

ĐẠI HỌC QUỐC GIA HÀ NỘI  
TRƯỜNG ĐẠI HỌC CÔNG NGHỆ

\*\*\*\*\*



XỬ LÍ NGÔN NGỮ TỰ NHIÊN  
BÁO CÁO BÀI TẬP LỚN

NHÓM 5  
CHỦ ĐỀ: QUESTION ANSWERING

Giảng viên: TS. Trần Hồng Việt

Sinh viên: Hoàng Vũ Duy Anh - 18020001

Lưu Hoàng Nam - 18020921

Nguyễn Công Thuận - 18021250

# Mục lục

<b>1 Giới thiệu</b>	<b>3</b>
1.1 Giới thiệu về Question Answering . . . . .	3
1.2 Các tập dữ liệu nổi tiếng . . . . .	3
1.3 Phương pháp đánh giá . . . . .	4
1.3.1 Thông số <i>accuracy</i> . . . . .	4
1.3.2 Thông số $F_1$ . . . . .	5
1.3.3 Các thông số đánh giá khác . . . . .	6
1.4 Kiến trúc của một hệ thống Open-domain Question Answering . . . . .	6
1.4.1 Kiến trúc Retriever - Reader . . . . .	6
1.4.1.1 Thành phần Retriever . . . . .	7
1.4.1.2 Thành phần Reader . . . . .	8
1.4.2 Kiến trúc Retriever - Generator . . . . .	8
1.4.3 Kiến trúc Generator . . . . .	10
<b>2 DrQA</b>	<b>12</b>
2.1 Tổng quan . . . . .	12
2.2 Document Retriever . . . . .	12
2.3 Document Reader . . . . .	13
2.4 Thí nghiệm - đánh giá . . . . .	14
2.4.1 Dánh giá Document Retriever . . . . .	14
2.4.2 Dánh giá pipeline . . . . .	15
2.5 Thảo luận . . . . .	15
<b>3 Cải tiến</b>	<b>16</b>
3.1 Các phương pháp . . . . .	16
3.1.1 BM25 Retriever . . . . .	16
3.1.2 Mô hình BERT . . . . .	17
3.1.3 BERTserini . . . . .	20
3.1.4 Multi-passage BERT . . . . .	21
3.1.5 Learning to Retrieve Reasoning Paths . . . . .	22
3.2 Thí nghiệm - đánh giá . . . . .	22
<b>4 Ứng dụng</b>	<b>24</b>
4.1 Ứng dụng cho tiếng Việt . . . . .	24
4.1.1 Chuẩn bị dữ liệu . . . . .	24
4.1.2 Huấn luyện mô hình . . . . .	25
4.2 Web UI . . . . .	27
4.2.1 Giao diện tiếng Anh . . . . .	28
4.2.2 Giao diện tiếng Việt . . . . .	31
<b>Tài liệu tham khảo</b>	<b>35</b>

## Danh sách hình vẽ

1	Tổng quan về ba kiến trúc của hệ thống Open-domain Question Answering . . . . .	7
2	Kiến trúc Retriever - Reader . . . . .	8
3	Mô hình BERT được sử dụng trong hệ thống Question Answering . . . . .	9
4	Kiến trúc Retriever - Generator . . . . .	9
5	Mô hình RAG . . . . .	9
6	Kích cỡ của một số mô hình ngôn ngữ phổ biến . . . . .	10
7	Mô hình T5 được fine-tuned để trả lời câu hỏi . . . . .	11
8	Khả năng của GPT-3 đối với tập dữ liệu TriviaQA . . . . .	11
9	Tổng quan về hệ thống DrQA . . . . .	12
10	Kiến trúc Transformer . . . . .	18
11	Tổng quan BERT . . . . .	18
12	Mô tả đầu vào mô hình BERT . . . . .	19
13	Minh họa pipeline hệ thống BERTserini . . . . .	20
14	Bảng so sánh kết quả giữa các chiến lược retrieval . . . . .	21
15	Minh họa pipeline hệ thống Multi-passage BERT . . . . .	21
16	Một số ví dụ dữ liệu đa ngôn ngữ . . . . .	24
17	Biểu đồ so sánh kết quả giữa các mô hình đọc hiểu dựa trên số lượng dữ liệu . . . . .	27
18	Giao diện chính của hệ thống . . . . .	28
19	Câu hỏi về Tổng thống Hoa Kỳ đương nhiệm . . . . .	29
20	Câu hỏi về Ca sĩ/Nhạc sĩ Ed Sheeran . . . . .	29
21	Câu hỏi về CEO của Tập đoàn Microsoft . . . . .	30
22	Giao diện tiếng Việt của hệ thống . . . . .	31
23	Câu hỏi về Cách thức Việt Nam chống dịch COVID-19 . . . . .	32
24	Câu hỏi về Chủ tịch nước đầu tiên của Việt Nam . . . . .	32
25	Câu hỏi về Trường Đại học Công nghệ, Đại học Quốc gia Hà Nội . . . . .	33
26	Câu hỏi về Trường Trung học phổ thông Chuyên Vĩnh Phúc . . . . .	33
27	Câu hỏi về Trường Trung học phổ thông Chu Văn An . . . . .	34
28	Câu hỏi về Trường Trung học phổ thông Chuyên Khoa học tự nhiên . . . . .	34

## Danh sách bảng

1	Confusion Matrix khi đánh giá phân lớp . . . . .	5
2	Kết quả so sánh giữa Wiki Search với DrQA . . . . .	15
3	Kết quả đánh giá DrQA trên bộ dữ liệu SQuAD với số lượng tài liệu được lấy ra từ 2 đến 5 . . . . .	15
4	So sánh kết quả của các hệ thống . . . . .	22
5	Thí nghiệm với DrQA retriever và các mô hình BERT trên tập Open SQuAD-dev . . . . .	23
6	Lượng cặp câu hỏi - trả lời trong bộ dữ liệu MLQA . . . . .	25
7	Bảng đánh giá mô hình đọc hiểu . . . . .	26

# 1 Giới thiệu

## 1.1 Giới thiệu về Question Answering

Lĩnh vực *Truy hồi thông tin* (*Information Retrieval*) là một lĩnh vực tập trung vào tìm kiếm tài liệu có bản chất *phi cấu trúc* (*unstructured*) như văn bản, hình ảnh, video... sao cho *phù hợp* (*relevant*) với một *nhu cầu thông tin* (*information need*) nào đó, từ một *tập hợp dữ liệu lớn* (*large collections*).

Trong lĩnh vực đó, nhiệm vụ *hỏi đáp* (*Question Answering – QA*) là nhiệm vụ tự động trả lời những câu hỏi được đưa ra dưới dạng ngôn ngữ tự nhiên. Các câu hỏi trong một miền ứng dụng cụ thể có thể được trả lời thông qua các kỹ thuật xử lý ngôn ngữ tự nhiên. Theo đó, một hệ thống Question Answering sẽ tìm ra câu trả lời cho một câu hỏi bằng cách truy vấn hoặc tìm kiếm ở trong một hệ tri thức nào đó, có thể là một nguồn thông tin, hay một kho kiến thức.... Thông thường, hệ tri thức đó sẽ được cung cấp cho hệ thống từ một vài thông tin trước đó, một vài tài liệu có sẵn, hay từ một trang web chứa một lượng thông tin rất đầy đủ như [Wikipedia](#).

Tuỳ thuộc vào lượng thông tin có sẵn để hỗ trợ trong việc đưa ra câu trả lời, hệ thống Question Answering có thể được chia thành hai loại, bao gồm:

- **Closed-domain Question Answering:** trả lời những câu hỏi liên quan đến một miền ứng dụng nhất định, chẳng hạn như trong lĩnh vực Y tế, hoặc lĩnh vực Chính trị...
- **Open-domain Question Answering:** trả lời những câu hỏi liên quan đến mọi thứ có thể, và cần phải dựa vào một kho kiến thức đủ rộng lớn. Một ví dụ của loại hệ thống này là [DrQA](#) [1], được phát triển bởi [Facebook Research](#). Theo đó, nguồn dữ liệu được cung cấp cho hệ thống DrQA là một lượng lớn các bài viết trên Wikipedia, và do những bài viết đó liên quan đến rất nhiều chủ đề khác nhau, nên hệ thống này có thể được coi là một hệ thống Open-domain Question Answering.

## 1.2 Các tập dữ liệu nổi tiếng

Đối với nhiệm vụ Question Answering, để có thể hỗ trợ trong việc huấn luyện và đánh giá một hệ thống, cộng đồng Xử lý ngôn ngữ tự nhiên đã tạo ra một số tập dữ liệu đã được chuẩn hoá. Trong các tập dữ liệu, dữ liệu của mỗi câu hỏi thường bao gồm ba phần cơ bản như sau:

- "**question**": là một câu hỏi để huấn luyện hoặc đánh giá.
- "**text**": là một đoạn văn có thể chứa câu trả lời cho câu hỏi, hoặc không chứa bất kỳ câu trả lời nào.
- "**answer**": là câu trả lời cho câu hỏi, có thể có hoặc không tuỳ vào tập dữ liệu huấn luyện hay đánh giá.

Trong số các tập dữ liệu đó, có một số tập dữ liệu được sử dụng phổ biến như sau, chủ yếu phù hợp với hệ thống Open-domain Question Answering:

1. **Stanford Question Answering Dataset (SQuAD)** [2]: là một bộ dữ liệu được tạo ra bởi các nhà nghiên cứu tại Đại học Stanford. Bộ dữ liệu này bao gồm hơn 100,000 câu hỏi liên quan đến một số chủ đề có trên Wikipedia, với câu trả lời cho mỗi câu hỏi là một hoặc một vài cụm

từ có trong đoạn văn tương ứng với chủ đề trên Wikipedia. Tuy nhiên, trong một vài dữ liệu, câu hỏi có thể không có câu trả lời.

2. **Natural Questions (NQ)** [3]: là một bộ dữ liệu được cung cấp bởi Google. Bộ dữ liệu này bao gồm hơn 300,000 câu hỏi được người dùng hỏi trong thực tế qua công cụ Google Search, với câu trả lời được trích xuất trong một đoạn của một bài viết tại Wikipedia.
3. **Question Answering in Context (QuAC)** [4]: là một bộ dữ liệu liên quan đến hành động tìm kiếm thông tin trong một cuộc hội thoại giữa một học sinh và một giáo viên. Trong bộ dữ liệu này, học sinh sẽ đưa ra một loạt các câu hỏi về một chủ đề trên Wikipedia, và câu trả lời sẽ được giáo viên đưa ra thông qua những đoạn ngắn của bài viết đó trên Wikipedia. Bộ dữ liệu này bao gồm khoảng 100,000 cặp câu hỏi - câu trả lời.
4. **ELI5 (Explain Like I'm Five)** [5]: là một bộ dữ liệu được tạo ra bởi Facebook Research, bao gồm hơn 270,000 cặp câu hỏi - câu trả lời từ [subreddit ELI5](#), cùng với một số dữ liệu trên CommonCrawl.
5. **NewsQA** [6]: là một tập dữ liệu bao gồm hơn 100,000 cặp câu hỏi - câu trả lời liên quan đến các chủ đề trên trang báo điện tử [CNN](#).

### 1.3 Phương pháp đánh giá

Trong lĩnh vực Truy hồi thông tin, cụ thể là đối với nhiệm vụ hỏi đáp, độ chính xác của một hệ thống thường được đánh giá dựa trên mức độ liên quan của câu trả lời đối với câu hỏi. Theo đó, các thông số đánh giá được sử dụng phổ biến trong nhiệm vụ hỏi đáp là  $F_1$  và *accuracy*. Các thông số này sẽ đánh giá xem liệu câu trả lời có liên quan đến câu hỏi hay không, với mức độ liên quan như thế nào.

#### 1.3.1 Thông số *accuracy*

Khi đánh giá dữ liệu, hệ thống sẽ phân loại dữ liệu vào 2 lớp, có thể gọi là 2 lớp *Positive* và lớp *Negative*, và sẽ xảy ra 4 trường hợp như sau:

- **True Positive:** dữ liệu thuộc lớp *Positive* trong thực tế được phân loại **chính xác** vào đúng lớp *Positive*
- **True Negative:** dữ liệu thuộc lớp *Negative* trong thực tế được phân loại **chính xác** vào đúng lớp *Negative*
- **False Positive:** dữ liệu thuộc lớp *Negative* trong thực tế được phân loại **không chính xác** vào lớp *Positive*
- **False Negative:** dữ liệu thuộc lớp *Positive* trong thực tế được phân loại **không chính xác** vào lớp *Negative*

Các trường hợp đó có thể được biểu diễn theo Confusion Matrix như Bảng 1 dưới đây.

Theo đó, thông số *accuracy* được tính theo công thức sau:

$$Accuracy = \frac{True\ Positive + True\ Negative}{True\ Positive + False\ Positive + False\ Negative + True\ Negative}$$

		Predicted	
		Positive	Negative
Actual	Positive	True Positive	False Negative
	Negative	False Positive	True Negative

Bảng 1: Confusion Matrix khi đánh giá phân lớp

### 1.3.2 Thông số $F_1$

Đối với một hệ thống hỏi đáp, khi nhận được một câu hỏi, thường chỉ có duy nhất **một** câu trả lời đúng, tức sẽ được phân lớp *Positive*, còn các câu trả lời khác được coi là sai, tức sẽ được phân lớp *Negative*. Chính vì vậy, có thể thấy được lượng câu trả lời True Negative thường có giá trị rất cao. Từ đó, khi đánh giá theo thông số *accuracy*, kết quả thu được có giá trị cao, tuy nhiên, thông số đó chỉ cao nhờ vào lượng câu trả lời True Negative, nên không mang lại ý nghĩa gì, khi chúng ta chỉ muốn đánh giá hệ thống dựa trên số câu trả lời True Positive.

