Table of contents

[Project overview 2](#_Toc193539630)

[Tools, methodologies and technologies used 3](#_Toc193539631)

[Installation instructions 4](#_Toc193539632)

[Usage guide 4](#_Toc193539633)

[Lessons learned 4](#_Toc193539634)

[References 4](#_Toc193539635)

# Project overview

This project was created a coursework for Tampere University’s course “COMP.CS.530: Capstone project on LLM fine-tuning”. The project is a “search query optimization” application that on high level takes users input (a search query), optimizes it using an SLM/LLM (SLM = Small Language Model), and makes a search towards an article index in Elastic Search with both the original and optimized queries returning the results. On top of building the application, the project included fine-tuning an existing t5-small language model for query optimization task using Low-Rank Adaptation (LoRa) method.

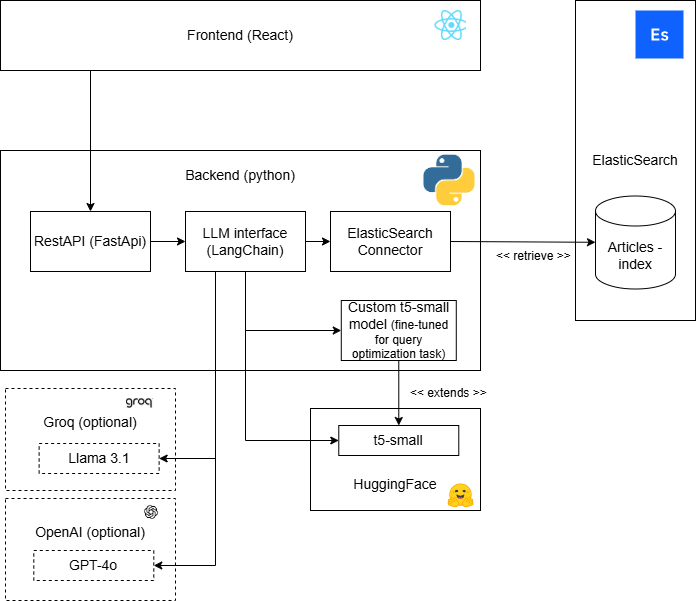


Diagram 1: Description of the components within the application

# Tools, methodologies and technologies used

Frontend of the application is generic react application, which consists of couple of UI elements. The frontend’s responsibility is to allow users to interact with the system and display the results. Frontend generates API calls towards the backend part of the application.

The backend is python application that consists of 3 main components: RestAPI, LLM interface and ElasticSearch connector. RestAPI includes two endpoints for frontend: GET endpoint for listing available AI-models, and POST endpoint for optimizing and searching. LLM interface is built using LangChain. When an “optimize\_and\_search” call has been made by user, two chains are constructed depending on the chosen model. The first chain includes inferencing with an SLM/LLM and then passing the output to elastic search runnable, the second chain just takes the original query and passes it to elastic search runnable. The two chains are run in parallel.

User is able to optimize their queries by using commercial options such as Groq (llama 3.1 model) or OpenAI (GPT-4o model). However the notable component that can be seen in the diagram 1 is the “Custom t5-small model”. This model was created within this project by fine-tuning the adapter layers of t5-small model[1]. The model was fine-tuned by using a dataset of 60000 examples, where each example was a human made query towards Bing -search engine with optimized version of that query. The human made examples were part of MS MARCO Queries(08/11/2020)[2] dataset where as the optimized versions were synthetically created using existing LLM’s such as Llama-3.3-70b and OpenAI’s GPT-4o. The dataset used for fine-tuning can be found from ./datasets folder within this repository. Fine-tuning was done in Kaggle using Low-Rank Adapter (LoRa) fine-tuning with 5 epochs and learning rate of 2e-4. The notebook used for training and test inferencing can be found from ./notebooks folder within this repository.

Since the fine-tuned custom t5-small only includes adapter layers of t5-small model, the original t5-small model is required to do inferencing towards the fine-tuned model. This is why the application loads the t5-small model from HuggingFace. User is also able to use the original t5-small model for query optimization when using the application, however due to the nature of the original t5-small model not being optimized for query optimization tasks, the model acts very poorly.

ElasticSearch connector retrieves data from “articles” index within ElasticSearch. The articles index includes 1000 articles that were scraped from Wikipedia randomly. The articles are included in the repository in .json format. When the application is started (using docker compose), the backend uploads all articles from the “./backend/articles” folder to ElasticSearch. As of this moment, there is no easy way for user to upload new articles, or any other data to ElasticSearch.

# Installation instructions

See Readme.MD from the repository’s root for installation instructions

# Usage guide

See Readme.MD from the repository’s root for usage guide

# References

[1] HuggingFace; T5-small language model <https://huggingface.co/google-t5/t5-small>

[2] Microsoft; MS MARCO; Queries(08/11/2020) dataset <https://microsoft.github.io/msmarco/>