the optimal choice for a given task.

	BLEU	**RougeL**	**RougeL**
count	1.1490	11490	11490
mean	0.0875	0.403	0.273
std	0.105	0.121	0.115
50%	0.0578	0.396	0.252
max	0.96	1	1

Table 2: DistilBART metrics on test dataset.

In the figures located in annex A the differences are even more notorious. It is clear that the model trained with the "CNN daily news" scores improve the performance of the model in this specific task.

5 Real World Case Scenario: A Twitter Bot deployment

To test our model, we decide to implement a real world case scenario. Twitter is a well known social media, with over 500 million tweets published per day. Millions of people inform themselves daily on Twitter and synthesizing properly and accurately is highly appreciate. Furthermore, the usage of bots is widely adopted by the public, with link providers, commenters, and other examples.

The Twitter bot architecture we propose is as per fig. 2. The prerequisite is to mention it with a public URL on the content of the tweet. Our bot is listening on streaming with a cool down of one minute. This means that every 60 seconds we check for new mentions. If we detect non-previously answered mentions, we put them on a queue. This queue turns on one of the modules of the bot: NewsScraper.

NewsScraper has the tools to parse the obtained URL. It leverages python *requests-html*. With this library we are able to retrieve the information of every web page, and solve possible problems derived from Selenium and simple Requests with JavaScript code. After loading the new, we remove all HTML labels and get a raw text. We experienced some issues after this step. All HTML labels have their own text: buttons, copyright footnotes, links, images... All these together supposes a big challenge for the model, since it won't have the presence of clean text to summarize, but a lot of noise is added to the equation.

Our proposal to mitigate the effect of noise consists on leveraging the Flesch Readability Test. This is a method of measuring the readability and understandability of a written text, developed by Rudolf Flesch in 1948 and based on two factors: the average sentence length and the average number of syllables per word. The Flesch readability score is calculated using a formula that takes these two factors into account, and produces a score between -inf and 100. A higher score indicates that the text is easier to read and understand, while a lower score indicates that the text is more difficult to read and understand [12].

$$F_{score} = 206.835 - 1.015(N_{words}/N_sentences) - 84.6(N_syllables/N_words) \quad (1)$$

Where N means total amount. If we split the text into sentences and put a threshold of readability of 0, we produce legible sentences and remove noisy parts. Apart from that, we also employ a "Legal" Bag of Words to get rid of legal terms appearing on web pages. With this technique, for instance, sentences like "All rights reserved" are deleted from the text.

Once the text is processed, TwitterBot turns on again (it is the module checking the number of mentions and mentioning the reply). It gets the text and selects a model to perform summarization, with certain summarizing ratio. Then, it outputs the resume via Twitter thread, mentioning the user who previously mentioned the bot.

We decide to not limits the number of characters to just a tweet because most news are dense and a summary in 280 characters is not enough, and produces unsatisfactory results. What fig. 2 shows is a real case in which a user mentions our bot, and we output a summary of the URL the user provided the bot.

