Retrieval Augmented Generation (RAG)

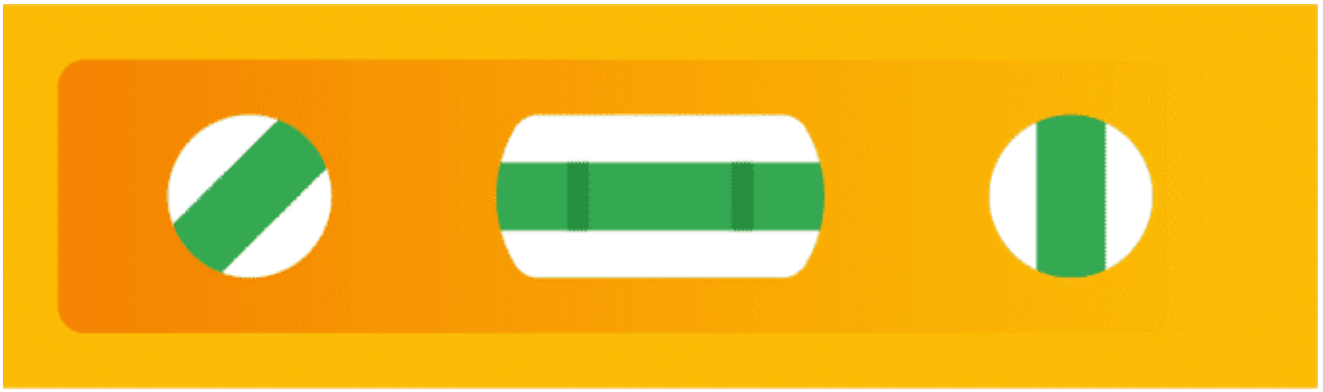
Tạo sinh được bổ sung bằng truy xuất thông tin - Tạo sinh tăng cường truy xuất

Cải thiện độ chính xác của các foundation models (mô hình nền tảng)

Các foundation models được huấn luyện cho các mục đích chung. Ngay cả khi bạn cung cấp cho chúng các prompts (lời nhắc) tốt, chúng có thể không thực hiện các tác vụ tốt như bạn mong muốn. Tuy nhiên, có nhiều phương pháp bạn có thể sử dụng để cải thiện hiệu suất của mô hình.

Model tuning (Điều chỉnh mô hình)

Bạn có thể sử dụng model tuning để cải thiện hiệu suất của mô hình cho các tác vụ cụ thể. Supervised tuning (điều chỉnh có giám sát) cải thiện hiệu suất của mô hình bằng cách dạy nó một kỹ năng mới. Dữ liệu chứa hàng trăm ví dụ được gắn nhãn được sử dụng để dạy mô hình bắt chước hành vi hoặc tác vụ mong muốn.



Khi bạn chạy một supervised tuning job trên Vertex AI, mô hình học thêm các tham số giúp nó mã hóa thông tin cần thiết để thực hiện tác vụ mong muốn hoặc học hành vi mong muốn. Mô hình trở nên chính xác hơn cho tác vụ được huấn luyện.

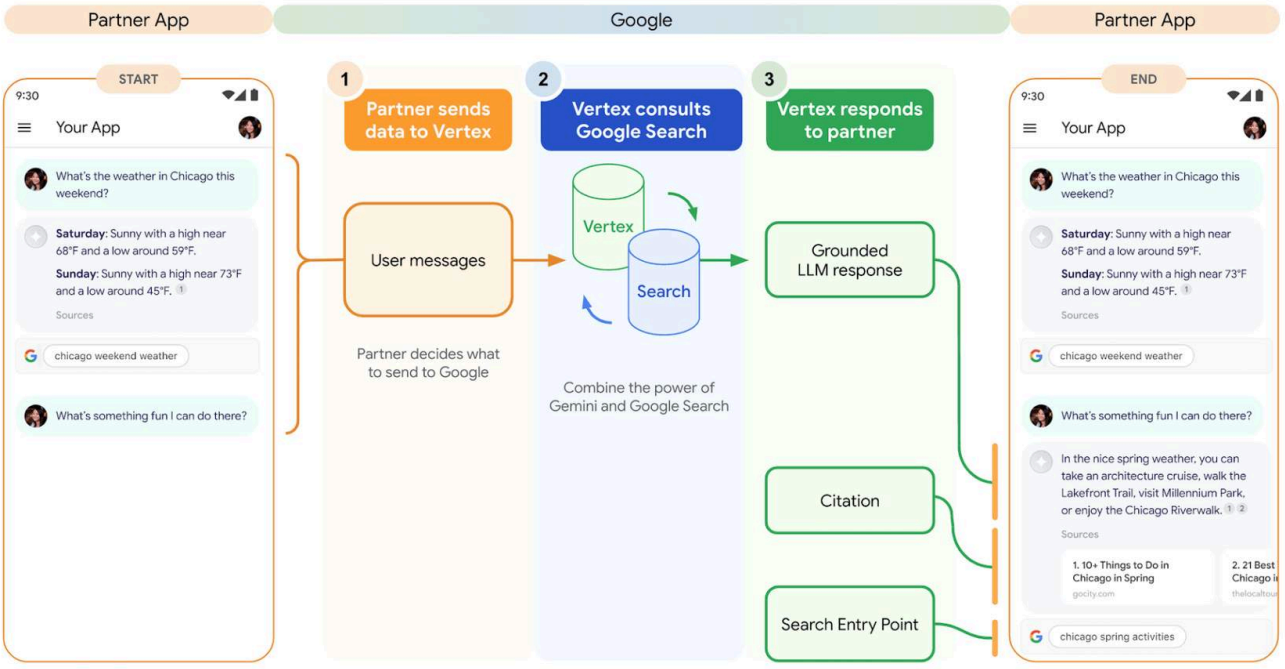
Việc tuning một mô hình có thể tốn kém, và mô hình đã được huấn luyện cũng là tĩnh. Nếu bạn muốn mô hình của mình tích hợp dữ liệu động, model tuning có thể không phải là giải pháp phù hợp.

Grounding responses with Google Search (Định hướng câu trả lời với Google Search)

> Trong generative AI (AI tổng hợp), grounding là khả năng kết nối đầu ra của mô hình với các nguồn thông tin có thể xác minh được.

Khi bạn cung cấp cho mô hình quyền truy cập vào các nguồn dữ liệu cụ thể, grounding có thể giảm khả năng tạo ra nội dung bịa đặt.

Trong Vertex AI, phương pháp định hướng đầu ra mô hình đầu tiên là Định hướng câu trả lời với Google Search. Sử dụng Định hướng câu trả lời với Google Search nếu bạn muốn kết nối mô hình với kiến thức thế giới, phạm vi rộng các chủ đề có thể, hoặc thông tin cập nhật trên internet.



1) Ứng dụng gửi dữ liệu tới Vertex AI

Ứng dụng quyết định nội dung nào cần được gửi đến mô hình trong Vertex AI.

Đối với ứng dụng chatbot, nội dung này thường bao gồm truy vấn hiện tại và lịch sử trò chuyện của người dùng.

2) Vertex AI tham khảo Google Search

Dựa trên thông tin được gửi tới Vertex AI, mô hình sử dụng Google Search để giúp xây dựng câu trả lời có thẩm quyền cho truy vấn.

3) Vertex AI phản hồi cho ứng dụng

Như với các tương tác mô hình nền tảng khác, Vertex AI cung cấp phản hồi cho truy vấn.

Phản hồi này sẽ được định hướng (grounded) với thông tin có thể truy cập từ Google Search:

- Citations (Trích dẫn): các liên kết đến trang web hỗ trợ các sự kiện cụ thể trong phản hồi.
- Search entry point (Điểm truy cập tìm kiếm): một truy vấn tìm kiếm có thể được sử dụng để tìm thêm thông tin liên quan.

Khi bạn định hướng với Google Search trong Gemini, câu trả lời thường bao gồm metadata bổ sung:

- Source links (Liên kết nguồn) là các URL web dẫn đến các trích dẫn nguồn cho thông tin được Gemini sử dụng trong câu trả lời. Hiển thị trích dẫn trong ứng dụng của bạn được khuyến nghị cao, vì nó nâng cao độ tin cậy của câu trả lời.
- Search entry point (Điểm truy cập tìm kiếm) là truy vấn ban đầu nên được sử dụng cho các tìm kiếm Google tiếp theo. Điểm truy cập không nên bị sửa đổi, và nhấp vào điểm truy cập sẽ đưa người dùng trực tiếp đến trang kết quả Google Search.

Định hướng câu trả lời với dữ liệu riêng bằng Vertex AI Search

Định hướng với Google Search sử dụng lượng thông tin khổng lồ trên internet, nhưng Google Search không cung cấp quyền truy cập vào dữ liệu độc quyền của riêng bạn.

Trong Vertex AI, phương pháp định hướng đầu ra mô hình thứ hai là định hướng với dữ liệu riêng của bạn. Với Vertex AI Search, bạn có thể xây dựng một ứng dụng tìm kiếm chất lượng Google trên dữ liệu bạn kiểm soát. Với recommendations (đề xuất), bạn có thể xây dựng một ứng dụng đề xuất trên dữ liệu của bạn để gợi ý nội dung tương tự với nội dung người dùng đang xem.

Bạn có thể định hướng các mô hình ngôn ngữ với dữ liệu văn bản riêng của mình bằng cách sử dụng Vertex AI Search như một kho dữ liệu. Bạn tích hợp dữ liệu riêng của mình, bất kể định dạng, để tinh chỉnh đầu ra của mô hình. Bằng cách liên kết với các kho dữ liệu được chỉ định trong Vertex AI Search, mô hình đã được định hướng có thể tạo ra các câu trả lời chính xác hơn, phù hợp hơn và liên quan trực tiếp đến trường hợp sử dụng của bạn.

Đây là ví dụ so sánh giữa câu trả lời không định hướng và có định hướng:

> **Không định hướng (Ungrounded)**
>
> **Prompt:** Chính sách nghỉ phép của công ty tôi là gì?
>
> **Phản hồi:** Tôi không thể trả lời chắc chắn về chính sách nghỉ phép cụ thể của công ty bạn. Chính sách nghỉ phép được thiết lập bởi từng công ty riêng lẻ và có thể khác nhau tùy thuộc vào quy mô, địa điểm và ngành nghề của công ty bạn.
>
> **Định hướng (Grounded)**
>
> **Prompt:** Chính sách nghỉ phép của công ty tôi là gì?
>
> **Phản hồi:** Công ty của bạn có chính sách nghỉ phép không giới hạn. Tuy nhiên vẫn cần được phê duyệt. Quản lý của bạn phải phê duyệt hoặc từ chối yêu cầu dựa trên khối lượng công việc, khả năng bảo hiểm và nhu cầu của công ty.

