Posted on Retrieval augmented generation (RAG) is a popular way to use current or proprietary data with Large Language Models (LLMs). There are many articles describing how to perform RAG. Typically, they involve encoding data as vectors and storing the vectors in a database. The database is queried and the data is placed in the context where it is tokenized (converted to vectors) along with the prompt to the LLM. At its simplest, RAG is placing data in the prompt for the LLM to process. For all practical purposes, the Internet is the database for the world and we can query it with search engines that have methods for returning relevant results. Sound familiar? We can use search to power a RAG application. For this example, we'll use DuckDuckGo for search, Langchain to retrieve web pages and process the data, and your choice of an Ollama with an open-source LLM or a LLM service like OpenAI. For the impatient, code (https://github.com/spara/RAG_step-by-step/) To get started, import the packages into your environment (https://docs.python.org/3/library/venv.html) . Let's dig into the code! Querying DuckDuckGo, retrieving the web pages, and formatting for insertion into the prompt are done by these three function. The ddg_search function queries DuckDuckGo. The get_page function uses Langchain's document loader to retrieve the pages from the search, extracts only the text between p HTML tags with the BeautifulSoupTransformer , and returns a list of Langchain documents (https://api.python.langchain.com/en/latest/documents/langchain_core.documents.base.Document.html) . The ddg_search function extracts the text from the documents and truncates them to ensure they fit within the context window of the LLM. Recent LLMs have larger context windows, and you can change the amount truncated and where to truncate by changing the values. For example, you may want to capture the end of the text which includes conclusions and summaries. The processed text from each document is returned as a list. The following section creates a prompt. Currently, there is no standard prompt format and each LLM implements it's own prompt format. The following section demonstrates how to create a prompt for a llama2 or llama3 LLM and an OpenAI LLM. Note how the prompt construction differs. Creating a completion (or response) for each LLM also differs. The application uses Streamlit to search DuckDuckGo, send the results to the LLM, and displays the completion. This is an example query critiquing Taylor Switft's new Tortured Poets Department album using OpenAI's GPT3.5. Ouch! A bit harsh. Can this be improved upon? Definitely! Most search engines have operators that support searching a specific site or excluding sites. For example, a search for Kubernete's Container Network Interfaces (CNI) can be limited to just kubernetes.io instead of all the other sites that address CNI. The BeautifulSoupTransformer supports extracting or excluding text by tag and the truncate function can be expanded to extract text from certain parts such as the end where conclusions and summaries are located. You can also change the LLM from a general chat model to an instruct model and use it as a coding assistant. Using web search with an LLM can help produce better search results and summaries. Be sure to check out the code long Github (https://github.com/spara/RAG_step-by-step/) . Templates let you quickly answer FAQs or store snippets for re-use. Are you sure you want to hide this comment? It will become hidden in your post, but will still be visible via the comment's permalink (#) . Hide child comments as well Confirm For further actions, you may consider blocking this person and/or reporting abuse (/report-abuse) Kavya Sahai - Jan 14 Jagroop Singh - Jan 14 Kudzai Murimi - Jan 14 Spencer Marx - Jan 14 Thank you to our Diamond Sponsor Neon (https://neon.tech/) for supporting our community. DEV Community (/) â€” A constructive and inclusive social network for software developers. With you every step of your journey. Built on Forem (https://www.forem.com) â€” the open source (https://dev.to/t/opensource) software that powers DEV (https://dev.to) and other inclusive communities. Made with love and Ruby on Rails (https://dev.to/t/rails) . DEV Community Â© 2016 - 2025. We're a place where coders share, stay up-to-date and grow their careers.