LLM Repurpose Experiment Report

GitHub Repository

GitHub Link

1 Experiment with English prompts and Gemini API

Implementation

I used Gemini-1.5-flash with API created from Google's AI studio.
I tested with this piece of paper "Rerankers and Two-Stage Retrieval" on 3 tasks: write newsletter,summary and X threads. It briefly addressed a limitation of LLM, proposed and explained a method to solve it - RAG. However, RAG contains its own limitations, the author introduced rerankers to tackle one of them. He also mentioned some representative rerankers and his one.

Result

Newsletter result:

Boosting RAG Performance with Rerankers

Retrieval Augmented Generation (RAG) often underperforms expectations. This week, I explored a simple yet powerful solution: rerankers. RAG typically uses vector search to find relevant documents, but this can miss crucial information. Rerankers act as a second stage, re-ordering retrieved documents based on their relevance to the *specific query*, significantly improving the quality of information fed to the LLM. This two-stage approach (fast initial retrieval, precise reranking) maximizes both retrieval and LLM recall, leading to more accurate and helpful responses. Think of it as refining the initial search results for optimal LLM input.

[Image suggestion: A simple diagram illustrating the two-stage retrieval process: Vector DB (fast) → Reranker (accurate) → LLM. Arrows should show the flow of data.]

The implementation involves using a bi-encoder for initial retrieval (e.g., multilingual-e5-large) and a reranker model (e.g., ‘bge-reranker-v2-m3‘ from Pinecone) for re-ordering. While rerankers are slower than initial retrieval, the improved accuracy far outweighs the slight performance hit, especially when dealing with large datasets.

[Image suggestion: A screenshot showcasing a comparison of results before and after reranking. Highlight how reranking brings more relevant information to the top.]

This technique dramatically improves RAG's recall, ensuring the LLM receives the most pertinent information for generating a superior response. It's a relatively straightforward method to significantly enhance the effectiveness of your RAG pipeline.

Summary result:

Retrieval Augmented Generation (RAG) often underperforms expectations due to limitations in retrieving and utilizing relevant information.

Vector search, while fast, loses information, causing relevant documents to be ranked lower than less relevant ones.

Increasing the number of retrieved documents ($\texttt{top}_k$) improves recall but exceeds LLMs' context window limits.

"Context stuffing"(exceeding the context window) harms LLM recall and performance.

The solution: use rerankers to maximize retrieval recall (retrieve many documents) and then maximize LLM recall (select the most relevant documents for the LLM).

Rerankers (cross-encoders) score query-document pairs, reordering documents by relevance. This is a two-stage process: fast retrieval followed by slower, more accurate reranking.

Rerankers are slower than bi-encoders but far more accurate because they avoid information loss and consider query context.

Bi-encoders pre-compute document embeddings, losing information and lacking query context. Rerankers process queries and documents together at query time, maintaining more information.

Implementing two-stage retrieval involves embedding data with a bi-encoder (e.g., multilingual-e5-large), indexing it in a vector database (e.g., Pinecone), and then using a reranker (e.g., bge-reranker-v2-m3) to reorder results.

Reranking significantly improves RAG performance by prioritizing highly relevant information, reducing noise for the LLM.

Visit my github repo to see emojis Gemini used in the response

X threads result:

1/10 RAG: Sounds amazing, right? But often, it falls short. Why? Let's dive into the surprisingly simple fix. #RAG #LLM #AI #Search

2/10 The problem? LLMs have limited "context windows."Trying to cram too many documents in makes them less accurate. Think of it like information overload! #AI #MachineLearning

3/10 The solution? Rerankers! These models prioritize the *most* relevant documents before feeding them to the LLM. Smart, right? #AI #Retrieval

4/10 It's a two-stage process: First, a quick search grabs many documents. Then, the reranker picks the best ones. Think of it as a highly efficient filter. #AI #Efficiency

5/10 Why are rerankers better? They analyze documents *in context* with your query. No more information loss from the initial compression! #AI #Data

6/10 Think of it like this: A quick, broad search followed by a precise edit. Much more accurate than a single, less-focused search. #AI #Optimization

7/10 Implementing this is easier than you think! We'll use Pinecone for vector databases and a cool reranking model. (More details in the thread!) #Pinecone #VectorDB

8/10 We'll use the 'multilingual-e5-large' model for embeddings and a custom reranker. It's a powerful combo for supercharged search results. #NLP #Embedding

9/10 Ready to boost your RAG pipeline? This two-stage approach maximizes both retrieval and LLM recall, leading to dramatically better results! #RAG #Improvement

10/10 Want to learn more about building this? Check out the full ebook and video companion! (link in bio) Let's build better AI together! #AI #LearnToCode

Future works

This experiment show that LLMs are capable of transform the given information into a different format. I have high hope that it can turn a piece of paper into exercises for educational purposes.

