Tampere University

Query analysis of Large Language Models Using LangChain for Multimedia Application

Group Name: Jannet

Janne Taskinen

A report submitted in partial fulfillment of the requirements for the

course

COMP.CS.530-2024-2025-1: Fine-tuning Large Language Models (LLM)

March 24, 2025

Abstract

This project was created a coursework for Tampere University's course "COMP.CS.530: Capstone project on LLM fine-tuning". The project resulted in a creation of "search query optimization" tool, that optimizes users search intent using fine-tuned t5-small language model.

The project included steps such as collecting search query data, fine-tuning t5-small language model, populating ElasticSearch – search index and building an application with user interface on top of the fine-tuned t5-small model and the ElasticSearch index.

The key features of the applications include being able to optimize a search query and compare the search results between the original search query and the optimized one. On top of using the custom t5-small model, user is able to use API-based model through Groq or OpenAI.

While the fine-tuned t5-small language model outperforms the base t5-small model in query optimization task, it was noted that models such as Llama 3.3 70B outperform the fine-tuned one. Therefore using the fine-tuned t5-small model could be considered to be used for query optimization task in very resource restricted environments.

Table of contents

List of Figures

List of Tables

Introduction

Background

When humans interact with search engines, they tend to write search queries non-optimally. The search queries can be too long, miss important keywords, or emphasize words that are not meaningful from the search engines point of view. This may result in non-optimal search results.

This project focuses on the issue of clarifying search intent of human-written search queries utilizing AI. While optimizing search queries is quite trivial task for commercial large language models (LLM's) such as GPT 4o, the project focuses on achieving better search intent results utilizing small language models (SML's) that can be easily run locally in resource constrained environments. The target is to demonstrate that this type of task can be achieved by a smaller model.

Objectives

The project included two main parts. Firstly, a SML was trained for search query optimization task. This part included finding suitable base model, finding suitable training data, training and evaluating the model performance. Secondly, a software stack was implemented on top of the model. This part included implanting user interface and backend. On top of that both the trained model and suitable search engine was integrated into the application.

Scope

Some limitations were set regarding the project. Firstly, it was decided that only one model would be pre-trained. To allow comparing results between models, an option was made for users to also try the optimization task with API-based existing models including Llama 3.3 and GPT 4o. Secondly, it was decided that it was good enough for the application to run locally and therefore no CI/CD pipelines were implemented that would deploy the application to external server or cloud platform. Thirdly, it was decided that for demonstration purposes including one type of data (Wikipedia articles) was good enough, so that not too much time would have to be spent on building the database for the search engine.

