NLP Summarization

Luca Reggiani
Master's Degree in Artificial Intelligence, University of Bologna

Abstract

The performance of numerous pre-trained models for automated summarization, including bert tiny, t5 small, distil roberta, and bart small, is investigated in this study. The goal is to compare the models' capacity to provide accurate and short summaries. Each model is tested on a sample dataset and its performance is evaluated using metrics such as ROUGE1, ROUGE2, ROUGEL, and ROUGE-LSum. The primary finding is that distil roberta beats the other models in terms of summarization quality. The conclusions of this paper are useful for anybody interested in automatic summarization and looking for advice on choosing a good pre-trained model.

1 Introduction

Text summary is the process of condensing a longer text content into a coherent, fluid, and brief summary while also emphasizing its key ideas. Text summarization can be used in a variety of applications, such as news articles, legal documents, scientific papers, and social media posts. It can also be used to summarize large amounts of data, such as in data analytics or business intelligence.

There are two main approaches to text summarization:

- **extractive summarization:** It takes the text as input, evaluates each sentence's comprehension and importance, and then provides the strongest phrases. This approach just shows the words and phrases that already exist, without producing any new words or phrases.
- **abstractive summarization:** at the contrary, involves generating new sentences that capture the essence of the original text. It makes an attempt to infer the meaning of the entire text before presenting it to you. The most important facts from the text are also added, along with new words and phrases that are combined in a coherent way. This makes abstractive summarizing strategies more intricate and costly in terms of computing than extractive summarization techniques.

What is certain, is that when it comes to automatic summaries in the NLP context, one has to face some challenges. These challenges may vary in complexity, but they are real problems to be taken into account when one wants to implement a service that performs the summary task, and they differ also depending on the kind of summary required (extractive or abstractive).

The approach used in this project is the abstractive one. It is the most innovative, but it also arises some issues:

1. **Domain-specific language:** One of the biggest challenges in abstractive summarization is dealing with domain-specific language. For example, summarizing scientific papers or legal documents requires knowledge of specialized terminology and concepts that may not be present in general language models.
2. **Inconsistencies in language:** Another major challenge in abstractive summarization is dealing with inconsistencies in language. This can arise from colloquialisms, idioms, or other linguistic phenomena that are not captured by standard language models.
3. **Generating coherent summaries:** A third challenge in abstractive summarization is generating summaries that are coherent and faithful to the original text. This requires not only identifying key concepts and ideas but also organizing them in a way that makes sense to the reader.

Overall, fine-tuning can help address these challenges by allowing the model to adapt to domain-specific language, capture a wider range of linguistic phenomena, and learn from high-quality examples of summaries (by learning from examples of high-quality summaries and adapt to the specific task of summarization).

2 Background

The field of artificial intelligence has paid a lot of attention lately to natural language processing (NLP). NLP focuses on how computers and human languages interact, allowing devices to comprehend, decipher, and produce human language. Text summarizing, which aims to reduce lengthy texts into shorter, more concise versions while retaining the most crucial information, is one of the primary difficulties in NLP. Overall, in this project four pre-trained models are used: BERT, BART, T5-Small and DistilRoberta. T5 model and BART have a transformer encoder layer and a decoder layer. BERT and DistilRoberta use a transformer architecture with a multilayer encoder. Using an evaluation measure called ROUGE (Recall-Oriented Understudy for Gisting Evaluation) [2], we will train the model on a sizable dataset of news and assess its performance. The ultimate purpose of this project is to develop a reliable and precise text summarization model that can be applied to articles texts in order to obtain a clear headline with an abstractive approach.

3 System description

The following four pre-trained models are tried:

- **BERT Tiny:** BERT [1] (Bidirectional Encoder Representations from Transformers) is a powerful language model that uses a transformer architecture to generate embeddings for text. BERT Tiny is a smaller, more lightweight version of BERT that is trained on a smaller dataset and has fewer parameters than the full BERT model.

