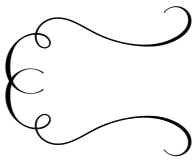
GlobalTech

NLP Project

Các kỹ thuật chunking trong hệ thống RAG hiện nay



Tác giả

Nguyễn Khánh Nhân

Contents

1 - Chiến lược Phân mảnh Xác định (Deterministic Strategies)

Mặc dù sự chú ý hiện nay đổ dồn vào AI, các chiến lược phân mảnh xác định vẫn đóng vai trò nền tảng (baseline) quan trọng nhờ tốc độ xử lý vượt trội và chi phí thấp. Hiểu rõ các phương pháp này là điều kiện tiên quyết để áp dụng các kỹ thuật nâng cao.

1.1 Fixed-Size Chunking: Tối ưu hóa Token và Overlap

Phương pháp phân mảnh theo kích thước cố định (Fixed-Size Chunking) là kỹ thuật phổ biến nhất, nơi văn bản được chia thành các đoạn có số lượng token hoặc ký tự nhất định. Tuy nhiên, sự đơn giản của nó ẩn chứa những sắc thái kỹ thuật quan trọng. Việc lựa chọn bộ tách từ (tokenizer) – ví dụ như tiktoken cho các mô hình OpenAI hay BertTokenizer cho các mô hình HuggingFace – có ảnh hưởng lớn đến việc bảo toàn ý nghĩa của từ. Một từ bị cắt đôi giữa hai chunk (do giới hạn ký tự cứng) có thể làm biến dạng hoàn toàn ngữ nghĩa vector của cả hai chunk đó.

Để khắc phục vấn đề đứt gãy ngữ nghĩa tại biên giới cắt, kỹ thuật Cửa sổ Trượt (Sliding Window) với độ chồng lấn (Overlap) được áp dụng. Dữ liệu thực nghiệm và các thực hành tốt nhất trong ngành (Industry Best Practices) đề xuất tỷ lệ chồng lấn từ 10-20%. Ví dụ, với kích thước chunk là 500 token, độ chồng lấn nên là 50-100 token. Sự chồng lấn này đảm bảo rằng các cấu trúc ngữ pháp quan trọng hoặc các mối quan hệ từ vựng nằm ở cuối chunk này sẽ xuất hiện lại ở đầu chunk kia, duy trì sự liên tục của mạch văn.

Tuy nhiên, Fixed-Size Chunking có nhược điểm chí mạng là "mù cấu trúc" (structure-blind). Nó đối xử với tiêu đề, đoạn văn, bảng biểu và chú thích như nhau. Điều này dẫn đến tình trạng các bảng dữ liệu bị cắt ngang, biến các dòng dữ liệu thành các chuỗi văn bản vô nghĩa khi tách khỏi hàng tiêu đề, hoặc mã nguồn (code) bị cắt giữa chừng, làm mất đi logic thực thi.

1.2 Recursive Character Splitting: LangChain

Recursive Character Splitting (Phân mảnh Đệ quy Ký tự) đại diện cho một bước tiến hóa thông minh hơn của Fixed-Size. Thay vì cắt một cách mù quáng tại token thứ 512, thuật toán này sử dụng một danh sách các ký tự phân cách theo thứ tự ưu tiên (thường là `["\n\n", "\n", " ", ""]`).

Cơ chế hoạt động như sau: Thuật toán đầu tiên cố gắng cắt văn bản tại các điểm ngắt đoạn (`\n\n`). Nếu đoạn văn bản kết quả vẫn lớn hơn kích thước chunk mục tiêu, nó sẽ chuyển sang cấp độ ưu tiên tiếp theo là ngắt câu (`\n`), và sau đó là khoảng trắng. Phương pháp này cố gắng giữ cho các đoạn văn và câu được nguyên vẹn nhất có thể, bảo toàn cấu trúc ngữ nghĩa tự nhiên của văn bản. Đây là lý do tại sao Recursive Character Splitting trở thành bộ chia mặc định trong các thư viện RAG phổ biến như LangChain. Mặc dù vậy, nó vẫn gặp khó khăn với các tài liệu có cấu trúc phức tạp hoặc định dạng không chuẩn.

