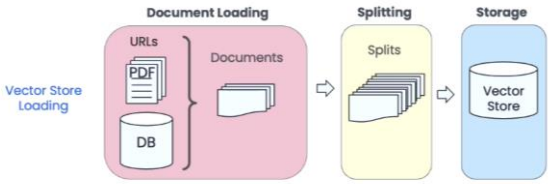
Based on course link: https://learn.deeplearning.ai/langchain-chat-with-your-data/lesson/1/introduction

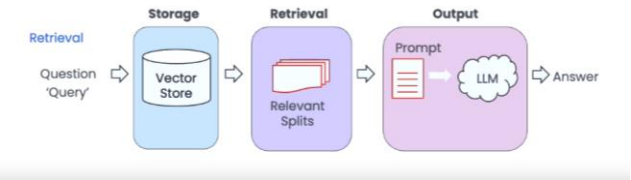
Introduction

Để sử dụng GPT với dữ liệu của mình, nó tồn tại 2 big step:

- **Step 1: Load dữ liệu vào bộ lưu trữ (Storage)**, dưới đây là workflow để load dữ liệu



- **Step 2: Truy xuất kết quả từ bộ lưu trữ (Storage)** qua câu truy vấn (query) của user.

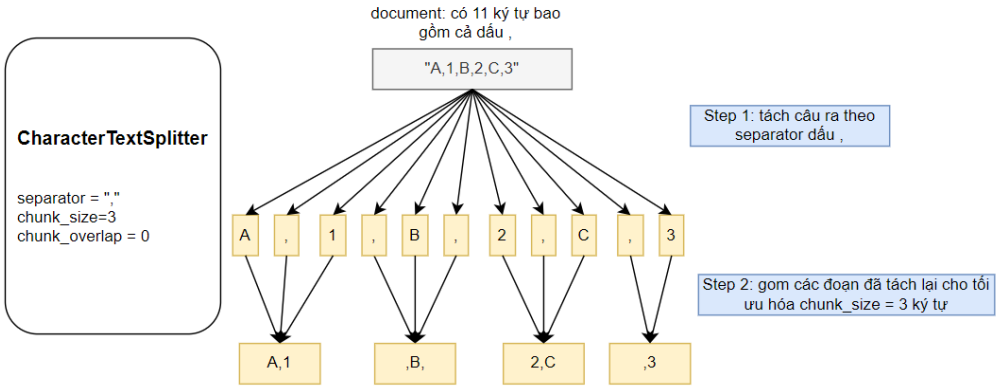


Part 1: Load dữ liệu vào bộ lưu trữ (Storage)

1. **Document loading:** Langchain có thể load dữ liệu từ rất nhiều nguồn khác nhau.
    a. Các dữ liệu unstructured data: Audio từ Youtube [2], Wikipedia [3], Notion [5], Word [4]
    b. Các dữ liệu structured data: OpenWeatherMap [1], Pandas [6]

File xanh dương là file local, File đỏ là file public trên mạng. Tất cả các type trên sau khi được load vào Langchain sẽ được lưu dưới dạng Document [7]

