Luận văn tốt nghiệp

Trần Gia Huy

Ngày 5 tháng 1 năm 2024

BỘ GIÁO DỤC VÀ ĐÀO TẠO
TRƯỜNG ĐẠI HỌC CẦN THƠ
TRƯỜNG CÔNG NGHỆ THÔNG TIN VÀ TRUYỀN THÔNG

TRẦN GIA HUY
MÃ SỐ SINH VIÊN: B2016968

HỖ TRỢ TƯ VẤN THỦ TỤC HÀNH CHÍNH TẠI CẦN THƠ BẰNG PHƯƠNG PHÁP RAG VÀ KIẾN TRÚC MICROSERVICES

ADVISE ADMINISTRATIVE PROCEDURES IN CAN THO CITY USING RAG METHOD AND MICROSERVICES ARCHITECTURE

LUẬN VĂN TỐT NGHIỆP
NGÀNH: KHOA HỌC MÁY TÍNH
MÃ SỐ: 748 101

GIẢNG VIÊN HƯỚNG DẪN
PGS. TS. GVCC. PHẠM NGUYÊN KHANG

NĂM 2024

TRƯỜNG CÔNG NGHỆ THÔNG TIN VÀ TRUYỀN THÔNG
KHOA KHOA HỌC MÁY TÍNH

XÁC NHẬN CHỈNH SỬA LUẬN VĂN THEO Ý KIẾN CỦA HỘI ĐỒNG

Tên luận văn: Hỗ trợ tư vấn thủ tục hành chính tại Cần Thơ bằng phương pháp RAG và kiến trúc Microservices. (Advise administrative procedures in Can Tho using RAG method and Microservices architecture.).
Họ tên sinh viên: Trần Gia Huy – Mã số sinh viên: B2016968.
Mã lớp: DI20Z6A1.
Đã báo cáo tại hội đồng: Khoa học máy tính.
Ngày báo cáo: –/–/—-.
Hội đồng báo cáo gồm:

1. ——— – Chủ tịch hội đồng.
2. ——— – Thành viên.
3. ——— – Thư ký.

Luận văn này đã được chỉnh sửa theo góp ý của Hội đồng.

Cần Thơ, ngày – tháng – năm 2024
Giảng viên hướng dẫn
(Ký, ghi rõ họ tên)

Phạm Nguyên Khang

Lời cảm ơn

Tôi xin gửi lời cảm ơn tới PGS. TS. GVCC. Phạm Nguyên Khang người đã tận tình giúp đỡ tôi trong quá trình học tập, nghiên cứu và hoàn thành luận văn này.

Tôi xin bày tỏ lòng biết ơn đến các Thầy Cô trong Khoa Khoa học máy tính, các Thầy Cô giảng dạy tại Trường Công nghệ thông tin và Truyền thông, Trường Đại học Cần Thơ đã giúp đỡ tôi trong suốt quá trình học tập và nghiên cứu tại Trường.

Sau cùng tôi xin gửi lời cảm ơn đến gia đình và bạn bè đã luôn ủng hộ tôi, động viên cũng như giúp đỡ tôi trong suốt thời gian qua.

Trong quá trình nghiên cứu và thực hiện đề tài luận văn sẽ không tránh khỏi nhiều sai sót và hạn chế, kính mong nhận được sự chỉ dẫn và đóng góp của quý Thầy Cô để bài luận văn của tôi được hoàn thiện hơn.

Tôi xin chân thành cảm ơn!

Cần Thơ, ngày – tháng – năm 2024

Tác giả

Trần Gia Huy

Tóm tắt

Abstract

Lời cam đoan

Chúng tôi xin cam đoan Luận văn tốt nghiệp "Hỗ trợ tư vấn thủ tục hành chính tại Cần Thơ bằng phương pháp RAG và kiến trúc Microservices" là công trình nghiên cứu của riêng chúng tôi. Ngoài các trích dẫn, tài liệu tham khảo đã được ghi nguồn đầy đủ. Các số liệu, kết quả nêu trong luận văn là trung thực và chưa từng được công bố trong các công trình khác. Nếu không đúng như đã nêu trên, chúng tôi xin hoàn toàn chịu trách nhiệm về đề tài của mình.

Cần Thơ, ngày – tháng – năm 2024
Người cam đoan
(Ký, ghi rõ họ tên)

Trần Gia Huy

Mục lục

Danh sách hình vẽ

Danh sách bảng

Danh mục từ viết tắt

LLM Large Language Model

RAG Retrieval-Augmented Generation

Chương 1

Giới thiệu

1.1 Lý do chọn đề tài

1.2 Mục tiêu nghiên cứu

1.3 Các nghiên cứu liên quan

1.4 Đối tượng và phạm vi nghiên cứu

1.5 Phương pháp nghiên cứu

1.6 Cấu trúc luận văn

Nội dung luận văn bao gồm 3 chương:

- Chương 1 – Giới thiệu
- Chương 2 – Nội dung: gồm 3 phần:
  - Phần 1 – Cơ sở lý thuyết
  - Phần 2 – Phương pháp thực hiện
  - Phần 3 – Kết quả thực nghiệm
- Chương 3 – Kết luận