Mỗi kho dữ liệu có thể chứa một loại dữ liệu, và mỗi kho dữ liệu có một hoặc nhiều bản ghi dữ liệu (tài liệu). Những gì một tài liệu thể hiện sẽ khác nhau tùy thuộc vào loại dữ liệu trong

kho dữ liệu.

Website data (Dữ liệu trang web)

Sử dụng trực tiếp nội dung từ trang web của bạn.

Bạn có thể cung cấp các tên miền như yoursite.com/faq và yoursite.com/events và kích hoạt tìm kiếm hoặc đề xuất trên nội dung tại các tên miền đó.

Một document (tài liệu) là một trang web.

Structured data (Dữ liệu có cấu trúc)

Sử dụng dữ liệu có tổ chức, có thể truy vấn được.

Một kho dữ liệu với dữ liệu có cấu trúc cho phép tìm kiếm kết hợp (từ khóa và ngữ nghĩa) hoặc đề xuất trên dữ liệu có cấu trúc như bảng BigQuery hoặc file NDJSON. Ví dụ, bạn có thể kích hoạt tìm kiếm hoặc đề xuất trên danh mục sản phẩm hoặc danh mục phim, hoặc trên danh bạ bác sĩ để tìm kiếm hoặc đề xuất nhà cung cấp.

Một document là một hàng trong bảng hoặc bản ghi JSON tuân theo schema (lược đồ) được định nghĩa trước. Bạn có thể tự cung cấp schema này, hoặc để Vertex AI suy ra schema từ dữ liệu được nạp vào.

Structured content (media) (Nội dung có cấu trúc - phương tiện)

Tích hợp nội dung phương tiện có cấu trúc.

Loại kho dữ liệu này sử dụng schema dữ liệu có cấu trúc dành riêng cho phương tiện. Ví dụ, loại dữ liệu này có thể chứa thông tin về video, bài báo, file nhạc hoặc podcast.

Một document chứa thông tin mô tả mục phương tiện, như tiêu đề, URI đến vị trí nội dung, mô tả, danh mục, ngôn ngữ, năm và xếp hạng.

Structured content for third-party data sources (Nội dung có cấu trúc cho nguồn dữ liệu bên thứ ba)

Kết hợp liền mạch dữ liệu từ nguồn bên ngoài.

Một document là một thực thể cụ thể cho nguồn dữ liệu bên thứ ba, như vấn đề Jira hoặc không gian Confluence.

Unstructured data (Dữ liệu phi cấu trúc)

Sử dụng dữ liệu thô, không định dạng.

Một document là một file ở định dạng HTML, PDF có nhúng văn bản, hoặc TXT.

Retrieval augmented generation (Tạo sinh tăng cường truy xuất)

Một phương pháp mạnh mẽ khác để cải thiện câu trả lời của bạn là retrieval augmented generation, còn được gọi là RAG. Với RAG, bạn sửa đổi prompt để cung cấp dữ liệu bổ sung cho mô hình. Không cần fine tuning (tinh chỉnh) mô hình.

Dữ liệu bổ sung này có thể bao gồm dữ liệu độc quyền không phải là một phần của foundation model. Ví dụ, Gemini đã được huấn luyện với dữ liệu công khai, vì vậy cơ sở kiến thức của tổ chức bạn và dữ liệu riêng khác không phải là một phần của mô hình Gemini.

Bạn cũng có thể cung cấp an toàn dữ liệu người dùng nhạy cảm cho người dùng đã đăng nhập vào mô hình trong prompt. Bạn không muốn dữ liệu của bất kỳ người dùng nào khác có sẵn cho mô hình, và không có cách nào để mô hình truy cập dữ liệu đó nếu nó không phải là một phần của mô hình và không được cung cấp trong prompt.

Câu hỏi kiểm tra

Câu hỏi 1: Phương pháp nào sau đây để cải thiện độ chính xác của foundation model yêu cầu sửa đổi trực tiếp foundation model? Chọn tất cả đáp án phù hợp.

- Fine tuning (Điều chỉnh tinh)
- Grounding with Google Search (Định hướng với Google Search)
- Grounding with Vertex AI Search (Định hướng với Vertex AI Search)
- Retrieval augmented generation (Tạo sinh tăng cường truy xuất)

Câu hỏi 2: Phương pháp nào sau đây cho phép bạn sử dụng dữ liệu độc quyền của mình để cải thiện mô hình? Chọn tất cả đáp án phù hợp.

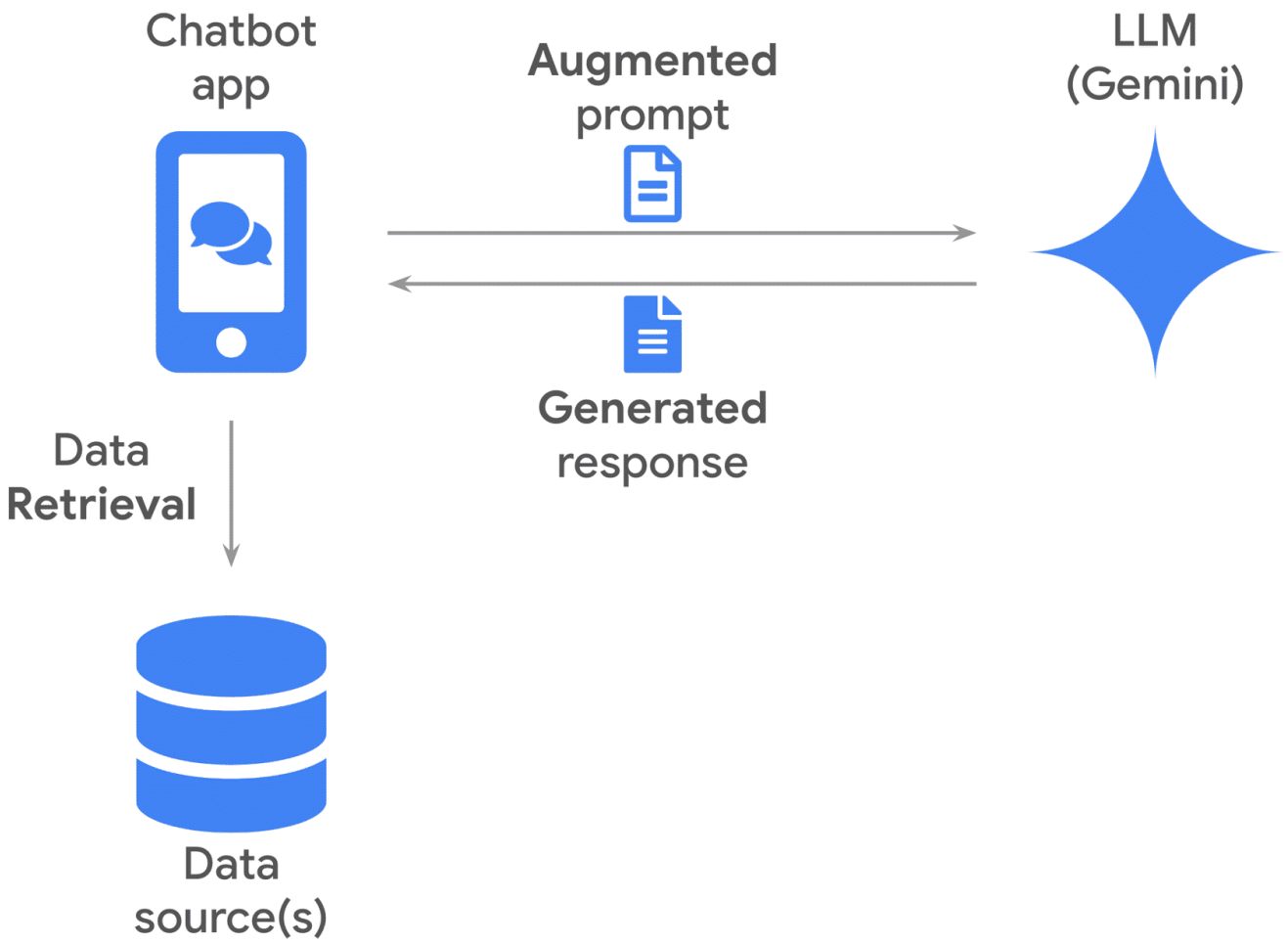
- Fine tuning
- Grounding with Google Search
- Grounding with Vertex AI Search
- Retrieval augmented generation

Retrieval augmented generation (Tạo sinh tăng cường truy xuất)

RAG (Tạo sinh tăng cường truy xuất) là một kỹ thuật nhằm cải thiện chất lượng đầu ra của LLM bằng cách định hướng nó với các nguồn kiến thức không có sẵn trong mô hình đã huấn luyện.

RAG **cung cấp cho mô hình quyền truy cập vào thông tin được truy xuất từ các cơ sở kiến thức hoặc tài liệu đáng tin cậy**, điều này cải thiện độ chính xác của câu trả lời. Thông tin này có thể bao gồm **dữ liệu độc quyền** không có trong mô hình, cũng như **dữ liệu cụ thể của người dùng** nhạy cảm không nên đưa vào mô hình huấn luyện.

Sau đây là ví dụ về cách kỹ thuật RAG được sử dụng để nâng cao câu trả lời từ LLM:



1) Data retrieval (Truy xuất dữ liệu)

Một LLM thường không bao gồm dữ liệu người dùng khi nó được huấn luyện. **Dữ liệu người dùng là động**, điều này thường có nghĩa là một LLM đã được huấn luyện không thể có dữ liệu mới nhất. Quan trọng hơn, **dữ liệu người dùng là nhạy cảm**, và bạn không muốn một LLM có khả năng cung cấp dữ liệu người dùng cho một người dùng khác trong câu trả lời của nó.

Ngoài ra, **các foundation models không được huấn luyện với dữ liệu độc quyền**, như cơ sở kiến thức và kho lưu trữ tài liệu. Ví dụ, khi bạn gửi một prompt trực tiếp đến Gemini, bạn không thể yêu cầu Gemini cung cấp thông tin không có sẵn trên internet.

Dữ liệu này có thể được truy xuất và cung cấp cho mô hình như một phần của prompt.

2) Augmented prompt (Prompt được tăng cường)

Ngoài truy vấn của người dùng và hướng dẫn được cung cấp cho mô hình, **prompt được tăng cường với dữ liệu đã truy xuất**. **Prompt hướng dẫn mô hình tin tưởng vào dữ liệu đã truy xuất**, và mô hình cũng có thể được yêu cầu **định hướng dữ liệu bằng cách cung cấp tham chiếu hoặc liên kết** đến dữ liệu để người dùng có thể truy cập vào tài liệu đầy đủ.

3) Generated response (Phản hồi được tạo ra)

Mô hình sử dụng tất cả thông tin trong prompt (bao gồm cả dữ liệu đã truy xuất) để tạo ra câu trả lời.