2 Experiment with Vietnamese prompts and Local Llama3.3

Implementation

I used the same paper (in English) as above but this time I used vietnamese prompts and hosted Llama3.3 locally. Keep in mind that Llama3.3 has 70 Billion parameters but to run it locally, I had to quantize its parameters to 4-bit precision which made it lost a lot of information.

Result

Newsletter result:

> Đầu ra là một bức thư quảng cáo về dịch vụ tự động hóa quy trình biến video Youtube thành dàn bài cho bài viết sử dụng SimplerLLM. Thư này giới thiệu về quy trình 3 bước để thực hiện việc này, bao gồm lấy đường link URL của video Youtube, lấy ra bản ghi, biến nó thành danh sách tóm tắt và cuối cùng biến bản tóm tắt trở thành dàn bài cho bài đăng.
>
> Thư cũng cung cấp thông tin về cách sử dụng API của OpenAI và Gemini để cải thiện quy trình, cũng như hướng dẫn cách tạo file ".env"để chứa các khóa API.
>
> Cuối cùng, thư cung cấp nút lấy đoạn code để người đọc có thể tải xuống và bắt đầu sử dụng dịch vụ tự động hóa này.

Summary result:

> Reranking là một phương pháp đơn giản để cải thiện hiệu suất của Retrieval Augmented Generation (RAG) hoặc bất kỳ pipeline dựa trên truy vấn nào khác.
> Một hệ thống truy vấn hai giai đoạn cho phép chúng ta tận dụng lợi thế của cả hai thế giới, cho phép tìm kiếm ở quy mô lớn đồng thời duy trì hiệu suất chất lượng.
> Reranking giúp loại bỏ thông tin không liên quan và giữ lại thông tin hữu ích, dẫn đến hiệu suất tốt hơn trong việc trả lời câu hỏi.
> Hệ thống truy vấn hai giai đoạn bao gồm truy vấn ban đầu và xếp hạng lại để cải thiện độ chính xác của kết quả.
> Reranking có thể được sử dụng với các mô hình ngôn ngữ khác nhau để cải thiện hiệu suất tổng thể của hệ thống.

X threads result:

Tôi hiểu rằng bạn muốn tạo một chuỗi bài đăng Twitter về chủ đề "Tăng tốc sáng tạo nội dung với công nghệ NoCode". Dưới đây là 5 bài đăng Twitter mẫu:

Bài đăng 1
Tăng tốc sáng tạo nội dung của bạn mà không cần viết code! Chúng tôi đã xây dựng hệ thống đột phá cho nghiên cứu từ khóa và chủ đề bằng cách sử dụng các công cụ #NoCode. Miễn phí và dễ thiết lập! #sáng_tạo #nội_dung

Bài đăng 2
Hệ thống của chúng tôi bao gồm 2 nhiệm vụ chính: nghiên cứu từ khóa sử dụng Google API và nghiên cứu chủ đề bằng sức mạnh AI. Tất cả dữ liệu được lưu trữ trong Google Sheet giúp dễ tiếp cận! #NoCode #sáng_tạo

Bài đăng 3
Bạn có thể bán hệ thống "tự động"này cho bất cứ công ty nào cần chúng! Hệ thống của chúng tôi giúp tiết kiệm thời gian và cũng là cơ hội lớn cho bạn! #NoCode #kinh_doanh

Bài đăng 4
Tiến sâu hơn 1 bước nữa: thêm vào hệ thống 1 AI phân tích dữ liệu để thu thập được những kiến thức sâu hơn. Tăng tốc sáng tạo nội dung của bạn ngay hôm nay! #AI #sáng_tạo

Bài đăng 5
Đừng bỏ lỡ cơ hội tăng tốc sáng tạo nội dung của mình! Hệ thống của chúng tôi đã giúp nhiều người tiết kiệm thời gian và tăng hiệu suất công việc. Thử ngay hôm nay! #NoCode #sáng_tạo

Future Works

This experiment with Vietnamese doesn't show the best results, but we can still improve our prompts and run Llama3.3 on GPUs with 32-bit float to unlock its true power.