Chương 2

Nội dung

2.1 Cơ sở lý thuyết

2.1.1 Xử lý ngôn ngữ tự nhiên

2.1.2 Phương pháp nhúng từ và văn bản

2.1.3 Mô hình Seq2Seq và cơ chế attention

2.1.4 Mô hình Transformer

2.1.5 Kiến trúc Transformer

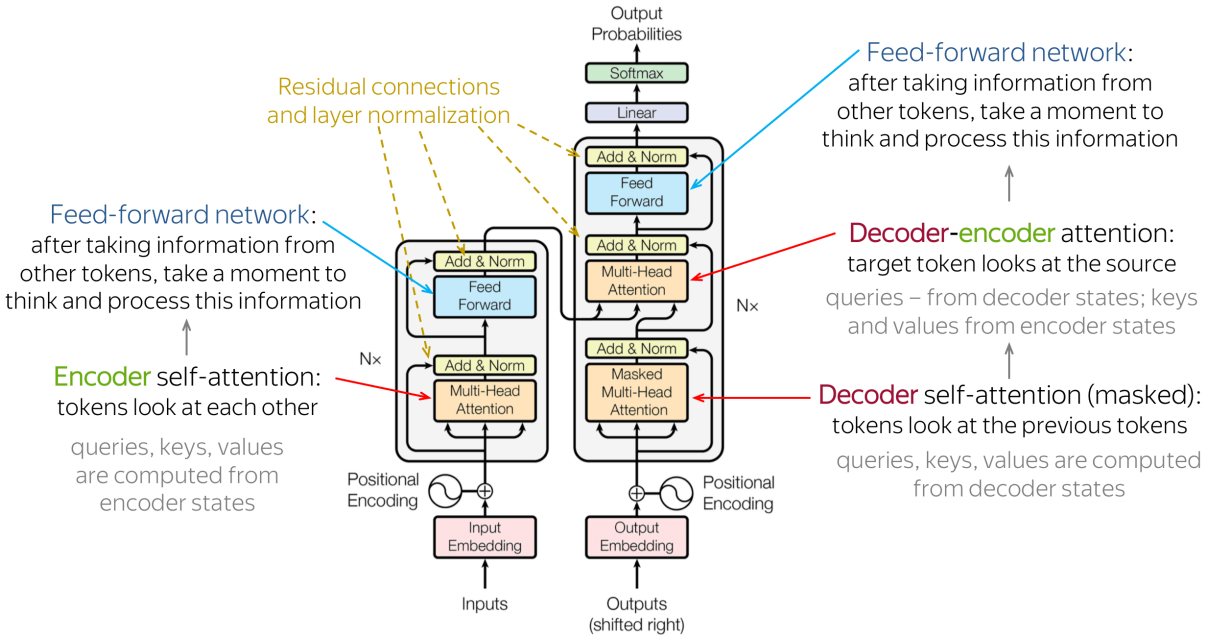
Tổng Quan

Transformer[1] là một mô hình học sâu được giới thiệu bởi Vaswani và các cộng sự [9] vào năm 2017, đây là mô hình đã đạt được kết quả nổi bật trong nhiều nhiệm vụ xử lý ngôn ngữ tự nhiên (NLP). Transformer được xây dựng dựa trên cơ chế Attention (Attention mechanism, tạm dịch: cơ chế tập trung), cho phép mô hình có khả năng "tập trung" vào các phần khác nhau của đầu vào (Input) trong quá trình học.

Mô hình Transformer sử dụng kiến thức đa đầu vào (Multi-Head Input) và đa đầu ra (Multi-Head Output) để xử lý đầu vào và đầu ra theo cách đồng thời, tức là không cần xử lý tuần tự từng phần như các mô hình trước đây. Điều này giúp giải quyết vấn đề độ dài phụ thuộc trong ngôn ngữ (Long-range Dependency) và giúp mô hình có khả năng hiểu ngữ cảnh (Contextual Understanding) của dữ liệu đầu vào (Input).

Mô hình Transformer đã được ứng dụng rộng rãi trong nhiều tác vụ xử lý ngôn ngữ tự nhiên, bao gồm dịch máy, phân loại văn bản, dự đoán từ, tóm tắt văn bản, hỏi đáp, và nhiều tác vụ ngôn ngữ tự nhiên khác. BERT**devlin2019bert** (Bidirectional Encoder Representations from Transformers), GPT**brown2020language** (Generative Pre-trained Transformer), và T5**raffel2023exploring** (Text-to-Text Transfer Transformer) là những mô hình NLP nổi tiếng được xây dựng trên kiến

trúc của mô hình Transformer.



Hình 2.1: Tổng quan kiến trúc Transformer**voita-etal-2019-good**

– Mô hình Transformer sử dụng định dạng mã hóa - giải mã (Encoder - Decoder) tương tự như Seq2Seq.

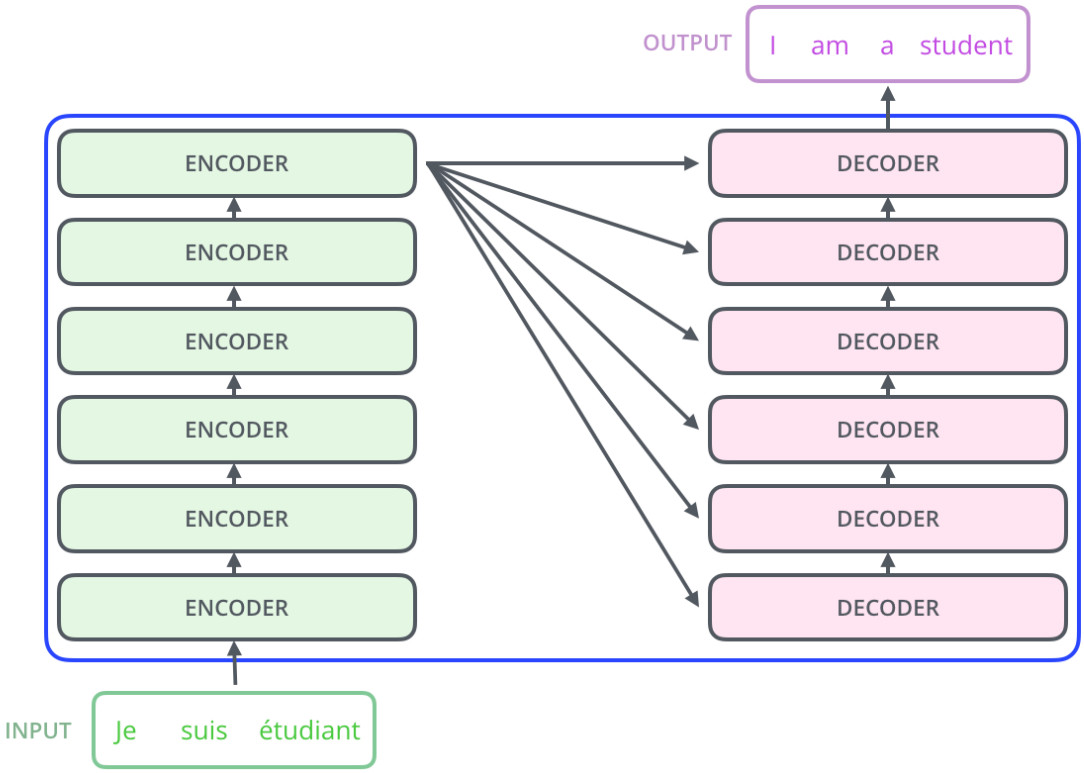
– Những khối được đề xuất trong Transformer, bao gồm Scaled Dot-Product Attention và Multi-Head Attention, là trọng tâm của kiến trúc này, và chúng được xếp hàng loạt và thực hiện song song (parallel) trong mô hình.

– Khác với kiến trúc của RNN,transformer, không có cấu trúc BPTT tương tự và tính toán có thể được thực hiện song song, do đó mô hình có khả năng hoạt động hiệu quả hơn so với kiến trúc RNN.

Hai khối Encoder và Decoder

Đầu vào và đầu ra của Encoder có cùng kích thước. Do đó, cấu trúc Encoder có thể được lặp lại nhiều lần để sử dụng một cách dễ dàng.

Tương tự, Decoder cũng có thể được lặp lại nhiều lần dưới dạng khối để giải mã đầu ra.



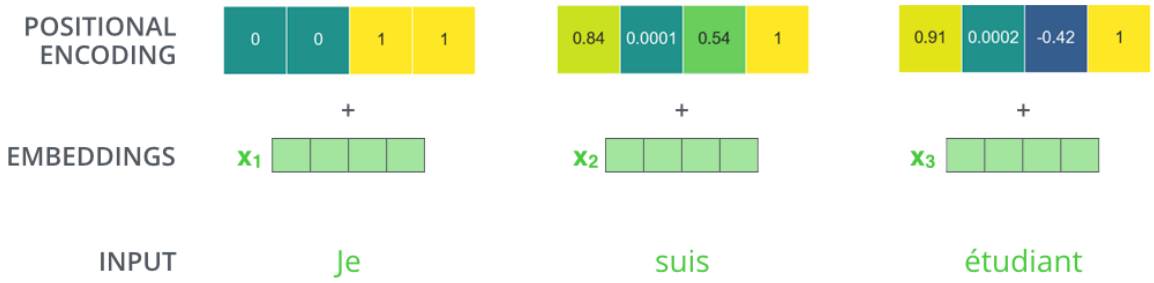
Hình 2.2: Chồng các khối Encoders và Decoders**https://jalammar.github.io/illustrated-transformer/**

Embedding và Positional Encoding

Input Embedding và Output Embedding Trong Transformer, Input Embedding và Output Embedding đều là tầng Embedding để chuyển đổi đầu vào thành vector với số thực.

