Study Buddy - Chatbots as Lecture Companions Using LLMs and Retrieval Augmented Generation

Students



Lucien Zimmermann



Florian Rohrer

Introduction: The emergence of large language models (LLM) changes the way we search for information. LLMs allow us to ask questions directly and receive answers in natural language. However, the knowledge of LLMs is limited to the information they have been trained on and is therefore often outdated. This limitation can be overcome by using the retrieval augmented generation (RAG) technique. This technique combines the user's prompt with contextual information from a custom knowledge base before asking the LLM to generate an answer. The technique relies on semantic search using embeddings to find relevant content related to the user's prompt in the knowledge base. RAG significantly improves the quality of the answers received from the LLM, especially when specific knowledge beyond what the LLM has been trained on is required.

Objective: The goal of this project was to implement a chatbot in Python and React that uses the RAG technique to answer a student's questions about lecture-related content, such as PDF lecture notes. In addition to providing correct answers, the bot should also list the sources used to generate the answers, allowing the student to verify the answer.

Result: A chatbot was implemented using opensource components. The focus was on the LLama2 LLM family and LLamaIndex, a data framework in Python for connecting LLM. The chatbot was tested using slides from the C++ and OOP lectures at OST. We found that the RAG technique works well for answering questions based on text-based notes. However, we encountered difficulties in retrieving relevant context when dealing with bullet points and images in lecture slides, resulting in the LLM generating inaccurate answers. To reduce the impact of these limitations, we conducted tests to evaluate an embedding model that best fits our use case. During our testing we could not find any model, including Llama2, that performed adequate with languages other than English. This problem can only be addressed by fine tuning a model. So we focused our evaluation on English texts. We have also provided a guide for lecturers and students on how to use chatbots like this one efficiently.

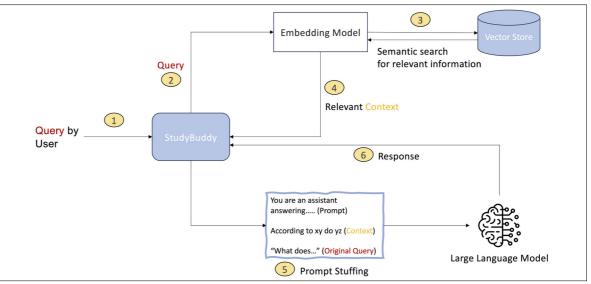
The web app is build using Python and React and is designed with simplicity in mind; it provides a simple chat interface Own presentment

ĥ	Study Buddy	
	Based on the provided context information, objects that are stored on the heap in C++ are:	
	Primitive values (int, foat, boolean) - These are not stored on the heap as they are primitive types and do not require any allocation of memory.	
	Dijects created using new operator - C++ allocates memory for these objects on definition, meaning no explicit heap memory is needed, and there is no indirection or space overhead. An example of such an object is a storechose an explore	
	Heap-allocated objects created using factory functions - If you need to support sur-time polymorphic container contents or class members that cannot be passed as references, you can use factory functions minuting that <u>involved appr</u> for heap-allocated objects. Howeve, it's important to check if alternatives are value before resuring to this approach.	
	Therefore, the answer to the query is: objects created using new operator or through factory functions are stored on the heap in C++.	
ь	(0, pp(,licture_slides-introduction.pdf (mp: 1)) (1), pp(,licture_slides-heap_memory_management.pdf (npr: 1))	
	What kind of printive types exist in C++7	8
	Based on the provided context information, we can identify that there are two primitive types mentioned in the lecture slides:	
	This is a signed integer type, which means it can hold both positive and negative values within a certain range (depending on the platform).	
	This is a floating-point number type, which can represent a wide range of values with decimal points.	
	Therefore, the answer to the query "What kind of primitive types exist in C++?" in: [10] and [7303].	
5	(0, cpi_lecture_sldee-introduction.pdf (row.l)) (0, cpi_lecture_sldee-namespaces_enums.pdf (row.l))	
٢		÷
		8

Answers provided by StudyBuddy are written using natural language and always contain a direct reference to the source Own presentment



09_cpl_lecture_slides-function_templates.pdf Page: 9) 07_cpl_lecture_slides-standard_containers.pdf Page



Visual representation of retrieval augmented generation. The query is enriched with relevant context and sent to the LLM Own presentment

Advisor Prof. Stefan F. Keller

Subject Area Software, Application Design