- **T5-Small:** T5 [4] (Text-to-Text Transfer Transformer) is a transformer-based language model that is designed to perform a wide range of natural language processing tasks, including text summarization. T5-Small is a smaller version of the T5 model that has fewer parameters than the full T5 model.

- **DistilRoBERTa:** RoBERTa [5] (Robustly Optimized BERT Pretraining Approach) is another transformer-based language model that is trained on a large corpus of text to generate high-quality embeddings. DistilRoBERTa is a smaller, more compact version of the RoBERTa model that is optimized for speed and efficiency.

- **BART Small:** BART (Bidirectional and Auto-Regressive Transformer) is a transformer-based language model that is designed for sequence-to-sequence tasks such as text summarization. BART Small is a smaller version of the BART model that has fewer parameters than the full BART model.

All of these models were selected for text summarization because they are all based on transformer architectures, which have been shown to be highly effective for generating high-quality text embeddings. Additionally, all of these models are smaller and more efficient than their full-sized counterparts, which makes them well-suited for use in applications where speed and efficiency are important considerations.

Table 1 shows the number of parameters for each model.

Table 1: Number of parameters for each model.

model	parameters
BERT Tiny	$4.4 \cdot 10^6$
t5-Small	$6.01 \cdot 10^7$
DistilRoBERTa	$8.2 \cdot 10^7$
BART Small	$1.5 \cdot 10^7$

Tokenization is carried out using the associated tokenizers of the pretrained models using AutoTokenizer.

After importing the dataset and analysing the available data, some preprocessing operations are performed, using the 'text_strip' function presented here. The various models undergo the fine-tuning operation, performed on the training data of the Xsum dataset [3], which stands for "Extreme Summarisation".

The performance of the various models is then compared using the 'ROUGE' metric, which uses n-grams to determine how far a predicted result deviates from the target result.

Of all of them, DistilRoBERTa appears to be the best performing model, although its performance is much worse than that of Pegasus, the model trained by Google, with which the final comparison is made.

4 Data

The data used in this project is the XSum dataset, which is used for extreme summarization NLP tasks. The dataset is composed by three features: ’document’, ’summary’ and ’id’.

Table 2: Statistics for Texts and Summaries

	Texts	Summaries
Mean Sentence Length	357.68	21.10
Standard Deviation	295.01	5.23
Longest Sentence	29059	86
Text sentences Longer Than 512 Words	24439	-
Summary sentences Longer Than 50 Words	-	58

Here below, two histograms, the first one related to the texts, the second one related to the summaries.

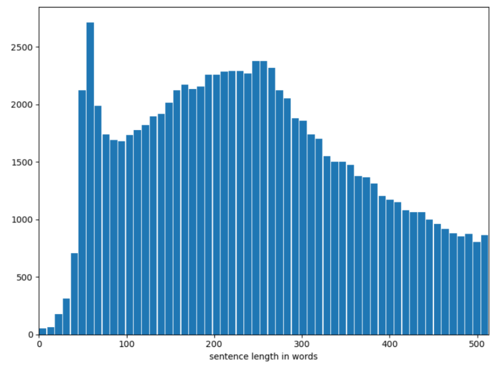

Figure 1: Number of text words (y-axis) with a specific length in words (x-axis)

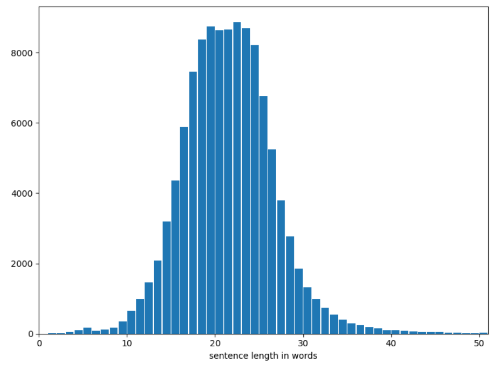

Figure 2: Number of summary words (y-axis) with a specific length in words (x-axis)

To load the dataset, the library dataset is used, which automatically divides the whole dataset into train, validation, and test sets (Table 3):

Table 3: Number of elements in the train, validation and test sets.

split	cardinality
train	204045
validation	11332
test	11334

Half of the data in the training set is kept, and the full dataset is used for validation and testing. The reason behind the choice of keeping half of the total training size is related to the computational time required by the training process. The training elements are randomly selected and saved into a file, so they are consistent across different runs. The code generates a list of indexes to be used for the training set, which is saved to a file. When the code is run again, the same list of indexes is used to ensure consistency.