Retrieving data for RAG (Truy xuất dữ liệu cho RAG)

Phần ***"retrieval"*** trong retrieval augmented generation cho phép mô hình sử dụng dữ liệu không có sẵn trong bản thân mô hình. RAG thường được sử dụng để cung cấp hai loại dữ liệu cho mô hình trong prompt:

- **Dữ liệu người dùng**: Đối với chatbot, đây là dữ liệu liên quan đến người dùng đưa ra truy vấn.
- **Dữ liệu khác không có trong mô hình**: Đối với chatbot, điều này thường bao gồm dữ liệu cơ sở kiến thức hoặc tài liệu liên quan không có sẵn trên internet.

Dữ liệu người dùng thường dễ truy xuất. ID của người dùng đã đăng nhập được ứng dụng biết. Phần lớn dữ liệu người dùng, bao gồm lịch sử trò chuyện hoặc các giao dịch gần đây với hệ thống, có thể được truy cập bằng cách sử dụng truy vấn SQL hoặc NoSQL của cơ sở dữ liệu, bằng cách sử dụng ID người dùng làm khóa.

Dữ liệu không phải của người dùng thường khó truy xuất hơn. Ví dụ, hãy tưởng tượng bạn có một cơ sở kiến thức có hàng nghìn tài liệu có thể giúp người dùng giải quyết vấn đề khi sử dụng sản phẩm của chúng ta. Việc truy xuất một tài liệu cụ thể từ cơ sở kiến thức bằng ID, từ khóa hoặc tiêu đề là dễ dàng. Có thể có những tài liệu sẽ hữu ích cho tình huống cụ thể mà người dùng đang gặp phải, **nhưng làm thế nào để biết tài liệu nào có khả năng hữu ích nhất cho truy vấn của người dùng?** Một foundation model được sử dụng chính xác vì nó giúp chúng ta phân tích và trả lời câu hỏi bằng ngôn ngữ tự nhiên. Sẽ rất chậm và tốn kém nếu bạn cung cấp toàn bộ cơ sở kiến thức cho mô hình với mọi prompt.

Truy xuất dữ liệu có liên quan (Retrieving relevant data)

Tìm kiếm ngữ nghĩa (Semantic search)

Theo truyền thống, hầu hết các tìm kiếm được thực hiện bởi ứng dụng đều là tìm kiếm từ khóa. **Tìm kiếm từ khóa (keyword search) tìm kiếm các kết quả phù hợp bằng cách so khớp văn bản.** Ví dụ, trong cơ sở dữ liệu, bạn có thể tìm kiếm một danh mục với một hoặc nhiều giá trị cụ thể:

```
SELECT id
FROM media
WHERE category IN ('audio', 'video')
```

Đối với một số trường hợp truy xuất dữ liệu, tìm kiếm từ khóa là hoàn toàn phù hợp. Tuy nhiên, khi bạn sử dụng các truy vấn ngôn ngữ tự nhiên, việc chỉ sử dụng tìm kiếm từ khóa có thể không mang lại kết quả tốt nhất.

> Tìm kiếm ngữ nghĩa cố gắng cung cấp kết quả dựa trên ý nghĩa của truy vấn, không phải bằng cách khớp trực tiếp các từ trong truy vấn.

Hãy tưởng tượng truy vấn này từ người dùng ứng dụng du lịch của bạn:
Điểm đến nghỉ dưỡng phổ biến phía nam xích đạo

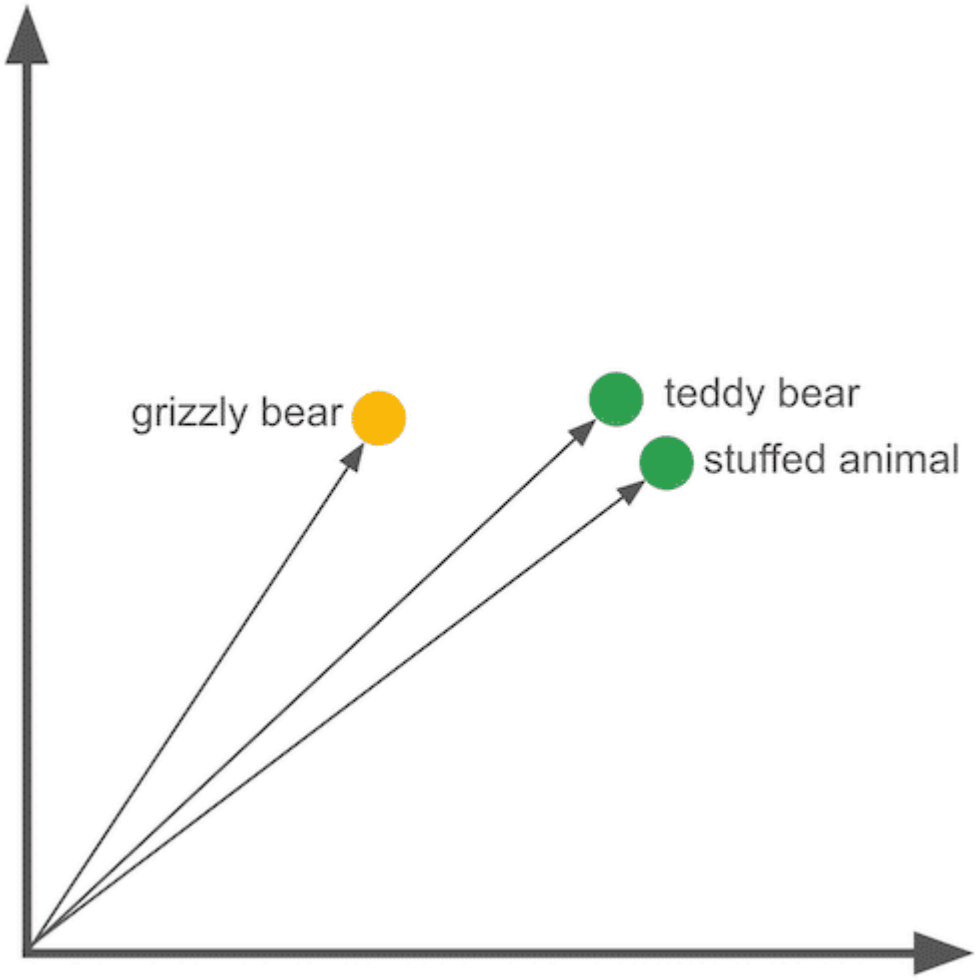
Tìm kiếm ngữ nghĩa có thể hiểu rằng "phía nam xích đạo" lọc các địa điểm nghỉ dưỡng phổ biến chỉ ở bán cầu nam, nhưng tìm kiếm từ khóa sẽ không hiểu được ý nghĩa đó.

Tìm kiếm vector (Vector search)

Để cung cấp tìm kiếm ngữ nghĩa trên dữ liệu của chúng ta, chúng ta có thể sử dụng một kỹ thuật gọi là tìm kiếm vector.

> Tìm kiếm vector sử dụng các vector (danh sách các số) để biểu diễn và tìm kiếm nội dung.

Sự kết hợp của các số xác định độ tương đồng với các chủ đề cụ thể. Đây là một ví dụ đơn giản về cách bạn có thể biểu diễn một số thuật ngữ bằng vector:



So sánh ngữ nghĩa của các thực thể sử dụng vector

Khi chỉ nhìn vào từ khóa, bạn có thể cho rằng *gấu xám (grizzly bear)* và *gấu bông (teddy bear)* gần nhau hơn so với *gấu bông (teddy bear)* và *thú nhồi bông (stuffed animal)*. Tuy

nhiên, về mặt ngữ nghĩa, hầu hết mọi người sẽ đồng ý rằng gấu xám không phải là sự thay thế gần gũi cho gấu bông!

Quy trình tìm kiếm vector (Vector search process)

Quy trình tìm kiếm vector cho Vertex AI Vector Search bao gồm ba bước:

1. Mã hóa dữ liệu (Encode the data)
2. Lập chỉ mục dữ liệu (Index the data)
3. Tìm kiếm dữ liệu (Search the data)

Mã hóa và lập chỉ mục dữ liệu khởi tạo tập dữ liệu, trước khi bạn có thể tìm kiếm dữ liệu. Khi muốn thêm dữ liệu mới vào tập dữ liệu, bạn có thể sử dụng cập nhật theo lô hoặc luồng để sửa đổi tập dữ liệu.

Vertex AI Vector Search dựa trên công nghệ tìm kiếm vector được phát triển bởi Google research.

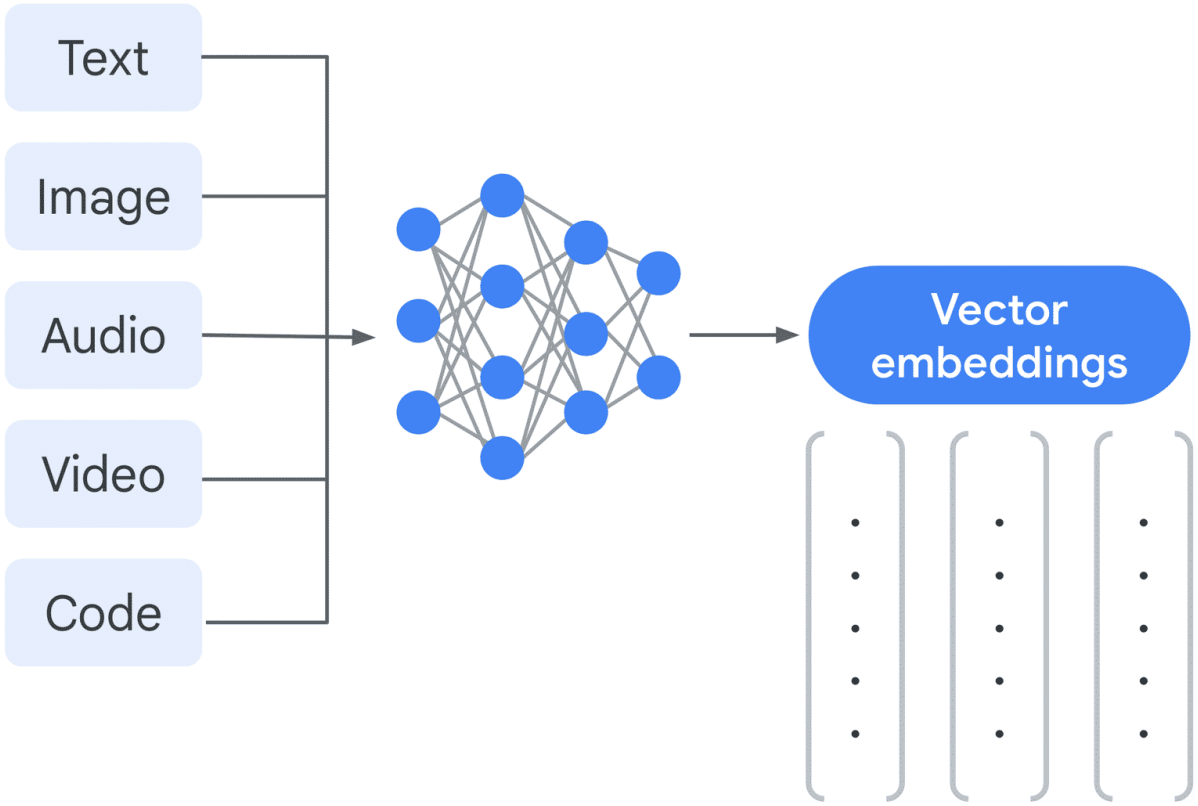
Mã hóa dữ liệu

Bước đầu tiên trong quy trình tìm kiếm vector là mã hóa dữ liệu để có thể tìm kiếm ngữ nghĩa.