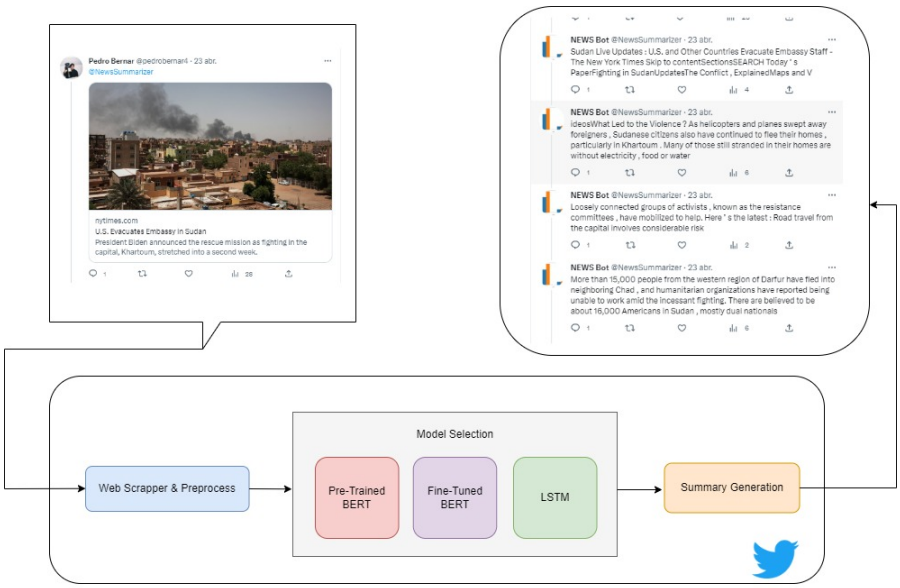

Figure 2: LSTM self-made model architecture

6 Conclusions and future work

In conclusion, our project successfully develops a generative model capable of generating concise and informative summaries for news articles. We experimented with various models, including a self-trained LSTM model, BERT extractive summarizer, and DistilBART. The evaluation metrics, BLEU, Rouge1, and RougeL, revealed that the DistilBART model outperformed the others in generating high-quality summaries.

To showcase the practical application of our model, we implement a real-world use case by deploying a Twitter bot that generates news summaries when provided with a URL. The bot processes the input text by employing the Flesch Readability Test and a "Legal" Bag of Words to filter out noise and generate legible summaries.

Overall, our project demonstrates the potential of generative models in enhancing news summarization and information retrieval. The successful implementation of a Twitter bot demonstrates the applicability of our model in real-world scenarios, providing users with concise and informative news summaries. The results of our project can serve as a foundation for further research and development in the field of generative models for text summarization.

6.1 Future work

While our project has achieved significant results, there are several directions for future work to further enhance the performance and applicability of our generative models for news summarization:

1. Fine-tuning and optimization: Further fine-tuning and optimization of the existing models

could lead to improvements in the quality of the generated summaries. This may involve exploring different pre-training techniques, hyperparameter tuning, or experimenting with different model architectures.

2. Exploration of abstractive summarization techniques: In our current work, we mainly focused on extractive summarization techniques. Future work could explore the use of abstractive summarization techniques, which involve generating new sentences to better capture the meaning of the original text, resulting in more concise and coherent summaries.

3. Multi-lingual and domain-specific summarization: Our current models are trained on English news articles. Expanding the models to support multi-lingual and domain-specific summarization would increase their applicability and utility. This may require the collection and curation of new datasets for training and evaluation.

4. Evaluation with user studies: In addition to the existing evaluation metrics, future work could include user studies to assess the perceived quality and usefulness of the generated summaries. This would provide valuable insights into how well the models meet the needs of real users.

5. Ethical considerations and potential biases: As with any machine learning model, it is important to consider potential biases and ethical implications. Future work could involve conducting thorough analyses to identify and mitigate any potential biases in the training data or generated summaries, as well as exploring approaches to ensure the responsible and ethical use of these models.

By addressing these future directions, we believe that the generative models for news summarization can be further improved and their potential impact on information retrieval and presentation can be more fully realized.

References

[1] *github repository*. URL: https://github.com/carlosmlosa/news_summarization.

[2] Kai Ma et al. "What is this article about? Generative summarization with the BERT model in the geosciences domain". In: *Earth Science Informatics* 15 (1 Mar. 2022), pp. 21–36. ISSN: 18650481. DOI: 10.1007/S12145-021-00695-2.

[3] Zhengpeng Li et al. "News headline generation based on improved decoder from transformer". In: *Scientific Reports 2022 12:1* 12 (1 July 2022), pp. 1–12. ISSN: 2045-2322. DOI: 10.1038/s41598-022-15817-z. URL: https://www.nature.com/articles/s41598-022-15817-z.

