Summarization of Legal Texts with Artificial Intelligence (Extractive summarization)

Cleto Pellegrino

June 4, 2024

Contents

1 Introduction

The goal of this project is to **analyze pre-trained models** in the task of summarizing EU legal texts, in order to *understand* and *extrapolate* in some few lines the main content of long-format texts, exploiting **extractive summarization technique**.

2 Dataset Description

The dataset used for the problem was taken from the **Huggingface** website [1], and was provided by Dennis Aumiller (dennlinger on the site) and it's described as follows:

"*The EUR-Lex-Sum dataset is a multilingual resource intended for text summarization in the legal domain. It is based on human-written summaries of legal acts issued by the European Union. It distinguishes itself by introducing a smaller set of high-quality human-written samples, each of which have much longer references (and summaries!) than comparable datasets.*"

Each instance in the dataset (which is already split into Train, Validation, and Test sets) has a column named "**Reference**", which represents the **whole text**, and a column named "**Summary**" that can be considered as a **label**.

The column containing the "*celex_id*" was used only for **preprocessing reasons**.

As the provider of the dataset wrote, **no one has ever used this data before for summarization problems** (the dataset is also very recent).

[...]"*We further noticed that no previous system had utilized the human-written samples from the EUR-Lex platform, which provide an excellent source for training instances suitable for summarization research.* "[...]

3 Natural Language Processing

The main issue faced while approaching the problem of summarizing long legal texts it's straightforward: **the size of the whole input is bigger than the allowed input by models** (sometime even if the input fits, a commodity GPU is not able to perform the operations required for fine-tuning or just generating results, due to **size limitations**).

To reduce the size of the input, the "*celex_id*" of some rows was exploited in order to search for **patterns** inside the original PDFs available in the EUR-Lex website [2].

As a result, what was found is that the main content of a legal text is bounded between lines like:

- HAS ADOPTED THIS REGULATION:
- HAVE AGREED AS FOLLOWS:

and ends with:

- "Done at", followed by the place;
- an ANNEX, which could be important to better understand the content of the text, but for a summary can be **excluded**.

The following plots (Figure 1, Figure 2, Figure 3) show the distributions ($bins = 30$) of the "*reference_tokens_unpreprocessed*", while Figure 4, Figure 5, Figure 6 shows the distributions of the "*reference_tokens_preprocessed*" calculated after doing this **first phase** of the preprocessing:

From this plots (and with the help of **Pandas library** in Python) one could notice that with new models, which can handle 16384 tokens (and exploits **global attention techniques** [3]), more than 75% of the texts could fit **entirely**. This information can be exploited for future **fine-tunings**.

The second phase was done in order to **simplify the job of the model** during the summarization procedure. Two simple operations were done:

- Rewriting the text in lower case;
- Lemmatization;

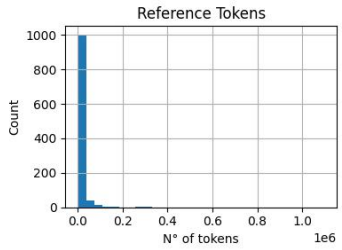


Figure 1: Train set unpreprocessed.

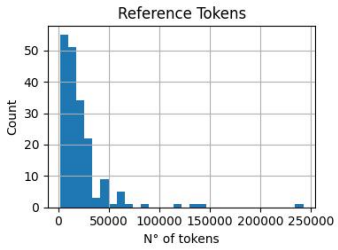


Figure 2: Validation set unpreprocessed.

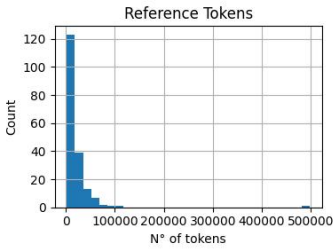


Figure 3: Test set unpreprocessed.

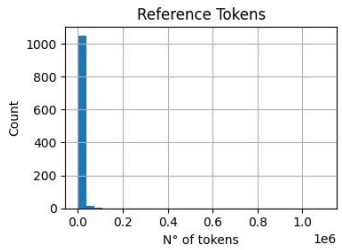


Figure 4: Train set preprocessed.