Để thực hiện tìm kiếm vector trên các thực thể, các thực thể phải được biểu diễn bằng vector embeddings. Vector embeddings là biểu diễn của bất kỳ nội dung nào sao cho độ tương đồng ngữ nghĩa được biểu diễn bằng khoảng cách giữa các thực thể trong không gian n chiều. Điều này giúp có thể tìm kiếm độ tương đồng, tìm nội dung liên quan trong cơ sở kiến thức, hoặc truy xuất mục phù hợp nhất với truy vấn phức tạp của người dùng.

Tạo vector embeddings cho dữ liệu đòi hỏi hiểu biết ngữ nghĩa về dữ liệu. Ví dụ, "gấu bông" (teddy bear) gần với "thú nhồi bông" (stuffed animal), và cũng gần với hình ảnh của một con gấu bông.

May mắn thay, tạo vector embeddings là một công việc hoàn hảo cho mô hình AI! Nhiều mô hình embedding được huấn luyện trước đã được tạo ra, bao gồm các mô hình làm việc với dữ liệu đa phương thức:



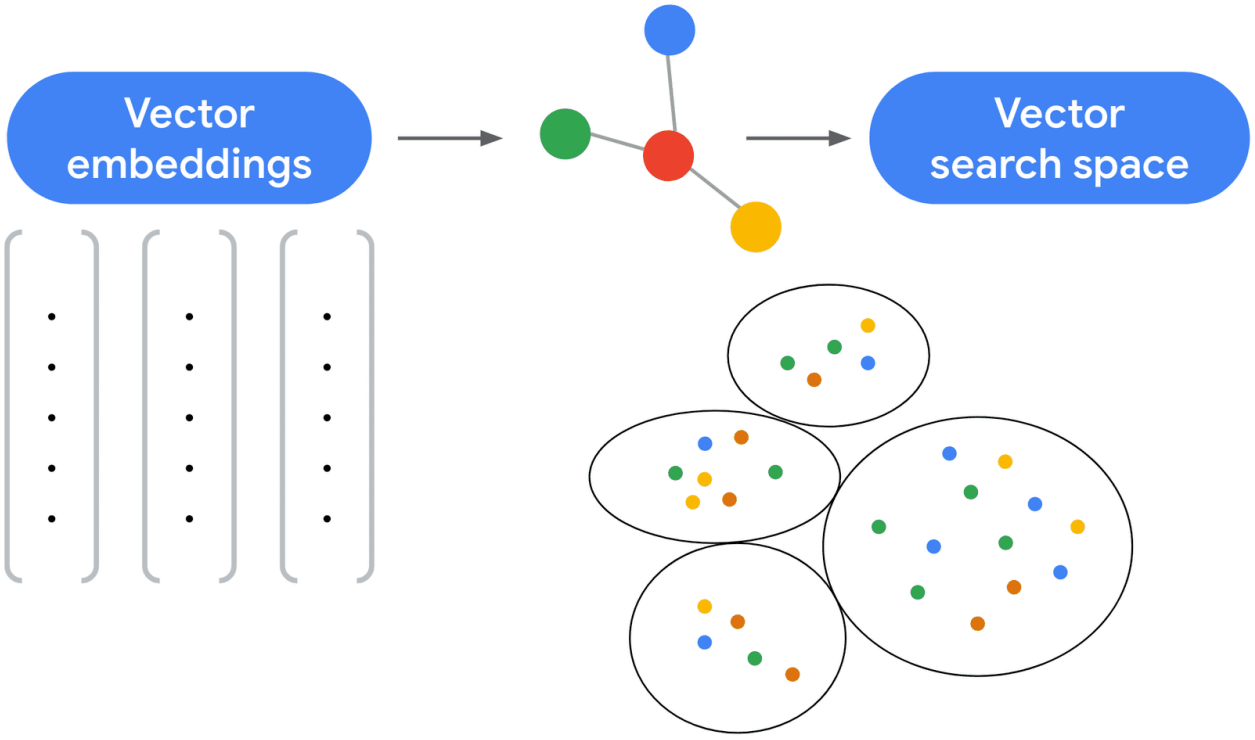
Một mô hình ngôn ngữ máy được hiển thị. Đầu vào của mô hình là văn bản, hình ảnh, âm thanh, video và mã. Đầu ra của mô hình là vector embeddings, một vector cho mỗi đầu vào.

Rất đơn giản để sử dụng các API mô hình embedding để tạo vector embeddings của bạn. Đây là ví dụ Python cho text embedding:

```
# import thư viện embedding
from vertexai.language_models import TextEmbeddingInput,
TextEmbeddingModel

# chọn mô hình embedding của bạn
# ví dụ: textembedding-gecko hoặc multimodalembedding
model = TextEmbeddingModel.from_pretrained(model_name)

# tạo đầu vào đối tượng (ví dụ nhúng một chuỗi văn bản đơn, nhưng có thể
tạo danh sách)
text_embedding_input = TextEmbeddingInput(
    task_type="RETRIEVAL_DOCUMENT",
    title=doc_id,
    text=text)

# lấy embeddings cho danh sách đầu vào văn bản
embeddings = model.get_embeddings([text_embedding_input])
```

Lập chỉ mục dữ liệu

Bước thứ hai trong quy trình tìm kiếm vector là tạo một chỉ mục chứa các embedding của bạn cho phép tìm kiếm nhanh và hiệu quả.



Vector embeddings được thêm vào một chỉ mục. Các phân vùng (nhóm các embedding) được hiển thị. Các phân vùng vector embeddings này là không gian tìm kiếm vector.

Khi bạn lưu trữ vector embeddings trong cơ sở dữ liệu, bạn sẽ cần khả năng truy xuất các vector embeddings tương tự về mặt ngữ nghĩa từ cơ sở dữ liệu. Không giống như nhiều kiểu dữ liệu khác, như văn bản và số, vector không có thứ tự tự nhiên tương ứng với hầu hết các trường hợp sử dụng. Nếu bạn có một cơ sở dữ liệu embeddings đủ lớn, thì tìm kiếm vét cạn để tìm các embedding gần nhất là không thực tế, và không có thuật toán chung nào để tìm hiệu quả danh sách các vector gần nhất, hay các láng giềng gần nhất chính xác.

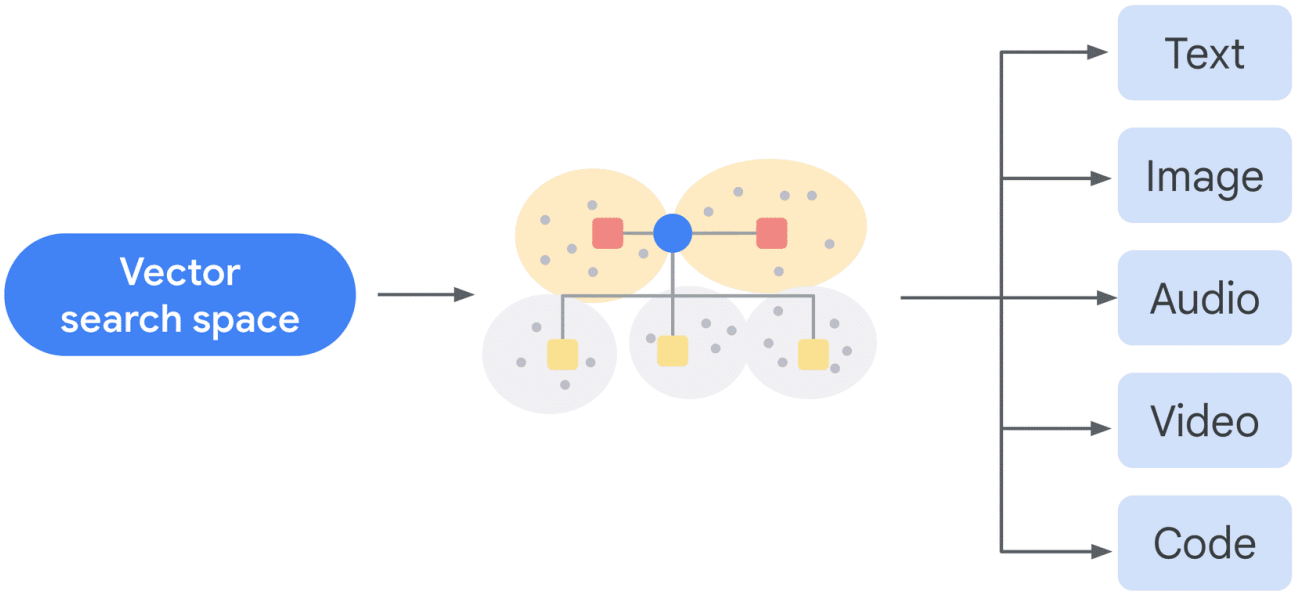
Các cơ sở dữ liệu vector được cung cấp bởi Google Cloud hỗ trợ các chỉ mục chuyên biệt cho vector. Hầu hết trong số này dựa trên Scalable Nearest Neighbor Search (ScaNN), một phiên bản thuật toán tìm kiếm được Google phát triển đã được sử dụng để cung cấp năng lượng cho nhiều sản phẩm của Google, bao gồm tìm kiếm và YouTube. ScaNN cung cấp khả năng trả về danh sách các láng giềng gần đúng một cách rất hiệu quả.

ScaNN sử dụng kỹ thuật phân cụm để nhóm các vector tương tự lại với nhau, gán cho mỗi nhóm, hay phân vùng, một vector đại diện. Điều này sẽ cho phép loại trừ nhanh chóng các phân vùng không tương tự trong quá trình tìm kiếm.

Chỉ mục chỉ định một không gian vector trên các embedding có thể tìm kiếm.