Để giải quyết vấn đề đó, có một thông số khác được sử dụng phổ biến hơn, đó là *f-measure*. Thông số này cũng được đánh giá dựa theo Confusion Matrix như Bảng 1, và được đánh giá bằng cách sử dụng 2 thông số phụ:

- **Precision:** là phần trăm các câu trả lời được phân lớp chính xác so với các câu trả lời được phân vào lớp *Positive*
- **Recall:** là phần trăm các câu trả lời được phân lớp chính xác so với các câu trả lời thực tế thuộc lớp *Positive*

Thông số *precision* được tính theo công thức sau:

$$\text{Precision} = \frac{\text{True Positive}}{\text{True Positive} + \text{False Positive}}$$

Thông số *recall* được tính theo công thức sau:

$$\text{Recall} = \frac{\text{True Positive}}{\text{True Positive} + \text{False Negative}}$$

Theo đó, với 2 thông số phụ này, có thể thấy rằng lượng câu trả lời True Negative dù cao cũng sẽ không ảnh hưởng đến độ chính xác của hệ thống.

Mặc dù vậy, việc quyết định thông số nào sẽ có ảnh hưởng nhiều hơn với độ chính xác sẽ tuỳ vào từng hệ thống. Ví dụ, đối với hệ thống hỏi đáp cần câu trả lời liên quan đến một chi tiết của sự kiện cần hỏi, thông số *recall* thường quan trọng hơn, do lượng câu trả lời False Positive sẽ không ảnh hưởng quá nhiều đến độ chính xác của hệ thống. Còn đối với hệ thống hỏi đáp cần câu trả lời liên quan đến một định nghĩa nào đó, thông số *precision* thường sẽ đánh giá độ chính xác tốt hơn. Do đó, thông số *f-measure* đã được giới thiệu để cân bằng 2 thông số này, được miêu tả như công thức dưới đây:

$$F_\beta = \frac{(1 + \beta^2) \times \text{precision} \times \text{recall}}{\beta^2 \times \text{precision} + \text{recall}}$$

Trong công thức trên,  $\beta$  là trọng số thể hiện mức độ quan trọng của thông số *recall* so với thông số *precision*. Ví dụ, với  $\beta = 0.5$ , thông số *recall* sẽ được coi là quan trọng một nửa so với thông số *precision*. Trong thực tế, giá trị  $\beta = 1$  thường được sử dụng để biểu diễn mức độ quan trọng tương đương nhau của thông số *recall* và thông số *precision*, để đánh giá thông số  $F_1$  như công thức dưới đây:

$$F_1 = 2 \times \frac{precision \times recall}{precision + recall}$$

### 1.3.3 Các thông số đánh giá khác

Ngoài thông số  $F_1$  và *accuracy*, có một số thông số khác cũng thường được sử dụng để đánh giá một hệ thống hỏi đáp thông qua việc tính toán số lượng câu trả lời liên quan đến câu hỏi, chẳng hạn như thông số *Mean Average Precision (MAP)*, và thông số *Mean Reciprocal Rank (MRR)*.

Cả 2 thông số *MAP* và *MRR* đều được tính dựa theo toàn bộ tập đánh giá gồm  $Q$  truy vấn. Theo đó, *MAP* được tính theo công thức sau:

$$MAP = \frac{\sum_{q=1}^Q AveP(q)}{Q}$$

Trong đó, *AveP* (*Average Precision*) được tính theo một truy vấn  $q$  bất kỳ theo công thức sau:

$$AveP = \frac{\sum_{k=1}^n P(k) \times rel(k)}{|\{relevant\}|}$$

Trong đó,  $k$  là hạng của đoạn văn, và  $rel(k) \in \{0, 1\}$  biểu diễn liệu đoạn văn có hạng  $k$  có liên quan đến câu hỏi hay không.  $P(k)$  là *precision* đối với đoạn văn có hạng  $k$ .

Ngược lại, thông số *MRR* chỉ quan tâm đến hạng của duy nhất **một** đoạn văn có độ liên quan cao nhất đến câu hỏi, và được tính như công thức dưới đây:

$$MRR = \frac{1}{Q} \sum_{q=1}^Q \frac{1}{rank_q}$$

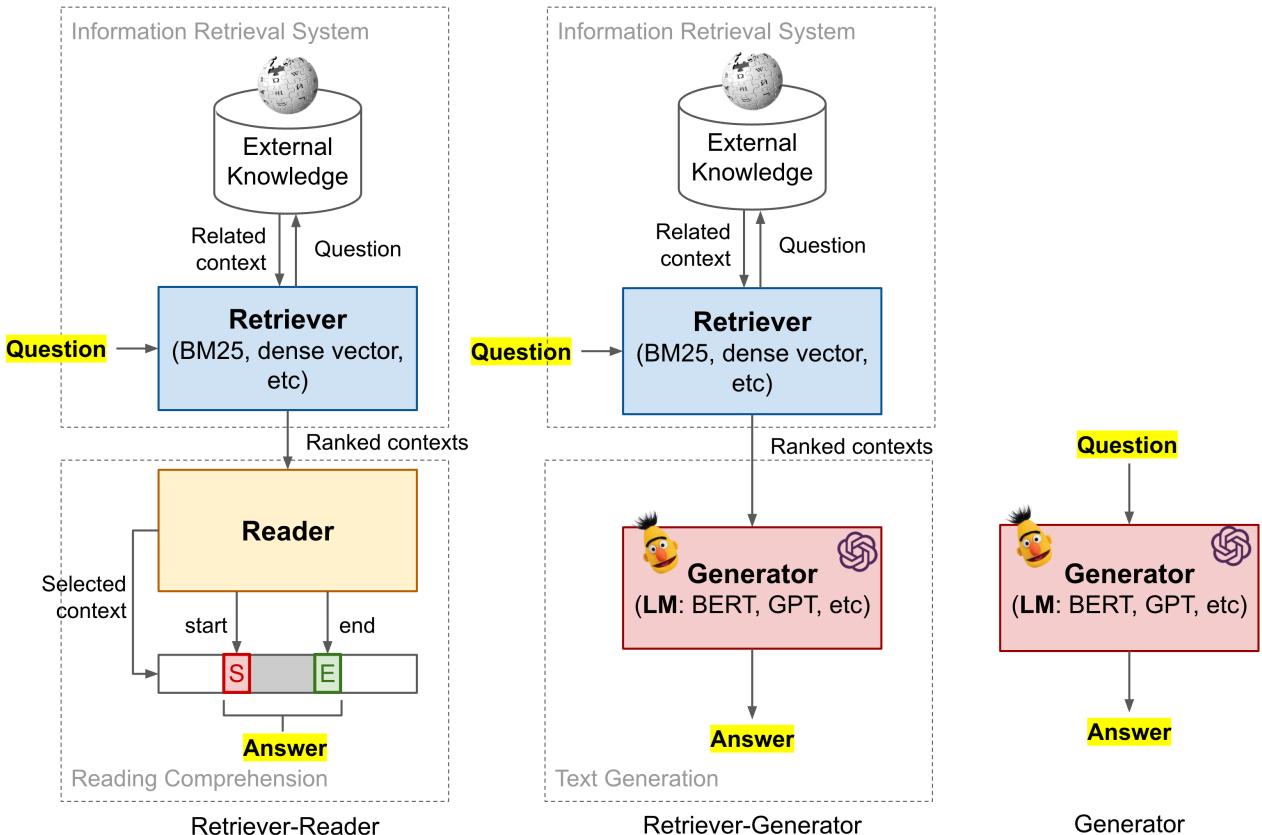
## 1.4 Kiến trúc của một hệ thống Open-domain Question Answering

Trong một bài báo được đưa ra bởi Chen, et al., 2020 [7], kiến trúc của hệ thống Open-domain Question Answering có thể được phân thành 3 loại, được mô tả như ở Hình 1 dưới đây.

- Retriever - Reader
- Retriever - Generator
- Generator

### 1.4.1 Kiến trúc Retriever - Reader

Đối với kiến trúc này, khi cần trả lời một câu hỏi, nếu một hệ thống không có bất kỳ thông tin nào, hoặc không đủ lớn để có thể ghi nhớ thông tin có trong tập dữ liệu huấn luyện, hệ thống đó sẽ khó



Hình 1: Tổng quan về ba kiến trúc của hệ thống Open-domain Question Answering

có thể đưa ra một câu trả lời chính xác cho câu hỏi đó. Chính vì vậy, với kiến trúc này, hệ thống cần phải được cung cấp một kho kiến thức rộng lớn, chẳng hạn như Wikipedia, để từ đó có thể xác định được những đoạn có thể liên quan đến câu trả lời.

Theo đó, quá trình tìm kiếm câu trả lời cho một câu hỏi nào đó có thể được chia ra thành 2 bước như sau:

1. Xác định được đoạn có thông tin liên quan đến câu hỏi trong kho kiến thức được cung cấp,
2. Trích xuất được câu trả lời trong đoạn thông tin nhận được.

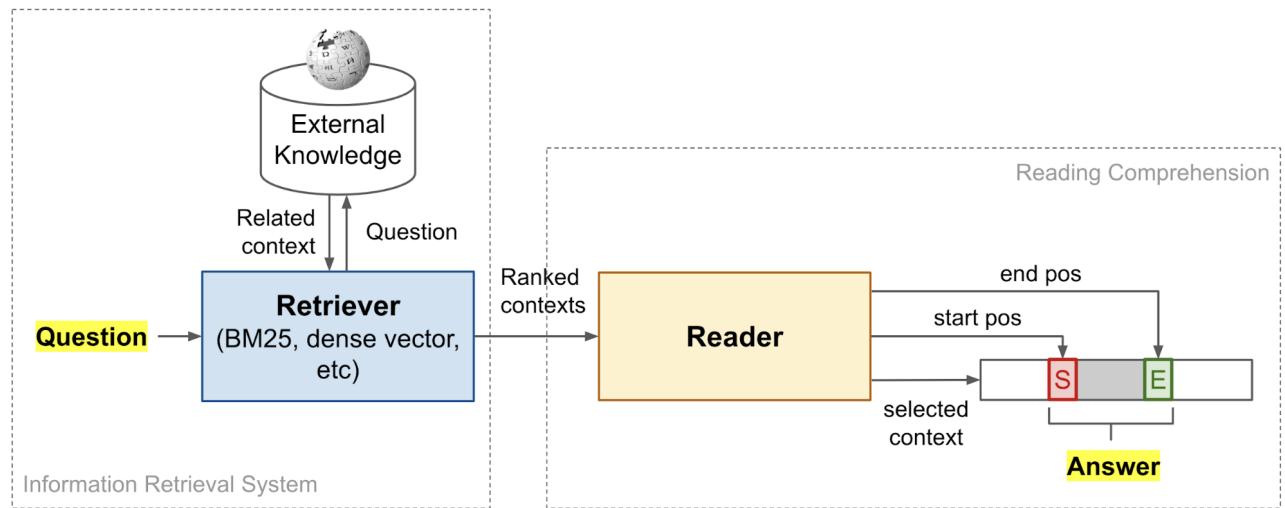
Kiến trúc này được đưa ra lần đầu tiên tại bài báo “Document retriever Question-Answering” bởi Chen et al., 2017 [1]. Theo đó, 2 thành phần Retriever và Reader có thể được huấn luyện và cài đặt một cách độc lập, hoặc được huấn luyện cùng nhau.

#### 1.4.1.1 Thành phần Retriever

Để tạo ra thành phần Retriever, có 2 cách tiếp cận phổ biến là sử dụng một hệ thống truy hỏi thông tin dựa vào các đặc trưng sau khi áp dụng TF-IDF (cách truyền thống), hoặc dựa vào các embedding vectors của đoạn văn được tạo bởi mạng nơ-ron (cách hiện đại).

#### Cách truyền thống

Mô hình **DrQA** sử dụng một cách tìm kiếm hiệu quả dựa vào mô hình vector space. Theo đó, mỗi truy vấn và mỗi đoạn văn được biểu diễn theo một vector bag-of-word, trong đó mỗi từ được biểu diễn



Hình 2: Kiến trúc Retriever - Reader

bởi một trọng số tính theo phương pháp TF-IDF (term frequency  $\times$  inverse document frequency). Chi tiết về mô hình này sẽ được mô tả tại **Mục 2**.

#### Cách hiện đại sử dụng mạng nơ-ron

Thay vì biểu diễn đoạn văn theo một vector space thừa, với các phương pháp tiên tiến như mạng nơ-ron (chẳng hạn như MLP, LSTM, bidirectional LSTM...), một đoạn văn có thể được biểu diễn dưới dạng một dense vector. Theo đó, một số mô hình đi theo cách tiếp cận sau:

$$h_x = E_x(x) \quad h_z = E_z(z) \quad score(x, z) = h_x^T h_z$$

1. Trích xuất dense vector biểu diễn câu hỏi  $x$  và đoạn văn  $z$  bằng cách sử dụng một mạng nơ-ron,
2. Nhân 2 vector với nhau để thu được điểm đánh giá tương quan, từ đó xếp hạng những đoạn văn có mức độ liên quan đến câu hỏi cao nhất.