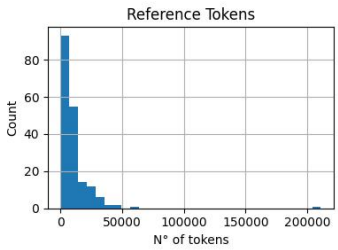


Figure 5: Validation set preprocessed.

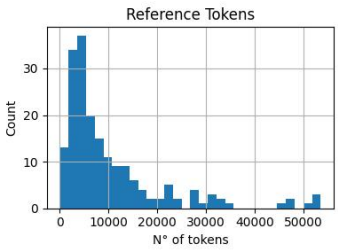


Figure 6: Test set preprocessed.

Further steps, like removing numbers or stop-words, proved to be **detrimental** while performing summarization, since the model which will be presented wants a text that maintains some **meaning**.

Here is an example of result of the **whole preprocessing procedure**:

- Reference:

  ```
  ha adopted this regulation : chapter i definition article 1 definition for the
  purpose of this regulation , the following definition shall apply : ( a ) ‘
  asset-backed security ’ mean non-equity security which either : ( i ) represent
  an interest in asset , including any right intended to ensure the servicing of
  those asset , the receipt or the timely receipt by holder of those asset of the
  amount payable under those asset ; ( ii ) are secured by asset and the term of
  the security provide for payment calculated by reference to those asset ; ( b )
  ‘ equivalent third country market ’ mean a third country market which ha been
  deemed equivalent to a regulated market in accordance with the requirement set
  out in third and fourth subparagraphs of article 25 ( 4 ) of directive 2014/65/eu
  of the european parliament and of the council ( 3 ) ; ( c ) ‘ profit estimate ’
  mean a profit forecast for a financial period which ha expired and for which
  result have not yet been published ; ( d ) ‘ profit 32019R0980 [...]
  ```

- Summary

  ```
  prospectus to be published when security are offered to the public or admitted
  to trading on a regulated market prospectus to be published when security are
  offered to the public or admitted to trading on a regulated market summary of :
  regulation ( eu ) 2017/1129 on the prospectus to be published when security are
  offered to the public or admitted to trading on a regulated market delegated
  regulation ( eu ) 2019/979 supplementing regulation ( eu ) 2017/1129 on regulatory
  technical standard on key financial information in the summary of a prospectus ,
  the publication and classification of prospectus , advertisement for security ,
  supplement to a prospectus , and the notification portal delegated regulation
  ( eu ) 2019/980 supplementing regulation ( eu ) 2017/1129 on format , content ,
  ```

```
scrutiny and approval of the prospectus to be published when security are offered
to the public or admitted to trading on a regulated market delegated regulation
( eu ) 2021/528 supplementing regulation ( 32019R0980 [...]
```

Some texts were dropped because they were **not compliant with the rules provided for preprocessing**. This are the results of the "drop None" procedure in terms of resulting instances:

- Train set, from 1129 to 1074
- Validation set, from 187 to 186
- Test set, from 188 to 187

4 Model Selection

Going back to the main issue, there are available models that can handle huge amounts of tokens. Reformer, LongFormer and LongNet are some of those, but they are **not pretrained for summarization tasks**.

Another issue is also the fact that they are either **not available** for usage or not able to **fit in a "commodity GPU"**, and so it's not possible to use them.

After doing some research, **T5** [4] was selected, which is a model provided by **Google**. The decision was driven not only by the possibility to run it **locally** by downloading it from the HuggingFace website [5], but also by the fact that it can handle **8192 tokens** (T5 has no limitation thanks to **relative position embeddings** [6], so the only limitation is **memory**, indeed putting more tokens, as expected, resulted in **OutOfMemory** exeptions)

5 Model Architecture Description

The following informations were extracted by displaying the **whole structure of the model** ("google-t5/t5-large") Listing 1.

The T5 model comprises two key components: the **T5StackEncoder** and the **T5StackDecoder**. The T5StackEncoder incorporates a **Self-Attention** layer, which evaluate relationships among **tokens within the same sequence**.

The T5StackDecoder not only features a Self-Attention layer, which evaluates token relationships within the output sequence, but also includes a **Cross-Attention** layer. This Cross-Attention layer evaluates the relationships **between the input tokens and the output tokens**.

Another worth mentioning step was done in the **prompt creation**: indeed we need to specify that we want to perform a summary by putting "summarize: " at the **beginning** of the text.

