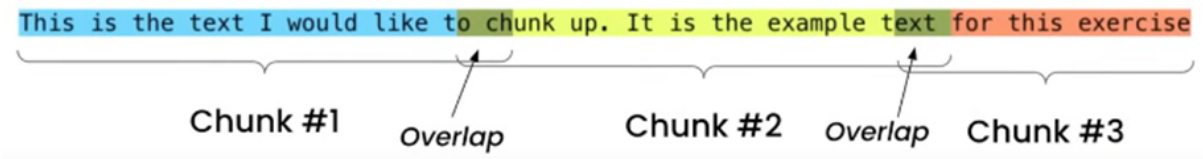
Chunk Optimization

1. Fixed-size chunking

- Phương pháp này sẽ chia văn bản thành các đoạn nhỏ hơn dựa vào số lượng từ hoặc số lượng ký tự, nó không quan tâm tới nội dung, cấu trúc hay ngữ nghĩa của văn bản. Phương pháp này cũng đã được tích hợp trong 2 framework Langchain và Llamaindex lần lượt là CharacterTextSplitter and SentenceSplitter

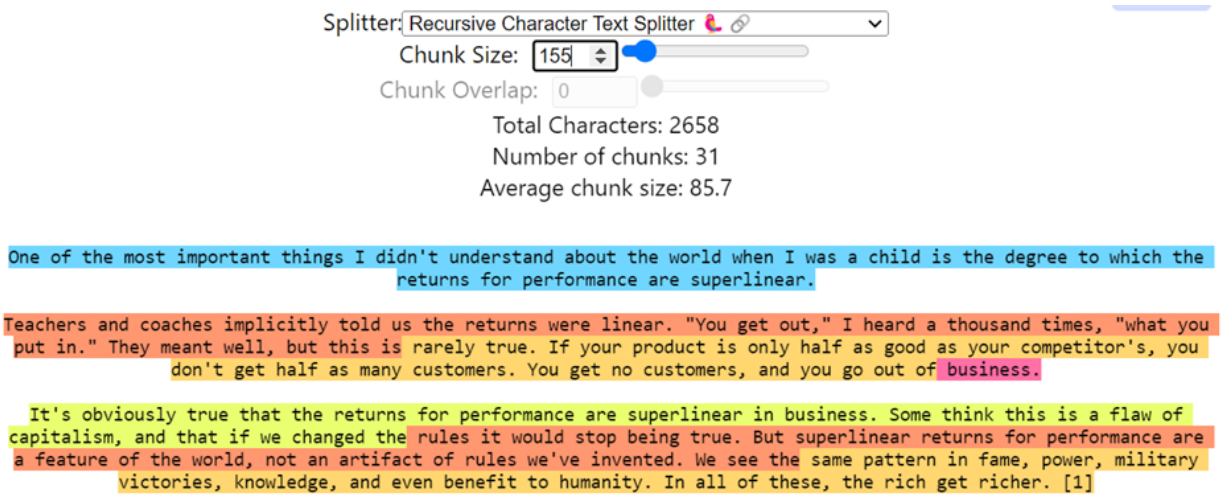


2. Context-aware chunking

- Context-aware chunking tận dụng cấu trúc bên trong của văn bản để phân tách các chunks tạo ra các chunks có ý nghĩa hơn và phù hợp với ngữ cảnh hơn.

2.1 Recursive Text Splitter

- Nhược điểm của phương pháp trên là sẽ cắt xem đúng số lượng token mình cung cấp mà không quan tâm tới nội dung bên trong của nó.
- Phương pháp này sẽ dựa vào những separators được thiết lập sẵn như: ["\n\n", "\n", " ", ""]. Và từ đó chúng sẽ cắt theo thứ tự ưu tiên dựa trên separators sao cho chunk được lấy ra vừa là dài nhất có thể vừa giữ được tính toàn vẹn của nội dung.



2.2. Semantic Chunking

- Semantic chunking được thực hiện dựa trên việc phân tích ngữ nghĩa của văn bản thay vì dựa trên cấu trúc của văn bản. Ý tưởng chính của semantic chunking chính là việc split văn bản dựa trên việc tính toán độ tương đồng giữa các sentences.
- Sự tương đồng này được tính bằng cách chia văn bản đã cho thành các sentences dựa vào các regex như [".", "?", "!"], sau đó chuyển tất cả các sentences này thành các vector embeddings và tính toán độ tương đồng giữa các sentences này bằng các phép tính similarity.
- Thực hiện các bước của Semantic Chunking như sau:
  - Bước 1: Đầu tiên sẽ tiến hành chia văn bản thành tập hợp các sentences dựa vào các regex như [".", "?", "!"].
  - Bước 2: Trong quá trình thực nghiệm tác giả của phương pháp này Greg Kamradt thấy rằng việc để từng sentence riêng lẻ sẽ cực kỳ noisy và mang lại kết quả không tốt khi tính toán similarity, vậy nên tác giả đã quyết định tạo ra thêm bước này để group các sentences nhỏ liền kề lại với nhau thành một sentence lớn hơn bằng một tham số buffer_size. Ví dụ với buffer_size = 1 có nghĩa rằng sẽ ghép câu trước nó và một câu sau nó lại với nhau, điều này có nghĩa chúng ta sẽ có 3 sentences nhỏ tạo thành một sentence lớn hơn.

- Bước 3: Bước tiếp theo sẽ tiến hành sử dụng các embedding model để embed các sentences thành vector embeddings.
- Bước 4: Tính toán độ tương đồng giữa #sentence 1 vs 2, #sentence 2 vs 3, #sentence 3 vs 4...v...v. Sau đó thu được distance = 1 - similarity score
- Bước 5: Thực hiện phân chia chunk dựa vào distance và break_point threshold. Khi distance < break_point threshold sẽ thoả mãn rằng chúng sẽ là một chunk.

NLTK (Natural Language Toolkit)

- NLTK là một thư viện phổ biến, toàn diện để xử lý ngôn ngữ tự nhiên. NLTK có bộ sentence tokenizer giúp phân tách văn bản text thành các câu một cách hiệu quả.

spaCy:

- spaCy là một thư viện nâng cao cho các tác vụ trong NLP, cung cấp khả năng phân đoạn các câu hiệu quả.