Đại học quốc gia TP.HCM
Trường đại học công nghệ thông tin

Khoa: Khoa học máy tính

Môn học: Nhận dạng

Lớp: CS338.O21

BÁO CÁO ĐỒ ÁN CUỐI KỲ

Document-Specific Retrieval Augmented Generation

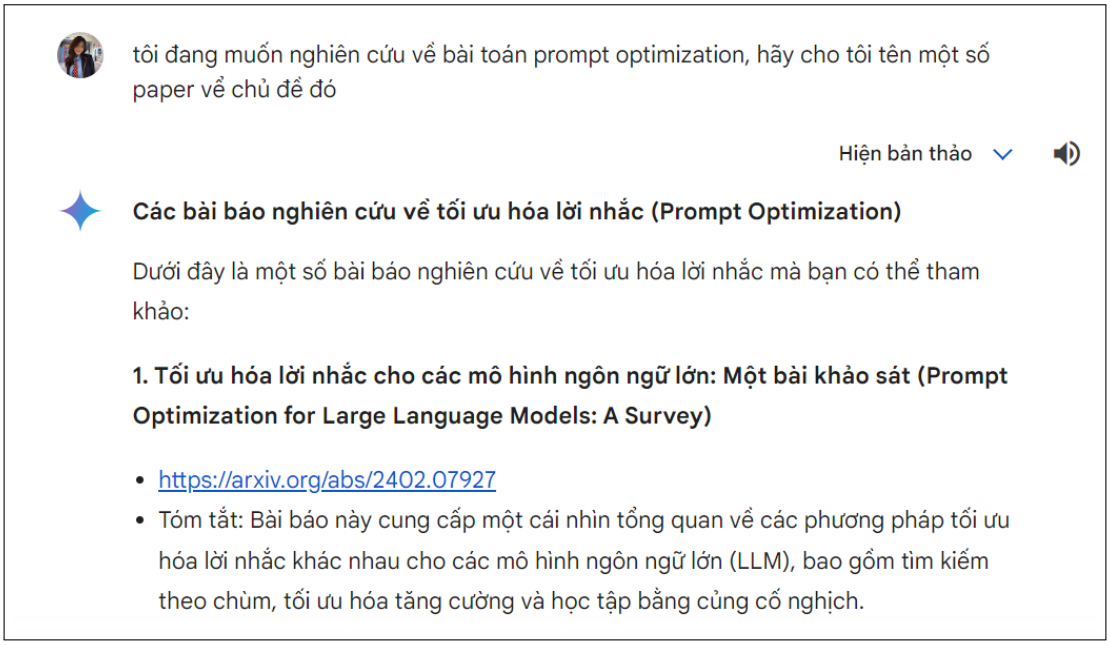
Nhóm sinh viên thực hiện:
21521024 - Nguyễn Trung Kiên
21522792 - Phạm Quốc Việt

Giảng viên hướng dẫn:
ThS. Đỗ Văn Tiến

Mục lục

1 Mục đích lựa chọn đề tài

Đặt ra tình huống như sau: "Kiên là một sinh viên thuộc lớp CS410 - Mạng neural và giải thuật di truyền. Để chuẩn bị cho đồ án cuối kỳ, Kiên đang muốn tìm kiếm các bài báo khoa học cho chủ đề Prompt Optimization. Vì đã quá quen với các công cụ LLMs (mô hình ngôn ngữ lớn) nên Kiên sử dụng Gemini[1] - một chatbot tạo sinh do Google tạo ra để có thể tìm kiếm các bài báo khoa học liên quan". Có thể thấy rằng các câu trả lời từ LLMs thoạt trông vô cùng thuyết phục và rất hợp lý (Hình 1).