##### 1.4.1.2 Thành phần Reader

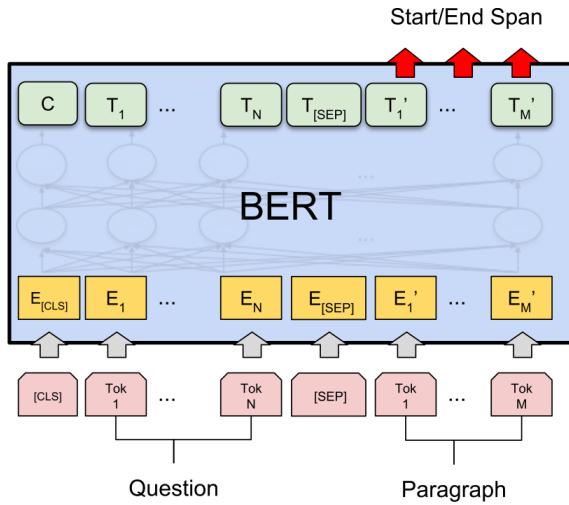
Thành phần Reader được sử dụng với mục đích trích xuất được câu trả lời từ một đoạn văn nhất định. Theo đó, có một số phương pháp sử dụng mô hình mạng nơ-ron được áp dụng phổ biến đối với bài toán này, bao gồm:

- **Bi-directional LSTM:** là mô hình được sử dụng trong DrQA.
- **Các mô hình BERT [8].**

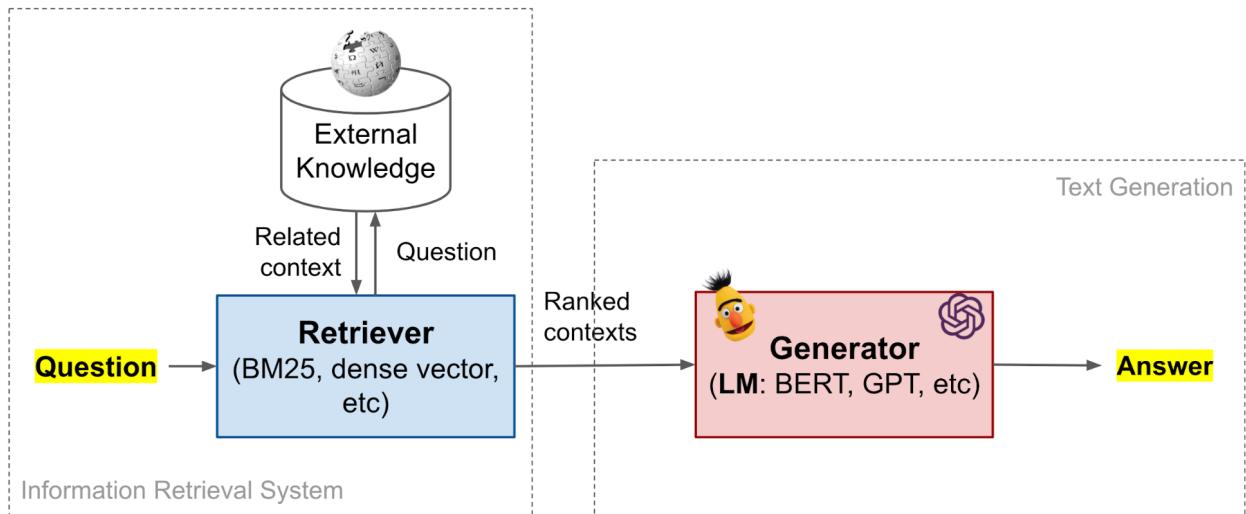
##### 1.4.2 Kiến trúc Retriever - Generator

Khác với kiến trúc Retriever - Reader, đối với kiến trúc Retriever - Generator, mục tiêu của bước thứ 2 là sinh ra một câu trả lời cho câu hỏi hơn là trích xuất câu trả lời dựa vào một đoạn văn.

Theo đó, sau khi những đoạn văn đã được tìm kiếm và xếp hạng, thành phần Generator sẽ sử dụng mô hình BERT, GPT hoặc các mô hình tương tự để có thể sinh ra câu trả lời cho câu hỏi một

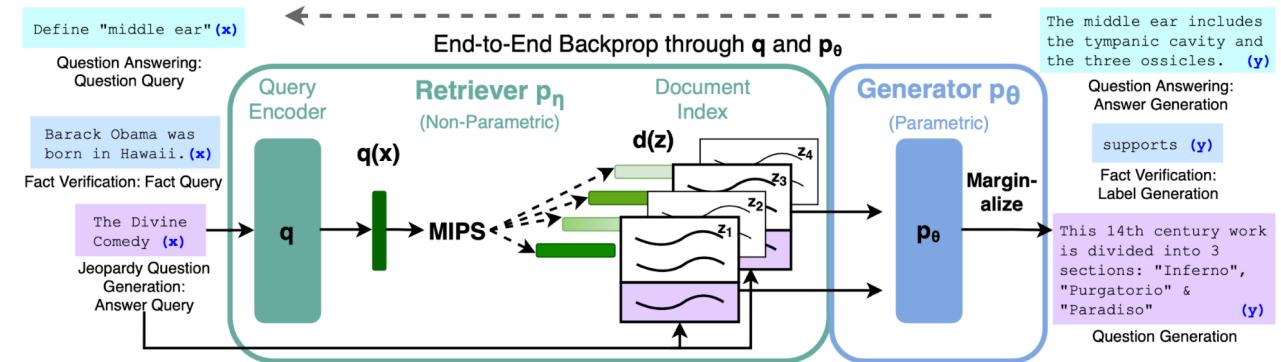


Hình 3: Mô hình BERT được sử dụng trong hệ thống Question Answering



Hình 4: Kiến trúc Retriever - Generator

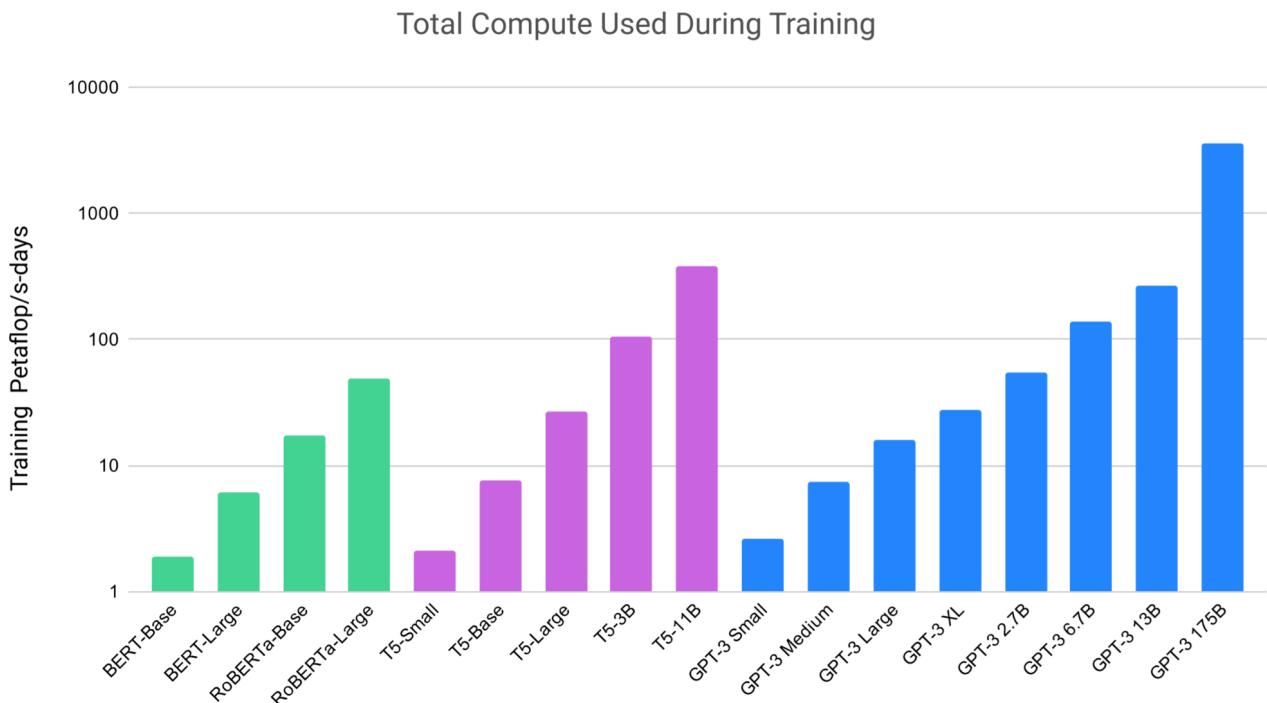
cách hiệu quả nhất. Một mô hình hệ thống đặc trưng cho kiến trúc này là *RAG* (Retrieval-Augmented Generation), được giới thiệu bởi Lewis et al., 2020 [9].



Hình 5: Mô hình RAG

### 1.4.3 Kiến trúc Generator

Đối với một số mô hình ngôn ngữ có kích cỡ lớn, chúng đã được huấn luyện dựa trên một tập lớn các dữ liệu văn bản, từ đó chúng có thể ghi nhớ những kiến thức thông qua các trọng số trong những mô hình đó. Chính vì vậy, những mô hình đó có thể được sử dụng để trả lời những câu hỏi mà không cần cung cấp một đoạn văn chứa câu trả lời cụ thể.



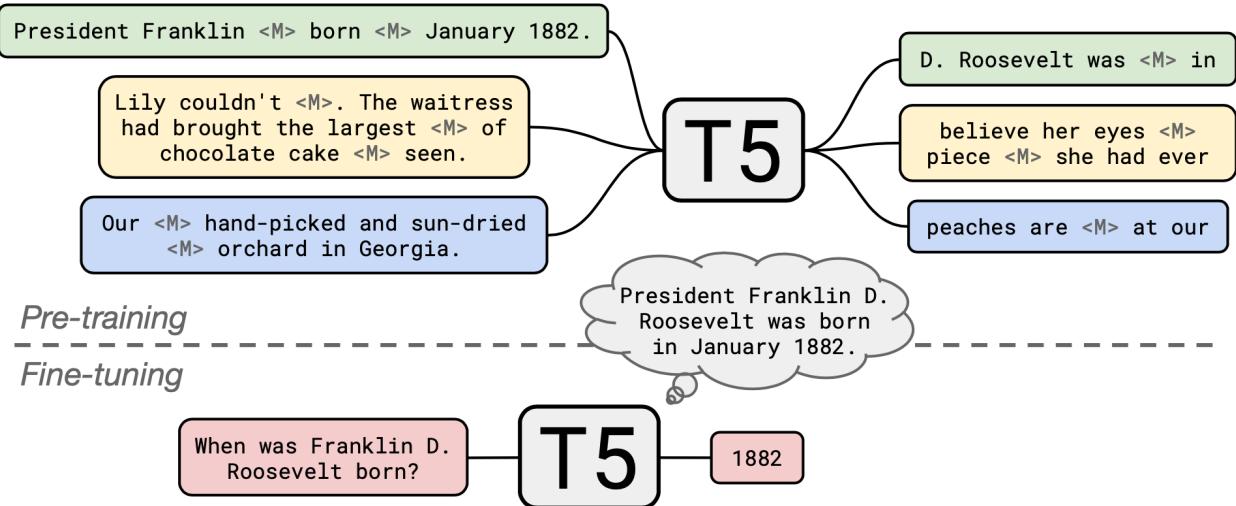
Hình 6: Kích cỡ của một số mô hình ngôn ngữ phổ biến

Theo đó, trong một bài báo được giới thiệu bởi Roberts, et al., 2020 [10], họ đã đánh giá khả năng thực tế của mô hình ngôn ngữ T5 sau khi đã được fine-tuned để trả lời câu hỏi trong điều kiện không biết trước đoạn văn chứa câu trả lời.