System Overview

System Description

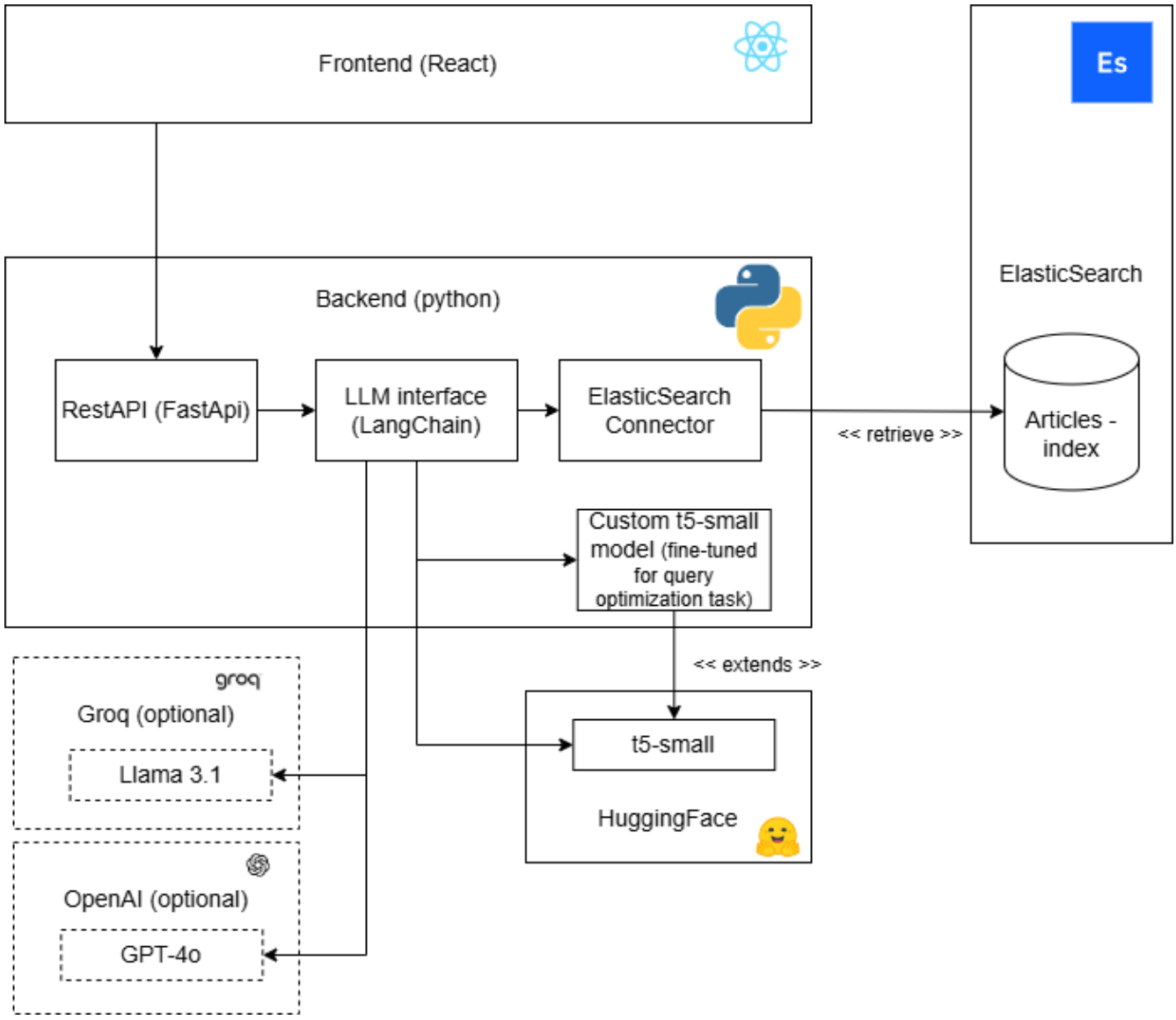

Figure 1: Description of the components within the application

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.

Query optimization is done by default using “custom t5-small” model, which is fine-tuned version of t5-small[1] model. The model was fine-tuned within this project. Optionally user choose to use Llama 3.1 model (provided by Groq API) or GPT-4o model (provided by Open AI). User has to include the API keys for the services if they wish to use these models. 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. The reason for including t5-small model as an option is to demonstrate the power of fine-tuning.

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.

Key Features

User is able to write a search query, chose a model that optimizes the search query and perform the search. Like mentioned in the “System Description”, the search query is executed towards an ElasticSearch index, which includes 1000 wikipedia articles. User is able to compare the search results between the optimized search query, and original search query. As of this moment, there is no easy way for user to upload new articles, or any other data to ElasticSearch.

Development Process

Tools and technologies used

T5-small[1] was chosen as base model for the SML. There were two main reasons for choosing this model. Firstly, it has been pre-trained for sequence-to-sequence tasks, which ultimately is what search query optimization is. Secondly, it was small enough to be fine-tuned efficiently considering the amount of available computational training resources and time for the project. Options such as Bert and Llama were looked into, but were rejected due to either their size or intended use case.

The model was fine-tuned by using a dataset of 60 000 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 the project's GitHub repository[3]. Fine-tuning was done in Kaggle using Low-Rank Adapter (LoRa) fine-tuning with 5 epochs and learning rate of 2e-4. Two GPU T4's were used as accelerators. The actual training took around two hours. The notebook used for training and test inferencing can be found from ./notebooks folder within the repository[3].

For the application part, the tools and technologies used are listed in "System description" -chapter.

Implementation details

Implementation details can be found from "System description" -chapter and also by inspecting the GitHub repository[3].

Testing and evaluation

Testing Methods

There was no formal number-based testing executed towards the results of the fine-tuned model using technologies such as RougeScore. Testing of the model was done manually.

Results

Table below presents results from tests comparing typical human-made search query towards the optimized version, where the optimization is performed by different models.