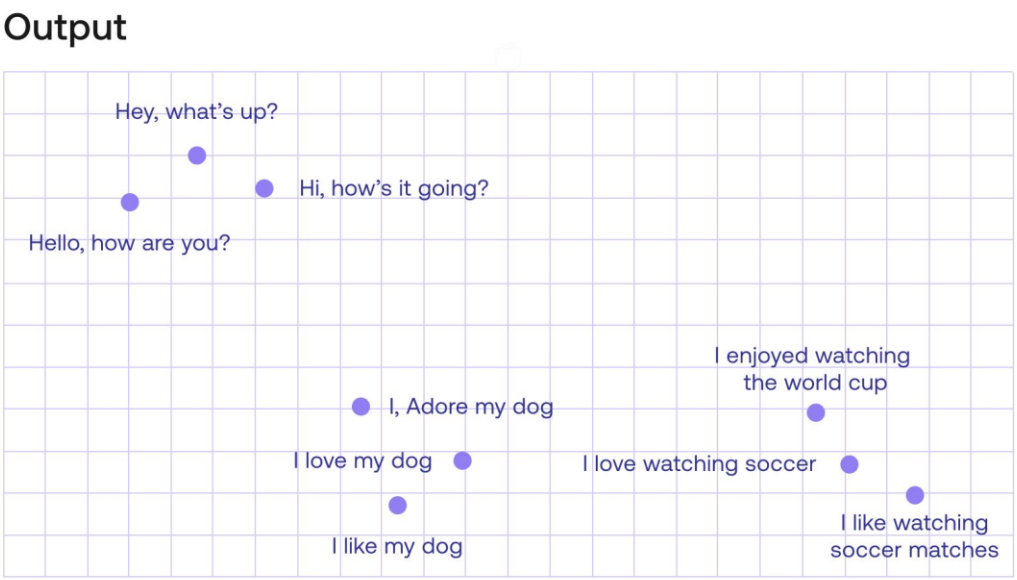
2. **Document Splitting:** Langchain split các document ra thành từng chunk và có dạng [str] (list của string).
    a. Thông số khi splitting: chunk_size và chunk_overlap:
        i. chunk_size: là số lượng ký tự tối đa có trong một trong chunk
        ii. chunk_overlap: số lượng ký tự được thêm vào từ những chữ cuối của chunk trước.
    b. Các dạng splitter: Sẽ cố gắng tách để tối ưu hóa chunk nhất, nghĩa là một chunk phải chứa nhiều ký tự nhất có thể. [14]



        i. Split by character [8] : Là dạng split đơn giản nhất, sẽ tách chunk dựa trên separator mà qua đó giảm thiểu tối đa số lượng các chunk dựa trên chunk_size(maximum character trong chunk có thể có) và chunk_overlap (số lượng character thêm vào từ chunk trước đó)
        ii. Recursively split by character [9]: Cũng tách trên ký tự (character) nhưng sẽ có thêm rule để tách.
        iii. Split by token [10]: LLM đếm số lượng token trong câu query, nên để kiểm soát số lượng token truyền vào, thì ta tách doc bằng token sẽ tránh việc bị lố token limit.

3. **Storage:** Dùng **vectorstore** để lưu trữ content và **embedding** của từng chunk.

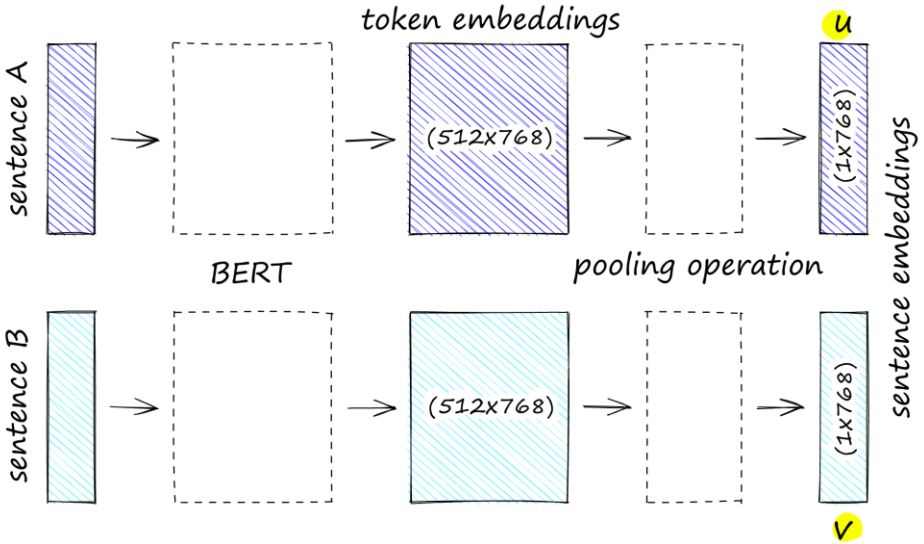
    a. **Embedding**: là dạng toán học vector của một câu văn bản, được compute từ một embedding model. Khoảng cách (distance, eg: **cosine similarity**) giữa hai vector đo lường mức độ liên quan
    b. Giải thích Embedding được tạo ra như thế nào? [15]