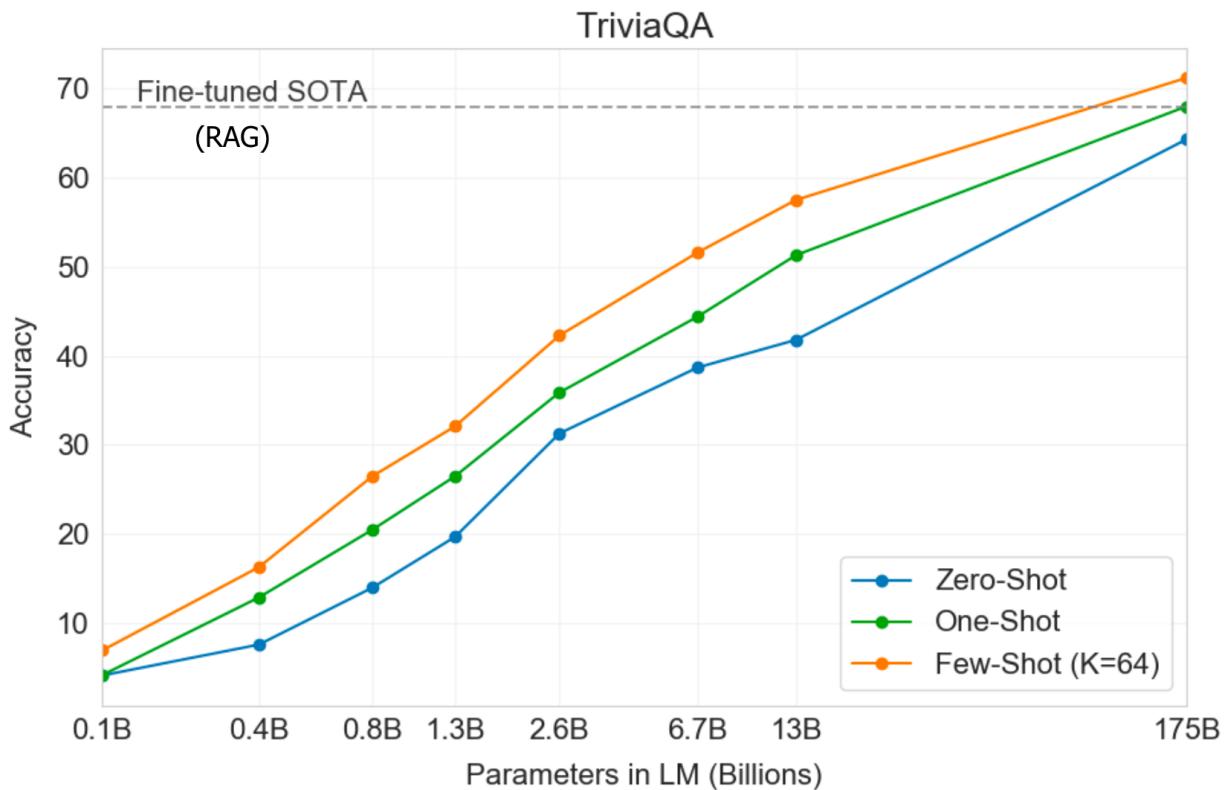
Ngoài ra, trong một số trường hợp, một mô hình không nhất thiết cần phải được fine-tuned để có thể sử dụng trong việc trả lời câu hỏi, chẳng hạn như mô hình GPT-3. Theo đó, trong một bài báo được giới thiệu bởi Brown et al., 2020 [11], họ đã đánh giá khả năng trả lời câu hỏi của mô hình qua 3 trường hợp cụ thể như sau:

1. **few-shot learning:** mô hình GPT-3 có thể nhận được một số chỉ dẫn về ngữ cảnh,
2. **one-shot learning:** mô hình GPT-3 chỉ được nhận duy nhất **một** chỉ dẫn về ngữ cảnh,
3. **zero-shot learning:** mô hình GPT-3 không được nhận bất kỳ một chỉ dẫn nào về ngữ cảnh.

Theo đó, với tập dữ liệu TriviaQA, khả năng của mô hình GPT-3 có thể đạt được tương đương, hoặc thậm chí còn vượt trội hơn so với một mô hình khác đã được fine-tuned. Ngoài ra, cũng có thể thấy khả năng hoạt động của GPT-3 tỉ lệ thuận với kích cỡ của mô hình, như Hình 8 dưới đây.



Hình 7: Mô hình T5 được fine-tuned để trả lời câu hỏi

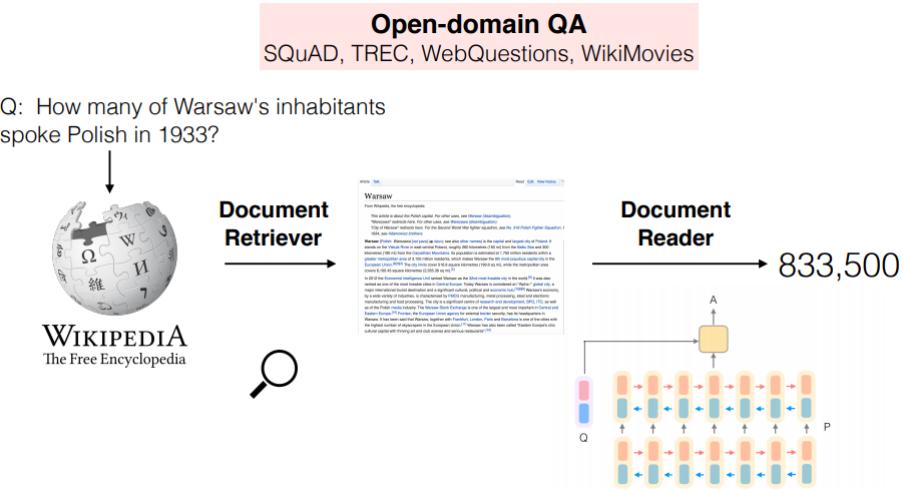


Hình 8: Khả năng của GPT-3 đối với tập dữ liệu TriviaQA

## 2 DrQA

### 2.1 Tổng quan

DrQA [1] là một trong những hệ thống QA tiêu biểu được Facebook Research công bố vào năm 2017, một năm sau khi các nghiên cứu trong bài toán machine reading comprehension và question answering nở rộ kể từ sự ra đời của bộ dữ liệu SQuAD [2]. Nó sử dụng kiến trúc Retriever - Reader. Mục tiêu của DrQA là trả lời các câu hỏi thuộc dạng open-domain. DrQA sử dụng nguồn dữ liệu từ Wikipedia làm cơ sở tri thức (knowledge base).



Hình 9: Tổng quan về hệ thống DrQA

Phương pháp bao gồm 2 phần chính là bộ lấy tài liệu (Document Retriever) và bộ đọc tài liệu (Document Reader). Trong đó

1. **Document Retriever** thu hẹp phạm vi tìm kiếm, từ tập các tài liệu (document) ban đầu, lấy ra một tập nhỏ các tài liệu có liên quan nhất. Cụ thể với DrQA, tập tài liệu là tập các bài viết trên Wikipedia, Document Retriever sẽ lấy ra  $k = 5$  tài liệu liên quan nhất đến câu hỏi.
2. **Document Reader** nhận đầu vào là tập các bài viết liên quan được lấy ra từ bước Document Retriever, sau đó tìm ra đoạn văn có khả năng cao nhất là đáp án.

### 2.2 Document Retriever

DrQA sử dụng cách đánh giá bằng TF-IDF, một phương pháp hiệu quả được sử dụng trong các công cụ tìm kiếm (search engine) mà không cần qua quá trình học. TF-IDF là viết tắt của term frequency - inverted document frequency. Giá trị này xuất phát từ một nhận xét rằng: một từ sẽ càng có ý nghĩa với việc tìm kiếm nếu nó xuất hiện nhiều lần trong một tài liệu, và nó chỉ xuất hiện trong một số ít các tài liệu.

Để tính toán giá trị này, trước tiên, truy vấn và các tài liệu được chuyển về dưới dạng vector

bag-of-word. Mỗi từ  $t$  ứng với tài liệu  $d$  và tập tài liệu  $D$  sẽ được tính giá trị tf-idf như sau:

$$tfidf(t, d, D) = tf(t, d) \times idf(t, D)$$

$$tf(t, d) = \log(1 + freq(t, d))$$

$$idf(t, D) = \log \frac{|D|}{|d \in D : t \in d|}$$

Với  $freq(t, d)$  là số lần từ  $t$  xuất hiện trong tài liệu  $d$ .  $t$  ở đây có thể là unigram hoặc bigram term. DrQA sử dụng một tokenizer để thực hiện phân đoạn câu cũng như xác định từ loại cho các từ trong câu. Đối với một tài liệu  $d$ , giá trị của tài liệu đối với truy vấn  $q$  được tính như sau:

$$score(q, d, D) = \sum_{t \in q} tfidf(t, d, D)$$

DrQA mặc định lấy ra  $k = 5$  tài liệu có điểm số cao nhất với câu hỏi.

## 2.3 Document Reader

Document Reader sẽ nhận đầu vào bao gồm:

- Một câu hỏi  $q$  được biểu diễn dưới dạng  $\{q_1, q_2, \dots, q_l\}$ , với  $l$  là số token ở trong câu hỏi
- Một tập các tài liệu, trong đó, mỗi tài liệu được chia thành các đoạn văn, mỗi đoạn văn  $p$  được biểu diễn bởi  $\{p_1, p_2, \dots, p_m\}$ , với  $m$  là số token của đoạn văn

Mô hình Document Reader sẽ có 3 phần chính: Paragraph Encoding, Question Encoding và Prediction.

### Paragraph Encoding

Ta biểu diễn các token  $p_i$  dưới dạng một vector đặc trưng  $\tilde{p}_i$ . Vector đặc trưng  $\tilde{p}_i$  bao gồm những thành phần sau:

- Nhúng từ (Word embedding):  $f_{emb}(p_i) = E(p_i)$ . Glove word embedding được sử dụng để thực hiện việc embedding. Đa số các từ pre-trained word embedding đều được cố định và chỉ thực hiện fine-tune 1000 từ thường xuyên xuất hiện nhất.
- Khớp chính xác (Exact match):  $f_{exact\_match}(p_i) = \prod(p_i \in q)$ . Sử dụng 3 binary features để biểu diễn  $p_i$  có khớp với một từ ở trong câu hỏi  $q$ , có thể ở dạng ban đầu (giống như trong câu hỏi), không phân biệt chữ hoa, thường (case-insensitive), hoặc ở dạng nguyên bản (lemma form, ví dụ như các từ break, breaks, broke và broken có chung dạng nguyên bản là break).
- Đặc trưng của token:  $f_{token}(p_i) = (POS(p_i), NER(p_i), TF(p_i))$ . Các đặc trưng về từ loại (POS - part of speech), loại thực thể của các đối tượng có tên (NER - named entity recognition), và tần suất xuất hiện của từ (TF - term frequency).
- Sự giống hàng với các từ trong câu hỏi (Aligned question embedding):  $f_{align}(p_i) = \sum_j a_{i,j} E(q_j)$ , trong đó trọng số attention  $a_{i,j}$  thể hiện sự giống hàng (alignment) giữa  $p_i$  và mỗi từ  $q_j$  trong

câu hỏi. Cụ thể,  $a_{ij}$  được tính bằng tích chấm giữa các ánh xạ phi tuyến (nonlinear mappings) của các word embedding:

$$a_{ij} = \frac{\exp(\alpha(E(p_i)) \cdot \alpha(E(q_j)))}{\sum_{j'} \exp(\alpha(E(p_i)) \cdot \alpha(E(q_{j'})))}$$

trong đó  $\alpha()$  là một lớp fully-connected với hàm activation phi tuyến ReLU. So với đặc trưng exact match, đặc trưng này bổ sung thêm các gióng hàng tương đối (soft alignment) giữa các từ tương tự nhưng không giống nhau (ví dụ: car và vehicle).

Tập vector đặc trưng  $\tilde{p}_1, \tilde{p}_2, \dots, \tilde{p}_m$  sau đó sẽ được đưa qua RNN để thu được tập vector  $\mathbf{p}_i$ , với kì vọng sẽ mã hóa được các thông tin về ngữ cảnh:

$$\{\mathbf{p}_1, \mathbf{p}_2, \dots, \mathbf{p}_m\} = \text{RNN}(\{\tilde{p}_1, \tilde{p}_2, \dots, \tilde{p}_m\})$$

## Question Encoding

Câu hỏi  $q$  được encode thành tổng có trọng số của các từ trong câu hỏi. Cụ thể, vector  $\mathbf{q}$  được biểu diễn dưới dạng  $\mathbf{q} = \sum b_j \times q_j$ , trong đó  $b_j$  thể hiện độ quan trọng của mỗi từ trong câu hỏi. Công thức tính  $b_j$  như sau:

$$b_j = \text{softmax}(w^T E(x_j))$$

Trong đó  $w$  là một vector có thể học được.

## Prediction

Sau khi có các vector đặc trưng của câu hỏi và các đoạn văn, với mỗi vị trí  $i$ , ta cần tính toán xác suất là vị trí bắt đầu  $P_{start}(i)$  và vị trí kết thúc  $P_{end}(i)$ .

$$\begin{aligned} P_{start}(i) &\propto \exp(\mathbf{p}_i \mathbf{W}_s \mathbf{q}) \\ P_{end}(i) &\propto \exp(\mathbf{p}_i \mathbf{W}_e \mathbf{q}) \end{aligned}$$

Trong đó,  $\mathbf{W}_s$  và  $\mathbf{W}_e$  là các tham số có thể học được. Ta sẽ cần tìm 2 chỉ số  $i_s$  và  $i_e$  sao cho  $P_{start}(i_s) \times P_{end}(i_e)$  đạt giá trị lớn nhất với điều kiện  $i_s \leq i_e \leq i_s + 15$ .

## 2.4 Thí nghiệm - đánh giá

### 2.4.1 Đánh giá Document Retriever

Việc đánh giá được thực hiện trên các bộ dữ liệu được nêu trong bài báo DrQA [1], trong đó tiêu biểu là bộ SQuAD. Độ chính xác của Document Retriever được tính dựa trên tỉ lệ đáp án của câu hỏi nằm trong top  $k = 5, 4, 3, 2$  bài viết.

Chi tiết cài đặt: Document Retriever sử dụng mô hình tách từ (tokenizer) spacy với các term được đẽ dưới dạng unigram, hash-size =  $2^{24}$ .

Kết quả so sánh giữa công cụ tìm kiếm của Wiki và DrQA như ở Bảng 2 dưới đây.

Dataset	Wiki Search	DrQA Document Retriever			
		$k = 5$	$k = 4$	$k = 3$	$k = 2$
SQuAD 1.1	62.7	70.8	66.38	63.80	59.15
CuratedTREC	81.0	75.26	71.83	67.24	64.30
WebQuestions	73.7	71.26	68.43	64.91	61.76
WikiMovies	61.7	68.55	65.05	64.29	60.66

Bảng 2: Kết quả so sánh giữa Wiki Search với DrQA

#### 2.4.2 Đánh giá pipeline

Với Document Retriever được sử dụng ở **Mục 2.2**, chúng em đã thử nghiệm hệ thống khi số lượng tài liệu được lấy ra sau bước retrieval khác nhau, và thu được kết quả như Bảng 3 dưới đây.