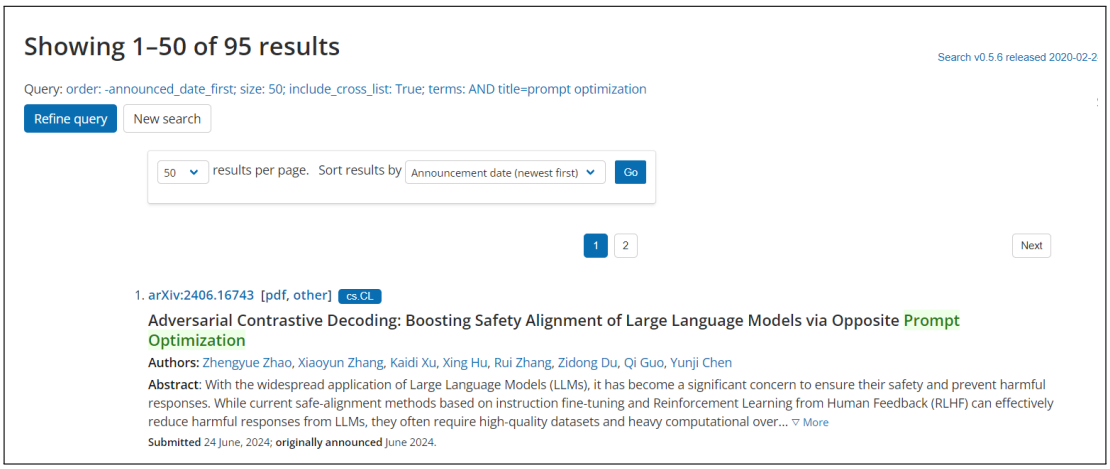


Hình 1: Kết quả của Gemini về các bài báo thuộc chủ đề Prompt Optimization

Tuy nhiên, khi nhấn vào đường dẫn đến bài báo khoa học đó, hay tìm kiếm tên bài báo mà Gemini trả về thì kết quả thu được lại là các bài báo này không tồn tại. Hiện tượng này được gọi là **AI Hallucination** (Ảo giác AI) - khi một LLMs trả về một thông tin sai lệch, không đúng sự thật. Điều này có thể là do các mô hình ngôn ngữ lớn chịu ảnh hưởng từ một lượng các dữ liệu không kiểm duyệt và chính xác trong lượng dữ liệu khổng lồ được sử dụng để huấn luyện.

[1] https://gemini.google.com/app

Nhận thấy dùng Gemini không hiệu quả, Kiên quyết định dùng Arxiv[2] - một nguồn cơ sở dữ liệu mở có chứa số lượng các bài báo khoa học lớn để tìm kiếm. Tuy nhiên vấn đề với Arxiv là ta chỉ có thể tìm kiếm theo từ khóa (hình 2), do đó sẽ rất khó nếu ta cần biết các thông tin cụ thể hơn ví dụ như các kỹ thuật để thực hiện Prompt Optimization, . . . Arxiv là một lượng dữ liệu đủ tin cậy nhưng chính Arxiv không thể nhìn vào lượng dữ liệu mà nó có để tạo sinh một câu trả lời chuẩn xác cho một vấn đề nào đó.



Hình 2: Tìm kiếm trên Arxiv bằng cách dùng từ khóa “Prompt Optimization”

Retrieval Augmented Generation[3] (Lewis et al. 2020) là một giải pháp có thể giải quyết cả hai vấn đề trên bằng cách kết hợp việc **nhận dạng được những nguồn thông tin bên trong một lượng dữ liệu đủ tin cậy và khả năng trả về các câu trả lời đầy thuyết phục của các LLMs**. Nhận thấy rằng đây là một đề tài phù hợp cho môn học, cũng như là có thể tận dụng một số các kiến thức mới từng được đề cập trên lớp như **LangChain**[4], ... Nhóm thực hiện mong muốn những kiến thức học được từ môn học có thể được sử dụng vào việc giải quyết một bài toán thực tế.

Nhóm vô cùng cảm ơn **ThS. Đỗ Văn Tiến** đã nhiệt tình giảng dạy, giải đáp những vấn đề còn chưa hiểu trong quá trình học tập. Nhóm hi vọng có thể nhận được những lời góp ý của thầy qua đồ án này để có thể ngày càng cải thiện hơn.

[2] https://arxiv.org/
[3] Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks
[4] https://www.langchain.com/

2 Giới thiệu đề tài

2.1 Định nghĩa bài toán

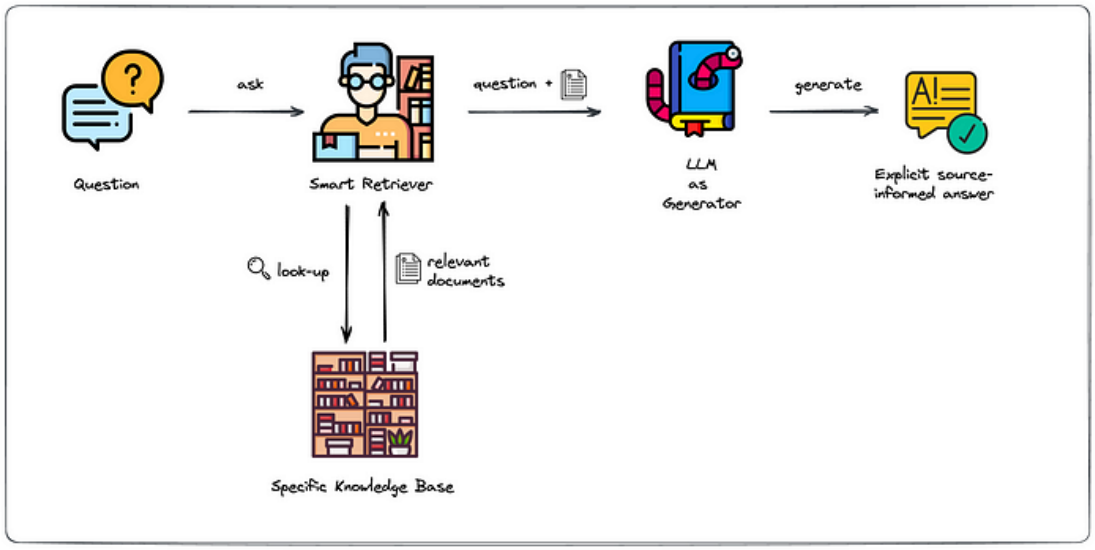
Với một lượng dữ liệu đáng tin cậy D và một mô hình ngôn ngữ lớn M (đã được pre-trained), làm thế nào để M có thể "nhìn vào" được các thông tin bên trong D và tạo sinh ra một câu trả lời hợp lý để không rơi vào tình trạng AI Hallucination. Do đó:

