Text Summarization Tool

Using small models as plugin to a large language model

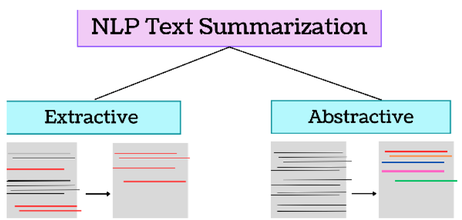
Sritha Hadassah Duddu, M.S., Dacia Rios, UG.,
UC Riverside BCOE, Riverside, CA, USA.
Arman Irani (Advisor), Prof. Esterling (Mentor), Prof. Mariam Salloum (Mentor)

UC RIVERSIDE UNIVERSITY OF CALIFORNIA
Marlan and Rosemary Bourns College of Engineering

Abstract

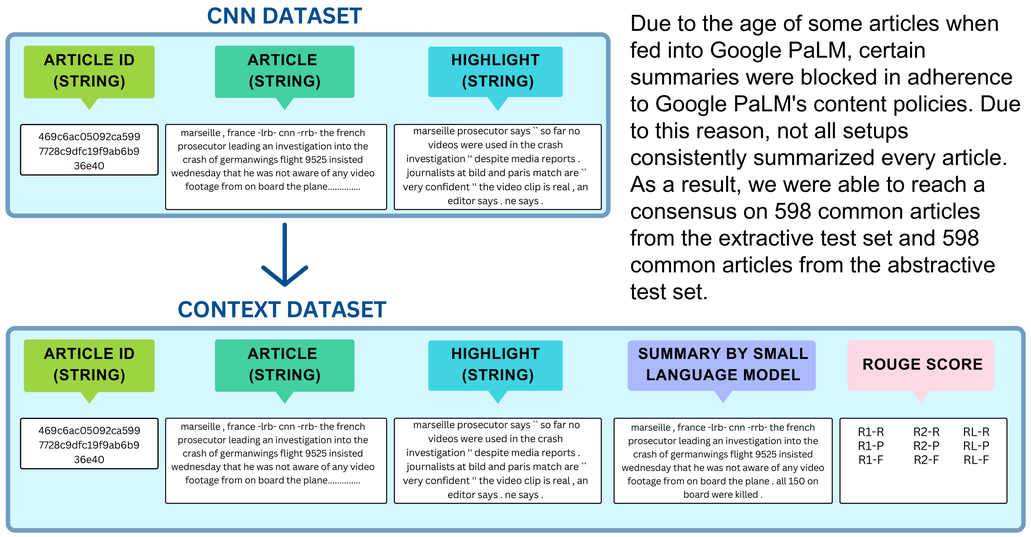
Our project introduces a novel text summarization tool that combines two specialized small language models: FastAbsRL (abstractive summarization) and BanditSum (extractive summarization) integrated as plug-ins within the large language model (LLM) Google PaLM (text-bison@001). Utilizing the CNN/Dailymail dataset as input, we gauged the summarization accuracy using rogue-1, rogue-2, and rogue-L metrics. We curated a new dataset 'Context' which contains the original articles, highlight, model-generated summaries, and their respective rogue scores. Using the Google Cloud Platform (GCP), we fine-tuned the large language model and evaluated its performance in three distinct scenarios: standalone evaluation, summarization, and refinement tasks, all benchmarked using the Rouge metric. Our study validates the hypothesis that fine-tuning a large language model with an equal mix of extractive and abstractive summaries from smaller models creates a hybrid model superior in the summarization task. This hybrid approach combines the best of both summarization techniques, leading to enhanced performance across various text inputs.

Methodology



The two small models that we picked are BanditSum and Fast Abs RL. BanditSum, created by our professor Yue Dong, is a two-layered LSTM and an extractive summarization model. Fast Abs RL, developed by Yen-Chun Chen, and Mohit Bansa. This model uses a bi-directional LSTM-RNN. It uses the advantages of both extractive and abstractive summarization by using a neutral network to select salient sentences then uses an encoder-aligner-decoder to rewrite them.

The dataset that we used is the CNN/Daily Mail dataset which is a widely used benchmark dataset in NLP for tasks related to text summarization. The standard training split contains 287,227 documents, 13,368 documents for validation, and 11,490 for testing. Using the models, we created a new dataset by making them summarize the test set of the CNN dataset. We call this dataset “Context” which has 10704 articles, we split this dataset into unique articles of train and test sets. The train has 4352 (articles of abstractive summaries from Fast Abs RL) + 4352 (articles of extractive summaries from BanditSum) and the test has 2000 articles. We fine-tuned PaLM by sending in the article, summary by small language model, and rouge-1 F score from the train set of "Context", providing it the instruction that a higher rouge score means a better summary. After fine-tuning, we moved on to creating our 3 setups.