Đây là tầng đầu tiên trong kiến trúc Transformer, nhận đầu vào là một one-hot-encoded vector là một vector có độ dài bằng với số từ trong từ điển,

Mỗi từ đi vào sẽ được tầng này tạo ra một vector đặc, có số chiều nhỏ hơn ban đầu. **Positional Encoding** Kết quả của tầng Input Embedding hoặc Output Embedding sẽ được đi qua một phép tính Positional Encoding.



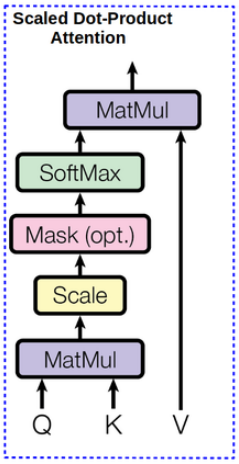
Hình 2.3: Tầng Input và Output của Transformers**Bramas2021AFV**

Do Transformer không tính toán tuần tự mà xử lý một cách song song , nên nó không có khả năng nhận biết được vị trí của từ trong câu. Mục tiêu của phép tính Positional Encoding là

để giữ lại vị trí cho câu input, không làm mất thứ tự và ngữ nghĩa câu.

Khối Multi-Headed Attention

Scaled Dot-Product Attention Attention là một cơ chế cho phép mô hình tập trung vào những phần quan trọng của đầu vào (Input) trong quá trình học. Cơ chế Attention được sử dụng trong nhiều mô hình NLP, bao gồm cả mô hình Transformer.



Hình 2.4: Khối ScaleDotProductAttention**DBLP:journals/corr/VaswaniSPUJGKP17**

Đầu vào của Scaled Dot-Product Attention là 3 ma trận Q, K, V có cùng số chiều.

Đầu ra của Scaled Dot-Product Attention là ma trận có cùng số hàng với ma trận Q và cùng số cột với ma trận V thể hiện trọng số attention của nó.

Dữ liệu trong khối này sẽ được xử lý như sau:

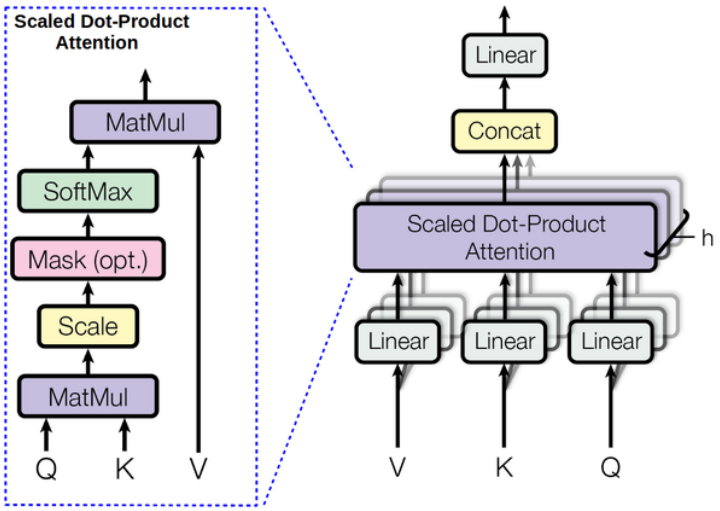
- Hai ma trận Q và K sẽ được nhân lại với nhau, sau đó chia cho căn bậc hai của số chiều của ma trận K (tầng Scale) nhằm ngăn giá trị tăng lên quá lớn .
- Sau tầng Scale, nếu có tầng Mask thì sẽ được thực hiện để ngăn chặn Attention đến các kết nối không hợp lệ (illegal connection).
- Sau đó, ma trận sau khi được nhân sẽ được đưa vào hàm softmax để chuẩn hóa giá trị Attention. Giúp mô hình tự tin hơn với các trọng số.
- Cuối cùng, ma trận sau khi được chuẩn hóa sẽ được nhân với ma trận V để tạo ra đầu ra của khối Scaled Dot-Product Attention. Đây chính là trọng số Attention của khối.

Multi-headed Attention

Thay vì chỉ sử dụng một khối Scaled Dot-Product Attention, Transformer sử dụng nhiều khối Scaled Dot-Product Attention song song (parallel) để tăng khả năng học của mô hình, hiểu ngữ nghĩa theo nhiều khối.

Đầu ra của mỗi khối Scaled Dot-Product Attention sẽ được nối với nhau và đi qua một tầng Linear để tạo ra đầu ra của khối Multi-Headed Attention.

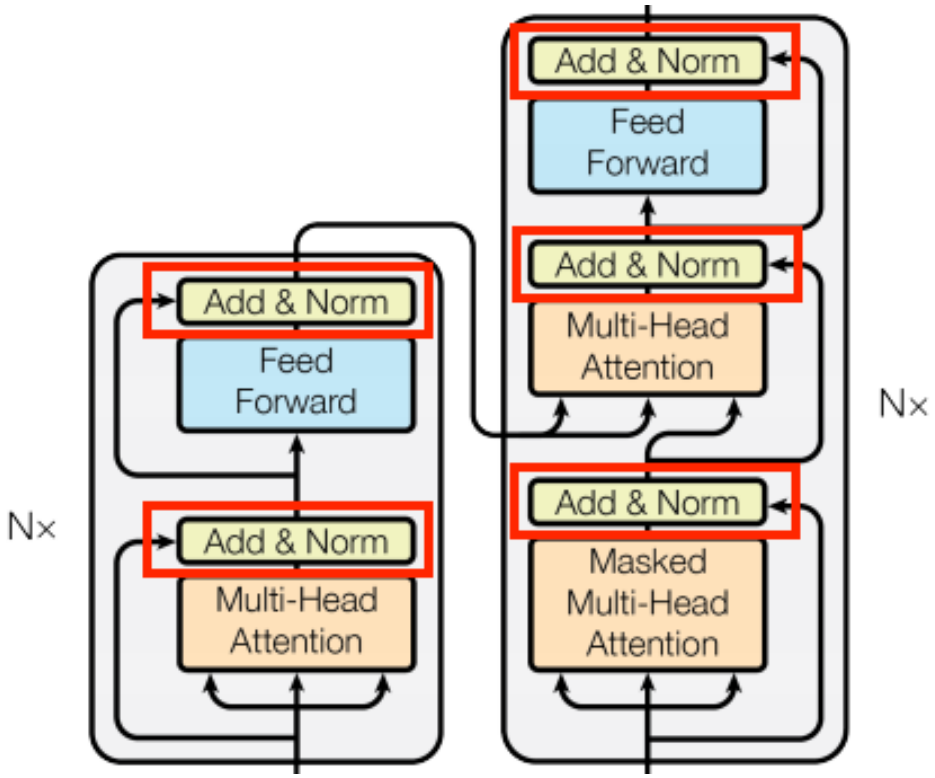
Kết quả của khối là một ma trận có cùng số hàng với ma trận Q và cùng số cột với ma trận V thể hiện trọng số attention của nó.