- Input: Tập dữ liệu D và một input prompt P.
- Output: Câu trả lời được tạo sinh bởi M cho prompt P.

Để thuận tiện cho việc diễn giải, ta gọi $Relevant(P, D)$ là tập các tài liệu bên trong D liên quan nhất với input prompt P và d_i là tài liệu thứ i trong D.

2.2 Retrieval Augmented Generation

Retrieval-Augmented Generation, hay RAG (Lewis et al. 2020) là một giải pháp có thể kết hợp sức mạnh từ khả năng nhận dạng được những nguồn thông tin bên trong một lượng dữ liệu đủ tin cậy và khả năng trả về các câu trả lời đầy thuyết phục của các LLMs.

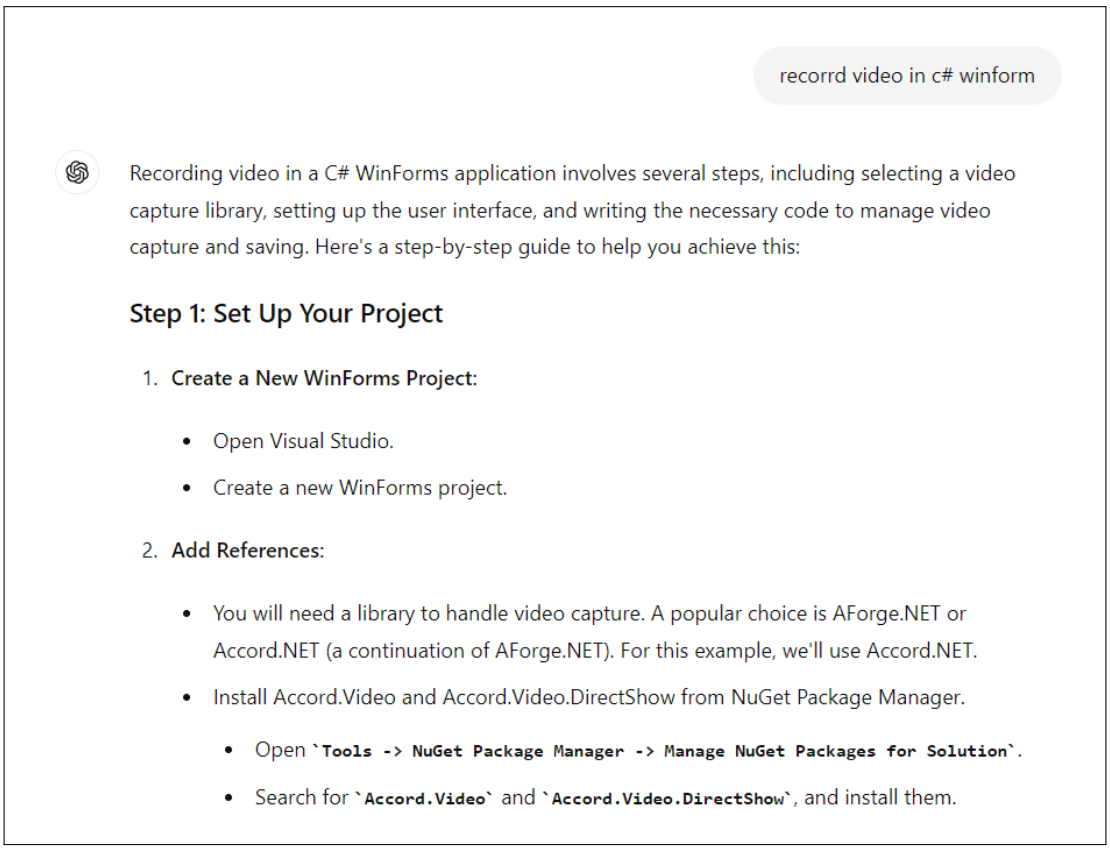


Hình 3: Pipeline của Retrieval Augmented Generation

Khác với quá trình mà ta sử dụng các công cụ LLMs như ChatGPT hay Gemini là việc đưa prompt do con người yêu cầu vào cho LLMs và để nó tạo sinh ra câu trả lời dựa trên lượng dữ liệu khổng lồ mà nó được pre-trained, RAG có thêm "Retriever" trước khi đưa vào LLMs với nhiệm vụ truy vấn các tài liệu liên quan bên trong D và kết hợp cả input prompt P để M tạo sinh ra câu trả lời.

2.3 Sản phẩm đề tài

Sản phẩm mà nhóm mong muốn thực hiện là một chatbot tương tự ChatGPT[5].



Hình 4: Ví dụ minh họa cho sản phẩm mà nhóm mong muốn thực hiện

Bằng cách sử dụng một tập dữ liệu đáng tin cậy, lựa chọn và kết hợp lại các công cụ phù hợp cho từng thành phần trong pipeline của RAG, nhóm sẽ hiện thực hóa RAG và demo với chatbot này.

[5] https://chatgpt.com/

3 Hiện thực hóa RAG

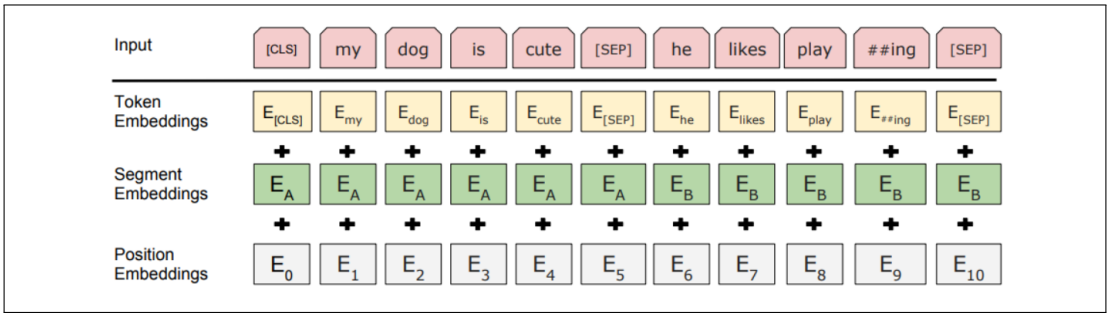
Ba câu hỏi chính được giải quyết khi hiện thực RAG trong phần này bao gồm:

- Xác định $Relevant(P, D)$ bằng cách nào?
- Lựa chọn công cụ nào phù hợp cho từng thành phần của RAG?
- Làm thế nào để liên kết các thành phần của RAG?

3.1 Truy vấn văn bản

Cùng với sự phát triển của lĩnh vực Truy vấn thông tin (Information Retrieval), bài toán truy vấn văn bản cũng được các nhà nghiên cứu thực hiện và đã có được những kết quả và phương hướng khác nhau. Trước khi các mô hình như Transformer hay BERT ra đời thì phương pháp thường được sử dụng là các phương pháp term-based như TF-IDF hay BM25, ...

Tuy nhiên, nhược điểm khi sử dụng các phương pháp term-based là nếu như với input prompt P, tồn tại một số các tài liệu liên quan có ý nghĩa tương đồng nhưng lại không chứa các từ giống với các từ trong P thì mức độ liên quan giữa P và các tài liệu đó theo các phương pháp term-based sẽ khá thấp. Nhận thấy điều đó và từ sự xuất hiện của các mô hình embedding như BERT, bài toán truy vấn văn bản dần chuyển hướng sang việc biến đổi nội dung tìm kiếm và các văn bản thành các vector có cùng chiều dữ liệu bằng các mô hình embedding. Khi này vector được số hóa mang nhiều thông tin ý nghĩa về mặt câu từ và ngữ nghĩa, dẫn đến việc nhiều thông tin được so với nhau hơn là chỉ với việc so khớp từ khóa của các phương pháp term-based.



Hình 5: Cách thức BERT embedding chuỗi từ

Ngoài ra cũng có một số nghiên cứu kết hợp cả việc sử dụng phương pháp term-based và mô hình pre-trained để giải quyết bài toán. Chẳng hạn, nhóm đã từng thực hiện đề tài truy vấn văn bản sử dụng kết hợp cả 2 hướng trên cho đồ án cuối kỳ môn Truy vấn thông tin đa phương tiện.