6 Moving Window Method

The issue of handling huge amounts of tokens **still remains**. The approach utilized for the summarization is based on a **moving window** method and can be summarized in this pseudo-code:

Algorithm 1: ReadingMethod(text, model, tokenizer)

```
if tokens_in(text) ≤ MAX_ALLOWED_SEQUENCE then
    summary ← GenerateSummary(text, model, tokenizer);
    return summary;
end
else
    chunks ← SplitTextIntoChunks(text);
    summaries ← [];
    for chunk in chunks do
        summary ← GenerateSummary(chunk, model, tokenizer);
        summaries.append(summary);
    end
    merged_text ← MergeSummaries(summaries);
    if tokens_in(merged_text) > MAX_ALLOWED_SEQUENCE then
        summary ← ReadingMethod(merged_text, model, tokenizer);
    end
    else
        final_summary ← GenerateSummary(merged_text, model, tokenizer);
        return final_summary;
    end
end
```

In short, this algorithm provides **little summaries** across the whole text while maintaining some information between chunks exploiting the SplitTextIntoChunks() function, that can be performed in **three different ways**:

- Overlapping a percentage of the the previous text and putting it after the new chunk;
- Overlapping a percentage of the the previous text and putting it before the new chunk;
- Reading text with disjointed windows;

The last part of recursion is done in order to give the model the possibility to return a final result that was made by **looking at a complete text**, instead of partially. If instead the whole text can be inserted as a unique input in the model, it **directly** generates a summary.

7 Metric for comparison

In order to compare the summaries provided by the model against the reference summaries available in the dataset, **BertScore** [7] was chosen.

This decision was motivated by the fact that BertScore, unlike other metrics such as **ROUGE** that only exploits the presence of words **without considering the context**, incorporates **semantic information**, making it better suited for assessing the quality of summaries.

8 Statistical analysis

The test was performed by giving as an *output maximum tokens length* the value **8192**, like the input size, to avoid OutOfMemory exceptions. Also, **no minimum number of tokens** was selected and **no beam search strategy** [8] was adopted. Finally, **no length penalty** was selected, in order to give the possibility to the model to write texts **without constraints**.

The statistical methods utilized are the **Kruskall-Wallis Test** [9] as a non parametric test and the **ANOVA test** [10] as a parametric test.

8.1 Validation of the Assumptions

For the Kruskall-Wallis test the assumption are **few**: The *scipy.stats.kruskall* page of Scipy states that we need at least **5 measurement for groups** [11], and indeed we have them; then we have to assure that the results are **independent from each other** and this can be assured quite firmly given how the results are produced; last assumption is to have **similar shape for the distributions** and indeed from this plot Figure 7 we can see that this condition is satisfied.

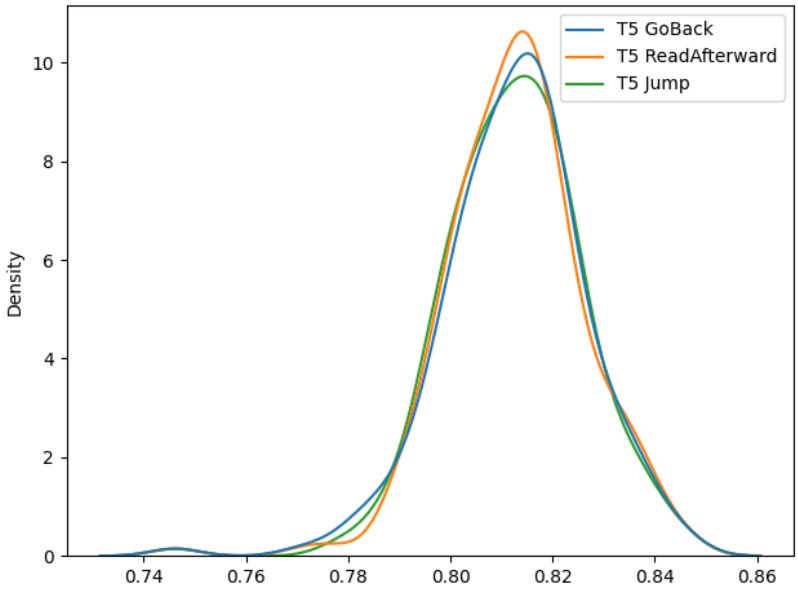


Figure 7: T5 with the three methods.