Hình 2.5: Multi-headed Attention**app13148473**

2.1.6 Add và LayerNormalization

Trong Transformer, Add & Norm được sử dụng để kết hợp thông tin từ các tầng khác nhau trong mạng. Trong quá trình này, đầu ra của một tầng sẽ được cộng với đầu vào ban đầu của tầng đó (skip-connection), sau đó chuẩn hóa lại với Layer Normalization để tạo ra đầu ra cuối cùng.



Hình 2.6: Tầng Add và LayerNormalization**372429372_A_Stock_Price_Prediction_Method_Based_on_**

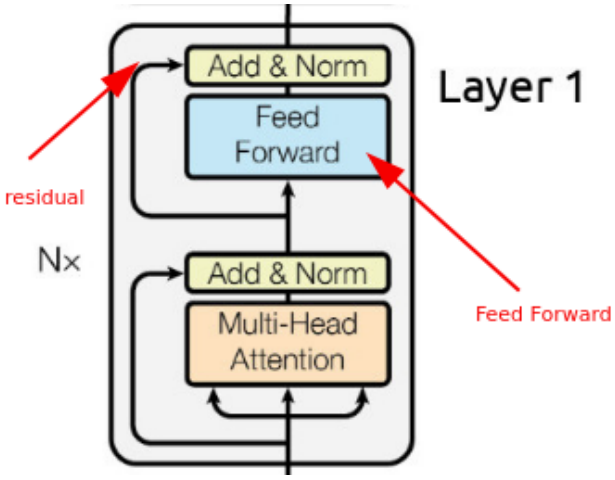
Add

Trong bước này, đầu ra của lớp sẽ được cộng với vector đầu vào ban đầu. Việc cộng này giúp cập nhật thông tin từ các phần khác nhau của kiến trúc và giúp tránh hiện tượng mất thông tin (Vanishing Gradient) trong quá trình huấn luyện.

LayerNormalization

Sau khi được cộng với đầu vào ban đầu, đầu ra của lớp sẽ được chuẩn hóa lại với Layer Normalization. Layer Normalization là một phép chuẩn hóa dữ liệu đầu ra của một tầng theo của ma trận đầu ra. Quá trình chuẩn hóa giúp cải thiện tính ổn định của mô hình

2.1.7 Position-wise Feed Forward Network



Hình 2.7: Khối Position-wise Feed Forward**341553895_Natural_language_processing_system_for_self-**

Các vector đầu vào (là các vector đại diện cho từ) sẽ được truyền qua tầng Fully Connected Layer với hàm kích hoạt ReLU, và cuối cùng đi qua một tầng Fully ConnectedLayer nữa. Đầu ra của tầng thứ hai cũng chính ra đầu ra của Position-wise Feed-Forward.

Mục đích chính là xử lý tiếp attention output từ tầng trước đó, giúp mô hình có thể học được các mối quan hệ giữa các từ trong câu.

2.1.8 Mô hình SBERT

Độ tương đồng giữa các embedding

Trong lĩnh vực xử lý ngôn ngữ tự nhiên, một embedding là một biểu diễn vector của một chuỗi, được tạo ra bởi một mô hình học máy. Embedding có thể được tạo ra bằng cách sử dụng các mô hình học máy như Word2Vec, GloVe, FastText, và nhiều mô hình khác. Ngoài ra, với sự phát triển của các mô hình học sâu và kiến trúc transformer, các embedding có thể được tạo ra bằng cách sử dụng khối encoder các mô hình sử dụng kiến trúc transformer hoặc các mô hình học sâu BERT, GPT, và nhiều mô hình khác.

Để đánh giá độ tương đồng giữa các embedding, chúng ta cần một phương pháp đánh giá độ tương đồng. Một trong những phương pháp đánh giá độ tương đồng giữa các embedding đó là độ tương đồng cosin. Công thức tính độ tương đồng cosin giữa hai embedding A và B được tính như sau:

$$\text{similarity}(A,B) = \cos(\theta) = \frac{A \cdot B}{\|A\|\|B\|} \tag{2.1}$$

Một số ưu điểm khi sử dụng độ tương đồng cosin để đánh giá độ tương đồng giữa các embedding:

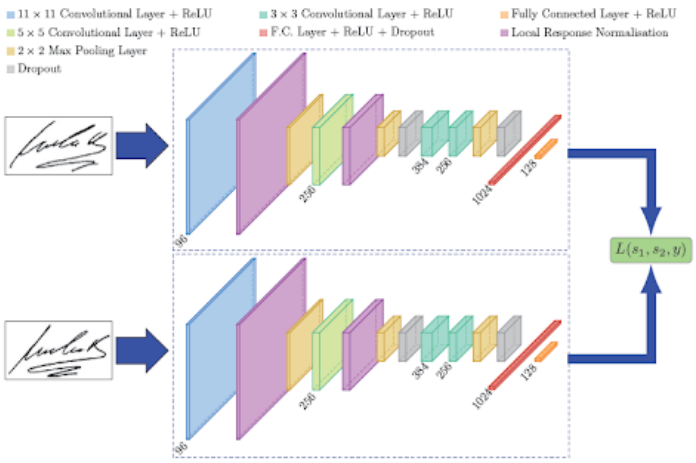
1. **Không Ảnh Hưởng bởi Độ Dài:** Một trong những ưu điểm chính của độ tương đồng cosine là nó không bị ảnh hưởng bởi độ dài của vectơ. Bất kể kích thước của các vectơ,

chỉ cần hướng của chúng giống nhau, độ tương đồng cosine sẽ là nhỏ nhất khi chúng đối lập và lớn nhất khi chúng trùng hướng.

2. **Đo Lường Hướng Tương Đồng:** Độ tương đồng cosine tập trung vào hướng của vectơ thay vì giá trị tuyệt đối. Điều này làm cho nó thích hợp cho các tác vụ như phân loại văn bản, xác định chủ đề, và tìm kiếm tương đồng ngữ cảnh.

3. **Hiệu Quả Cho Dữ Liệu Nhiều Chiều:** Trong không gian chiều cao, nơi mỗi chiều biểu diễn một đặc trưng khác nhau, độ tương đồng cosine thường hiệu quả hơn so với các phương pháp đo tương đồng khác. Điều này giúp giảm hiệu ứng "hiệu ứng chiều cao"khi sử dụng các phương pháp dựa trên khoảng cách Euclidean.