Giá trị $nDCG@10$ của từng hướng tiếp cận trong bảng dưới (kết quả được làm tròn đến chữ số thập phân thứ 4). Kết quả cao nhất của bảng 1 được **in đậm**, cao thứ nhì được gạch chân.

Bảng 1: Kết quả $nDCG@10$ của một số phương pháp

Hướng tiếp cận	Phương pháp	ndcg@10
Phương pháp term-based	TF-IDF	0.9259
	BM25	0.7433
Pre-trained language model	Vietnamese-SBERT	0.9256
	PhoBERT	0.7383
Multi-stage document ranking	TF-IDF + Vietnamese-SBERT	**0.9582**
	TF-IDF + PhoBERT	0.8019
	BM25 + Vietnamese-SBERT	0.9256
	BM25 + PhoBERT	0.7825

Hình 6: Trích lại kết quả thực nghiệm trong đề tài truy vấn văn bản sử dụng kết hợp cả phương pháp term-based và mô hình pre-trained cho đồ án cuối kỳ môn Truy vấn thông tin đa phương tiện của nhóm

Trong đồ án này, nhóm sử dụng một mô hình pre-trained giúp embedding dữ liệu dạng text cho tác vụ truy vấn các tài liệu liên quan. Với P và D, các bước thực hiện để xác định $Relevant(P, D)$ bao gồm:

- Bước 1: Biến đổi P và các tài liệu trong D thành các vector có cùng chiều dữ liệu bằng một mô hình embedding.
- Bước 2: Với mỗi d_i trong D và P, ta xác định giá trị $Cosine(d_i, P)$ (nằm trong khoảng $[0, 1]$) bằng cách sử dụng vector được chuyển hóa tương ứng. Nếu giá trị đó càng gần 1, ta có thể khẳng định nội dung câu từ và ngữ nghĩa của d_i và P càng đồng nghĩa nhau và ngược lại.

- Bước 3: Chọn ra k tài liệu liên quan nhất, cũng chính là k tài liệu có giá trị Cosine cao nhất khi so với P để cùng P đưa vào LLMs nhằm mục đích tạo sinh ra câu trả lời.

Khi thực hiện thực nghiệm, nhóm chọn giá trị $k = 3$. Vậy $Relevant(P, D)$ sẽ là tập hợp 3 văn bản trong D có giá trị Cosine cao nhất với P.

3.2 Retriever

Trong phần 3.1, nhóm đã trình bày về cách thức mà nhóm truy vấn các văn bản trong D liên quan nhất với P. Phần này sẽ trình bày về các công cụ nhóm sử dụng để phục vụ cho việc truy vấn văn bản.

3.2.1 Mô hình embedding

Từ hình 6, ta thấy Vietnamese-SBERT (một mô hình được fine-tune từ SBERT để có thể sử dụng cho ngôn ngữ Việt) cho ra được kết quả $ndcg@10$ tốt nhất khi xét hướng tiếp cận sử dụng mô hình pre-trained. Nhận thấy được sức mạnh của các mô hình SBERT và việc ngôn ngữ sử dụng cho sản phẩm đề tài là tiêng Anh nên nhóm lựa chọn một phiên bản của SBERT có tên **all-MiniLM-L6-v2**[6] cho tác vụ embedding P và các d_i trên D.

3.2.2 So sánh sự tương đồng giữa 2 vector

Để so sánh sự tương đồng giữa 2 vector, nhóm sử dụng độ đo **Cosine**:

- Xét 2 vector u và v (có thể được padding về cùng kích thước), cosine giữa u và v được tính theo công thức:

$$Cosine(u, v) = \frac{\mathbf{u} \cdot \mathbf{v}}{\|\mathbf{u}\|\|\mathbf{v}\|}$$