For the parametric test **more assumptions** were needed to validate in order to state that the results were **statistically relevant**: first of all we need to assure that the values are **normally distributed**. This can be showed by plotting the **Q-Q plot of the groups** (Figure 8, Figure 9, Figure 10);

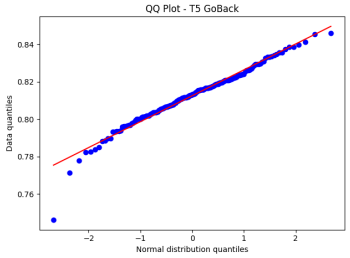


Figure 8: Train set.

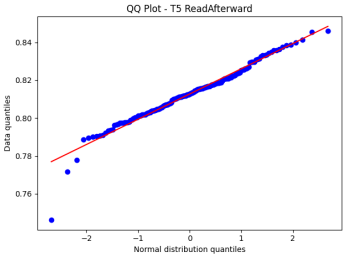


Figure 9: Validation set.

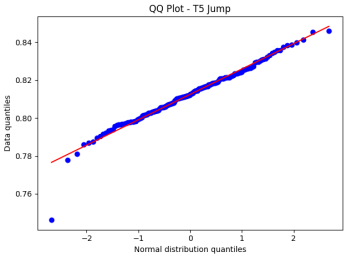


Figure 10: Test set.

we then skip the part about independence since was proved before and we focus on the last assumption which is the **homoscedasticity**, so they have the same **finite variance**. This can be proved using the **Levene's test** [12] which indeed proves this property:

```
Levene's statistic: 0.10322449421879756
p value: 0.9019416925501255
There is no evidence to reject the null hypothesis of equal variances (homoscedasticity).
```

8.2 Conclusions

Both the he statistical analysis showed **no significant difference between the reading methods**. Indeed we have:

- Kruskall-Wallis:

```
Kruskal-Wallis statistic: 0.02846787556307589
p value: 0.9858668857718969
No statistically relevant difference between groups.
```

- ANOVA:

```
ANOVA statistic: 0.01182275904328721
p value: 0.9882471027047812
No statistically relevant difference between groups.
```

Given this result, we can say that overall the model is capable of summarizing legal texts maintaining similar information provided by the **human summaries** with all the provided method, that were created to challenge it in the task of summarizing the same content **arranged in different ways**.

9 Further Testing

Differently from before, now we want to test the **full capability** of the model in generating quality summaries, so in order to search for the best output, the **beam search** strategy was utilized.

The selected method was the JumpingWindow since overall it has less outliers (Figure 11) with respect of the others, and so it's less likely to have very bad results.

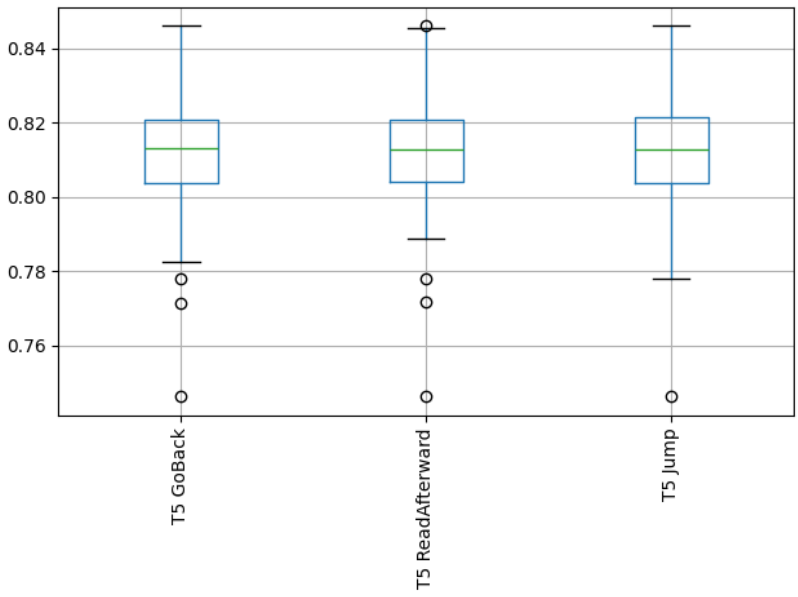


Figure 11: T5 with the three methods.