Siamese Networks



Hình 2.8: Siamese Networks[2]

Mạng Siamese là một kiến trúc mạng nơ-ron đặc biệt được thiết kế để xác định mức độ tương đồng giữa hai đối tượng hoặc hai đầu vào. Đặc điểm chính của mạng Siamese là sử dụng hai nhánh đồng nhất (tương tự nhau) chia sẻ trọng số, mỗi nhánh xử lý một đầu vào khác nhau. Mục tiêu của mạng Siamese là học cách biểu diễn và so sánh đặc trưng giữa các cặp đối tượng.

Mạng Siamese thường được sử dụng trong các nhiệm vụ như nhận dạng khuôn mặt, phân loại văn bản, hay tìm kiếm hình ảnh, văn bản tương đồng. Để đạt được mục tiêu này, mạng Siamese thực hiện các bước cơ bản như sau:

– **Trích Xuất Đặc Trưng:** Mỗi nhánh của mạng Siamese nhận một đầu vào và trích xuất đặc trưng từ đối tượng đó bằng cách sử dụng các lớp tích chập và lớp pooling.

– **So Sánh Đặc Trưng:** Các đặc trưng được trích xuất từ cả hai nhánh sau đó được so sánh để đo lường mức độ tương đồng giữa chúng. Thông thường, một hàm khoảng cách như Euclidean hoặc độ tương đồng cosine được sử dụng để đo lường khoảng cách giữa hai vectơ đặc trưng.

– **Huấn Luyện và Tối Ưu Hóa:** Mạng Siamese được huấn luyện bằng cách sử dụng các cặp dữ liệu huấn luyện được gán nhãn với thông tin về mức độ tương đồng. Mục tiêu là

tối ưu hóa mô hình để đặc trưng của các cặp giống nhau gần nhau và của các cặp khác nhau xa nhau.

Mạng SNN tập trung vào việc cải thiện embedding sao cho các lớp giống nhau sẽ nằm gần nhau hơn. Với việc phải bình phương số lượng dữ liệu để tạo ra các cặp so sánh, việc huấn luyện nó sẽ mất thời gian hơn là phân lớp thông thường. Chi tiết cách huấn luyện mạng SNN như sau:

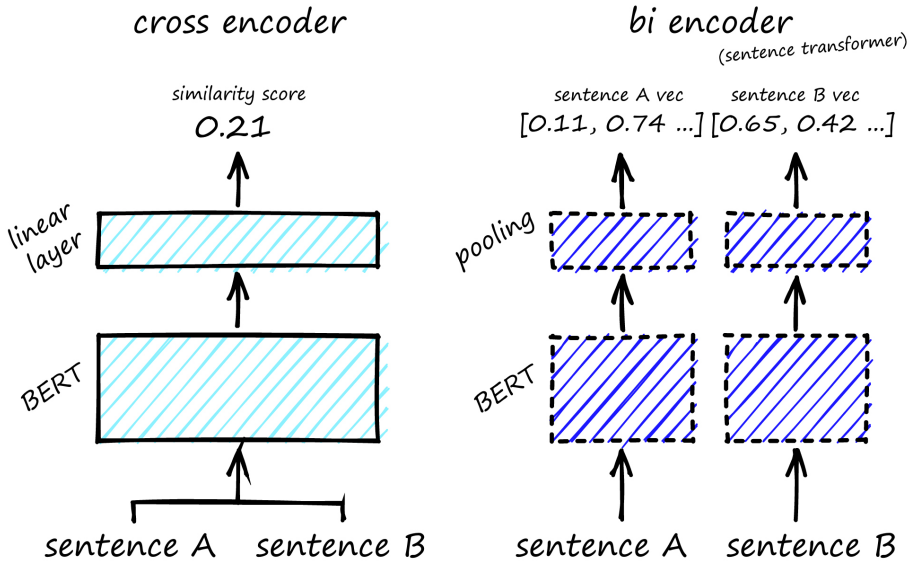
1. **Xây dựng** kiến trúc mạng, hàm loss và optimizer.
2. **Đưa dữ liệu** 1 trong cặp qua network.
3. **Đưa dữ liệu** 2 trong cặp qua network.
4. **Tình lỗi** dựa vào output từ dữ liệu 1 và dữ liệu 2
5. **Lan truyền ngược** để tính lại đạo hàm của các trọng số.
6. **Cập nhật trọng số** bằng optimizer.

Tổng quan mô hình

SBERT , hay Sentence-BERT, là một tiến bộ quan trọng trong lĩnh vực xử lý ngôn ngữ tự nhiên (NLP) và biểu diễn văn bản. Được phát triển dựa trên ý tưởng của BERT (Bidirectional Encoder Representations from Transformers), SBERT tập trung vào việc tối ưu hóa biểu diễn cho các câu trong văn bản.

Với nhược điểm về thời gian so sánh giữa các cặp câu của BERT ,phương pháp phổ biến để giải quyết các vấn đề nhóm và tìm kiếm ý nghĩa là ánh xạ mỗi câu vào không gian vectơ sao cho các câu có ý nghĩa tương tự sẽ gần nhau. Các nhà nghiên cứu đã bắt đầu đưa từng câu vào BERT và rút trích các vectơ nhúng cố định cho câu đó. Phương pháp phổ biến nhất là lấy trung bình của lớp đầu ra của BERT (được biết đến là nhúng BERT) hoặc bằng cách sử dụng đầu ra của ký tự đầu tiên (ký tự [CLS]). Nhưng thực nghiệm cho rằng, phương pháp này tạo ra các vectơ nhúng câu khá kém, thường xấp xỉ hoặc kém hơn so với việc lấy trung bình vectơ nhúng GloVe.

Sentence-BERT (SBERT), một phiên bản chỉnh sửa của mạng BERT sử dụng mô hình siamese và triplet với khả năng tạo ra nhúng câu mang ý nghĩa ngữ nghĩa. Điều này cho phép BERT được sử dụng cho một số nhiệm vụ mới, mà cho đến nay chưa áp dụng được cho BERT. Các nhiệm vụ này bao gồm so sánh ý nghĩa ngữ cảnh quy mô lớn, phân nhóm, và truy xuất thông tin thông qua tìm kiếm ý nghĩa ngữ.



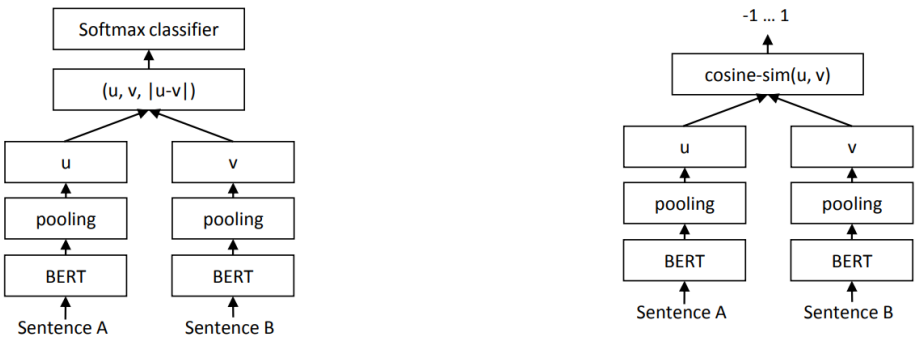
Hình 2.9: Sự khác biệt giữa cách tiếp cận của mô hình BERT và SBERT trong bài toán so sánh sự tương đồng ngữ nghĩa

SBERT thêm vào một tầng pooling cho tầng output của BERT để lấy ra được một embedding vector size cố định cho câu đó. Đã thử nghiệm trên cả 3 cách pooling và chọn phương thức MEAN Pooling:

- Sử dụng output của CLS token.
- Sử dụng MEAN pooling.
- Sử dụng MAX pooling.