[4] Rajeev Kumar Singh et al. "SHEG: summarization and headline generation of news articles using deep learning". In: *Neural Computing and Applications* 33 (8 Apr. 2021), pp. 3251–3265. ISSN: 14333058. DOI: 10.1007/S00521-020-05188-9/TABLES/8. URL: https://link.springer.com/article/10.1007/s00521-020-05188-9.

[5] *deepmind/rc-data: Question answering dataset featured in "Teaching Machines to Read and Comprehend*. URL: https://github.com/deepmind/rc-data.

[6] *abisee/cnn-dailymail: Code to obtain the CNN / Daily Mail dataset (non-anonymized) for summarization*. URL: https://github.com/abisee/cnn-dailymail.

[7] *CNN-DailyMail News Text Summarization — Kaggle*. URL: https://www.kaggle.com/datasets/gowrishankarp/newspaper-text-summarization-cnn-dailymail?resource=download.

[8] *bert extractive summarizer model*. URL: https://pypi.org/project/bert-extractive-summarizer/.

[9] Derek Miller. "Leveraging BERT for Extractive Text Summarization on Lectures". In: (2019). arXiv: 1906.04165 [cs.CL].

[10] *Evaluation Metrics for text summarization*. URL: https://direct.mit.edu/tacl/article/doi/10.1162/tacl_a_00373/100686/SummEval-Re-evaluating-Summarization-Evaluation.

[11] *Evaluation Metrics for text summarization*. URL: https://www.freecodecamp.org/news/what-is-rouge-and-how-it-works-for-evaluation-of-summaries-e059fb8ac840/.

[12] Jordan Lovett et al. "Online information on dysmenorrhoea: An evaluation of readability, credibility, quality and usability". In: *Journal of Clinical Nursing* 28 (19-20 Oct. 2019), pp. 3590–3598. ISSN: 1365-2702. DOI: 10.1111/JOCN.14954. URL: https://onlinelibrary.wiley.com/doi/full/10.1111/jocn.14954%20https://onlinelibrary.wiley.com/doi/abs/10.1111/jocn.14954%20https://onlinelibrary.wiley.com/doi/10.1111/jocn.14954.