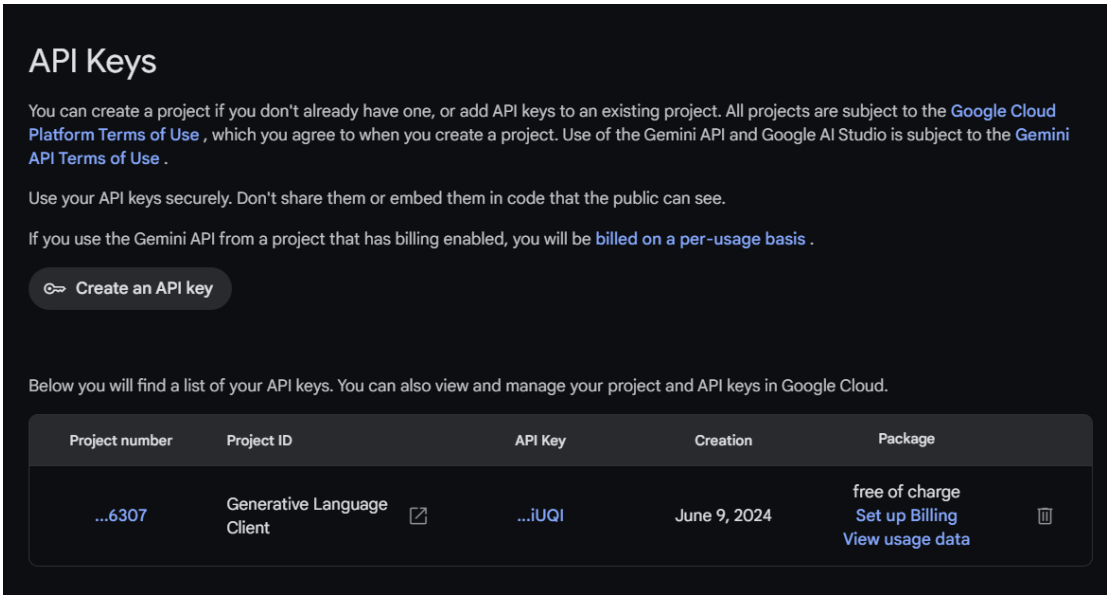
- Trong đó: $\mathbf{u} \cdot \mathbf{v}$ là tích vô hướng của u và v, $\|\mathbf{u}\|$ và $\|\mathbf{v}\|$ lần lượt là chuẩn 2 của vector u và v. Nếu giá trị Cosine của 2 vector tương ứng với 2 văn bản càng gần 1 thì 2 văn bản này càng đồng nghĩa với nhau và ngược lại, nếu càng gần -1 thì 2 văn bản này có ý nghĩa ngược nhau.

[6] https://huggingface.co/sentence-transformers/all-MiniLM-L6-v2

3.3 Mô hình ngôn ngữ lớn

LLM mà nhóm lựa chọn có tên: **Gemini-pro**[7] - một mô hình được phát triển bởi Google Deepmind. Lý do nhóm lựa chọn mô hình này là vì đây là LLM mà nhóm có thể sử dụng ổn định trong khi các LLM khác khi thực hiện gọi API đều gặp vấn đề về giới hạn số lần gọi API trong một khoảng thời gian nhất định.

Cách mà nhóm sử dụng mô hình này khá đơn giản, nhóm tạo API key từ Google AI Studio[8], sau đó kết hợp P và $Relevant(P, D)$ như là input prompt vào mô hình và sau đó việc mà mô hình cần thực hiện là tạo sinh một câu trả lời với văn phong phù hợp dựa vào thông tin mà nó được nhận.

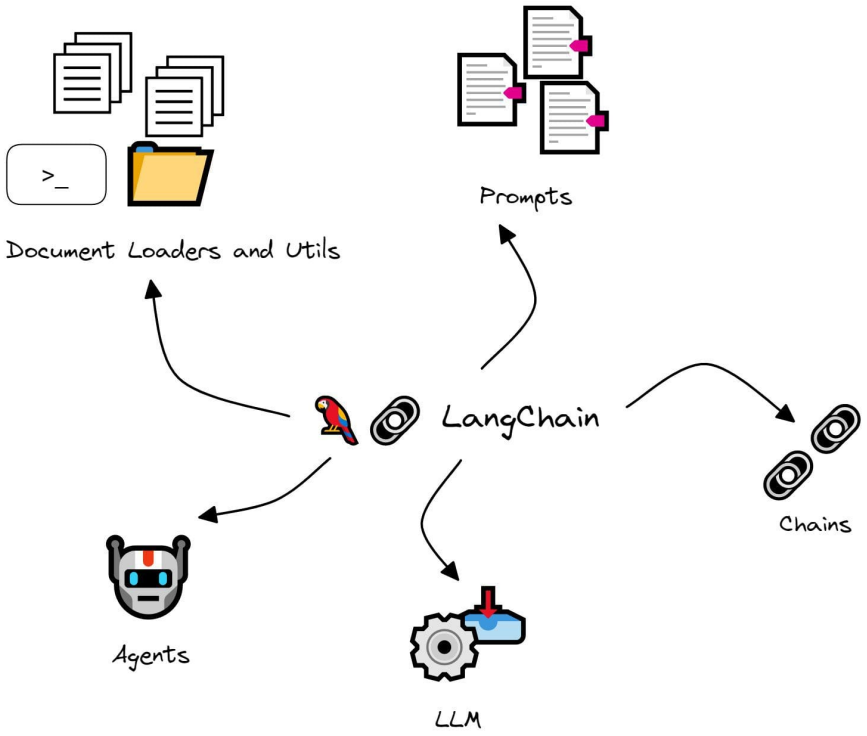


Hình 7: Lấy API key để sử dụng mô hình Gemini-pro

3.4 LangChain

Để liên kết các thành phần mà nhóm đã chọn và kết hợp tất cả theo pipeline của RAG (hình 3), phương pháp mà nhóm sử dụng có tên là **LangChain**. LangChain là một framework được thiết kế để đơn giản hóa việc tạo các ứng dụng bằng các mô hình ngôn ngữ lớn.