Finetune từ BERT, RoBERTa

Và để fine-tune lại các mô hình BERT và RoBERTa, SBERT sử dụng kiến trúc siamese và triplet network. Trong quá trình fine-tuning sẽ cập nhật trọng số sao cho các vector embedding đầu ra có ngữ nghĩa và có thể được so sánh bằng cosine similarity.



Hình 2.10: SBERT fine-tuned lại BERT sử dụng hàm đánh giá Softmax classifier (1) Hình 2.11: SBERT fine-tuned lại BERT sử dụng hàm đánh giá (2)

Trong quá trình fine-tuning, SBERT đã thử nghiệm với 2 hàm đánh giá khác nhau là Softmax Classifier và Cosine Similarity:

– Classification:

  + Nối embedding của u, v và phép tính element-wise |u - v|.
  + Nhân với trọng số $W_t \in \mathbb{R}^{3n \times k}$, và cho kết quả đi qua hàm softmax:

$$o = \text{softmax}(W_t(u, v, |u - v|)) \tag{2.2}$$

– Regression: Sử dụng cosine similarity để tính độ tương đồng giữa u và v, Sau đó dùng hàm loss là MSE (Mean Square Error) để tính độ lỗi.

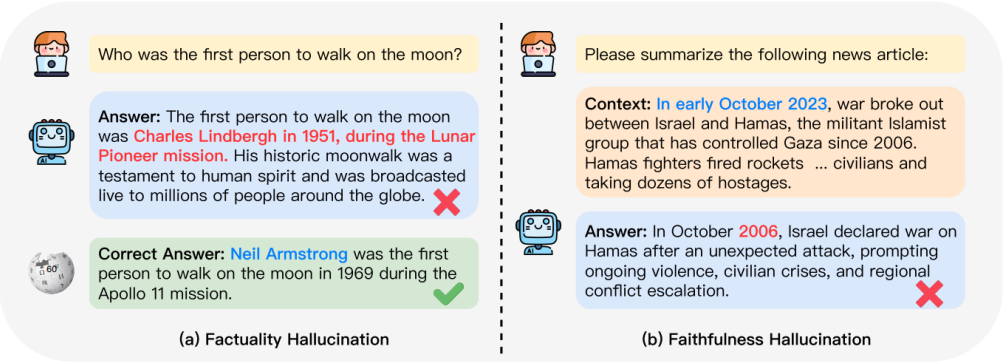
Huấn luyện SBERT được diễn ra trên sự kết hợp của các tập dữ liệu SNLI **snli:emnlp2015** (Bowman et al., 2015) và Multi-Genre NLI **N18-1101** (Williams et al., 2018). SNLI là một bộ sưu tập gồm **570,000** cặp câu được chú thích với các nhãn "contradiction"(mâu thuẫn), "entailment"(hỗ trợ), và "neutral"(trung lập). MultiNLI chứa **430,000** cặp câu và bao gồm nhiều thể loại văn bản nói và viết khác nhau. Tinh chỉnh SBERT với một hàm mục tiêu là **Classification Softmax 3 chiều** trong một epoch. Sử dụng **batch size là 16**, tối ưu hóa Adam với tốc độ học **2e-5**, và tăng tốc tốc độ học tuyến tính trên 10% dữ liệu huấn luyện. Chiến lược pooling sử dụng là **"MEAN"** (trung bình).

2.1.9 Phương pháp RAG

Thực trạng của các mô hình ngôn ngữ

Với sự phát triển bùng nổ của các mô hình ngôn ngữ lớn hiện nay đã mở đường cho những tiến bộ vượt bậc trong lĩnh vực xử lý ngôn ngữ tự nhiên cũng như đẩy mạnh việc ứng dụng AI tạo sinh vào cuộc sống

Tuy nhiên, những mô hình mạnh mẽ này cũng đi kèm với một số thách thức cần phải giải quyết. Một trong những vấn đề lớn là hiện tượng "hallucination ảo giác, tức việc Large Language Model (LLM) tạo ra các thông tin không chính xác, không đúng sự thật hoặc không được hỗ trợ bởi dữ liệu có sẵn. Hiện tượng này rất nguy hiểm bởi nó có thể dẫn đến việc cung cấp các thông tin sai lệch gây hậu quả nghiêm trọng và làm giảm độ tin cậy của các hệ thống dựa trên AI



Hình 2.12: Định nghĩa về vấn đề "hallucination"ở LLMs.

Hallucination [3] theo bài báo "*A Survey on Hallucination in Large Language Models: Principles, Taxonomy, Challenges, and Open Questions*"sẽ chia làm 2 loại: Ảo giác sự thật (Factuality Hallucination) và Ảo giác tính trung thực (Faithfulness Hallucination).

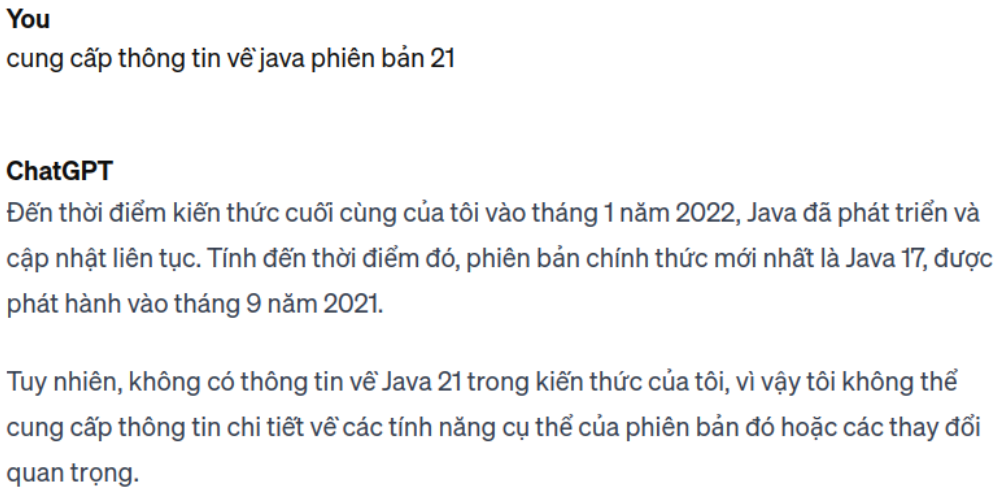
Ảo giác sự thật (Factuality Hallucination) thể hiện khi thông tin cung cấp từ LLM có sự sai lệch so với thông tin đã được xác nhận trong thế giới thực, những thông tin này được coi như là một sự không nhất quán với sự thật hoặc là sự ngụy tạo do mô hình ngôn ngữ.