A Annex. Additional figures

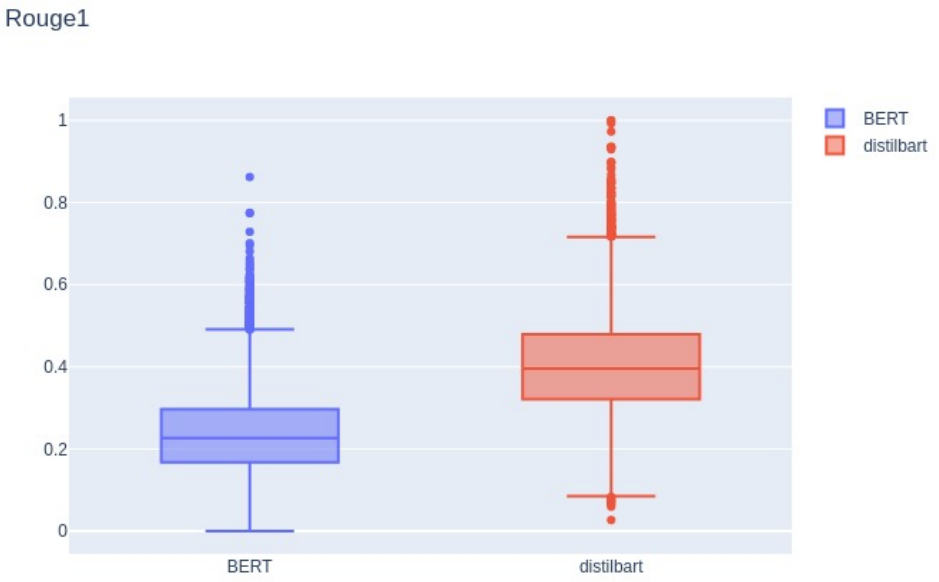

Figure 3: Rouge1 metric Boxplot

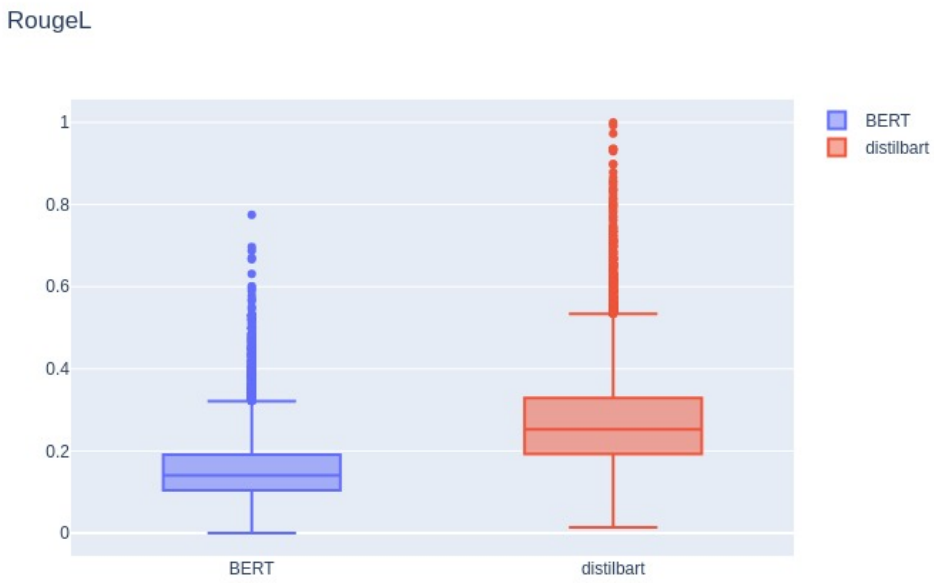

Figure 4: RougeL metric Boxplot

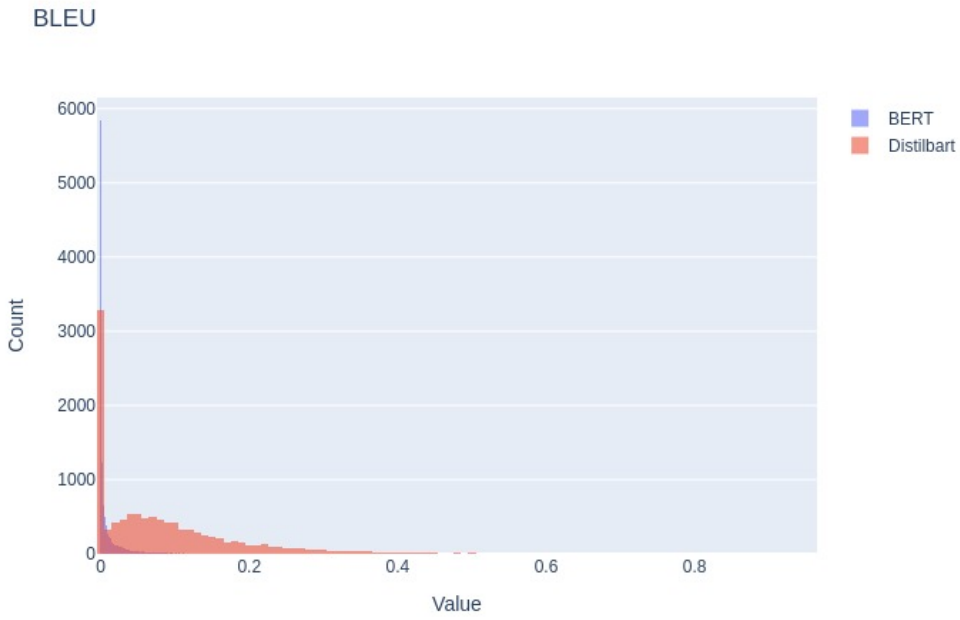