9.1 Texts with less than 8192 Tokens

The mean BertScore of the model on texts that were less than 8192, so that **we don't need to split in chunks**, is approximately 0.815039.

```
Example)
each member state shall submit to the commission ( eurostat ) data on job vacancy
at least for business unit with one employee or more. data shall cover all economic
activity defined by the common classification system for economic activity in the
community.
```

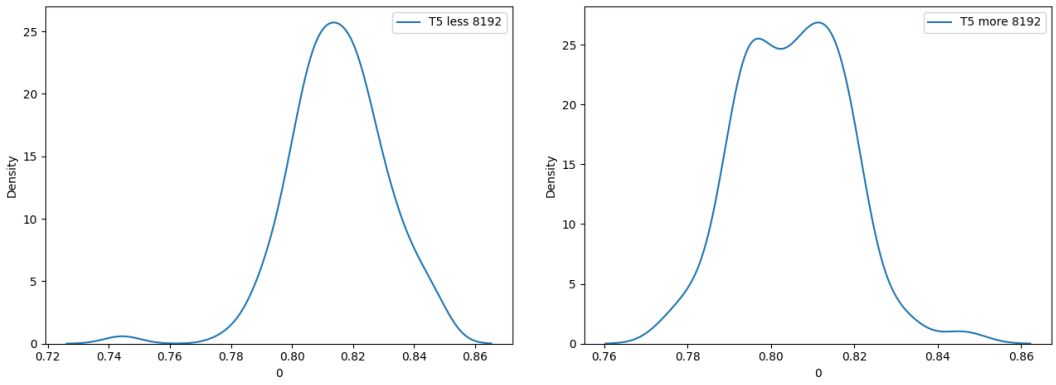
9.2 Texts with more than 8192 Tokens

Similar results as before were obtained, indeed the mean BertScore is approximately 0.8053786. We can conclude that the model is able to extract useful information **both were he can directly generate summaries and when it has to summarize chunk by chunk**.

```
Example)
eiopa shall monitor the market for insurance-based investment product. it shall
provide a key information document to enable retail investor to understand and
compare the key feature and risk of a priip. it shall ensure that action taken
is proportionate and dissuasive.
```

9.3 Analysis of the Results

From Figure 12



(a) Less than 8192.

(b) More than 8192.

Figure 12: Comparison of KDE plots for values less than and more than 8192.

it's clear that, when increasing the value for the beam search, the distribution of the BertScore of the plot Figure 14a is **shifted more to the right** rather then the other. When instead we compare the model with beam search equal 4, against the previous with beam search equal 1, we notice that the model performs **almost the same** (Figure 13).

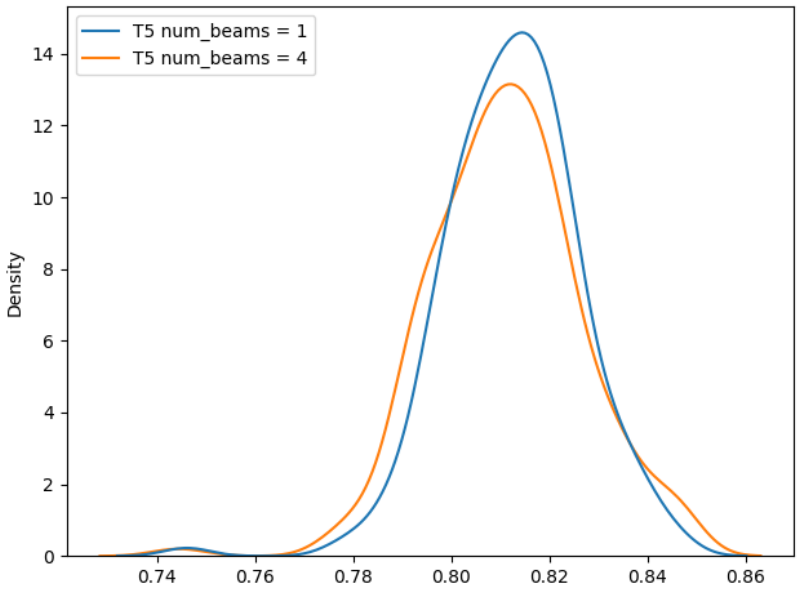


Figure 13: T5 with different values of the beam search.