Còn với ảo giác tính trung thực (Faithfullness Hallucination) đề cập đến sự chệch lệch của nội dung được tạo ra so với hướng dẫn của người dùng hoặc bối cảnh được cung cấp bởi đầu vào, cũng như tính nhất quán trong nội dung được tạo ra.

Ví dụ, như trong Hình 2.12(a) thể hiện ảo giác sự thật (Factuality Hallucination), khi được hỏi về người đầu tiên đi bộ trên mặt trăng, mô hình có thể khẳng định rằng đó là Charles Lindbergh vào năm 1951. Trong khi sự thật là Neil Armstrong là người đầu tiên đi bộ trên mặt trăng vào năm 1969 trong chuyến bay Apollo 11.

Ngược lại, như minh họa trong Hình 1(b) thể hiện ảo giác tính trung thực (Faithfullness Hallucination), khi được yêu cầu tóm tắt một bài báo, mô hình đã không chính xác về ngày diễn ra xung đột giữa Israel và Hamas từ tháng 10 năm 2023 thành tháng 10 năm 2006.

Một vấn đề khác với các LLM, chính là kiến thức bị lỗi thời. Ví dụ như dữ liệu đào tạo của ChatGPT 3.5 bị đóng băng ở thời điểm tháng 1/2022. Nếu hỏi ChatGPT về một sự kiện gì đó vào 2023, tình hình thời tiết, chính sách mới, vv. thì nó sẽ không trả lời được một thông tin chính xác.



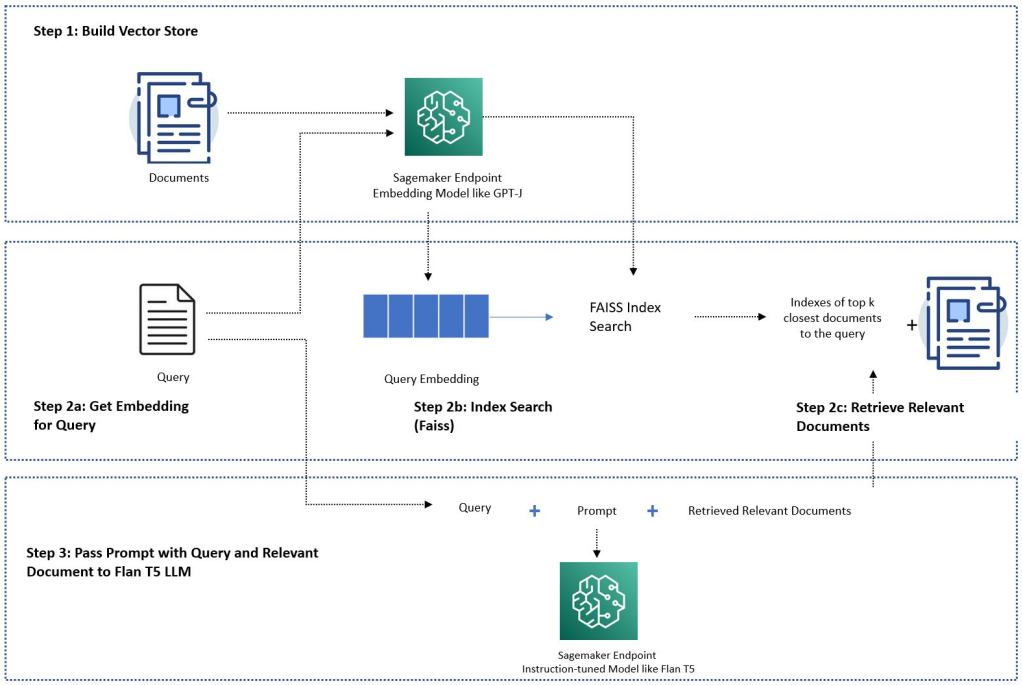
Hình 2.13: Đặt một câu hỏi nằm ngoài thời gian kiến thức huấn luyện của ChatGPT 3.5.

Ngoài ra, các LLM được huấn luyện cho các nhiệm vụ tổng quát, (generalized tasks), nghĩa là nó sẽ không biết dữ liệu riêng tư của cá nhân hay tổ chức nào đó, cũng như thiếu các kiến thức ngành chuyên sâu hoặc thông tin về một đối tượng rất cụ thể nào đó (ví dụ sản phẩm mới ABC của một công ty XYZ).

Một hạn chế khác cũng vô cùng quan trọng, chính là việc các LLM hoạt động như một hộp đen: nghĩa là không thể biết được LLM đã sử dụng những nguồn thông tin nào để đưa ra được câu trả lời. Điều này sẽ gây ra khó khăn trong việc quản lý kiến thức của LLM, hạn chế những nguồn thông tin sai lệch hoặc không phù hợp.

Định Nghĩa

Phương pháp Retrieval-Augmented Generation (RAG) là một phương pháp kết hợp giữa phương pháp truy vấn kết hợp và phương pháp sinh câu trả lời. Phương pháp RAG [4] này được giới thiệu lần đầu bởi các nhà nghiên cứu của Meta AI để giải quyết các nhiệm vụ yêu cầu nhiều kiến thức. Nó là thành phần kết hợp giữa thành phần truy xuất thông tin (Retrieval) và một mô hình tạo sinh văn bản (Generation).



Hình 2.14: Quá trình thực hiện phương pháp RAG.

Phương pháp RAG ra đời để giải quyết cho việc ảo giác (hallucination) của các mô hình ngôn ngữ lớn. Ý tưởng của phương pháp chính là kết hợp một phương pháp truy vấn để tìm ra được các ngữ cảnh (context) có liên quan cho câu hỏi của người dùng. Sau đó sử dụng chính ngữ cảnh này để yêu cầu (prompt) cho các mô hình ngôn ngữ lớn mạnh mẽ như GPT, Llama2, Gemini,...

Bằng cách này, phương pháp RAG có thể giải quyết được một số hạn chế của LLM. Đầu tiên, nó đã giúp cho LLM có được những kiến thức mới hoặc những tri thức chuyên sâu về một tác vụ cụ thể nào đó. Những kiến thức có thể đã không được sử dụng để huấn luyện cho LLM đó.

Kế tiếp, phương pháp RAG này giúp kiểm soát được những thông tin mà LLM sẽ trả lời, biết được nguồn thông tin mà mô hình ngôn ngữ sẽ sử dụng để sinh từ. Nhờ đó, hệ thống sẽ có được khả năng kiểm soát cao hơn với câu trả lời được sinh ra từ các LLM.

2.1.10 Xếp hạng lại trong RAG (Re-ranking)

2.1.11 Truy vấn kết hợp trong RAG (RAG Fusion)

2.2 Phương pháp thực hiện

Trong chương này, đầu tiên, luận văn sẽ trình bày chi tiết phương pháp ứng dụng RAG vào trong hệ thống tư vấn thủ tục hành chính. Hệ thống sử dụng mô hình SBERT để nhúng văn bản

và mô hình ngôn ngữ lớn GPT-3.5 để sinh câu trả lời. Với đầu vào là một câu hỏi của người dùng, hệ thống cần trả về một câu trả lời giải quyết được vấn đề của người dùng đi kèm với các trích dẫn thông tin về thủ tục hành chính phù hợp tại Cần Thơ.