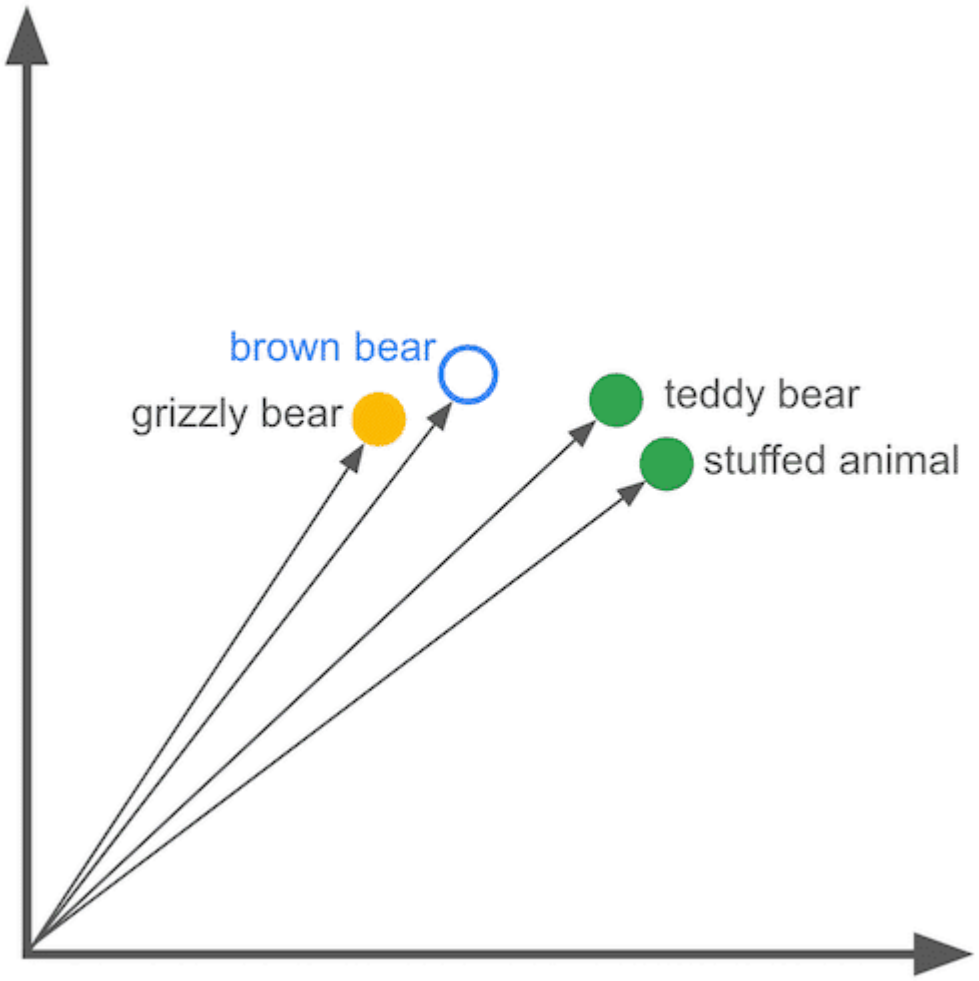
Tìm kiếm dữ liệu

Các embedding giờ đã được mã hóa, lưu trữ và lập chỉ mục trong cơ sở dữ liệu. Phần cuối cùng của quy trình tìm kiếm vector là tìm kiếm ngữ nghĩa trong không gian vector cho các embedding tương tự với truy vấn.



Một tìm kiếm trong không gian tìm kiếm vector hiển thị các phân vùng được đánh dấu gần với vector truy vấn, và bỏ qua các phân vùng ở xa hơn. Các thực thể gần từ các phân vùng được kích hoạt sẽ được trả về.

Trước khi tìm kiếm, bạn sử dụng API embedding để tạo một vector embedding cho truy vấn. Vector cho truy vấn này sẽ gần với các vector của các thực thể tương tự về mặt ngữ nghĩa.



Bốn vector được hiển thị. Vector mới cho "gấu nâu" (brown bear) gần với vector cho "gấu xám" (grizzly bear) nhưng không gần với các vector cho "gấu bông" (teddy bear) và "thú nhồi bông" (stuffed animal).

Gửi một vector truy vấn để tìm các láng giềng gần nhất

Vertex AI Vector Search có thể tìm kiếm hàng tỷ thực thể, được mã hóa dưới dạng vector embeddings, để tìm các thực thể tương tự về mặt ngữ nghĩa, hay các láng giềng gần nhất. Bước đầu tiên của thuật toán tìm kiếm tính toán khoảng cách xấp xỉ giữa vector truy vấn và các vector đại diện cho các phân vùng, và chọn một vài phân vùng gần với vector truy vấn, bỏ qua phần còn lại. Điều này loại bỏ một phần lớn tập dữ liệu, vì các vector trong các phân vùng xa sẽ không đủ gần với vector truy vấn để được đưa vào kết quả.

Tiếp theo, các tính toán nhanh được sử dụng để xấp xỉ khoảng cách giữa vector truy vấn và mỗi vector trong các phân vùng còn lại. Các vector gần nhất được trả về từ tìm kiếm.

Bạn có thể sắp xếp danh sách trả về bằng cách sử dụng các thuật toán như tính toán khoảng cách vector chính xác. Bạn cũng có thể lọc danh sách bằng cách sử dụng các tính năng hoặc thuộc tính bổ sung của các thực thể dữ liệu. Ví dụ, khi tìm kiếm sách mà một người có thể thích, bạn có thể sắp xếp kết quả bằng cách ưu tiên các thể loại hoặc tác giả được người đọc ưa thích.

Chi tiết thêm về tìm kiếm vector nằm ngoài phạm vi của khóa học này. Khóa học Vector Search and Embeddings cung cấp cái nhìn sâu hơn về hoạt động của embedding và thuật toán tìm kiếm vector.

Tăng cường prompt

Trong RAG, các kết quả được truy xuất được truyền đến mô hình như một phần của chính prompt.

Đây là một ví dụ về prompt bao gồm dữ liệu RAG và hướng dẫn cho mô hình về dữ liệu đó:

```
Bạn là một chatbot đại diện trả lời câu hỏi hỗ trợ khách hàng trong chat.
Nhiệm vụ của bạn là trả lời câu hỏi của khách hàng sử dụng dữ liệu được
cung cấp trong phần <DATA>.
- Ngày hiện tại nằm trong <CURRENTDATE>.
- Lịch sử chat của khách hàng nằm trong <CHATHISTORY>.
- Mỗi cuộc chat riêng biệt trong một ngày nằm trong một khối bao gồm ngày
và giống như <CHAT date="2024/06/01">.
- Mỗi tin nhắn chat riêng biệt được chỉ định trong một khối với thẻ <USER>
hoặc <BOT>, tùy thuộc vào nguồn của tin nhắn chat.
- Các tài liệu có thể hữu ích trong việc trả lời nằm trong <DOCS>.
- Mỗi tài liệu riêng biệt nằm trong một khối bao gồm tiêu đề và URL. Thẻ
khối giống như <DOC title="TITLE" URL="https://example.com/...">.

<DATA>
<CURRENTDATE>2024/06/01</CURRENTDATE>
<CHATHISTORY>
<CHAT date="2024/05/24"><USER>...</USER><BOT>...</BOT></CHAT>
...
</CHATHISTORY>
<DOCS>
<DOC title="Đặt lại mật khẩu của bạn"
URL="https://example.com/docs/U4A22">
Làm theo các hướng dẫn này để đặt lại mật khẩu của bạn...
</DOC>
...
</DOCS>
</DATA>
```

HƯỚNG DẪN:

1. Nếu không có dữ liệu để giúp trả lời câu hỏi, hãy trả lời "Tôi không có thông tin này. Vui lòng liên hệ dịch vụ khách hàng."
2. Bạn có thể hỏi câu hỏi tiếp theo nếu nó giúp xác định thông tin cần trả về.

3. Tất cả các khuyến nghị hỗ trợ phải được lấy từ các tài liệu này, và bạn phải trả về tiêu đề và URL của bất kỳ tài liệu nào được sử dụng trong câu trả lời.

CÂU HỎI: Tôi đăng nhập thất bại quá nhiều lần và tôi bị khóa tài khoản. Vui lòng giúp đỡ!
TRẢ LỜI:

Trong prompt này, phần đầu của prompt chỉ định cấu trúc và ý nghĩa của dữ liệu RAG động. Định nghĩa các dấu phân cách định dạng giúp mô hình phân tích cú pháp dữ liệu.

Toàn bộ phần `<DATA>` được tạo động dựa trên dữ liệu được truy xuất. Lịch sử chat được truy xuất từ cơ sở dữ liệu sử dụng ID của người dùng đã đăng nhập, nhưng tìm kiếm vector là cần thiết để tìm các tài liệu có thể hữu ích để trả lời câu hỏi ngôn ngữ tự nhiên.

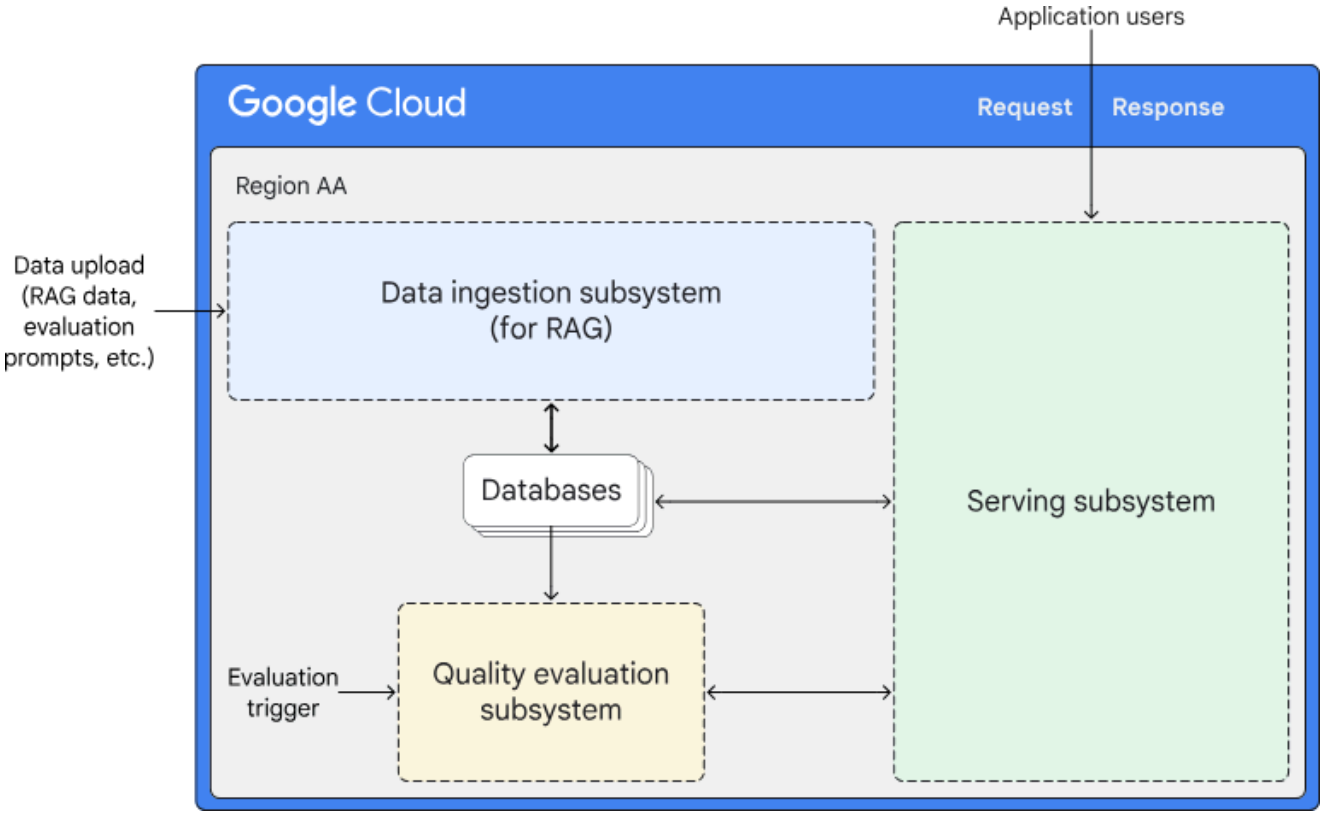
Hãy nhớ sử dụng kỹ thuật prompt để xây dựng prompt cung cấp phản hồi mô hình tốt nhất.