Query #	Search query	T5-small (base model)	T5-custom (fine-tuned model)	Llama 3.3 70b versatile
1	how can I find instructions to play quitar?	Optimieren wie kann ich die Anleitungen für die Spiele, wie Optimieren optimieren, wie können ich die Anleitungen für die Spiele spielen quitar?	find instructions to play quitar	guitar playing instructions
2	how to get to new york from la	optimize how to optimize how to get to new york from la	new york from la	Los Angeles to New York travel options
3	price of coconut oil in europe	optimize price of coconut oil in europe	coconut oil price europe	coconut oil prices europe
4	how is crime and punishment defined	Optimize how is defined crime and punishment	crime and punishment definition	crime and punishment definition
5	How to lose belly fat fast?	Optimieren Wie lose belly fat fast?	lose belly fat fast	fast belly fat loss
6	Difference between machine	optim optim Difference between machine	machine learning deep learning difference	machine learning vs deep learning

	learning and deep learning	learning and deep learning		
7	What is the James Webb Telescope seeing now?	optim optim optim Wie optim optim optim Wie sieht jetzt das James Webb Telescope?	James Webb Telescope viewing date	James Webb Space Telescope current observations
8	Best TV shows on Netflix right now	Optimieren Best TV shows on Netflix right now.	best TV shows on Netflix right now	Top Netflix TV shows currently available
9	What does check engine light mean?	optimize Optimiz Optimiz	check engine light meaning	check engine light meaning
10	Best skincare products for acne-prone skin	optimize Best skincare products for acne-prone skin	best skincare products acne-prone skin	acne prone skin care products

Table 1: Comparison of models in search query optimization task

Based on the table 1, following conclusions can be made. The t5-small language model performs unacceptably poorly regarding query optimization task. The results can be classified as hallucinations since the model has not been trained for this task. We can see that fine-tuning the model with 60 000 examples allowed the model to understand that this is an optimization tasks and the results are significantly better than with the base model.

While the t5-custom model outperforms the t5-small model, we can clearly see that the results are not perfect by any means. In most examples, what the model achieved was removing unnecessary words such as “how can”, “how to”, “how is”, and “what is” (Queries 1, 2, 4, 5, 7, 9). It also can rephrase the query by moving words (Queries 3, 6). While it can also rewrite words to some extend (Queries 4, 7, 9), it fails to fully understand the query and do meaningful optimizations compared to Llama 3.3. For example in query 7, the custom t5-small model replaced “seeing now” with “viewing date”, fully misunderstanding the intent.

Lessons learned

Challenges Encountered

Finding proper datasets with examples of human-made search queries and optimized versions turned out to be a challenge. I settled for creating the optimized versions synthetically using both Llama 3.3 model and GPT-4o.

Finding proper language model to fine-tune turned out to be bit challenging due to the resource constraints. There were couple of hick-ups in the fine-tuning process itself, mainly related to the structure of the t5-small model. For example deciding, which layers to fine-tune and finding out that the first token for each “decored_input_id” passed to the model within the training dataset should be a padding token.

Building the application was quite straightforward, though it required getting familiar with technologies such as LangChain and ElasticSearch.

Solutions and Strategies

Synthetically created decently sized dataset with LoRa – fine-tuning method turned out to be good compromise between time and resource constraints versus good enough results.

Key Takeaways

While the objectives and goals of the project were fulfilled to some extent – in my opinion, there are certain areas of improvement within the process, if one were to implement this project again. Firstly, more time and effort should be used to find a proper dataset. Since the model is only as good as the data is, getting proper data is crucial and can make or break the project. Secondly, one should really pay attention to the requirements of the solution and find out what are the most important aspects of it. For example, is it better to have better performing model over clean user interface. Would more basic search engine or database be good enough for demonstration purposes over elastic search, if it meant that more time and effort could be spent on model training? In hindsight, I believe I could have also paid more attention to the data within the search index, since demonstrating with random data gives random results. However, the reasoning for using random data was to demonstrate that a simple change within the search query can affect the results quite dramatically.

Conclusion and Future work

Summary

The application, that was formed as a result of this project should be considered as proof of concept (PoC) that it is possible with quite small effort to train a small language model for specific task such as query optimization. However, it is worth noting that many large language models can outperform fine-tuned small models so trade-offs between using an existing LLM and fine-tuning a model should be considered carefully.

Future Enhancements

There are many enhancements that the solution could benefit from. Firstly the custom t5-small model could be fine-tuned further, perhaps with different dataset or using completely different base model (eg. T5-large, llama 1.1B). Secondly, it should be made possible for user to upload information to ElasticSearch.

The application naturally lacks features such as user account support. One improvement idea could be to allow users to register to application and also restrict the some of the data for certain users or user groups. The application could also benefit from “Bring your own model” -type of feature, where user is able to define the model endpoint and possible system prompt in the UI, to evaluate differences between models.

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/

[3] Github; repository for Capstone project https://github.com/taskijanne/capstone/tree/main