Tiếp theo, ở chương này sẽ trình bày từng bước phương pháp xây dựng hệ thống theo kiến trúc Microservices. Hệ thống cần đảm bảo khả năng mở rộng, cô lập các tính năng và khả năng chịu lỗi cao.

Các bước thực hiện cho hệ thống sẽ được trình bày dựa vào 2 sơ đồ sau:

2.2.1 Thu thập dữ liệu

Mô tả dữ liệu

Dữ liệu được sử dụng cho phương pháp RAG - cũng là dữ liệu hệ thống dựa vào để có thể sinh câu trả lời cho câu hỏi hoặc tình huống của người dùng chính là các thủ tục hành chính tại Cần Thơ.

Tất cả các trường dữ liệu thông tin của một thủ tục hành chính đều ở dạng chuỗi. Một thủ tục hành chính bao gồm các thông tin như tên, lĩnh vực, trình tự thực hiện, cách thức thực hiện, thành phần hồ sơ, căn cứ pháp lý, ... Dữ liệu của một thủ tục hành chính có thể được tìm thấy ở phần phụ lục.

Trường	**Thông tin**
Mã thủ tục	1.004269.000.00.00.H13
Tên	Thủ tục cung cấp dữ liệu đất đai (cấp tỉnh)
Trình tự thực hiện	Bước 1: - Tổ chức, cá nhân có nhu cầu khai thác dữ liệu đất đai nộp phiếu yêu cầu hoặc gửi văn bản yêu cầu đến Trung tâm Dữ liệu và Thông tin đất đai thuộc Tổng cục Quản lý đất đai hoặc Văn phòng đăng ký đất đai. Đối với địa phương chưa xây dựng cơ sở dữ liệu đất đai, Văn phòng đăng ký đất đai, Ủy ban nhân dân cấp xã; Bước 2: - Khi nhận được phiếu yêu cầu, văn bản yêu cầu hợp lệ của tổ chức, cá nhân, cơ quan cung cấp dữ liệu đất đai thực hiện việc cung cấp dữ liệu cho tổ chức, cá nhân có yêu cầu khai thác dữ liệu. Cơ quan cung cấp dữ liệu đất đai tiếp nhận, xử lý và thông báo nghĩa vụ tài chính (trường hợp phải thực hiện nghĩa vụ tài chính) cho tổ chức, cá nhân. Trường hợp từ chối cung cấp dữ liệu thì phải có văn bản trả lời nêu rõ lý do; ... ...

Bảng 2.1: Giới thiệu một thủ tục hành chính

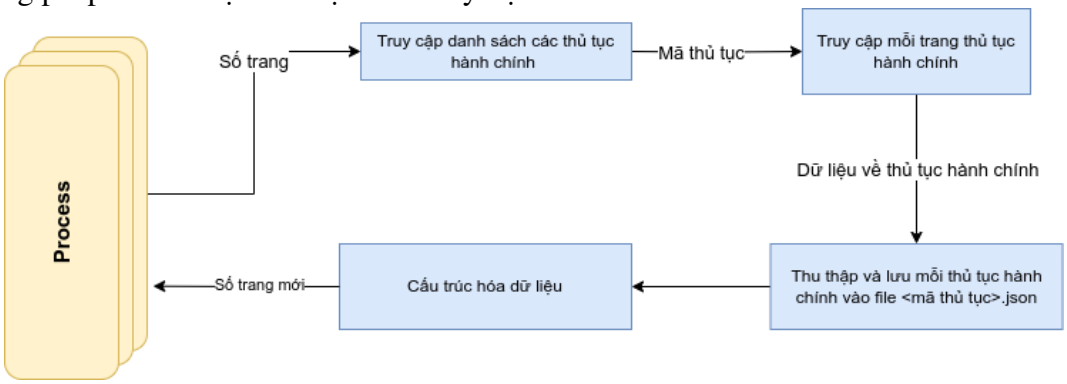
Nguồn dữ liệu sẽ được cào trực tiếp từ hệ thống thông tin giải quyết thủ tục hành chính

thành phố Cần Thơ. Với tổng cộng 1,827 thủ tục hành chính chia vào 99 nhóm dịch vụ công.

Cào dữ liệu các thủ tục hành chính

Dữ liệu các thủ tục hành chính sẽ được cào sử dụng thư viện Selenium với ngôn ngữ lập trình Python. Selenium là một thư viện hỗ trợ tự động hóa việc điều khiển trình duyệt web, bằng cách mở và điều khiển trình duyệt web bằng Selenium Webdriver. Quá trình cào dữ liệu sẽ được thực hiện đa luồng, mỗi luồng sẽ điều khiển một trình duyệt web riêng biệt giúp tăng tốc độ cào dữ liệu.

Phương pháp cào dữ liệu sẽ được trình bày dựa vào sơ đồ bên dưới:



Hình 2.15: Sơ đồ quy trình cào dữ liệu các thủ tục hành chính.

Đầu tiên, webdriver truy cập vào trang dịch vụ công của thành phố Cần Thơ. Tại đây, webdriver sẽ lấy danh sách các nhóm dịch vụ công ở mỗi trang. Tiếp theo, chương trình chia làm 10 luồng để truy cập vào từng trang chi tiết của thủ tục hành chính, sử dụng thư viện BeautifulSoup để lấy dữ liệu, cấu trúc hóa và lưu vào các tệp file json tương ứng với mã thủ tục.

Trong quá trình thu thập dữ liệu, có một số trường hợp bị lỗi do cách thức hiển thị của trang web. Những trường hợp này sẽ được lưu lại trong tệp log để xử lý thủ công. Dữ liệu thu thập cuối cùng sẽ đầy đủ 1,827 thủ tục hành chính từ hệ thống thông tin giải quyết thủ tục hành chính thành phố Cần Thơ.

2.2.2 Tiền xử lý dữ liệu

2.2.3 Xây dựng hệ thống ứng dụng phương pháp RAG

2.2.4 Đánh giá kết quả

2.2.5 Triển khai hỗ trợ tư vấn thủ tục hành chính

2.3 Kết quả thực nghiệm

Chương 3

Kết luận

3.1 Kết quả đạt được

3.2 Hạn chế và hướng phát triển

Tài liệu tham khảo

[1] T. Wolf, L. Debut, V. Sanh **andothers**, "HuggingFace's Transformers: State-of-the-art Natural Language Processing," *ArXiv*, **jourvol** abs/1910.03771, 2019. **url**: https://api.semanticscholar.org/CorpusID:208117506.

[2] S. Benhur, *A Friendly Introduction to Siamese Networks*, https://builtin.com/machine-learning/siamese-network, 2022.

[3] L. Huang, W. Yu, W. Ma **andothers**, *A Survey on Hallucination in Large Language Models: Principles, Taxonomy, Challenges, and Open Questions*, 2023. arXiv: 2311.05232 [cs.CL].

[4] P. Lewis, E. Perez, A. Piktus **andothers**, *Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks*, 2021. arXiv: 2005.11401 [cs.CL].

Phụ lục

File json chứa thông tin một thủ tục hành chính