Kiến trúc RAG sử dụng Vertex AI

Kiến trúc tổng quan

Sau khi bạn đã tìm hiểu về các ứng dụng AI tạo sinh và sinh bổ sung bằng truy xuất (RAG), sẽ hữu ích khi thấy cách một ứng dụng AI tạo sinh có khả năng RAG có thể được xây dựng trên Google Cloud.

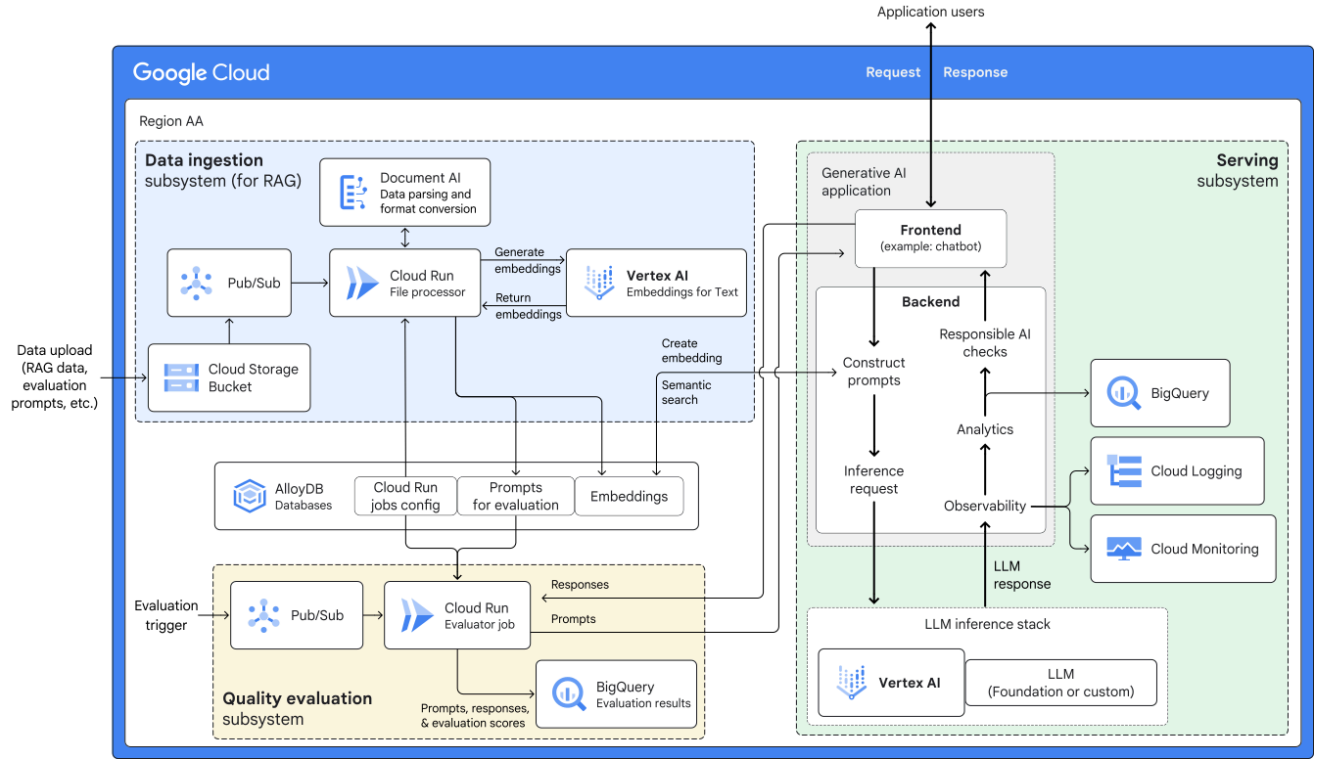
Đây là sơ đồ kiến trúc tổng quan cho một ứng dụng AI tạo sinh bao gồm RAG:



Kiến trúc bao gồm các thành phần sau:

Thành phần	Mục đích	Tương tác
Hệ thống tiếp nhận dữ liệu (Data ingestion subsystem)	Chuẩn bị và xử lý dữ liệu bên ngoài được sử dụng để kích hoạt khả năng RAG	Hệ thống tiếp nhận dữ liệu chỉ tương tác với các hệ thống con khác thông qua lớp cơ sở dữ liệu
Hệ thống phục vụ (Serving subsystem)	Xử lý luồng yêu cầu-phản hồi giữa ứng dụng AI tạo sinh và người dùng	Hệ thống phục vụ truy cập dữ liệu đã tiếp nhận thông qua lớp cơ sở dữ liệu
Hệ thống đánh giá chất lượng (Quality evaluation subsystem - tùy chọn)	Đánh giá chất lượng phản hồi được tạo bởi hệ thống phục vụ	Hệ thống đánh giá chất lượng tương tác trực tiếp với hệ thống phục vụ và với hệ thống tiếp nhận dữ liệu thông qua lớp cơ sở dữ liệu
Cơ sở dữ liệu	Lưu trữ các dữ liệu sau: • Prompts • Vector embeddings của dữ liệu được sử dụng cho RAG • Cấu hình của các công việc serverless trong hệ thống tiếp nhận dữ liệu và đánh giá chất lượng	Tất cả các hệ thống con trong kiến trúc tương tác với cơ sở dữ liệu

Sơ đồ sau đây cho thấy **cái nhìn chi tiết về kiến trúc RAG trên Google Cloud**:



Kiến trúc chi tiết cho một ứng dụng AI tạo sinh có khả năng RAG trên Google Cloud

Các phần sau đây cung cấp mô tả chi tiết về các thành phần và luồng dữ liệu trong từng hệ thống con của kiến trúc.

Hệ thống tiếp nhận dữ liệu

Hệ thống tiếp nhận dữ liệu nhận dữ liệu từ các nguồn bên ngoài như tệp, cơ sở dữ liệu và dịch vụ truyền phát. Dữ liệu tải lên bao gồm các prompt cho đánh giá chất lượng. Hệ thống tiếp nhận dữ liệu cung cấp khả năng RAG trong kiến trúc. Các bước của quy trình tiếp nhận dữ liệu được giải thích ở đây:

1. **Tải dữ liệu lên**

- Dữ liệu được tải lên Bucket Cloud Storage
- Nguồn dữ liệu có thể là người dùng ứng dụng thực hiện tải lên, tiếp nhận cơ sở dữ liệu hoặc dữ liệu truyền phát

2. **Thông báo tải lên**

- Khi dữ liệu được tải lên, một thông báo được gửi đến một chủ đề Pub/Sub

3. **Cloud Run job được kích hoạt**

- Pub/Sub kích hoạt một Cloud Run job để xử lý dữ liệu đã tải lên

4. **Truy xuất dữ liệu cấu hình công việc**

- Cloud Run bắt đầu công việc bằng cách sử dụng dữ liệu cấu hình được lưu trữ trong cơ sở dữ liệu AlloyDB for PostgreSQL

5. **Phân tích và phân đoạn dữ liệu**

- Cloud Run job sử dụng Document AI để chuẩn bị dữ liệu cho xử lý thêm
- Ví dụ, việc chuẩn bị có thể bao gồm phân tích dữ liệu, chuyển đổi dữ liệu sang định dạng yêu cầu và chia dữ liệu thành các đoạn