The statistical tests confirms this result. Now we use the Wilcoxon Test and the T-Test since we are comparing just 2 distributions.

- Wilcoxon:

```
Wilcoxon statistic: 7929.0
p value: 0.245906657194636
No statistically relevant difference between groups.
```

- T-Test:

```
T-Test statistic: 0.9209711188929747
p value: 0.35766236569941423
No statistically relevant difference between groups.
```

9.4 Toy example of Fine-Tuning

This experiment was done since implementing some sort of application would be useless at this stage, and so it was better to allocate the time in testing even further the capabilities of the model. The experiment, due to size limitation, was done by considering only texts with less than **2048** tokens and the small version of the model.

The resulting dataset was composed by:

- 438 test instances;
- 14 validation instances;
- 18 test instances.

The following table (Table 1) shows the results obtained at each epoch (some of them were not displayed by the code and so are missing).

This little fine-tuning was able to **demonstrate how the style of the summary changes**, according to the labels in the dataset (no more just a list of information but rather a more human way to summarize).

Epoch	Rouge1	Rouge2	Rougel	Rougelsum	Bert Precision	Bert Recall	Bert F1	Gen Len
0	12.289500	5.144900	8.961100	11.513300	85.387000	78.257300	81.656500	76.571400
2	12.838300	5.154300	9.334700	12.083200	85.051000	78.272100	81.512300	82.214300
4	13.478700	6.071500	9.887900	12.627200	85.453000	78.533200	81.838900	84.214300
6	13.445600	5.365900	9.654100	12.493300	84.434000	78.127400	81.145800	90.500000
8	13.250900	5.327500	9.368000	12.306500	84.620700	78.221600	81.287200	86.357100
10	14.125000	6.034500	10.256500	13.148500	84.682700	78.435400	81.430800	92.857100
12	14.291800	5.918900	10.097500	13.164500	84.597400	78.430000	81.387400	94.857100
14	14.423800	5.967700	10.207400	13.358400	84.510400	78.429400	81.348000	95.571400
16	14.106100	5.789400	10.064800	13.053400	84.427100	78.350800	81.266600	95.214300
18	14.129200	5.787200	10.091100	13.114000	84.428900	78.352400	81.268300	95.285700
19	14.129200	5.787200	10.091100	13.114000	84.428900	78.352400	81.268300	95.285700

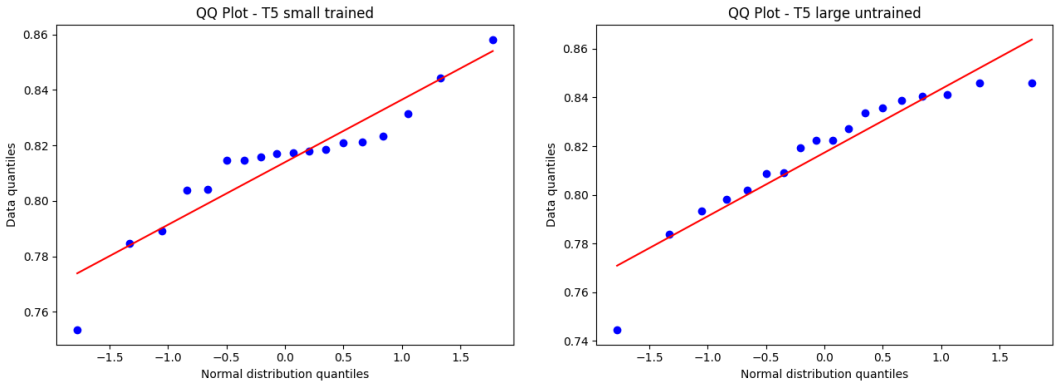
Table 1: Validation Metrics with BERT Scores and ROUGE

```
the european union recovery instrument ( the ‘ instrument ’ ) shall be carried out
under specific union programme and in accordance with the objective of the instrument.
the instrument shall be financed up to an amount of eur 750 000 million in 2018 price
on the basis of the empowerment provided for in article 5 of the own resource decision.
the measure shall be carried out under specific union programme and in accordance with
the relevant union act laying down rule for those programme.
```

Also, the small version of the model is now able to obtain similar results, in terms of BertScore, compared with the large one.

```
Wilcoxon statistic: 74.0
p value: 0.6396942138671875
No statistically relevant difference between groups.
```

We cannot use the parametric test since one of the distributions is no more normal.