Figure 5: BLEU metric histogram

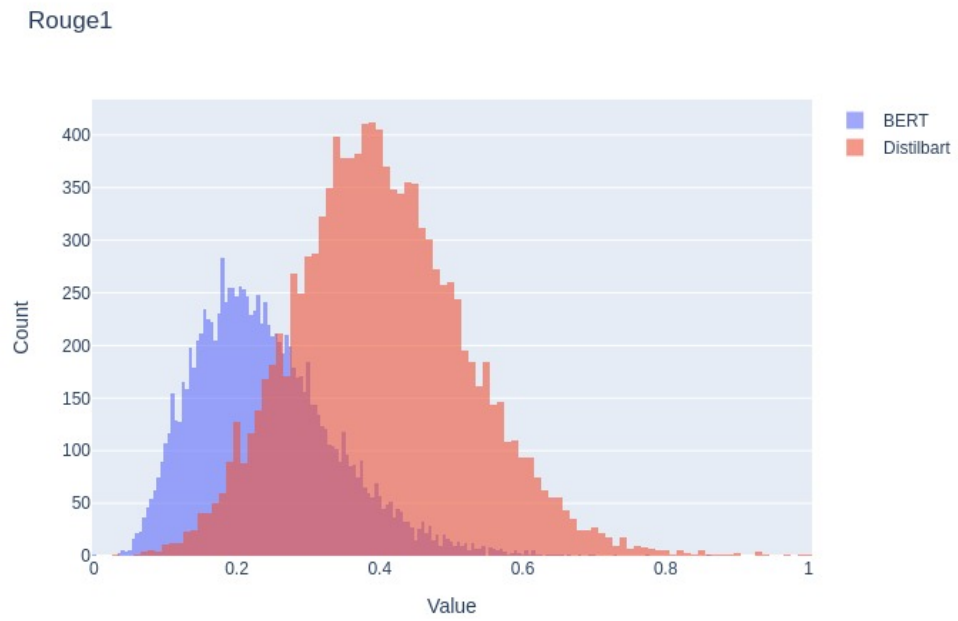

Figure 6: Rouge1 metric histogram

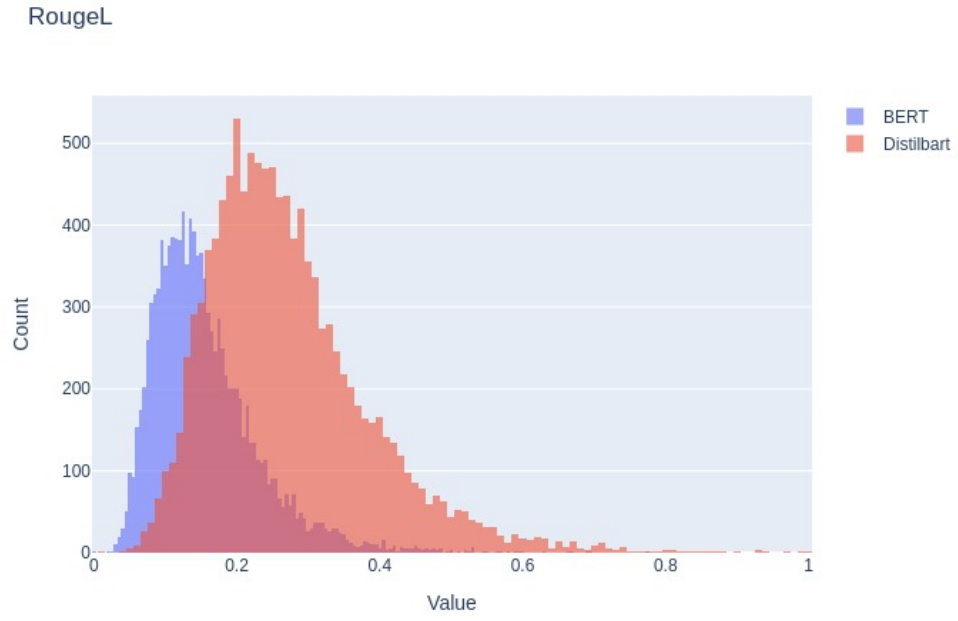

Figure 7: RougeL metric histogram