About preprocessing, a function is used to remove non-alphabetic characters and clean the text. This function removes escape characters, multiple consecutive occurrences of certain characters like _, -, ~, +, and ., as well as various special characters like <, >, (,), |, &, ©, ø, [,], ’, ", ?, ~, *, and !. It also replaces certain patterns of characters with labels, such as replacing "INC" numbers with "INC_NUM", and "CM" and "CHG" numbers with "CM_NUM". Additionally, it removes any single characters hanging between two spaces, replaces URLs with their domain names, and removes multiple spaces. Finally, the text is converted to lowercase.

5 Experimental setup and results

In this project, we conducted experiments to fine-tune four different pre-trained models, namely distilroberta, bert tiny, bart small, and t5-small, on a training set of more than 100000 samples.

- Optimizer: Adam optimizer
- Learning rate: 5e-5
- Epochs: 3
- Max text length: 512
- Max summary length: 32

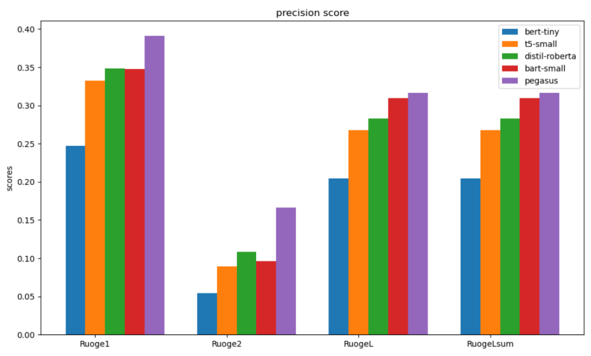

Figure 3: Precision for all the classes for all the models on the test set.

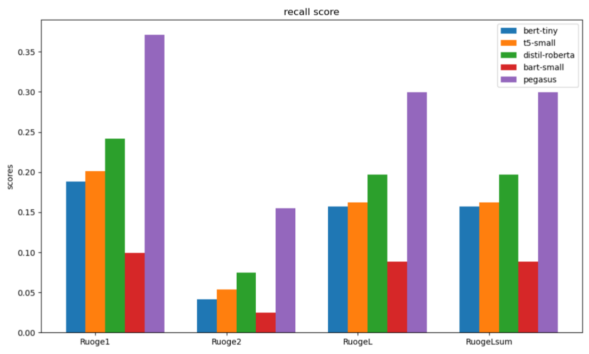

Figure 4: Recall for all the classes for all the models on the test set.

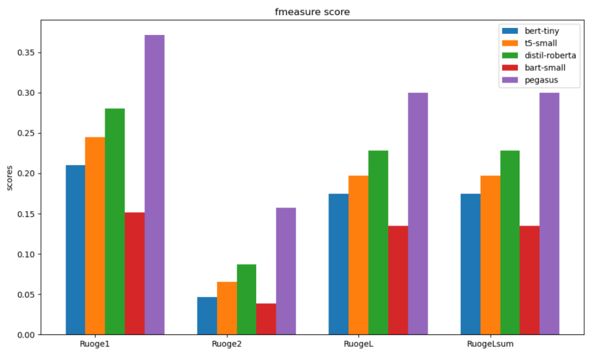

Figure 5: F1-score for all the classes for all the models on the test set.

- Batch size: 16

To evaluate the performance of the models, we measured their performance on validation and test sets and summarized the findings in tables. These tables presented the numerical results for various metrics, such as precision, recall, and F1 score, for each model on both the validation and test sets. Overall, our experiments aimed to identify the most effective pre-trained model and hyperparameters for the task at hand, and we analyzed the results in detail to gain insights into the performance of each model.