(a) Trained model.

(b) Untrained model.

10 Future Works

There are many things that can be done to improve the project in its entirety. First of all, what should be done is to exploit the "freedom" from the token limitation of T5 to fine-tune the model in order to increase it's capability of learning the human way of writing text.

Another thing is to try the *Long* version of T5 (LongT5 [13]), which exploit the previous mentioned global attention layer to improve the quality of the summary.

Also fine-tuning this model and testing it against T5 can be a way to see how the global attention affects the summarization task after training.

Last thing that can be done is to increase the quality of the preprocessing phase with the collaboration of lawyers in order to really capture the content of a legal text.

Listing 1: T5 Large Architecture

```
  T5ForConditionalGeneration(
(shared): Embedding(32128, 1024)
(encoder): T5Stack(
  (embed_tokens): Embedding(32128, 1024)
  (block): ModuleList(
    (0): T5Block(
      (layer): ModuleList(
        (0): T5LayerSelfAttention(
          (SelfAttention): T5Attention(
            (q): Linear(in_features=1024, out_features=1024, bias=False)
            (k): Linear(in_features=1024, out_features=1024, bias=False)
            (v): Linear(in_features=1024, out_features=1024, bias=False)
            (o): Linear(in_features=1024, out_features=1024, bias=False)
            (relative_attention_bias): Embedding(32, 16)
          )
          (layer_norm): T5LayerNorm()
          (dropout): Dropout(p=0.1, inplace=False)
        )
        (1): T5LayerFF(
          (DenseReluDense): T5DenseActDense(
            (wi): Linear(in_features=1024, out_features=4096, bias=False)
            (wo): Linear(in_features=4096, out_features=1024, bias=False)
            (dropout): Dropout(p=0.1, inplace=False)
            (act): ReLU()
          )
          (layer_norm): T5LayerNorm()
          (dropout): Dropout(p=0.1, inplace=False)
        )
      )
    )
    (1-23): 23 x T5Block(
      (layer): ModuleList(
        (0): T5LayerSelfAttention(
          (SelfAttention): T5Attention(
            (q): Linear(in_features=1024, out_features=1024, bias=False)
            (k): Linear(in_features=1024, out_features=1024, bias=False)
            (v): Linear(in_features=1024, out_features=1024, bias=False)
            (o): Linear(in_features=1024, out_features=1024, bias=False)
          )
          (layer_norm): T5LayerNorm()
          (dropout): Dropout(p=0.1, inplace=False)
        )
        (1): T5LayerFF(
          (DenseReluDense): T5DenseActDense(
            (wi): Linear(in_features=1024, out_features=4096, bias=False)
            (wo): Linear(in_features=4096, out_features=1024, bias=False)
            (dropout): Dropout(p=0.1, inplace=False)
            (act): ReLU()
          )
          (layer_norm): T5LayerNorm()
          (dropout): Dropout(p=0.1, inplace=False)
        )
      )
    )
  )
```

```
  (final_layer_norm): T5LayerNorm()
  (dropout): Dropout(p=0.1, inplace=False)
)
(decoder): T5Stack(
  (embed_tokens): Embedding(32128, 1024)
  (block): ModuleList(
    (0): T5Block(
      (layer): ModuleList(
        (0): T5LayerSelfAttention(
          (SelfAttention): T5Attention(
            (q): Linear(in_features=1024, out_features=1024, bias=False)
            (k): Linear(in_features=1024, out_features=1024, bias=False)
            (v): Linear(in_features=1024, out_features=1024, bias=False)
            (o): Linear(in_features=1024, out_features=1024, bias=False)
            (relative_attention_bias): Embedding(32, 16)
          )
          (layer_norm): T5LayerNorm()
          (dropout): Dropout(p=0.1, inplace=False)
        )
        (1): T5LayerCrossAttention(
          (EncDecAttention): T5Attention(
            (q): Linear(in_features=1024, out_features=1024, bias=False)
            (k): Linear(in_features=1024, out_features=1024, bias=False)
            (v): Linear(in_features=1024, out_features=1024, bias=False)
            (o): Linear(in_features=1024, out_features=1024, bias=False)
          )
          (layer_norm): T5LayerNorm()
          (dropout): Dropout(p=0.1, inplace=False)
        )
        (2): T5LayerFF(
          (DenseReluDense): T5DenseActDense(
            (wi): Linear(in_features=1024, out_features=4096, bias=False)
            (wo): Linear(in_features=4096, out_features=1024, bias=False)
            (dropout): Dropout(p=0.1, inplace=False)
            (act): ReLU()
          )
          (layer_norm): T5LayerNorm()
          (dropout): Dropout(p=0.1, inplace=False)
        )
      )
    )
    (1-23): 23 x T5Block(
      (layer): ModuleList(
        (0): T5LayerSelfAttention(
          (SelfAttention): T5Attention(
            (q): Linear(in_features=1024, out_features=1024, bias=False)
            (k): Linear(in_features=1024, out_features=1024, bias=False)
            (v): Linear(in_features=1024, out_features=1024, bias=False)
            (o): Linear(in_features=1024, out_features=1024, bias=False)
          )
          (layer_norm): T5LayerNorm()
          (dropout): Dropout(p=0.1, inplace=False)
        )
        (1): T5LayerCrossAttention(
          (EncDecAttention): T5Attention(
            (q): Linear(in_features=1024, out_features=1024, bias=False)
```

```
            (k): Linear(in_features=1024, out_features=1024, bias=False)
            (v): Linear(in_features=1024, out_features=1024, bias=False)
            (o): Linear(in_features=1024, out_features=1024, bias=False)
          )
          (layer_norm): T5LayerNorm()
          (dropout): Dropout(p=0.1, inplace=False)
        )
        (2): T5LayerFF(
          (DenseReluDense): T5DenseActDense(
            (wi): Linear(in_features=1024, out_features=4096, bias=False)
            (wo): Linear(in_features=4096, out_features=1024, bias=False)
            (dropout): Dropout(p=0.1, inplace=False)
            (act): ReLU()
          )
          (layer_norm): T5LayerNorm()
          (dropout): Dropout(p=0.1, inplace=False)
        )
      )
    )
    (final_layer_norm): T5LayerNorm()
    (dropout): Dropout(p=0.1, inplace=False)
  )
  (lm_head): Linear(in_features=1024, out_features=32128, bias=False)
)
```

