Controllable Text Summarization via Prompt Injection

Data Mining, Text Mining and Big Data Analytics course project 2023/2024

Matteo Belletti: matteo.belletti5@studio.unibo.it
Parsa Mastouri Kashani: parsa.mastouri@studio.unibo.it
Stricescu Razvan Ciprian: razvancipr.stricesc@studio.unibo.it

Project description

Guiding **LLMs** to generate accurate **summaries of specific lengths** and format by using **prompt injection** strategies. More specifically this projects explores and analyses different LLMs, of different parameter size and architecture, to have a comparison and exhibit the best performing one.

We experimented on 14 different models and asked them to generate summaries of either fixed length, word count or number of sentences.

TinyLlama

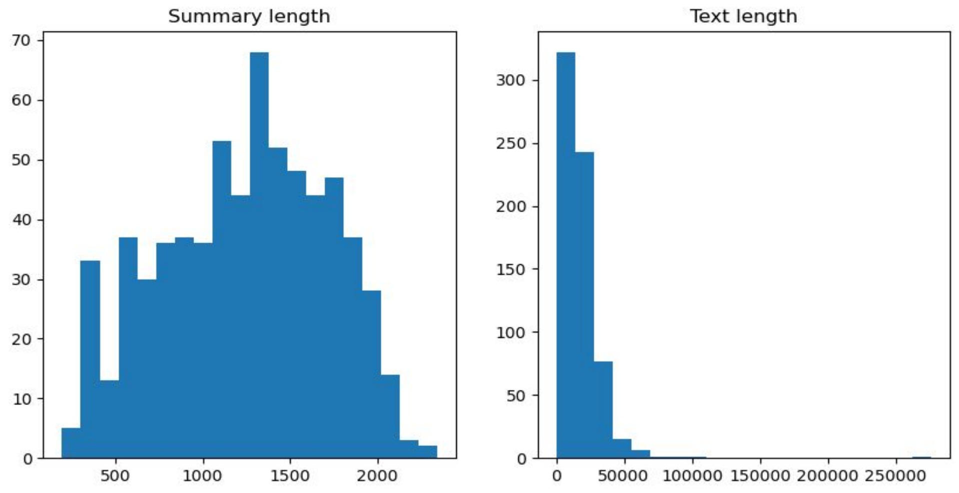
Notus

Mistral

Dataset

We did all the experiments on PubMed, a well-known biomedical dataset for long document summarization, as our goal was to produce different type of summaries for long documents. As the original dataset is massive, we were provided a **representative subset** extracted with stratified random sampling to retain less than 1000 documents.

Dataset length analysis after tokenization



Dataset

Before actually using the data from the dataset we had to execute three main steps:

1.Tokenization

In order to compute metrics such as token length we tokenized the data using the tokenizer given by the NLTK library.

2. Filtering

Depending on model card we filter documents which are too long to present as a prompt, also taking into consideration other parts.

3. Sanitization

We remove data points in which the target summary is longer than the actual document.

Prompts

In order to compare performances we selected 25 different standard prompts belonging to two main macro categories:

- **Reasoning based**: The model has to generate a summary with a variable length dependant on original document length.
- **Instruction based**: The model has to generate a summary with a fixed number of words, number of sentences or number of bullet points.

Example of a reasoning based prompt:

"Generate a summary 10 times shorter for the following text: {Document} …"

Prompts

The first thing we considered is adding a prompt before and after we actually present the document. We think that even a small prompt such as “Summary:” could help the model.

Example of an instruction based prompt:

“Summarise this text in 1 sentence: {Document}. Summary: ”

We also experimented with one-shot inference, a different strategy in which we also present an example summary to the model, almost like a template.

Example of one-shot inference prompt:

“Generate a summary 10 times shorter for the following text: {Example document}. Example summary: {Example summary}.

Generate a summary 10 times shorter for the following text: {Document}. Summary:”

Prompts

After further experiments we noticed some common problems to all the models. In order to try and prevent, or at least improve, these phenomenons we started experimenting with assistant and system prompts in all the previously defined ones. These prompts explain the task at hand and the behavior we want for the model in a direct way, thus strongly favoring an output in a desired manner.

Example of a prompt containing system and assistant prompts:

"System: You are an expert agent in information extraction and summarization.

User: Summarise this text in 1 sentence: {Document}.

Assistant: According to the context information, the summary in English is: "

Unfortunately we couldn't fully experiment with it as inference run times and number of models is way bigger than we could timely handle.