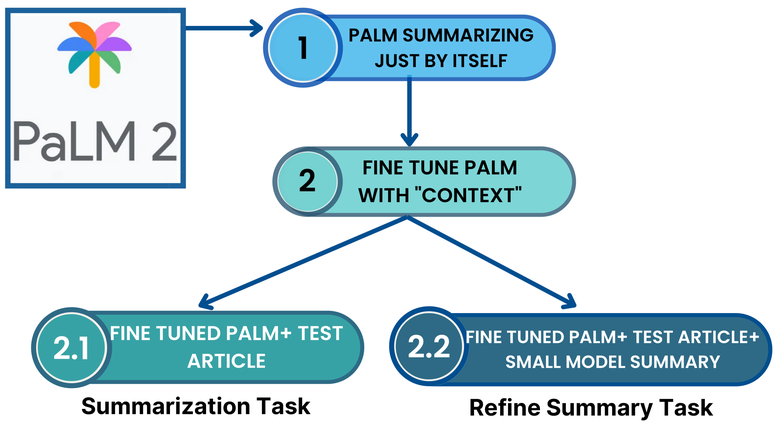
Due to the age of some articles when fed into Google PaLM, certain summaries were blocked in adherence to Google PaLM's content policies. Due to this reason, not all setups consistently summarized every article. As a result, we were able to reach a consensus on 598 common articles from the extractive test set and 598 common articles from the abstractive test set.

Discussion

We conducted an evaluation of the large language model's performance in three setups:

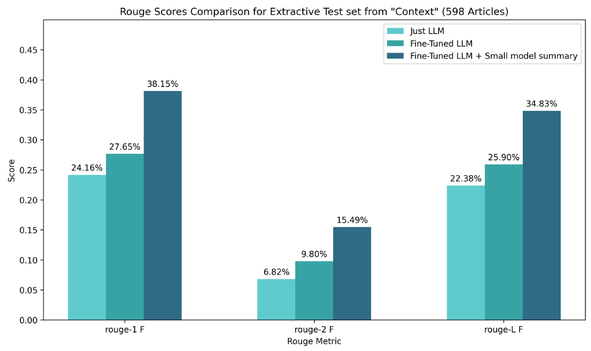
1. **Standalone Evaluation:** The large language model was tested on the "Context" dataset without any fine-tuning.
2. **Summarization Task:** After fine-tuning the model with the "Context" dataset, we assessed its ability to summarize articles without any additional input on the test set of "Context".
3. **Refinement Task:** After fine-tuning with the "Context" dataset, the model was tasked with refining summaries by considering both the article and the summary generated by a smaller model on the test set of "Context".

To assess the model's performance, we utilized the Rouge score metric.



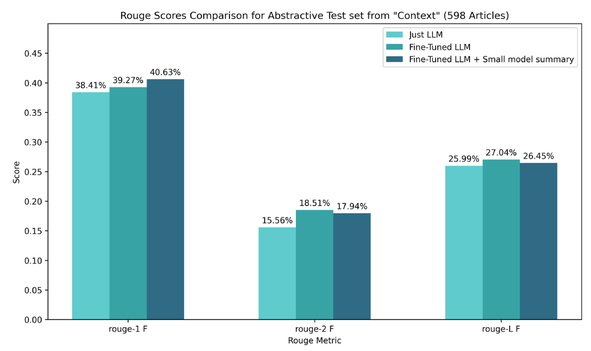
Results

Extractive Test Set of "CONTEXT"



- The results for the extractive test set show that there is a consistent increase in the rouge scores from set up-1, set up-2.1, and set up-2.2 after fine-tuning.
- This indicates that fine-tuning the model on the "CONTEXT" train set resulted in improving the performance of the model in the "summarising task".It is observed that there is a 14.44% increase in the performance of the model when it's fine-tuned
- For the "refine summary" task, the fine-tuned model performed even better than by itself when we also input the summary by BanditSum (extractive model) along with the article and there is a 57.9% increase in performance when it also received the small model summary along with the article as input.

Abstractive Test Set of "CONTEXT"



- The results for the abstractive test set show that there isn’t a significant increase after fine-tuning the model.
- We see an increase in the rouge scores for all the setups for the "summarization task"
- For the "refine summary" task, it is observed that there is an increase in rouge-1 for setup 1 but this trend does not remain consistent for the rouge-2, and rouge-L metrics.
- There is a notable increase in all the rouges for set-up 2.1, which implies that fine-tuning did have some effect on the way it summarized after.

Conclusion

We can conclude our hypothesis that fine-tuning an LLM with the results from small models can indeed have an effect on the way an LLM handles summarization tasks through our experiment. Based on the plots, it has an increasingly positive effect on creating better summaries for the "CONTEXT" test set altogether. However, there's a disparity in performance outcomes for the "refine summary task" (set up 2.1 vs 2.2). While the extractive test set showed positive results, the abstractive set did not follow suit. In the future, we hope to have more time to further evaluate the effects that fine-tuning can have on LLM's capabilities.

References

QR to BanditSum and Fast AbsRL papers + Github