Ví dụ về các điểm vector embedding của các câu văn bản

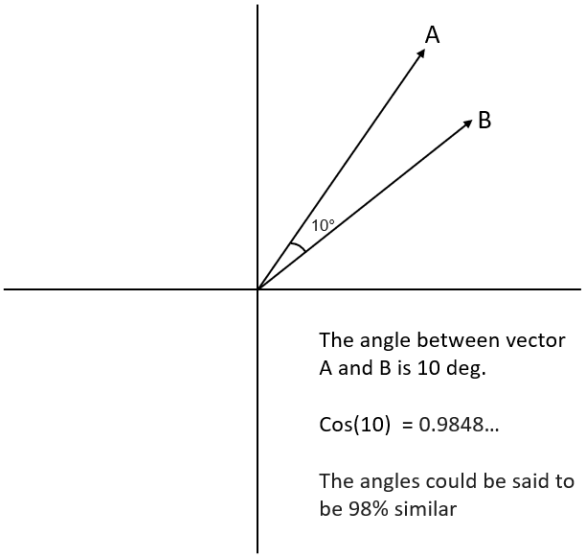
i. Bản default hiện tại là bản "`text-embedding-ada-002`" : Là bản không được public opensource [11] mà chỉ có thể dùng qua API.

ii. Phần nâng cao: Flow chạy để sinh ra embedding vector cho câu



1. *BERT là một mô hình embedding, tạo ra embedding vector cho từng chữ của câu*
2. *Embedding của từng chữ có shape (1,768), một câu có 512 ký tự -> embedding của một câu qua BERT là (512,768)*
3. *ChatGPT lấy trung bình cộng (mean) của chiều thứ nhất -> (1,768)*

iii. **Hàm so sánh cosine similarity** [12]: Cosine similarity biểu diễn độ giống nhau (độ trùng lặp) giữa 2 vector khi nó càng gần nhau.



Góc giữa vector A và B có số càng nhỏ, thì chỉ số cosine similarity càng về 1

$$\cos\theta = \frac{\vec{a} \cdot \vec{b}}{\|\vec{a}\| \cdot \|\vec{b}\|}$$

Công thức tính góc (tính cosine similarity) của 2 vector

b. Vectorstore [13]**:** Là database dùng chuyên cho việc lưu trữ và truy xuất embedding. Vectorstore có thể tìm kiếm vector embedding nhanh chóng.

i. Chroma DB: Là một **opensource vectorstore** được Langchain sử dụng để lưu trữ embedding và original data, và dùng công thức **cosine distance** làm thước đo để tìm ra docs có độ giống nhau nhất với query.

References:

[1] https://python.langchain.com/docs/integrations/providers/weather
[2] https://python.langchain.com/docs/integrations/providers/youtube
[3] https://python.langchain.com/docs/integrations/providers/wikipedia
[4] https://python.langchain.com/docs/integrations/document_loaders/microsoft_word
[5] https://python.langchain.com/docs/integrations/document_loaders/notion
[6] https://python.langchain.com/docs/integrations/document_loaders/pandas_dataframe
[7] https://docs.langchain.com/docs/components/schema/document

[8] https://python.langchain.com/docs/modules/data_connection/document_transformers/text_splitters/character_text_splitter
[9] https://python.langchain.com/docs/modules/data_connection/document_transformers/text_splitters/recursive_text_splitter
[10] https://python.langchain.com/docs/modules/data_connection/document_transformers/text_splitters/split_by_token
[11] https://arxiv.org/ftp/arxiv/papers/2306/2306.12689.pdf
[12] https://builtin.com/machine-learning/cosine-similarity
[13] https://www.datacamp.com/tutorial/chromadb-tutorial-step-by-step-guide
[14] https://learn.deeplearning.ai/langchain-chat-with-your-data/lesson/3/document-splitting
[15] https://docs.google.com/document/d/1642iHY0JqGK_lcn59MXjLjQjISBtkRdG/edit?usp=sharing&ouid=108537208205025925654&rtpof=true&sd=true

Bài tập

1. Giả sử, Châu đang tổng hợp thêm về thông tin của nhóm nhạc BTS mới với phần youtube dưới đây:
    a. BTS Documentary 2023: https://www.youtube.com/watch?v=vVjE_AHRWMo
2. Splitting các bài nói này ra các chunk khác nhau để đưa vào VectorDB