6. **Tạo vector embeddings**

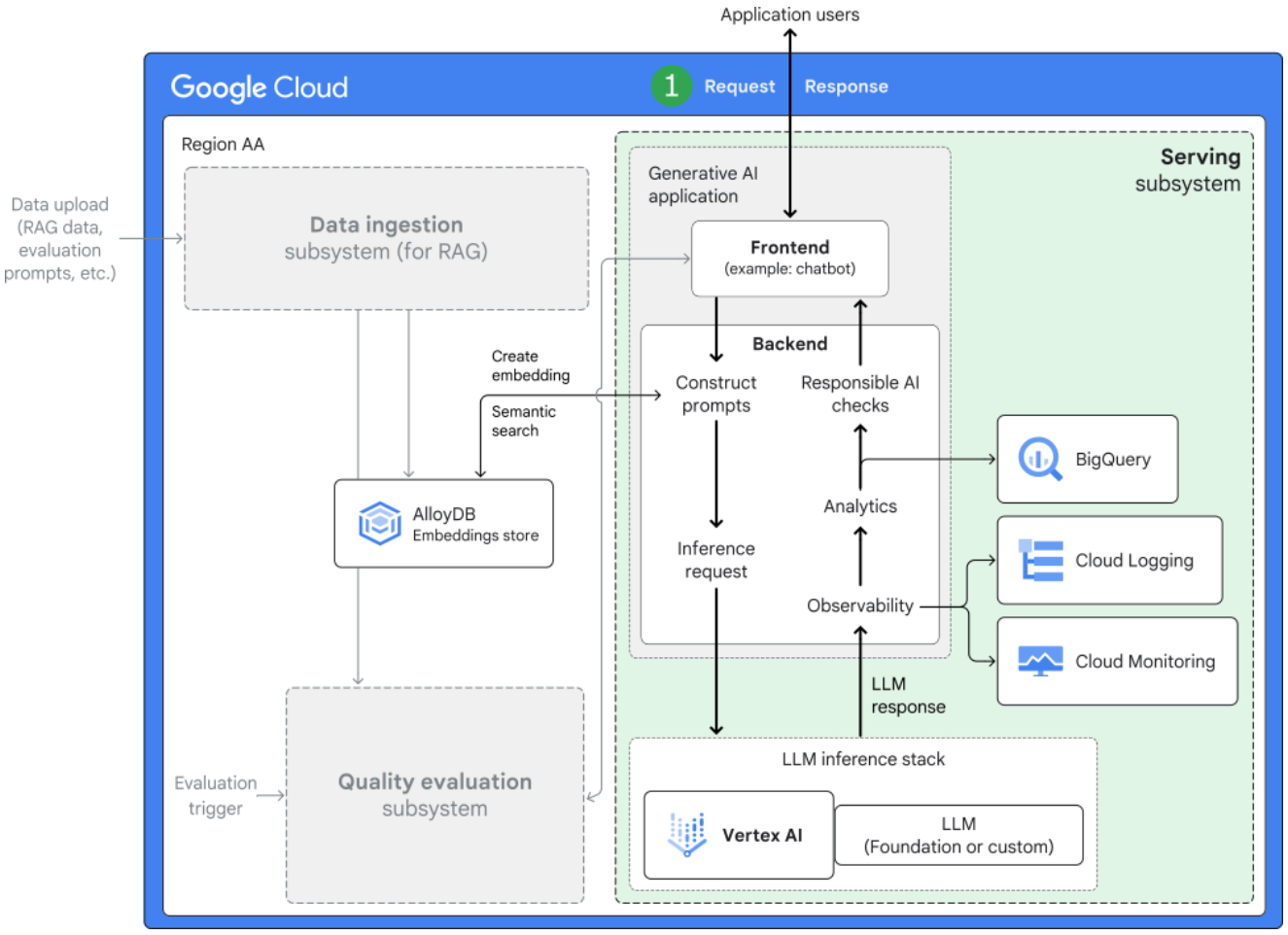
- Cloud Run job sử dụng Vertex AI Embeddings for Text model để tạo vector embeddings của dữ liệu được tiếp nhận
- Lưu ý: Ứng dụng AI tạo sinh nên sử dụng cùng mô hình embeddings và tham số để chuyển đổi các yêu cầu ngôn ngữ tự nhiên thành embeddings

7. **Lưu trữ embeddings**

- Cloud Run lưu trữ embeddings trong cơ sở dữ liệu AlloyDB for PostgreSQL có kích hoạt phần mở rộng pgvector
- Khi hệ thống phục vụ xử lý yêu cầu của người dùng, nó sử dụng embeddings trong cơ sở dữ liệu vector để truy xuất dữ liệu chuyên ngành có liên quan

Hệ thống phục vụ

Hệ thống phục vụ xử lý luồng yêu cầu-phản hồi giữa ứng dụng AI tạo sinh và người dùng. Các bước của quy trình phục vụ được giải thích ở đây:



1. **Yêu cầu được gửi đến ứng dụng**

- Người dùng gửi yêu cầu đến ứng dụng AI tạo sinh thông qua frontend (ví dụ: chatbot hoặc ứng dụng di động)

2. **Tạo embedding**

- Ứng dụng AI tạo sinh chuyển đổi yêu cầu ngôn ngữ tự nhiên thành embeddings
- Lưu ý: Ứng dụng AI tạo sinh nên sử dụng cùng mô hình embeddings và tham số được sử dụng bởi hệ thống tiếp nhận dữ liệu

3. **Truy xuất RAG**

- Ứng dụng thực hiện tìm kiếm ngữ nghĩa cho embedding trong kho vector AlloyDB for PostgreSQL được duy trì bởi hệ thống tiếp nhận dữ liệu
- Tìm kiếm ngữ nghĩa giúp tìm embeddings dựa trên ý định của prompt thay vì nội dung văn bản
- Ứng dụng kết hợp yêu cầu gốc với dữ liệu thô được truy xuất dựa trên embedding phù hợp để tạo prompt theo ngữ cảnh

4. **Gửi prompt được tăng cường**

- Ứng dụng gửi prompt theo ngữ cảnh đến ngăn xếp suy luận LLM chạy trên Vertex AI

5. **Phản hồi từ LLM**

- Ngăn xếp suy luận large language model (LLM) sử dụng LLM tạo sinh, có thể là LLM nền tảng hoặc LLM tùy chỉnh, và tạo phản hồi bị giới hạn trong ngữ cảnh được cung cấp

6. **Lưu trữ nhật ký**

- Ứng dụng có thể lưu trữ nhật ký hoạt động yêu cầu-phản hồi trong Cloud Logging
- Bạn có thể xem và sử dụng nhật ký để giám sát bằng Cloud Monitoring
- Google không truy cập hoặc sử dụng dữ liệu nhật ký

7. **Lưu cho phân tích ngoại tuyến**

- Ứng dụng gửi phản hồi đến BigQuery để phân tích ngoại tuyến

8. **Kiểm tra AI có trách nhiệm**

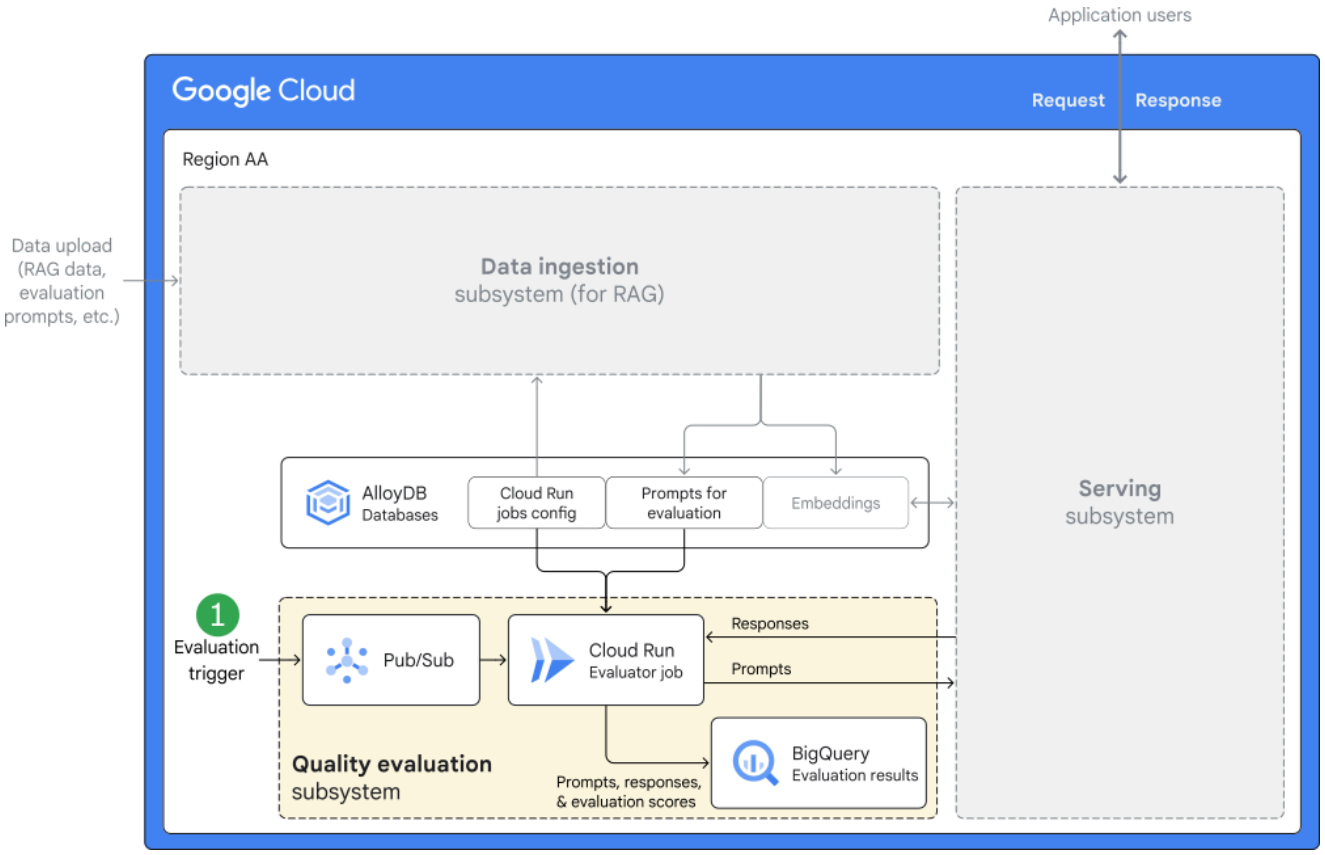
- Ứng dụng lọc phản hồi bằng cách sử dụng bộ lọc AI có trách nhiệm
- Tính linh hoạt của LLM khiến khó dự đoán chính xác loại đầu ra ngoài ý muốn hoặc không lường trước được chúng có thể tạo ra
- Vertex AI Studio có bộ lọc nội dung tích hợp và tính điểm thuộc tính an toàn để giúp đánh giá rủi ro an toàn

9. **Trả về phản hồi**

- Ứng dụng gửi phản hồi đã được lọc đến người dùng thông qua frontend

Hệ thống đánh giá chất lượng

Hệ thống đánh giá chất lượng tùy chọn cung cấp phương pháp tự động để đánh giá chất lượng phản hồi được tạo ra. Các bước đánh giá chất lượng được giải thích ở đây:



1. **Kích hoạt đánh giá**

- Đánh giá được kích hoạt bằng cách gửi tin nhắn Pub/Sub đến một chủ đề Pub/Sub

2. **Bắt đầu công việc đánh giá**

- Pub/Sub kích hoạt một Cloud Run job

3. **Truy xuất cấu hình công việc**

- Cloud Run bắt đầu công việc bằng cách sử dụng dữ liệu cấu hình được lưu trữ trong cơ sở dữ liệu AlloyDB for PostgreSQL

4. **Kéo prompt đánh giá**

- Cloud Run job kéo prompt đánh giá từ cơ sở dữ liệu AlloyDB for PostgreSQL
- Các prompt đã được tải lên cơ sở dữ liệu trước đó bởi hệ thống tiếp nhận dữ liệu

