FAKULTA
APLIKOVANÝCH VĚD
ZÁPADOČESKÉ
UNIVERZITY
V PLZNI

KIV-OP

Semester Project

`StackExchange Siamese Posts Search`

Daniel Schnurpfeil (dschnurp@students.zcu.cz)

August 22, 2023

Contents

1 Project Assignment **2**

2 Introduction **2**

3 Analysis **3**

4 Indexing StackExchange data with Faiss **3**
4.1 MQDD Encoder & Faiss Indexing 4
4.2 Large Indexes for Big Data 4

5 Web Application extending **4**
5.1 MQDD and indexed data with Faiss 4
5.2 Docker 5
5.3 Automatic question upload to Stack Exchange 5

6 Tests **5**

7 Conclusion **6**
7.1 Future Work 6

1 Project Assignment

Explore the model provided, look for similar questions and learn how to input and output. Index the offline Stack Overflow database in a suitable search engine (Elasticsearch is recommended). Create a web application that allows you to submit a question and accompanying source code. The application will retrieve the most similar question for this input from the off-line Stack Overflow database. Finally, it will allow the question to be automatically entered into the Stack Overflow portal.

2 Introduction

The aim of this project is develop web application for duplicity search on Stack Overflow (or random Stack Exchange) portal. Stack Overflow and the Stack Exchange network help people find the answers they need, when they need them. Comprising 173 Q&A communities, including Stack Overflow, over 100 million people visit every month to ask questions, learn, and share technical knowledge.[1]

The web application will be designed to address the issue of duplicated questions or answers that may exist within the Stack Overflow or Stack Exchange network. Duplicity in this context refers to questions or answers that are similar or identical to each other. Such duplication can result in redundant information, confusion among users, and difficulties in locating the most relevant and accurate answers.

To achieve the project's objective, the web application will employ various techniques and technologies, including natural language processing (NLP) and information retrieval. The application will utilize machine learning algorithms and models to identify duplicate questions or answers based on their textual content. It will analyze the similarities and patterns within the questions and answers, considering factors such as the use of keywords, sentence structure, and semantic meaning.

By developing this web application, users of Stack Overflow or Stack Exchange will have a powerful tool to assist them in finding the most relevant and non-duplicate answers to their questions. It will help reduce redundancy, improve the quality of information, and enhance the overall user experience within the platform.

[1] from: https://stackexchange.com/about

3 Analysis

At first, we need to describe which technologies are needed to solve this issue. The application itself needs a database where the possible duplicate posts are located. That is given in the assignment. It is Elasticsearch. Then we need some mechanism to find the duplicate questions. That is provided too. On huggingface.co/... MQDD... there is fine tuned model. So we will use it. According to paper MQDD: Pre-training of Multimodal Question Duplicity Detection for Software Engineering Domain the fine tuned model (Figure 1) consists two Longformer encoders and classification head.

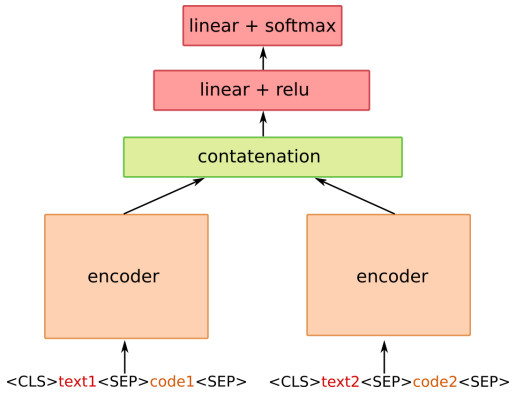

Figure 1: MQDD architecture from huggingface.co/UWB-AIR/MQDD-duplicates

Finally the data, we can find them on archive.org website, which is American not only internet archive.

4 Indexing StackExchange data with Faiss

For searching the duplicity questions, it is much faster using only encoders and use Faiss library accroding the MQDD: Pre-training of Multimodal Question Duplicity Detection for Software Engineering Domain paper. Faiss is a library for efficient similarity search and clustering of dense vectors.[2] Faiss is particularly useful in scenarios where there is a need to perform similarity-based operations on high-dimensional data.

The primary goal of Faiss is to accelerate similarity search tasks, such as nearest neighbor search and similarity ranking, by leveraging various indexing structures and optimization techniques. These operations involve finding the vectors that are most similar to a given query vector, based on a chosen similarity metric.

Faiss supports different similarity metrics, including Euclidean distance, cosine similarity, and inner product similarity. The library implements several indexing methods, such as IVF (Inverted File), HNSW (Hierarchical Navigable Small World), and PQ (Product Quantization), to efficiently organize and search through the vectors.