[7] https://deepmind.google/technologies/gemini/pro/
[8] https://aistudio.google.com/



Hình 8: Các thành phần chính của LangChain

Các thành phần chính của LangChain được áp dụng cho đồ án bao gồm:

- Document Loaders and Utils: Đọc dữ liệu D từ nguồn nào đó sao cho M có thể truy cập dễ dàng.
- Prompts: $P + Relevant(P, D)$ đưa vào M.
- LLM: Mô hình ngôn ngữ lớn, là M.
- Agents (Agent): Là những thực thể được hỗ trợ cho các thành phần của LangChain. Chúng có thể là chatbot, hệ thống trả lời câu hỏi hoặc bất kỳ loại ứng dụng AI nào khác tương tác với người dùng hoặc giải quyết các vấn đề cụ thể.
- Chains: Xâu chuỗi 4 thành phần trên một cách hợp lý để tạo thành 1 workflow.

3.5 Bộ dữ liệu sử dụng

Nguồn dữ liệu mà nhóm sử dụng có tên **Wikipedia**[9]. Phiên bản mà nhóm sử dụng chứa 205328 bài viết khác nhau được crawl từ Wikipedia. Mỗi bài viết có 4 trường bao gồm id, url, tiêu đề và nội dung bài viết.

```
{'id': '1',
 'url': 'https://simple.wikipedia.org/wiki/April',
 'title': 'April',
 'text': 'April is the fourth month...'
}
```

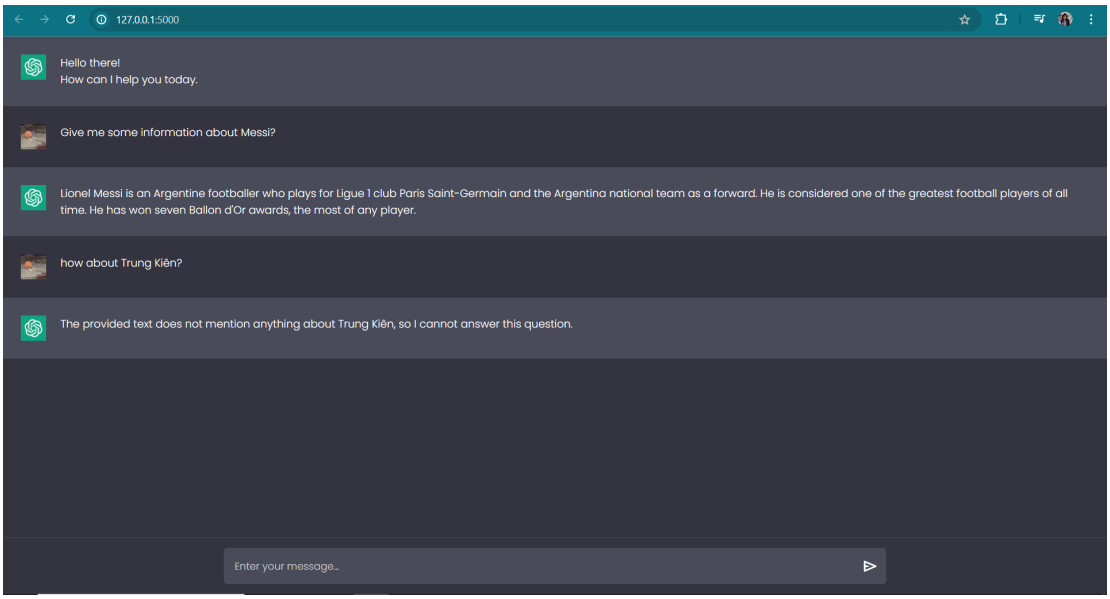
Hình 9: Một ví dụ trong tập dữ liệu về định nghĩa của "April" (Tháng 4)

Toàn bộ nội dung tập dữ liệu D và input prompt P được đưa qua mô hình embedding. Các tài liệu liên quan nhất có được bằng cách sử dụng Cosine Similarity thông qua thư viện **FAISS**[10] - một thư viện hỗ trợ việc truy vấn vector được xây dựng bởi Facebook.

[9] https://huggingface.co/datasets/legacy-datasets/wikipedia
[10] https://python.langchain.com/v0.2/docs/integrations/vectorstores/faiss/

4 Kết quả thu được

Nhóm đã hoàn thành một ứng dụng web (localhost) sử dụng Flask trên framework bootstrap5 nhằm triển khai mô hình RAG thành ứng dụng thực tế.