Model	EM	F <sub>1</sub>
DrQA ( $k = 2$ )	24.67	25.6
DrQA ( $k = 3$ )	25.34	27.96
DrQA ( $k = 4$ )	27.78	30.89
DrQA ( $k = 5$ )	29.0	33.07
DrQA (Chen et al., 2017) [1]	29.5	-

Bảng 3: Kết quả đánh giá DrQA trên bộ dữ liệu SQuAD với số lượng tài liệu được lấy ra từ 2 đến 5

### 2.5 Thảo luận

Chúng em nhận thấy hệ thống DrQA gặp phải những điểm yếu như sau:

- Chưa cân nhắc tối độ dài và số lượng của văn bản trong bước retrieval. Việc truy hồi top 5 bài viết trên Wikipedia là chưa thực sự tối ưu.
- Số lượng lớn đoạn văn mà mô hình reader phải xử lý cho mỗi câu hỏi có nhiều thông tin thừa, gây ra độ nhiễu lớn.
- Mô hình reader dựa trên bi-LSTM hiện đã lỗi thời, bộc lộ nhiều nhược điểm so với mô hình Transformer (sẽ được đề cập ở phần tiếp theo).

Chính vì vậy, chúng em đã nghiên cứu các phương pháp, các hướng tiếp cận hiện đại hơn để cải tiến Hệ thống DrQA. Chi tiết về chúng được viết trong **Mục 3**.

### 3 Cải tiến

#### 3.1 Các phương pháp

##### 3.1.1 BM25 Retriever

BM25 là một phương pháp xếp hạng tương tự như tf-idf, được sử dụng rộng rãi trong tìm kiếm và truy hồi thông tin.

Cho một câu hỏi  $Q$ , chứa các từ khóa  $\{q_1, q_2, \dots, q_n\}$ , BM25 tính điểm cho một văn bản theo công thức:

$$\text{score}(D, Q) = \sum_{i=1}^n \text{IDF}(q_i, D) \cdot \frac{f(q_i, D) \cdot (k_1 + 1)}{f(q_i, D) + k_1 \cdot \left(1 - b + b \cdot \frac{|D|}{avgdl}\right)}$$

Trong đó, trọng số IDF của từ khóa  $q_i$  ứng với văn bản  $D$  được tính bởi:

$$\text{IDF}(q_i, D) = \log \frac{N - n(q_i) + 0.5}{n(q_i) + 0.5}$$

- $N$  : Số lượng văn bản trong kho ngữ liệu
- $n(q_i)$  : số văn bản có chứa từ
- $f(q_i, D)$  : (Term frequency) Tần suất xuất hiện của từ  $q_i$  trong văn bản  $D$
- $|D|$  : Tổng số từ trong văn bản  $D$
- $avgdl$  : Average document length, độ dài trung bình của các văn bản trong kho dữ liệu
- $k_1$  : Tham số tự chọn, thường là 2
- $b$  : Tham số tự chọn, thường là 0.75

Công thức tính IDF trên có một nhược điểm. Đối với những từ phổ biến, xuất hiện trong hơn một nửa số văn bản trong tập văn bản, IDF sẽ có giá trị âm. Như vậy đối với 2 văn bản tương tự bất kỳ, 1 văn bản có chứa 1 từ, còn văn bản khác không chứa từ đó, thì văn bản không chứa từ đó sẽ có điểm cao hơn. Như vậy những từ phổ biến sẽ có ảnh hưởng không tốt đến đánh giá văn bản. Vì thế một số trường hợp công thức sẽ được điều chỉnh như sau:

1. Bỏ qua tất cả những số hạng có giá trị âm (việc này tương đương với việc cho những từ phổ biến vào stop-list và bỏ qua tất cả những từ phổ biến).
2. Quy định giá trị thấp nhất cho  $\text{IDF} = \epsilon$ , nếu giá trị IDF nhỏ hơn  $\epsilon$  thì tính IDF bằng  $\epsilon$ .
3. Sử dụng những công thức IDF không cho giá trị âm.

Trong bài tập này, BM25 retriever mà chúng em sử dụng được cài đặt trong thư viện Pyserini [12], dựa trên Anserini với Lucene index. Một đặc điểm tốt của phần cài đặt này so với DrQA tf-idf retriever là nó truy hồi các văn bản từ file index được lưu trên đĩa thay vì lưu một ma trận lên tới 13GB trong RAM để reranking. Dù vậy tốc độ truy hồi của Pyserini vẫn đảm bảo real-time (dưới 1s cho mỗi câu hỏi). Vậy chúng em đã không gặp phải vấn đề về thiếu bộ nhớ RAM mà vẫn đảm bảo tốc độ xử lý.

### 3.1.2 Mô hình BERT

BERT, viết tắt của Bidirectional Encoder Representations from Transformers, được đề xuất bởi Devlin et al., 2018 [8], là một mô hình cho mục đích hiểu ngôn ngữ nói chung, được huấn luyện trên một lượng dữ liệu lớn bao gồm tập BooksCorpus và Wikipedia tiếng Anh. BERT có thể được huấn luyện tinh chỉnh với các tác vụ xử lý ngôn ngữ tự nhiên cụ thể như là question answering (hỏi đáp), natural language inference (suy diễn ngôn ngữ), sentiment analysis (phân tích sắc thái văn bản), NER ... và đã đạt được các kết quả SOTA trong 11 bài toán tính ở thời điểm bài báo được công bố, trong đó có bài toán question answering.

Trong các mục tiếp theo chúng em sẽ mô tả chi tiết hơn về BERT, về kiến trúc và việc áp dụng vào bài toán question answering.

#### Các hướng tiếp cận trước

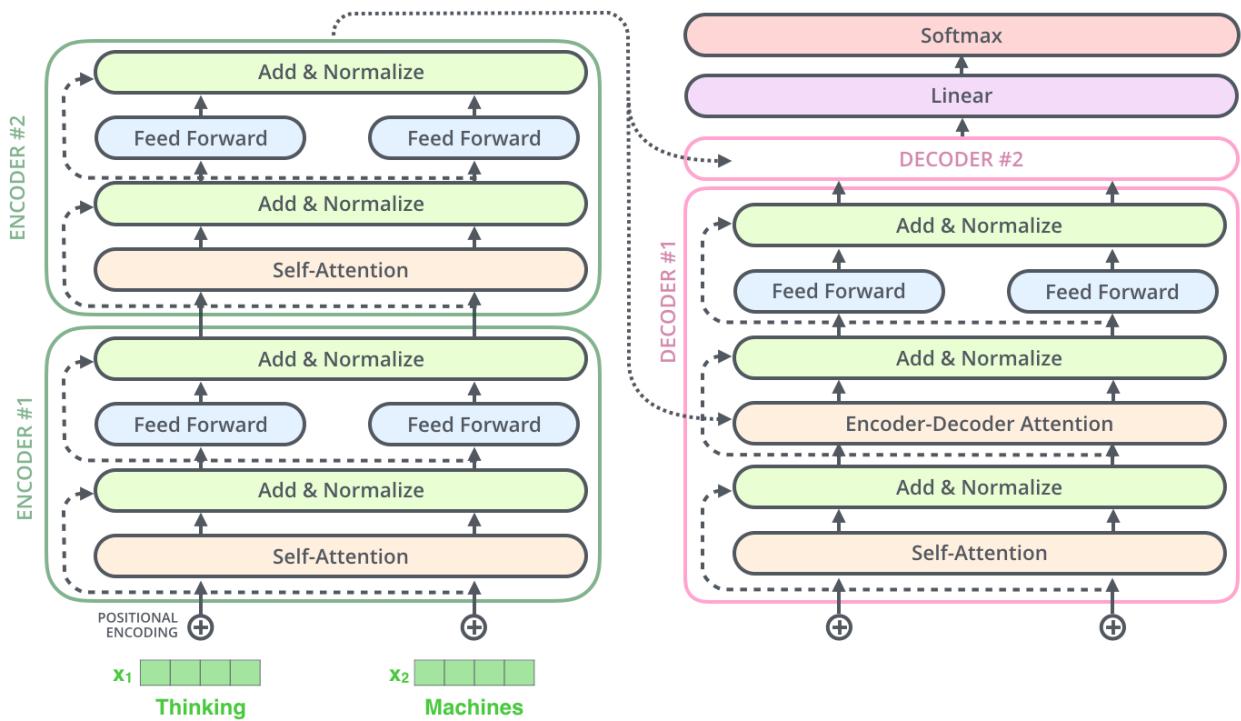
Các biểu diễn ngôn ngữ đã được pretrained trước đây có thể là độc lập với ngữ cảnh (ví dụ như là WordNet hoặc là Glove word embeddings, trong đó một từ chỉ có một biểu diễn vector cố định, vậy ví dụ từ “bank” sẽ có biểu diễn như nhau trong hai ngữ cảnh “river bank” và “bank deposit”) hoặc có thể là phụ thuộc theo ngữ cảnh. Các mô hình ngôn ngữ phụ thuộc ngữ cảnh có thể chỉ theo một chiều (hoặc là trái qua phải hoặc ngược lại, hoặc là kết hợp cả hai chiều nhưng theo một cách tuần tự đối với bi-RNN).

Các tác giả của mô hình BERT cho rằng các phương pháp trước đó hạn chế sự hiểu văn bản của mô hình. Chẳng hạn như với bài toán đọc hiểu, hỏi đáp, nó cần nhiều sự liên hệ, kết hợp ngữ cảnh xung quanh. BERT là mô hình ngôn ngữ đầu tiên hiểu, biểu diễn, nhúng (encode) văn bản một cách “deeply bidirectional”, hay còn có thể nói là “nondirectional” (vô hướng). Điều này là nhờ kiến trúc Transformer và phương pháp pretrain hợp lý, khác với các phương pháp trước mà trong đó mô hình chỉ được huấn luyện theo một cách có hướng.

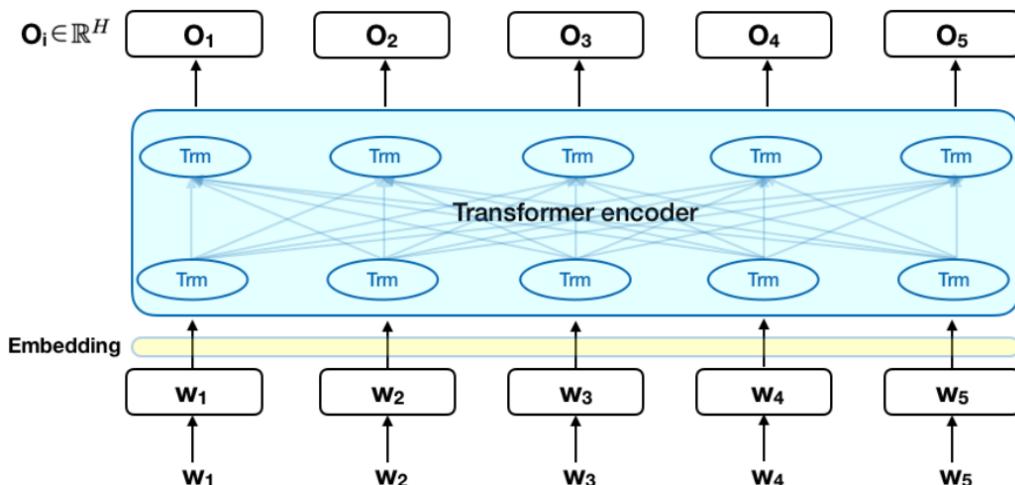
#### Kiến trúc mô hình BERT

Kiến trúc của mô hình BERT bao gồm nhiều lớp bidirectional Transformer Encoder thuộc kiến trúc Transformer được đề xuất bởi Vaswani et al., 2017 [13]. Transformer (Hình 10) là mô hình đầu tiên dựa theo cơ chế self-attention, độc lập khác biệt hoàn toàn với mạng neural hồi quy (recurrent neural network - RNN) mà đã từng rất phổ biến trước đây. Việc không xử lý tuần tự các step như RNN giúp Transformer tận dụng được khả năng tính toán song song tốt hơn, từ đó tăng tốc độ huấn luyện và xử lý thông tin. Mô hình Transformer tuân theo kiến trúc encoder - decoder, nhưng tác vụ chính của BERT là hiểu ngôn ngữ, là học được cách biểu diễn hay nhúng ngôn ngữ đầu vào thành một vector dày nên nó chỉ có những khối encoder của Transformer mà thôi.

Để trình bày chi tiết về các cơ chế sâu bên trong của mô hình Transformer thì thực sự còn rất nhiều thứ để viết (Cơ chế self-attention, tích chấm attention, multi-head attention, positional encoding...). Nhưng nhóm chúng em xin được tập trung vào nội dung chính là chủ đề question answering. Vì vậy chúng em sẽ refer đến bài báo Attention is all you need bởi Vaswani et al., 2017 [13] và bài viết The Illustrated Transformer với các hình minh họa rất trực quan bởi Jay Alammar [14].