2 - Semantic Chunking: Phân mảnh dựa trên Ngữ nghĩa

Chiến lược	Cơ chế Hoạt động	Ưu điểm Chính	Nhược điểm Chính	Trường hợp Sử dụng Tối ưu
Fixed-Size	Cắt theo số lượng token/ký tự cố định + Overlap.	Tốc độ cao, chi phí thấp, dễ triển khai.	Mất ngữ cảnh, phá vỡ cấu trúc câu/bảng.	Dữ liệu văn bản thô, prototyping nhanh.
Recursive	Cắt theo danh sách ưu tiên (`\n\n` , `\n`).	Giữ nguyên vẹn đoạn văn/câu tốt hơn.	Vẫn có thể cắt sai cấu trúc phức tạp.	Văn bản văn xuôi thông thường.

Sự ra đời của Semantic Chunking đánh dấu bước chuyển mình từ việc xử lý văn bản như một chuỗi ký tự sang xử lý văn bản như một luồng ý nghĩa. Phương pháp này không quan tâm đến số lượng ký tự, mà quan tâm đến việc khi nào chủ đề của văn bản thay đổi.

2.1 Cơ chế Phân tích Độ tương đồng (Similarity Analysis)

Quy trình Semantic Chunking thường bao gồm bốn giai đoạn chính:

1. Phân đoạn câu (Sentence Segmentation): Tài liệu được chia thành các câu riêng lẻ bằng cách sử dụng các mô hình NLP chuyên biệt (như NLTK hoặc SpaCy).
2. Tạo Embedding (Embedding Generation): Mỗi câu được chuyển đổi thành một vector embedding.
3. Tính toán Khoảng cách (Distance Calculation): Hệ thống tính toán độ tương đồng cosine (cosine similarity) giữa các vector của các câu liền kề.
4. Hình thành Chunk (Chunk Formation): Hệ thống quét qua chuỗi các độ tương đồng này. Khi độ tương đồng giữa câu t và câu $t+1$ giảm xuống dưới một ngưỡng (threshold) nhất định, điều đó báo hiệu sự thay đổi về chủ đề hoặc ngữ cảnh. Một điểm ngắt (break point) được tạo ra tại đó để bắt đầu một chunk mới.

2.2 Các Biến thể và Thách thức Kỹ thuật

Thách thức lớn nhất của Semantic Chunking là việc xác định ngưỡng cắt. Một ngưỡng cố định (fixed threshold) thường không hiệu quả do độ biến thiên ngữ nghĩa khác nhau giữa các loại tài liệu. Do đó, các phương pháp thống kê như "Percentile-based Splitting" (chia dựa trên phân vị) được sử dụng. Ví dụ, LlamaIndex cho phép thiết lập ngưỡng dựa trên phân vị thứ 95 của độ khác biệt, giúp thuật toán tự thích nghi với từng tài liệu cụ thể.

Các kỹ thuật nâng cao hơn sử dụng thuật toán gom cụm (Clustering) như K-Means hoặc DBSCAN để nhóm các câu không liền kề nhưng có cùng chủ đề vào một chunk logic, mặc dù điều này làm mất đi tính tuần tự của văn bản gốc

3 - Agentic Chunking: Trí tuệ Nhân tạo trong Phân mảnh

Agentic Chunking đại diện cho đỉnh cao hiện tại của sự linh hoạt, nơi việc phân mảnh không còn dựa vào thuật toán cứng nhắc mà được giao phó cho một Mô hình Ngôn ngữ Lớn (LLM) đóng vai trò như một tác nhân (agent) thông minh.

3.1 Mô hình "LLM-as-a-Judge" trong Phân mảnh

Trong phương pháp này, LLM phân tích văn bản giống như một biên tập viên con người. Nó đọc qua tài liệu, hiểu cấu trúc lập luận, và quyết định chính xác đâu là điểm kết thúc của một ý tưởng và bắt đầu của ý tưởng khác. Quy trình này thường được triển khai dưới dạng một chuỗi suy luận (chain-of-thought):

1. Phân tích Mệnh đề (Proposition Extraction): LLM tách văn bản thành các mệnh đề độc lập.
2. Đánh giá Ngữ cảnh: LLM xem xét mệnh đề hiện tại có liên quan chặt chẽ với mệnh đề trước đó không.
3. Quyết định: Nếu có, gộp vào chunk hiện tại. Nếu không, tạo chunk mới.
4. Làm giàu (Enrichment): Đồng thời, LLM có thể tạo ra tiêu đề hoặc tóm tắt cho chunk đó ngay lập tức.