Hình 10: Ứng dụng web Chatbot dựa trên bộ dữ liệu Wikipedia

Khi nhập xong câu prompt để đưa vào Chatbot và bấm nút gửi, Chatbot sẽ tạo sinh câu trả lời. Trong hình 10 có thể thấy khi đưa vào câu hỏi "Give me some information about Messi?" thì ta đã nhận được câu trả lời gồm một số các thông tin về cầu thủ Messi. Tuy nhiên, vì RAG phụ thuộc vào tập dữ liệu mà nó được cung cấp nên có thể có trường hợp câu trả lời đã lỗi thời (Trong hình 10, Messi được cho là đang chơi cho câu lạc bộ Paris Saint-Germain trong vai trò tiền đạo nhưng ở thời điểm hiện tại thì Messi đã chuyển sang thi đấu cho câu lạc bộ Inter Miami).

Ngoài ra, cũng có trường hợp câu hỏi mà ta đưa vào nằm ngoài phạm vi của bộ dữ liệu được cung cấp. Cũng trong hình 10, khi ta muốn biết thông tin về người tên "Trung Kiên", mô hình trả về kết quả "The provided text does not mention anything about Trung Kiên, so I cannot answer this question." (Tạm dịch: "Dữ liệu được cung cấp không hề đề cập đến Trung Kiên, do đó tôi không thể trả lời câu hỏi này").

5 Hướng mở rộng và phát triển

Đề tài được thực hiện với mục tiêu chính là áp dụng những gì được học trong môn học, đồng thời mở rộng thêm từ những kiến thức cá nhân, trên mạng hay được giới thiệu trên lớp. Từ quá trình nghiên cứu, tìm hiểu và thực hiện đề tài, nhóm nhận thấy có một số hướng mở rộng và phát triển mà nhóm có thể thực hiện như sau:

- Từ phần 4, có thể thấy rằng câu trả lời từ Chatbot phụ thuộc vào bộ dữ liệu được cung cấp. Nếu bộ dữ liệu này bị lỗi thời thì câu trả lời có thể sẽ không còn chính xác nữa. Do đó, ta có thể xây dựng một hệ thống giúp update dữ liệu real-time, chẳng hạn như với bộ dữ liệu Wikipedia thì ta có thể thực hiện crawl lại dữ liệu sau một khoảng thời gian nào đó.

- Khi thực hiện truy vấn văn bản, nhóm sử dụng mô hình embedding rồi so sánh độ tương đồng giữa 2 vector. Một số phương pháp khác như các phương pháp term-based (TF-IDF, BM25, ...) hay kết hợp cả 2 phương pháp kể trên có thể được thực hiện và so sánh hiệu năng của từng phương pháp.

- Nhóm giữ nguyên input prompt được đưa vào pipeline của RAG. Các phương pháp prompt engineering cũng có thể được áp dụng để nâng cao hiệu năng truy vấn văn bản nói riêng và của RAG.

- Từng văn bản từ bộ dữ liệu Wikipedia được nhóm embedding ngay lập tức mà không qua bất kỳ các bước tiền xử lý văn bản. Một số các phương pháp như lowercase, loại bỏ stopwords hay Chunking - một kỹ thuật chia văn bản dài thành nhiều phần nhỏ hơn có thể được sử dụng cho đề tài.

6 Phân công công việc

Nguyễn Trung Kiên

- Phân chia công việc cho từng người.
- Lựa chọn các thành phần cài đặt RAG.
- Tìm hiểu và cài đặt mô hình embedding: all-MiniLM-L6-v2, thư viện Faiss, mô hình ngôn ngữ lớn: Gemini-pro và LangChain để hiện thực hóa RAG trên Colab.
- Thực nghiệm RAG với bộ dữ liệu Wikipedia.
- Soạn slide báo cáo và cuốn báo cáo cho môn học bằng Latex.

Phạm Quốc Việt

- Chỉnh sửa code đã được thực thi trên colab và áp dụng vào ứng dụng web demo.
- Tìm hiểu Flask và tạo ứng dụng web Chatbot với framework bootstrap5 để minh họa cho hoạt động của RAG.
- Hỗ trợ tìm hiểu và cài đặt các thành phần đã lựa chọn cho RAG.

7 Tài liệu tham khảo

[1] Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks
[2] Tìm hiểu và áp dụng LangChain
[3] Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks
[4] Cài đặt mô hình embedding all-MiniLM-L6-v2
[5] Thư viện FAISS
[6] Sử dụng mô hình Gemini-pro
[7] Bộ dữ liệu Wikipedia
[8] BootStrap 5.0 Documentation
[9] Flask documentation