Hình 10: Kiến trúc Transformer



Hình 11: Tổng quan BERT

## Mô tả đầu vào và đầu ra

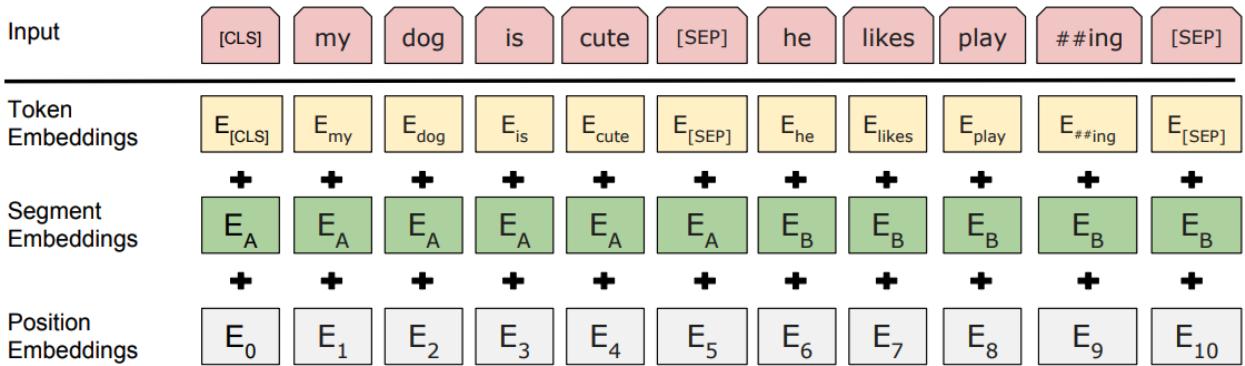
Xét góc nhìn tổng quan tại Hình 11:

- Đầu vào của mô hình là một chuỗi các token  $\{w_1, w_2, \dots, w_n\}$ , đầu tiên chúng được nhúng thành các vector embedding và sau đó được mô hình xử lý.
- Đầu ra là một chuỗi các vector  $\{o_1, o_2, \dots, o_n\}$ , mỗi vector có kích thước  $H$ , và tương ứng với một token đầu vào cùng vị trí. Với  $H$  là số hidden state, một tham số khi khởi tạo mô hình, ví dụ là 768.

Đầu vào của BERT phải làm rõ được văn bản đơn (cho bài toán như là phân loại) và các cặp văn bản (cho bài toán như question answering với cặp câu hỏi - câu trả lời) trong một chuỗi đầu vào. Vì vậy, mô hình sử dụng [CLS], [SEP] là các token đặc biệt. [CLS] viết tắt của classification, luôn được

đặt ở đầu chuỗi, vector output ở vị trí token này sẽ được sử dụng cho bài toán phân loại. [SEP] viết tắt của separate, là token đặc biệt cho việc đánh dấu kết thúc câu và phân tách giữa hai câu nếu mô hình nhận input là hai câu (ví dụ như ở Task 2, hoặc bài toán question answering).

Với mỗi token đầu vào, vector đầu vào ứng với token đó được tạo bởi 3 yếu tố như trong Hình 12:



Hình 12: Mô tả đầu vào mô hình BERT

- Chuỗi đầu vào được xử lý thành chuỗi các token thông qua WordPiece embeddings (được viết rõ hơn phía dưới). Kí tự ## đánh dấu sự phân tách của từ.
- Khi đầu vào là một cặp câu, segment embeddings phân biệt token thuộc câu thứ nhất hay câu thứ hai.
- Positional embeddings giúp mô hình có ý thức về vị trí, thứ tự của các token. (xem thêm trong bài viết về mô hình Transformer)

WordPiece [15] là một thuật toán subword tokenizer được huấn luyện với phương pháp không giám sát (unsupervised), được dùng để xử lý chuỗi đầu vào cho các mô hình xử lý ngôn ngữ dựa trên mạng neural. Tập từ vựng được xác định trong quá trình huấn luyện tokenizer, trước khi huấn luyện mạng neural. WordPiece biểu diễn được các mảnh con của một từ (subword) (ví dụ như là giống với byte-pair-encoding (BPE) [16]). WordPiece ban đầu khởi tạo tập từ vựng bao gồm tất cả các ký tự có trong kho ngữ liệu huấn luyện và dần dần kết hợp các ký tự để tạo nên các subword sao cho cực đại hóa sự hợp lý (likelihood) của dữ liệu huấn luyện.

## Pretraining BERT

Trong quá trình pretrain BERT, nhóm tác giả sử dụng hai phương pháp huấn luyện unsupervised là masked Language model và next sentence prediction trên lượng dữ liệu lớn bao gồm tập BooksCorpus và Wikipedia tiếng Anh. Việc được huấn luyện như vậy giúp cho mô hình BERT đạt được kết quả tốt đáng kể.

### Task 1: Masked Language Model

Trong tác vụ này, họ che giấu (mask) một số từ khỏi input, và yêu cầu mô hình đưa ra các dự đoán để khôi phục những chỗ bị che giấu từ các từ và ngữ cảnh xung quanh. Tác vụ này còn được gọi là Cloze task. Dưới đây là ví dụ về dữ liệu được sinh ra để pretrain mô hình.

Input: [CLS] the man went to [MASK1] store and he bought a gallon [MASK2] milk [SEP]  
Label: [MASK1] = grocery ; [MASK2] = of

### Task 2: Next Sentence Prediction

Pretrain mô hình với phương pháp masked language model là chưa đủ để giúp cho việc fine-tuning đạt kết quả tốt ở các bài toán con, vì mô hình có thể chưa học được mối liên hệ giữa các câu với nhau. Để giải quyết vấn đề này, nhóm tác giả đã tiếp cận bằng bài toán phân loại nhị phân, mô hình nhận đầu vào là hai câu A và B nối với nhau qua token [SEP] và dự đoán xem câu B có phải là câu nối tiếp của A không.

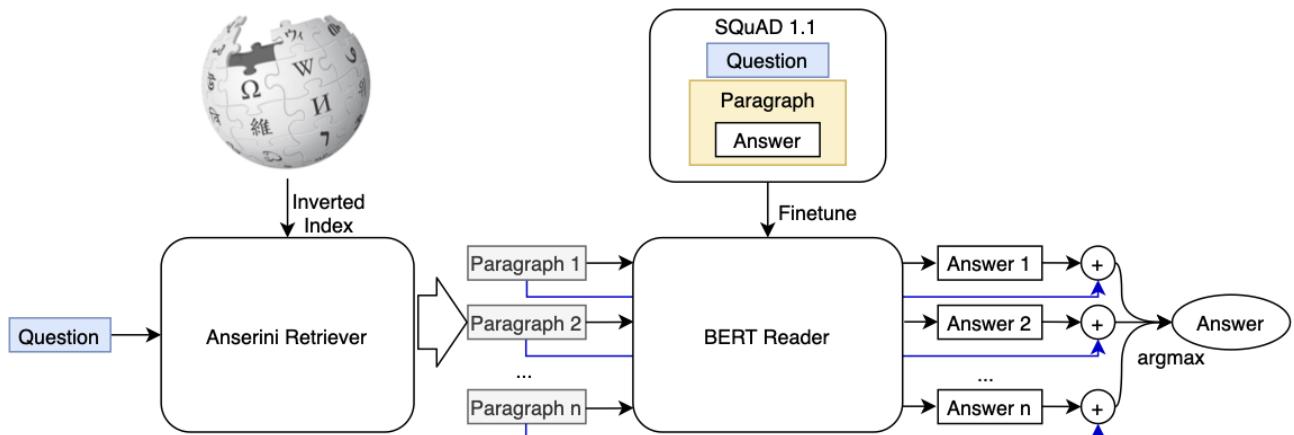
Input: [CLS] the man went to [MASK] store [SEP] he bought a gallon [MASK] milk [SEP]  
Label: IsNext  
Input: [CLS] the man [MASK] to the store [SEP] penguin [MASK] are flight ##less birds [SEP]  
Label: NotNext

### BERT trong bài toán question answering

Với bài toán question answering (Hình 3), mô hình BERT sẽ gắn thêm một lớp mạng neural phân loại cho mỗi hidden vector  $T_i$  tương ứng với token  $i^{th}$  xem token nào bắt đầu câu trả lời và token nào kết thúc câu trả lời. Lớp này có thể được biểu diễn bằng hai vector tham số  $S, E \in R^H$ , được học trong quá trình fine-tune mô hình. Cuối cùng xác suất để token  $i^{th}$  là bắt đầu hoặc kết thúc câu trả lời sẽ được tính bằng tích chấm giữa  $T_i$  và  $S$  hoặc  $E$ , cùng với hàm softmax:

$$P_{start}^i = \frac{e^{S \cdot T_i}}{\sum_i e^{S \cdot T_i}} \quad \text{and} \quad P_{end}^i = \frac{e^{E \cdot T_i}}{\sum_i e^{E \cdot T_i}}$$

#### 3.1.3 BERTserini



Hình 13: Minh họa pipeline hệ thống BERTserini

BERTserini phát triển bởi Yang et al., 2019 [17] kết hợp công cụ mã nguồn mở Anserini IR toolkit làm thành phần retriever với mô hình BERT đã được tinh chỉnh trên bộ SQuAD để làm thành phần reader. Retriever hoạt động theo thuật toán BM25, thu về  $k$  ( $k = 100$ ) đoạn văn có độ tương quan

lớn nhất và đưa vào mô hình reader xử lý.

Bài báo này còn thử nghiệm tác động của độ chi tiết của văn bản thu về vào kết quả của hệ thống (Hình 14). Kết luận cuối cùng là truy hồi các đoạn văn cho kết quả tốt nhất (với  $k = 100$ ) và nhìn chung tốt hơn truy hồi các câu riêng rẽ, và tốt hơn rất nhiều so với truy hồi cả các bài viết như là DrQA ( $k = 5$ ).

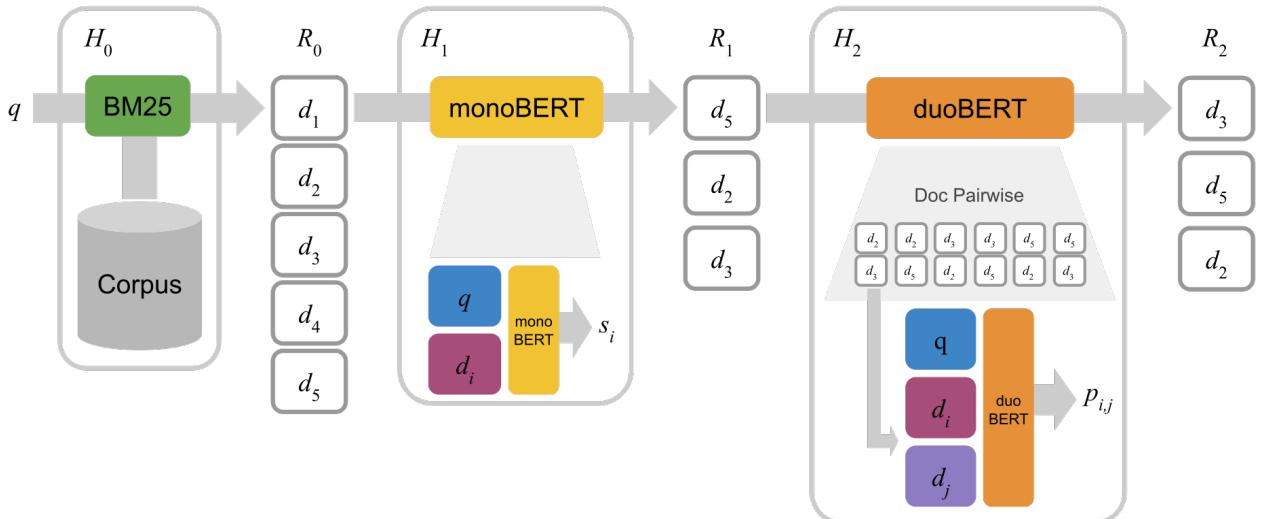
Model	EM	F1	R
BERTserini (Article, $k = 5$ )	19.1	25.9	63.1
BERTserini (Paragraph, $k = 29$ )	36.6	44.0	75.0
BERTserini (Sentence, $k = 78$ )	34.0	41.0	67.5
<b>BERTserini (Paragraph, <math>k = 100</math>)</b>	<b>38.6</b>	<b>46.1</b>	<b>85.8</b>

Hình 14: Bảng so sánh kết quả giữa các chiến lược retrieval

Điểm khác biệt của mô hình đọc BERTserini so với BERT nguyên gốc là lớp softmax cuối cùng bị lược bỏ. Điểm của mô hình đọc sẽ chính là điểm logits thay vì xác suất. Điều này là hợp lý vì khi normalize theo hàm softmax cho mỗi câu hỏi và đoạn văn, ta coi như câu hỏi và đoạn văn như một instance riêng rẽ. Nhưng trong bài toán này, một câu hỏi ứng với  $k$  đoạn văn, vì vậy ta cần tổng hợp (aggregate) kết quả trên toàn bộ  $k$  instance.

Ngoài ra khi đánh giá các câu trả lời có thể, điểm cuối cùng của câu trả lời được tính bằng tổng của điểm retriever và điểm của mô hình reader với các trọng số nhất định (ví dụ 50 - 50). Câu trả lời với điểm cuối cùng cao nhất sẽ được chọn.