Metrics

To ensure clarity and technical validity when evaluating a generated summary without gold targets for each prompt type, we employed five distinct metrics. Here are the proposed metrics:

Margin of error allowed

- **Percent check:** We check whether the generated summary is a percentage of the original document.
- **Length check:** We check whether the generated summary is of a given length.
- **Sentence check:** We check whether the generated summary has a given number of sentences.
- **Bullet check:** We check whether the generated summary has a given number of bullet points.
- **BARTScore:** We compute the BART score which takes into consideration factualness and semantic similarity with a target summary.

Uses NLTK functions

Experimental Setup

In order to perform the experiments we decided to run all the prompts in a single “run”, meaning that all prompts were generated and fed to the model one by one consecutively.

All our experiments were run on Google Colab notebooks using either a V100 runtime or A100 runtime, depending on availability and model size. Length of the document and prompt had also a significant part on hardware requirements.

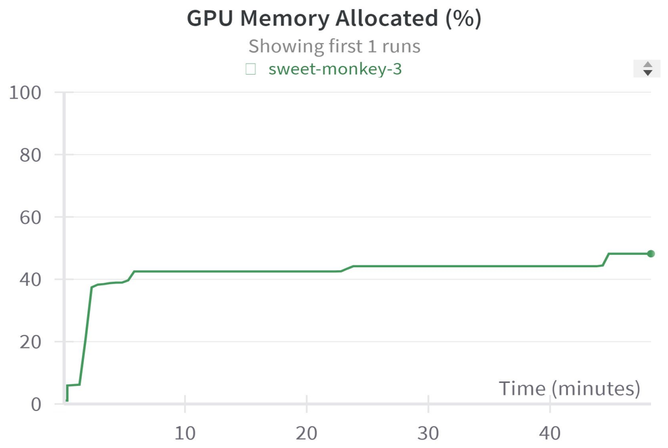
As we were concerned about model parameters size we opted to use a quantized version, more specifically GPTQ, in order to diminish it and further speed-up inference for most models.

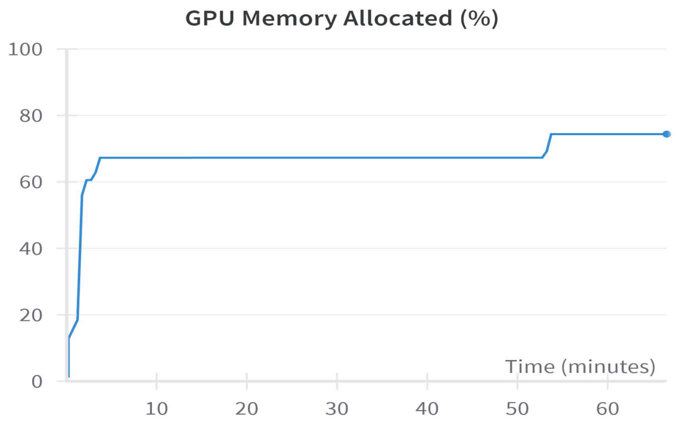
All inference runs took between 35-75 minutes per model and were fully logged on WandB.

Experimental Setup

A100 runtime:
- System RAM 51 GB
- GPU RAM 40 GB
- Disk 201.2 GB

V100 runtime:
- System RAM 51 GB
- GPU RAM 16 GB
- Disk 201.2 GB





Results

On the **best** five models

Zero-shot inference without system or assistant prompt

Model	Checks passed	Average BART score
CapybaraHermes-2.5-Mistral-7B-GPTQ	7	-2.7965
LLAMA-2-13B-GPTQ	6	-2.7483
Starling-LM-7B-alpha-GPTQ	6	-2.8729
WizardLM-1.0-Uncensored-Llama2-13B-GPTQ	5	-3.2340
Orca-2-7B-GPTQ	4	-2.8040

Results

On the **worst** five models

Zero-shot inference without system or assistant prompt

Model	Checks passed	Average BART score
Led-large-book-summary-460M	0	-3.0553
Zephyr-7B-alpha	1	-3.1956
Bigbird-pegasus-large-pubmed	2	-3.5049
SOLAR-10.7B-Instruct-v1.0-uncensored-GPTQ	3	-4.3151
WestLake-7B-v2-GPTQ	3	-3.4265

Conclusions

We believe the scale of the experiment was vast and our limited computation power and the expenses regarding the hardware requirements really limited us in exploring different types of prompts. For future improvements we plan to:

- Research using different temperatures and top_k numbers for the transformer models.
- Avoid using quantization which cripples the models to some extent but grants faster inference.
- Add one-shot and multiple shot inference for all models.