11 References

1. dennlinger/eur-lex-sum . Dataset
   https://huggingface.co/datasets/dennlinger/eur-lex-sum
2. Diritto dell'UE - EUR-Lex
   https://eur-lex.europa.eu/homepage.html?locale=it
3. Gentle Introduction to Global Attention for Encoder-Decoder Recurrent Neural Networks
   https://machinelearningmastery.com/global-attention-for-encoder-decoder-recurrent-neural-networ
4. T5 Explained — Papers With Code
   https://paperswithcode.com/method/t5
5. T5
   https://huggingface.co/docs/transformers/model_doc/t5
6. Relative Position Encodings
   https://paperswithcode.com/method/relative-position-encodings#:~:text=Relative%20Position%20Encodings%20are%20a%20type%20of%20position,the%20model%20on%20two%20levels%3A%20values%20and%20keys.
7. BERTScore: Evaluating Text Generation with BERT
   https://arxiv.org/abs/1904.09675
8. Beam search
   https://en.wikipedia.org/wiki/Beam_search
9. Kruskall-Wallis Test
   https://www.statology.org/kruskal-wallis-test/#:~:text=Before%20we%20can%20conduct%20a%20Kruskal-Wallis%20test%2C%20we,each%20group%20need%20to%20have%20a%20similar%20shape.
10. One-way analysis of variance
    https://www.statology.org/one-way-anova/#:~:text=For%20the%20results%20of%20a%20one-way%20ANOVA%20to,within%20groups%20were%20obtained%20by%20a%20random%20sample.
11. scipy.stats.kruskal
    https://docs.scipy.org/doc/scipy/reference/generated/scipy.stats.kruskal.html
12. Levene's test
    https://en.wikipedia.org/wiki/Levene%27s_test
13. LongT5: Efficient Text-To-Text Transformer for Long Sequences
    https://arxiv.org/abs/2112.07916