### 3.1.4 Multi-passage BERT



Hình 15: Minh họa pipeline hệ thống Multi-passage BERT

ElasticSearch + BM25 được sử dụng bởi Multi-passage BERT QA (Wang et al., 2019 [18]) trong

bước đầu để lấy ra  $k_1$  đoạn văn ( $k_1 = 100$ ). Sau đó sử dụng một passage ranker bắn chắt là một mô hình BERT nữa để tính điểm phù hợp của đoạn văn với câu hỏi và giới hạn lại còn  $k_2$  đoạn văn có điểm cao nhất ( $k_2 = 30$ ). Cuối cùng sử dụng mô hình BERT đã được tinh chỉnh cho bài toán question answering để đưa ra câu trả lời. Cũng gần giống với BERTserini, điểm cuối cùng của câu trả lời được tổng hợp từ điểm passage ranker và điểm của mô hình reader.

Passage ranker được cài đặt với BERT thông thường và với một lớp output tính điểm cho mỗi đoạn văn dựa vào hidden vector của token đặc biệt [CLS]. Nhóm tác giả áp dụng hàm softmax để normalize điểm của toàn bộ passage ứng với mỗi câu hỏi và huấn luyện mô hình để cực đại hóa log-likelihood của đoạn văn chứa câu trả lời đúng. Sử dụng passage ranker đã cải thiện kết quả trên tập Open SQuAD-dev 2% so với là không sử dụng.

### 3.1.5 Learning to Retrieve Reasoning Paths

Phương pháp này được đề xuất bởi Asai et al., 2020 [19] với ý tưởng chính là tận dụng sơ đồ liên kết của các hyperlink trong các văn bản trên Wikipedia để có thể thu về được các đoạn văn bản phù hợp hơn, chính xác hơn với câu hỏi. Với điều này, hệ thống có thể xem xét các hyperlink dẫn đến các văn bản có liên quan trên Wikipedia để tìm thêm các nội dung chính xác hơn cho câu hỏi. Cải tiến này giúp hệ thống đạt kết quả tốt hơn Multi-passage BERT, theo như Bảng 4.

Model	Open SQuAD-dev	
	EM	F <sub>1</sub>
DrQA (Chen et al., 2017) [1]	29.5	-
BERTserini (Yang et al., 2019) [17]	40.6	48.1
Multi-passage BERT (Wang et al., 2019) [18]	53.0	60.9
Path Retriever (Asai et al., 2020) [19]	56.5	63.8

Bảng 4: So sánh kết quả của các hệ thống

## 3.2 Thí nghiệm - đánh giá

### DrQA retriever và BERT reader

Trong thí nghiệm này chúng em cài tiến hệ thống DrQA bằng cách thay thành phần reader cũ là mạng bi-LSTM thành một số mô hình cải tiến của BERT. Chúng em không sử dụng BERT bởi thời gian xử lý của mô hình này với một lượng lớn văn bản (lên tới 5 bài viết Wikipedia) là khá lâu, trên 5s trên một GPU tốt trên Google Colab. Vì vậy chúng em sử dụng những phiên bản nhẹ hơn của BERT mà cho kết quả không kém hơn một cách đáng kể. Chúng bao gồm: DistilBERT bởi Sanh et al. [20], MobileBERT bởi Sun et al. [21], ALBERT bởi Lan et al. [22].

Về chi tiết cài đặt của thành phần reader BERT, chúng em sử dụng framework transformers [23] phát triển bởi HuggingFace. Các mô hình được chúng em thử nghiệm đã được pretrained, fine-tuned và công bố trên kho các mô hình của HuggingFace:

- DistilBERT: [distilbert-base-cased-distilled-squad](#)

- MobileBERT: [csarron/mobilebert-uncased-squad-v1](#)
- ALBERT: [madlag/albert-base-v2-squad](#)

Model	EM	F <sub>1</sub>	Latency
DrQA (Chen et al., 2017) [1]	29.5	-	~0.5s
DrQA retriever + DistilBERT-base (n_docs = 5)	31.0	35.2	2.58s
n_docs = 4	<b>31.9</b>	<b>36.9</b>	<b>2.07s</b>
n_docs = 3	31.6	35.5	1.53s
n_docs = 2	30.3	35.1	1.03s
DrQA retriever + MobileBERT	<b>32.0</b>	<b>37.5</b>	<b>2.35s</b>
DrQA retriever + ALBERT	-	-	-

Bảng 5: Thí nghiệm với DrQA retriever và các mô hình BERT trên tập Open SQuAD-dev

Kết quả thí nghiệm ở Bảng 5 cho thấy việc sử dụng BERT đã cho kết quả tốt hơn bi-LSTM nhưng không quá đáng kể. Vấn đề của hệ thống đó là thành phần retriever của DrQA hoạt động chưa tối ưu. Theo cấu hình mặc định thì có tới 5 bài viết dài của Wikipedia được truy hồi, tính theo số đoạn văn sẽ là rất lớn, lên tới 200 - 500 đoạn văn, vậy sẽ có rất nhiều đoạn thừa, không liên quan, gây nhiễu và trễ cho mô hình BERT. Chúng em đã thực hiện giảm lượng bài viết được truy hồi và kết quả nhận được là tốt hơn, tốt nhất với n\_docs = 4.

### Pyserini và BERT reader

Để tiếp tục cải tiến hệ thống, chúng em dựa theo ý tưởng BERTserini, với việc truy hồi các đoạn văn thay vì cả bài viết. Chúng em đặt số lượng đoạn văn truy hồi với mỗi câu hỏi là 30, kết quả trên độ đo EM / F<sub>1</sub> với tập Open SQuAD-dev thu được là **37.3 / 43.9**.

## 4 Ứng dụng

### 4.1 Ứng dụng cho tiếng Việt

#### 4.1.1 Chuẩn bị dữ liệu

Chúng em đã tổng hợp được một số nguồn dữ liệu sau:

- SQuAD-translate:** chính là bộ SQuAD [2] ban đầu được dịch sang tiếng Việt qua Google API. Tuy có số lượng dữ liệu khá lớn (hơn 100 nghìn câu), điểm yếu của tập dữ liệu này là chất lượng bởi nó được dịch bằng máy do đó dẫn đến việc dữ liệu thiếu tự nhiên hoặc bị thiên lệch.
- vi-wiki** [24]: là một bộ dữ liệu tiếng Việt được tạo nên bởi Nguyễn Việt Nam trong quá trình làm luận án tốt nghiệp về chủ đề hệ thống hỏi đáp. Bộ dữ liệu này có tổng cộng 710 cặp câu hỏi - trả lời, dựa trên 81 đoạn văn được lấy từ các bài viết trên Wikipedia tiếng Việt.
- XQuAD** [25]: là một bộ dữ liệu nhằm đánh giá tác vụ cross-lingual question answering, hỏi đáp đa ngôn ngữ. Nó bao gồm 240 đoạn văn và 1190 cặp câu hỏi - câu trả lời được trích từ tập validation của bộ SQuAD v1.1 [2] và sau đó được dịch bởi người có chuyên môn thành 10 ngôn ngữ khác nhau: tiếng Anh, tiếng Ả-rập, tiếng Đức, tiếng Tây Ban Nha, tiếng Ấn Độ, tiếng Việt, tiếng Trung Quốc, tiếng Hy Lạp, tiếng Nga, tiếng Thổ Nhĩ Kỳ, và tiếng Thái.

En	During what time period did the Angles migrate to Great Britain?
	The name "England" is derived from the Old English name Englaland [...] The Angles were one of the Germanic tribes that settled in Great Britain during the <b>Early Middle Ages</b> . [...] The Welsh name for the English language is "Saesneg"
De	Während welcher Zeitsperiode migrierten die Angeln nach Großbritannien?
	Der Name England leitet sich vom altenglischen Wort Englaland [...] Die Angeln waren ein germanischer Stamm, der das Land im <b>Frühmittelalter</b> besiedelte. [...] ein Verweis auf die weißen Klippen von Dover.
Ar	في أي حقبة هاجر الأنجل إلى بريطانيا العظمى؟
	والتي تعنى "[أرض الأنجل]"، والأنجل كانت واحدة، يشتقت اسم "إنجلترا" من الكلمة الإنجليزية القديمة من القبائل герمانية التي استقرت في إنجلترا خلال <b>وقت العصور الوسطى</b> . [...] وقد سماها العرب قديماً بالأنكلن.
Vi	Trong khoảng thời gian nào người Angles di cư đến Anh?
	Tên gọi của Anh trong tiếng Việt bắt nguồn từ tiếng Trung. [...] Người Angle là một trong những bộ tộc German định cư tại Anh trong <b>Thời đại Trung Cổ</b> . [...] dường như nó liên quan tới phong tục gọi người German tại Anh là Angli Saxones hay Anh - Sachsen.

(a)

(b)

Hình 16: Một số ví dụ dữ liệu đa ngôn ngữ

- MLQA** [26]: MultiLingual Question Answering là một bộ dữ liệu nhằm đánh giá tác vụ cross-lingual question answering, hỏi đáp đa ngôn ngữ. MLQA bao gồm hơn 5 nghìn cặp câu hỏi - câu trả lời ở cấu trúc tương tự bộ SQuAD, với mỗi một trong 7 ngôn ngữ khác nhau trong đó có tiếng Việt. MLQA được cho là có tính song song về mặt ngữ nghĩa cao. Bảng 6 dưới đây thể hiện lượng dữ liệu trong tập MLQA.
- UIT-ViQuAD** [27]: Vietnamese Question Answering Dataset, là một bộ dữ liệu do nhóm nghiên cứu NLP thuộc Trường Đại học Công nghệ Thông tin, Đại học Quốc gia Thành phố Hồ Chí Minh

tạo ra, a new dataset for the low-resource language as Vietnamese to evaluate MRC models. Bộ dữ liệu này gồm có hơn 23,000 cặp câu hỏi - câu trả lời được người dán nhãn, dựa trên 5,109 đoạn văn của 174 bài viết tiếng Việt trên Wikipedia.

Loại tập	en	de	es	ar	zh	vi	hi
dev	1148	512	500	517	504	511	507
test	11590	4517	5254	5335	5137	5495	4918

Bảng 6: Lượng cặp câu hỏi - trả lời trong bộ dữ liệu MLQA

Về dữ liệu làm knowledge base cho hệ thống hỏi đáp, chúng em sử dụng tập các văn bản trên Wikipedia tiếng Việt. Tập dữ liệu này sau khi xử lý bằng công cụ WikiExtractor [28] để lấy raw text, nặng khoảng 1GB. Sau đó chúng em lưu vào một cơ sở dữ liệu SQLite và tạo file tfidf cho việc retrieval theo như DrQA. Đối với phương pháp retrieval bằng BM25, chúng em tách các văn bản thành các đoạn văn và index chúng bằng công cụ Pyserini [12].

#### 4.1.2 Huấn luyện mô hình

Nhóm đã thử nghiệm với 3 mô hình đó là ALBERT, PhoBERT và XLM-RoBERTa.

##### ALBERT-vi

ALBERT (A Lite BERT) [22] là một phiên bản cải tiến của BERT [8]. Bài báo công bố kết quả mà ALBERT đạt được cao hơn và có số lượng tham số bé hơn nhiều so với BERT nhờ chiến lược chia sẻ tham số. Nhóm chúng em sử dụng bản pretrained được open-sourced tại [https://github.com/ngoanpv/albert\\_vi](https://github.com/ngoanpv/albert_vi). Tuy nhiên thì thực tế cho kết quả không khả quan hơn và dù nó có lượng tham số bé hơn hẳn, thời gian chạy cho kết quả của ALBERT cũng không nhanh hơn. Điều này có thể do nó chỉ được thực hiện pretrained trên bộ dữ liệu khá khiêm tốn là Wiki Tiếng Việt (chỉ với khoảng 1GB) nên kết quả có thể chưa thực sự tốt.

##### PhoBERT

PhoBERT [29] được viện nghiên cứu VinAI giới thiệu trong năm 2020. Mô hình này được pretrained trên lượng dữ liệu lớn đáng kể là 20GB. Nhìn chung mô hình này cho kết quả khá tốt, tuy nhiên nó cũng có một số điểm trừ. Nó không được cài đặt kèm với fast tokenizer, điều này làm cho tốc độ xử lý giảm một phần nhỏ và gây khó khăn trong việc cài đặt pipeline. Ngoài ra khi sử dụng PhoBERT cần phải có thêm một bước tiền xử lý nữa, đó là sử dụng công cụ tách từ (word segmentator) của VnCoreNlp [30] nên khá bất tiện.

##### XLM-RoBERTa

Mô hình XLM-RoBERTa được đề xuất bởi Conneau et al. [31], với một phương pháp hợp lý trong việc pretraining các mô hình đa ngôn ngữ với dữ liệu lớn. Điều này giúp cho mô hình cải thiện đáng kể khả năng ở nhiều tác vụ đa ngôn ngữ khác nhau. Mô hình này tốt hơn đáng kể so với mô hình Bert đa ngôn ngữ (mBERT) trên tất cả các điểm đánh giá, như là XNLI, MLQA, NER... XLM-RoBERTa được pretrained trên lượng dữ liệu khổng lồ, lên tới 2.5TB dữ liệu của 100 ngôn ngữ, từ hai nguồn

chính là Wikipedia và CommonCrawl [32].

Mô hình XLM-RoBERTa hiện đang là SOTA cho tiếng Việt ở bài toán question answering dựa theo bài báo công bố bộ dữ liệu UIT-ViQuAD [27] và cùng với đó là mã nguồn mở [vireader](#) ứng với nghiên cứu này. Hơn nữa, trong công bố gần đây hơn của Do et al., [33] nhóm nghiên cứu NLP tại Trường Đại học Công nghệ Thông tin, Đại học Quốc gia Thành phố Hồ Chí Minh đã thực hiện so sánh khả năng của XLM-RoBERTa với PhoBERT, cùng với một số mô hình khác. Kết quả là mô hình XLM-RoBERTa vẫn nắm vị trí số một.