3.2 Hiệu quả Vượt trội và Rào cản Chi phí

Agentic Chunking tạo ra các chunk có chất lượng ngữ nghĩa cao nhất, gần với tư duy con người nhất. Nó có khả năng xử lý các tài liệu phi cấu trúc phức tạp mà các bộ regex hay semantic threshold thất bại. Tuy nhiên, đây là phương pháp tốn kém nhất. Việc sử dụng LLM (đặc biệt là các model mạnh như GPT-4o hay Claude 3.5 Sonnet) để xử lý toàn bộ kho dữ liệu là cực kỳ đắt đỏ và chậm chạp. Do đó, Agentic Chunking thường chỉ được khuyến nghị cho các bộ dữ liệu nhỏ, giá trị cao (high-value corpus), hoặc các ứng dụng "offline" nơi độ trễ indexing không phải là vấn đề nghiêm trọng.

4 - Các Kiến trúc Bảo toàn Ngữ cảnh (Contextual Architectures)

Một trong những vấn đề lớn nhất của RAG truyền thống là "mất ngữ cảnh cục bộ". Khi một chunk bị tách khỏi tài liệu gốc, nó thường mất đi các thông tin tiền đề quan trọng (ví dụ: đại từ "nó", "dự án này", "bà ấy"). Năm 2024 và 2025 chứng kiến sự ra đời của hai kỹ thuật đột phá để giải quyết vấn đề này: Late Chunking và Contextual Retrieval.

4.1 Late Chunking (Jina AI): Đảo ngược Quy trình Embedding

Jina AI đã giới thiệu khái niệm "Late Chunking" để khắc phục nhược điểm của việc cắt văn bản trước khi embedding (Early Chunking).

Cơ chế Kỹ thuật: Trong Early Chunking, các chunk được cắt rời rạc và đưa vào mô hình embedding một cách độc lập. Do cơ chế Attention của Transformer chỉ hoạt động trong phạm vi đầu vào, các token trong Chunk 2 không thể "nhìn thấy" các token trong Chunk 1. Late Chunking đảo ngược quy trình này:

1. Long-Context Embedding: Sử dụng một mô hình embedding hỗ trợ ngữ cảnh dài (ví dụ: 8192 token của jina-embeddings-v2) để mã hóa toàn bộ tài liệu hoặc một phần lớn của nó trong một lần duy nhất.

2. Ma trận Attention Toàn cục: Trong quá trình này, cơ chế Attention cho phép mỗi token trong tài liệu tương tác với mọi token khác, bất kể khoảng cách. Vector đại diện cho mỗi token lúc này đã chứa thông tin ngữ cảnh của toàn bộ tài liệu.

3. Late Splitting & Pooling: Chỉ sau khi embedding xong, hệ thống mới cắt chuỗi vector token này thành các đoạn nhỏ (chunk) và thực hiện Mean Pooling để tạo ra vector cho chunk.

Kết quả: Vector của mỗi chunk trong Late Chunking mang đậm dấu ấn ngữ cảnh của toàn bài. Thử nghiệm cho thấy Late Chunking vượt trội so với Naive Chunking trong việc truy xuất các thông tin phụ thuộc ngữ cảnh xa mà không làm tăng kích thước lưu trữ index.

4.2 Contextual Retrieval (Anthropic): Làm giàu Ngữ cảnh bằng LLM

Anthropic tiếp cận vấn đề từ góc độ tiền xử lý dữ liệu (Preprocessing). Thay vì thay đổi cách embedding, họ thay đổi nội dung của chunk.

Quy trình Hoạt động: Với mỗi chunk, hệ thống sử dụng một LLM để tạo ra một bản tóm tắt ngắn gọn về ngữ cảnh của tài liệu mẹ và gắn (prepend) nó vào đầu chunk đó trước khi embedding.

- Chunk gốc: "Doanh thu tăng 20% so với quý trước."
- Contextual Chunk: "Trong báo cáo tài chính Q3/2024 của Công ty ABC, mảng dịch vụ đám mây ghi nhận tăng trưởng mạnh. Doanh thu tăng 20% so với quý trước."

4.3 Parent Document Retrieval (Small-to-Big)

Đây là kỹ thuật tách biệt giữa "đơn vị tìm kiếm" (search unit) và "đơn vị sinh" (generation unit).

Cơ chế: Hệ thống chia tài liệu thành các chunk con rất nhỏ (child chunks) để tối ưu hóa độ chính xác tìm kiếm vector. Tuy nhiên, mỗi chunk con được liên kết với một chunk cha (parent chunk) lớn hơn hoặc thậm chí là toàn bộ tài liệu.

Truy xuất: Khi tìm thấy chunk con, hệ thống không trả về chunk con đó cho LLM mà trả về chunk cha. Điều này đảm bảo LLM có đầy đủ ngữ cảnh xung quanh thông tin tìm được để trả lời câu hỏi một cách toàn diện.

END