Each pre-trained model uses the same loss function during fine-tuning: the cross-entropy loss function. The cross-entropy loss measures the difference between the predicted and actual probability distributions.

Finally, to minimize the influence of randomness on the final performance scores, each pre-trained model was fine-tuned three times using different random seeds: 15, 42, 97. The scores obtained from each of these runs were then averaged to obtain a final score.

To quantify the uncertainty in the final score, we used the standard deviation of the three runs, which was divided by the square root of three. The rouge values, according to the precision, the recall and the F1 measure, can be seen in 3, 4 and 5.

6 Discussion

In this project, we aimed to develop a text summarization model using various transformer-based architectures such as BERT, T5, DistilRoberta, BART, and Pegasus. We trained each model on a dataset of news articles and evaluated their performance using four different ROUGE metrics.

- ROUGE-1: Measures the overlap of unigrams (single words) between the generated summary and the reference summary.
- ROUGE-2: Measures the overlap of bigrams (pairs of adjacent words) between the generated summary and the reference summary.
- ROUGE-L: Measures the longest common subsequence between the generated summary and the reference summary, taking into account word order and sentence length.
- ROUGE-SUM: Calculates the average of the ROUGE-1, ROUGE-2, and ROUGE-L scores. This metric provides an overall evaluation of the quality of the generated summary compared to the reference summary.

Looking at the quantitative results, we can see that Pegasus outperformed all other models in terms of ROUGE scores, achieving the highest scores in all four metrics. This suggests that Pegasus is better at capturing important information and generating summaries that are more similar to the reference summaries. This is not surprising as Pegasus is a state-of-the-art model that has been trained on large amounts of data and has shown to perform well on various summarization tasks. DistilRoberta followed closely, with significant improvements compared to other models, especially in the ROUGE-2 metric. T5 and BERT performed similarly, achieving comparable scores across all four metrics.

An interesting observation is that while all models perform well on the ROUGE-1 metric, which measures the overlap between the model's summary and the reference summary, they all struggle with the ROUGE-2 metric, which requires the model's summary to contain bigrams from the reference summary. This could be due to the inherent difficulty of generating grammatically and semantically coherent bigrams.

In terms of error analysis, we observed that our models sometimes struggled with handling proper names, dates, and locations, often producing incorrect or missing information. In some cases, the generated summaries were not grammatically correct and lacked coherence. Upon examining these examples, we concluded that the models struggled with identifying the most important information to include in the summary and maintaining coherence.

7 Conclusion

Given the limited amount of training data, and considering the big difference of the parameters number between DistilRoberta and the SOTA Pegasus, we noticed that the former one performs pretty well.

About future developments of this work, we could explore techniques such as entity recognition and named entity disambiguation to improve the models' ability to handle proper names and locations. Additionally, we could improve the model's performance by incorporating additional features, such as attention mechanisms and pointer networks, which would allow the model to focus more closely on important information while maintaining coherence. We could also consider fine-tuning the models on a larger and more diverse dataset to further enhance its performance. Overall, our results show promising potential for abstractive text summarization using transformer-based models and highlight areas for future improvement.

8 Links to external resources

XSum Datasets
Text_strip Function

References

[1] Xiaoqi Jiao, Yichun Yin, Lifeng Shang, Xin Jiang, Xiao Chen, Linlin Li, Fang Wang, and Qun Liu. 2019. Tinybert: Distilling bert for natural language understanding. *arXiv preprint arXiv:1909.10351*.

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

[3] Shashi Narayan, Shay B Cohen, and Mirella Lapata. 2018. Don't give me the details, just the summary! topic-aware convolutional neural networks for extreme summarization. *arXiv preprint arXiv:1808.08745*.

[4] Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J Liu. 2020. Exploring the limits of transfer learning with a unified text-to-text transformer. *The Journal of Machine Learning Research*, 21(1):5485–5551.

[5] Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. 2019. Distilbert, a distilled version of bert: smaller, faster, cheaper and lighter. *ArXiv*, abs/1910.01108.