5. **Đánh giá chất lượng phản hồi prompt**

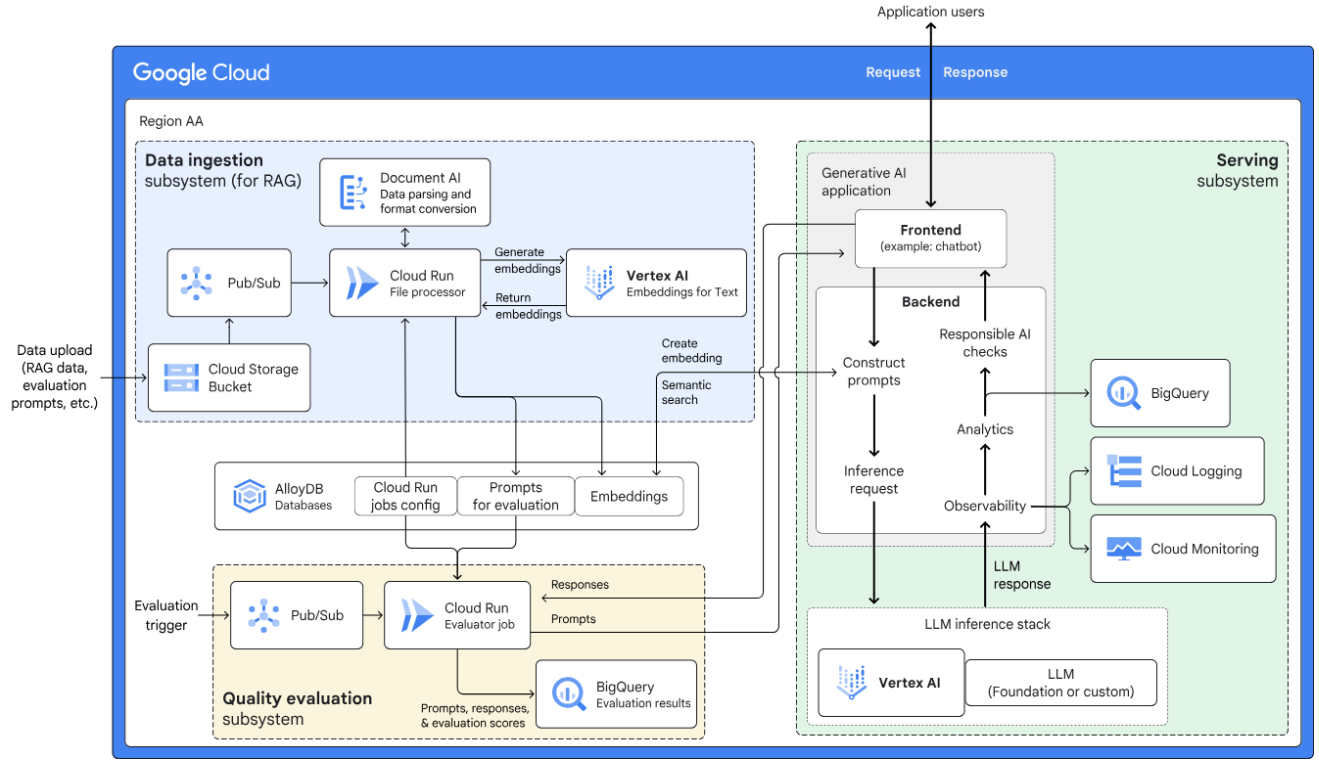
- Cloud Run job sử dụng prompt đánh giá để đánh giá chất lượng phản hồi mà hệ thống phục vụ tạo ra
- Đầu ra của đánh giá này bao gồm điểm đánh giá cho các số liệu như độ chính xác thực tế và tính phù hợp

6. **Lưu trữ điểm prompt**

- Cloud Run lưu trữ điểm đánh giá và prompt và phản hồi đã được đánh giá vào BigQuery để phân tích trong tương lai

Các sản phẩm được sử dụng trong kiến trúc

Đây là kiến trúc tham khảo cho một ứng dụng AI tạo sinh có khả năng RAG sử dụng Vertex AI. Bạn có thể chọn sửa đổi bất kỳ phần nào của kiến trúc. Ví dụ, việc sử dụng AlloyDB là tùy chọn. Hầu hết các cơ sở dữ liệu Google Cloud đều cung cấp khả năng lưu trữ dữ liệu vector embedding.



Cloud Storage

Cloud Storage là kho lưu trữ đối tượng chi phí thấp, không giới hạn cho các loại dữ liệu đa dạng. Dữ liệu có thể được truy cập từ bên trong và bên ngoài Google Cloud, và được sao chép qua nhiều vị trí để dự phòng.

Pub/Sub

Pub/Sub là dịch vụ nhắn tin không đồng bộ và có thể mở rộng, tách biệt các dịch vụ tạo tin nhắn khỏi các dịch vụ xử lý những tin nhắn đó.

Cloud Run

Cloud Run là nền tảng điện toán serverless cho phép bạn chạy các container trực tiếp trên hạ tầng có thể mở rộng của Google.

Document AI

Document AI là nền tảng xử lý tài liệu, lấy dữ liệu phi cấu trúc từ tài liệu và chuyển đổi nó thành dữ liệu có cấu trúc.

Vertex AI
Vertex AI là nền tảng ML cho phép bạn huấn luyện và triển khai các mô hình ML và ứng dụng AI, đồng thời tùy chỉnh LLM để sử dụng trong các ứng dụng hỗ trợ AI.

AlloyDB for PostgreSQL
AlloyDB for PostgreSQL là dịch vụ cơ sở dữ liệu tương thích với PostgreSQL được quản lý hoàn toàn, được thiết kế cho các khối lượng công việc đòi hỏi cao nhất của bạn, bao gồm cả xử lý giao dịch và phân tích kết hợp.

BigQuery
BigQuery là kho dữ liệu doanh nghiệp giúp bạn quản lý và phân tích dữ liệu với các tính năng tích hợp như máy học, phân tích địa lý không gian và phân tích kinh doanh.

Cloud Logging
Cloud Logging là hệ thống quản lý nhật ký thời gian thực với khả năng lưu trữ, tìm kiếm, phân tích và cảnh báo.

Cloud Monitoring
Cloud Monitoring là dịch vụ cung cấp khả năng hiển thị về hiệu suất, tính khả dụng và tình trạng của ứng dụng và cơ sở hạ tầng của bạn.

Tóm tắt

Trong module này, bạn đã học về sinh bổ sung bằng truy xuất (RAG), tìm kiếm vector và vector embeddings, và bạn đã xem xét một kiến trúc tham khảo có khả năng RAG trên Google Cloud.

Tóm tắt những gì bạn đã học:

✓ Mô tả các phương pháp cải thiện độ chính xác của mô hình nền tảng (foundation model):

- Sử dụng RAG để bổ sung thông tin có liên quan từ nguồn dữ liệu đáng tin cậy
- Tối ưu hóa prompt để hướng dẫn mô hình tạo ra kết quả chính xác hơn
- Áp dụng các kỹ thuật đánh giá chất lượng để kiểm tra và cải thiện độ chính xác

✓ Mô tả cách vector embeddings có thể được sử dụng để truy xuất dữ liệu RAG có liên quan cho một truy vấn cụ thể:

- Chuyển đổi văn bản và dữ liệu thành vector embeddings để nắm bắt ý nghĩa ngữ nghĩa
- Sử dụng tìm kiếm vector để tìm thông tin tương tự về mặt ngữ nghĩa
- Áp dụng các kỹ thuật như ScaNN để tìm kiếm láng giềng gần nhất một cách hiệu quả

✓ Mô tả các thành phần của ứng dụng AI tạo sinh có khả năng RAG trên Google Cloud:

- Hệ thống tiếp nhận dữ liệu để xử lý và chuẩn bị dữ liệu RAG
- Hệ thống phục vụ để xử lý yêu cầu và tạo phản hồi

- Hệ thống đánh giá chất lượng để đảm bảo chất lượng phản hồi
- Cơ sở dữ liệu để lưu trữ prompt, vector embeddings và cấu hình

Câu hỏi kiểm tra

Câu hỏi 1
Mục tiêu chính của Retrieval Augmented Generation (RAG) là gì?

- Để giảm chi phí đào tạo và triển khai các mô hình nền tảng
- Để thay thế các mô hình nền tảng bằng các mô hình nhỏ hơn, hiệu quả hơn
- Để loại bỏ nhu cầu kỹ thuật prompt trong các ứng dụng AI tạo sinh
- Để cải thiện độ chính xác và sự phù hợp của phản hồi từ các mô hình nền tảng

Câu hỏi 2
Vai trò của vector embeddings trong hệ thống RAG là gì?

- Để biểu diễn văn bản và các loại dữ liệu khác theo cách nắm bắt ý nghĩa ngữ nghĩa
- Để tạo phản hồi văn bản giống con người dựa trên prompt đã cho
- Để đánh giá chất lượng và độ chính xác của phản hồi của mô hình
- Để nén các tài liệu lớn thành các phần nhỏ hơn, dễ quản lý hơn

Câu hỏi 3
Trong ngữ cảnh RAG, "grounding" đề cập đến điều gì?

- Tinh chỉnh mô hình để cải thiện hiệu suất trên một nhiệm vụ cụ thể
- Đào tạo mô hình nền tảng trên một tập dữ liệu cụ thể
- Giảm sự phụ thuộc của mô hình vào các nguồn kiến thức bên ngoài
- Kết nối đầu ra của mô hình với các nguồn thông tin có thể xác minh

Câu hỏi 4
Mục đích của phần mở rộng pgvector trong AlloyDB for PostgreSQL là gì?

- Để kích hoạt việc lưu trữ và truy vấn vector embeddings
- Để tạo phản hồi ngôn ngữ tự nhiên dựa trên truy vấn của người dùng
- Để đánh giá chất lượng của phản hồi được tạo ra bởi mô hình
- Để cải thiện hiệu suất của tìm kiếm từ khóa

Câu hỏi 5
Thành phần nào của kiến trúc RAG chịu trách nhiệm thêm dữ liệu bên ngoài vào quy trình RAG?

- Hệ thống đánh giá chất lượng
- Hệ thống tiếp nhận dữ liệu
- Hệ thống phục vụ
- Lớp cơ sở dữ liệu

Câu hỏi 6
Loại dữ liệu nào sau đây nên được lưu trữ ở vị trí khác với kho dữ liệu Vertex AI Search?

- Dữ liệu huấn luyện
- Dữ liệu phi cấu trúc
- Dữ liệu trang web
- Dữ liệu có cấu trúc

Câu hỏi 7
Mục đích của việc sử dụng tìm kiếm ngữ nghĩa trong hệ thống phục vụ RAG là gì?

- Để tạo phản hồi sáng tạo không liên quan trực tiếp đến truy vấn
- Để lọc ra các tài liệu không liên quan hoặc chất lượng thấp từ kết quả tìm kiếm
- Để tìm các kết quả khớp từ khóa chính xác trong truy vấn của người dùng
- Để truy xuất tài liệu dựa trên ý nghĩa và ý định của truy vấn

Câu hỏi 8
Kỹ thuật nào sau đây được sử dụng để cải thiện hiệu suất của mô hình nền tảng cho các tác vụ cụ thể?

- Tìm kiếm vector
- Grounding với Google Search
- Supervised tuning
- Retrieval Augmented Generation (RAG)