[2]https://github.com/facebookresearch/faiss#faiss

Faiss offers GPU acceleration, allowing for even faster similarity search on compatible hardware. By leveraging the parallel processing power of GPUs, Faiss significantly reduces the search time for large-scale datasets.

4.1 MQDD Encoder & Faiss Indexing

One of the aims of this project is implement scripts that index data, which are the posts bodies of question for searching the duplicates, with IndexFlatL2. IndexFlatL2 is prepared for exact L2 search, so it could be suitable for similarity search among posted question bodies. The script takes the path where posted question bodies are located, then sanitizes them from malicious HTML tags and encodes them with MQDD Longformer. Encoding takes quite a lot of time, so CUDA (with PyTorch) is supported. Once the posted question bodies are encoded, the indexing process with Faiss begins. Finally, the index is saved to disk. User reference can be found on the project GitHub site.

4.2 Large Indexes for Big Data

The primary objective of the script is to detect duplicity within Big Data, specifically targeting datasets like Stack Overflow, where the Posts.xml file can be as large as 100 GB. To efficiently handle such massive datasets, the script incorporates a feature to create parts of the overall index. This approach allows for the systematic processing of the XML file in smaller, manageable partitions. By employing a streaming method, the script reads the XML file in chunks or lines, thereby preventing memory overflow issues. The partitioning strategy facilitates the creation of independent indexes for each partition, enabling parallel processing and optimized resource utilization. The script's implementation takes into account the specific characteristics of large XML files, such as Stack Overflow, and ensures efficient parsing, indexing, and duplicate detection.

5 Web Application extending

There is already an implemented web application that visualizes indexed data from Elastic search by user-defined query via web interface. The app source codes can be found on Jan Pašek's GitHub.

5.1 MQDD and indexed data with Faiss

The main issue is to connect Faiss indexed data to be able to perform a search for duplicates when the question post detail is opened and show the first few potential links of duplicity question posts. The first approach is to encode the question again with an encoder using Hugging Face transformers and PyTorch, but it needs to import a large amount of library data and time performance, including the web application start. To avoid this issue, there is an option to get already encoded

post by ID in the index and search for nearest neighbors in the state space of clusters.

5.2 Docker

Application is fully dockerized. That means, there is needed only to place Faiss index and data for Elastic Search with read-me manual on this GitHub site.

5.3 Automatic question upload to Stack Exchange

There is implemented a form to automatic upload to Stack Exchange site eg. (LaTeX, Signal Processing, Gamedev, Tor, English Language & Usage, Mathematics...), which need to provide authentication of web application. So there is also implemented generic link to login page based on indexed data.

6 Tests

There are also implemented cases to test how the indexed data with Faiss are usable to perform similarity search. Test data are from Stack Overflow Duplicity Dataset, which was created along with the MQDD. In this data-set is used only significant classes: duplicate (108 posts) and different (289 posts).

	Positive	Negative	Total
Positive	20	2	22
Negative	88	287	375
Total	108	289	397

Table 1: Confusion Matrix of Faiss `IndexFlatL2`

Test results above show that Faiss index has extremely good precision, which means if the index tells it is a duplicate, then that is a duplicate. However, it found not many duplicates, and there are used top 5 results from similarity search so that reduces the precision. Computed accuracy (≈ 0.75) of Faiss index is similar like in MQDD paper.

	Positive	Negative	Total
Positive	27	52	79
Negative	81	237	318
Total	108	289	397

Table 2: Confusion matrix of original MQDD

The original **MQDD** test took **4 minutes and 34 seconds** to run. In this test were encoded tuples of questions and computed by classification head of MQDD.

The Faiss `IndexFlatL2` test took **2 minutes and 6 seconds**. In Faiss test was performed search in index. Both algorithms were run on the same data and on an M1 processor. Details can be found at Faiss test and MQDD test.

7 Conclusion

In conclusion, this assignment aimed to explore a provided model, learn about input and output mechanisms, and utilize an offline Stack Overflow database to create a web application. The key objectives were to index the database using a suitable search engine, develop a user-friendly interface that enables users to submit questions and source code, retrieve the most similar question from the offline database based on the input, and automate the process of entering the question into the Stack Overflow portal.

Despite the fact that the base of web application was already implemented, it still took time to learn about neural MQDD network, Elasticsearch and extend application by duplicity search with Faiss index, Docker support and finally automatic entering the question into the Stack Overflow portal.

Full datasets from gamedev, digital signal processing and tor from the Stack Exchange portal were indexed as an extension of the assignment.[3]

7.1 Future Work

Finally, there is an option to get better results with Faiss index by improving MQDD encoder.

[3]Entire text was corrected by AI(GPT4) for spell checks and sememantics