## Quá trình huấn luyện

Việc huấn luyện các mô hình được tiến hành trên môi trường của Google Colab với GPU NVIDIA Tesla T4, với các tham số như sau:

- "learning-rate" =  $3 \times 10^{-5}$
- "batch-size" = 16
- "max-seq-length" = 384 (256 với PhoBERT)
- "doc-stride" = 128

Với bộ dữ liệu SQuAD-translate, số step warm-up là: 1000 step đầu (tương ứng với khoảng 1/10 tổng số step), số epoch là 4, tương ứng với hơn 10000 step. Với bộ dữ liệu tổng hợp của MLQA, XQuAD và vi-wiki thì số step warm-up là 500, số epoch là 5, tương ứng với gần 3000 step.

Tổng thời gian quá trình huấn luyện diễn ra trong khoảng 2 - 4 giờ, tùy vào mỗi mô hình và dữ liệu huấn luyện.

## Đánh giá kết quả

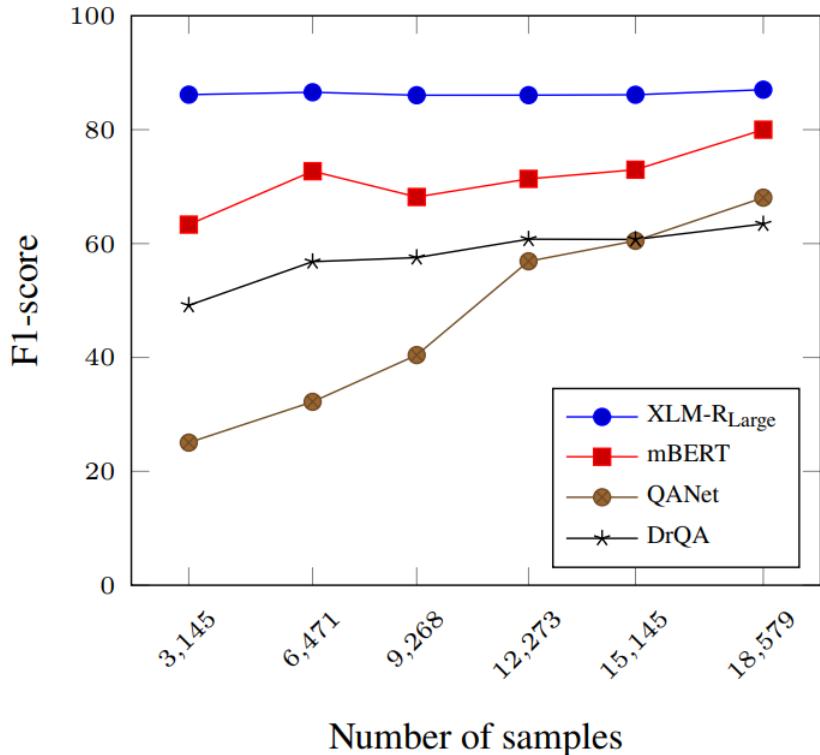
Chúng em đánh giá mô hình thông qua độ đo EM / F<sub>1</sub> trên 2 tập dữ liệu là vi-wiki-test và MLQA-dev với lần lượt 111 và 511 câu hỏi. Để chọn ra mô hình phù hợp với tính real-time của hệ thống hỏi đáp, chúng em cũng xem xét đến tốc độ xử lý của các mô hình, dựa theo throughput được tính bằng số lượng cặp câu trên giây (Throughput này được đo trên GPU NVIDIA Tesla K80 trên môi trường Google Colab).

Dữ liệu huấn luyện	Mô hình	Tham số	Throughput	vi-wiki-test	MLQA-dev
SQuAD-translate (~100k pairs)	ALBERT-vi-base	12M	12.2/s	32.4 / 48.8	26.2 / 42.1
	PhoBERT-base	135M	17.6/s	45.0 / 63.6	37.6 / 57.2
	XLM-R-base	270M	15.1/s	45.9 / 65.5	40.9 / 59.8
MLQA + XQuAD (6685 pairs)	XLM-R-base	270M	15.1/s	<b>52.3 / 67.0</b>	<b>44.4 / 64.5</b>

Bảng 7: Bảng đánh giá mô hình đọc hiểu

Từ nghiên cứu của Trường Đại học Công nghệ Thông tin, Đại học Quốc gia Thành phố Hồ Chí Minh [27], một điểm cộng của mô hình XLM-RoBERTa cho tiếng Việt nữa đó là nó có thể cho kết

quả tốt ngay cả khi không được cung cấp quá nhiều dữ liệu huấn luyện. Điều này một phần lớn do việc được pretrained với dữ liệu lớn và khả năng xử lý đa ngôn ngữ của mô hình này. Kết quả thực nghiệm được thể hiện ở Hình 17 dưới đây:



Hình 17: Biểu đồ so sánh kết quả giữa các mô hình đọc hiểu dựa trên số lượng dữ liệu

Bảng 7 cũng cho thấy mặc dù mô hình XLM-RoBERTa được huấn luyện trên tập dữ liệu khá bé là MLQA + XQuAD với chỉ hơn 6 nghìn dòng dữ liệu nhưng nó lại cho kết quả tốt hơn đáng kể so với bộ SQuAD-translate với gần 100 nghìn dòng dữ liệu. Kết quả này có thể được giải thích bởi hai yếu tố là bộ SQuAD-translate có chất lượng không tốt và mô hình XLM-RoBERTa có thể đạt kết quả tốt chỉ với ít dữ liệu.

Từ đây nhóm chúng em nhận thấy rằng mô hình XLM-RoBERTa đang có kết quả tốt nhất cho tác vụ question answering cho tiếng Việt. Vì thế, chúng em chọn sử dụng mô hình này cho ứng dụng hỏi đáp cho tiếng Việt.

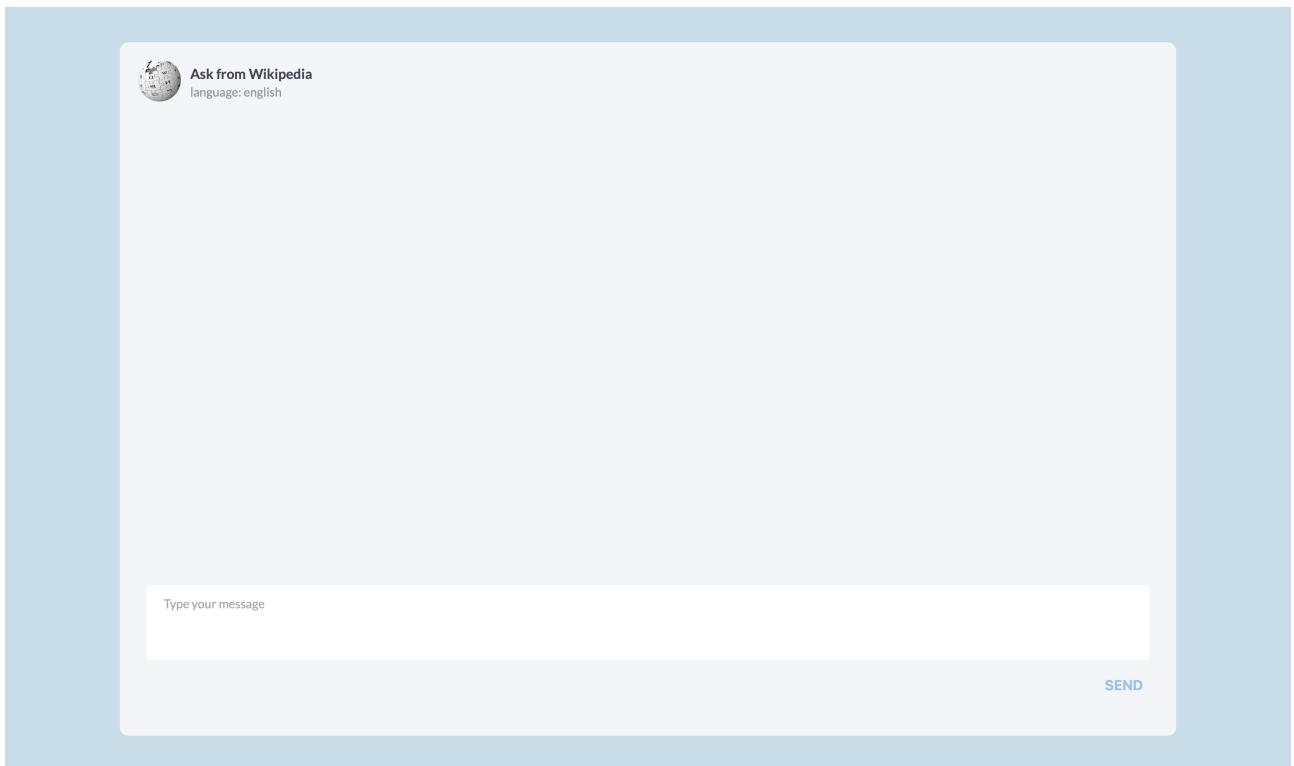
## 4.2 Web UI

Dựa vào dự án mã nguồn mở [zagzaghi/drqa-webui](#), chúng em đã có chỉnh sửa lại một phần mã nguồn tương ứng với những cải tiến chúng em đã đề cập tới tại **Mục 3** và **Mục 4.1**. Ngoài ra, chúng em cũng đã thay đổi giao diện người dùng sao cho phù hợp với kích thước lớn của các thiết bị máy tính.

Mã nguồn đã chỉnh sửa được lưu trữ tại repo [namlh201/drqa-webui](#), và được đóng gói thành một submodule trong repo chính tại [hoangvuduyanh33/QA](#).

#### 4.2.1 Giao diện tiếng Anh

Giao diện chính của hệ thống sau khi truy cập như Hình 18 dưới đây:



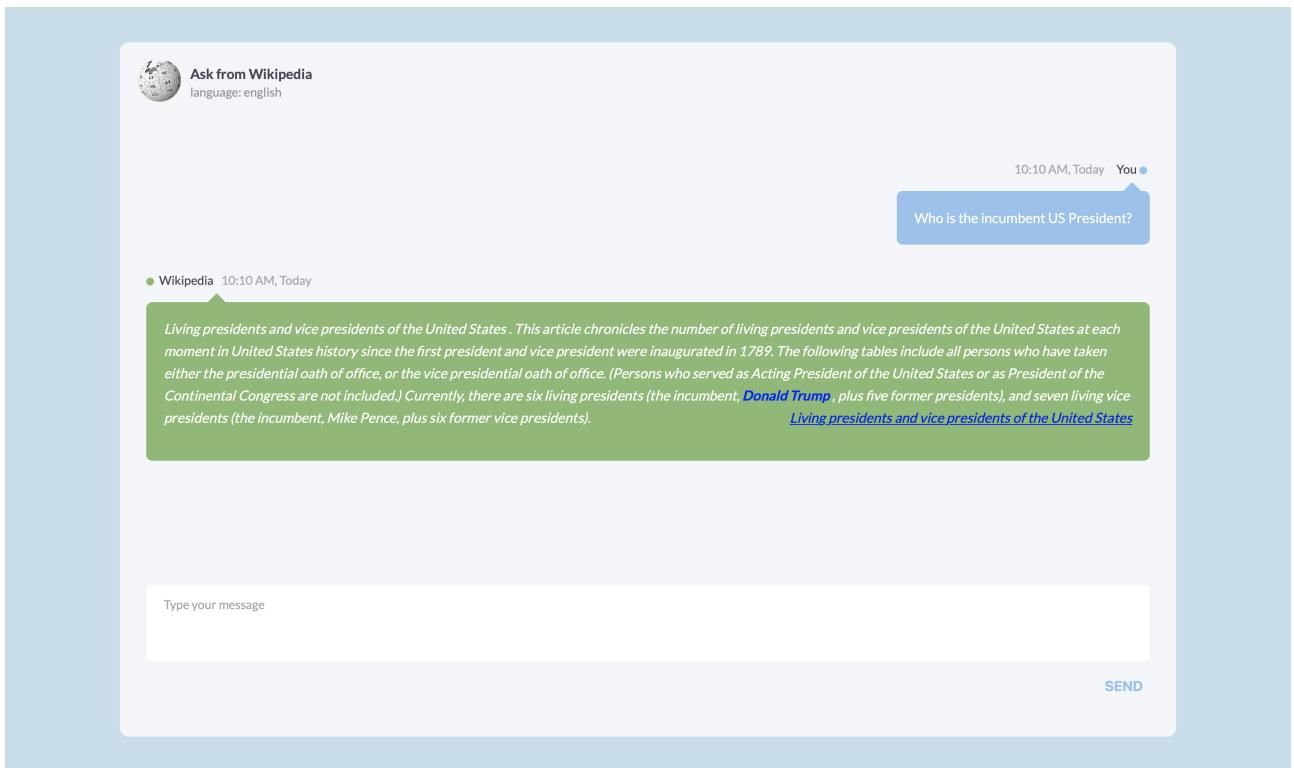
Hình 18: Giao diện chính của hệ thống

Khi đó, hệ thống sẽ cung cấp một khu vực để người dùng có thể điền câu hỏi vào. Sau khi được hỏi, hệ thống sẽ phản hồi lại đoạn văn chứa câu trả lời, với câu trả lời sẽ được đánh dấu **màu xanh** và **in đậm**.

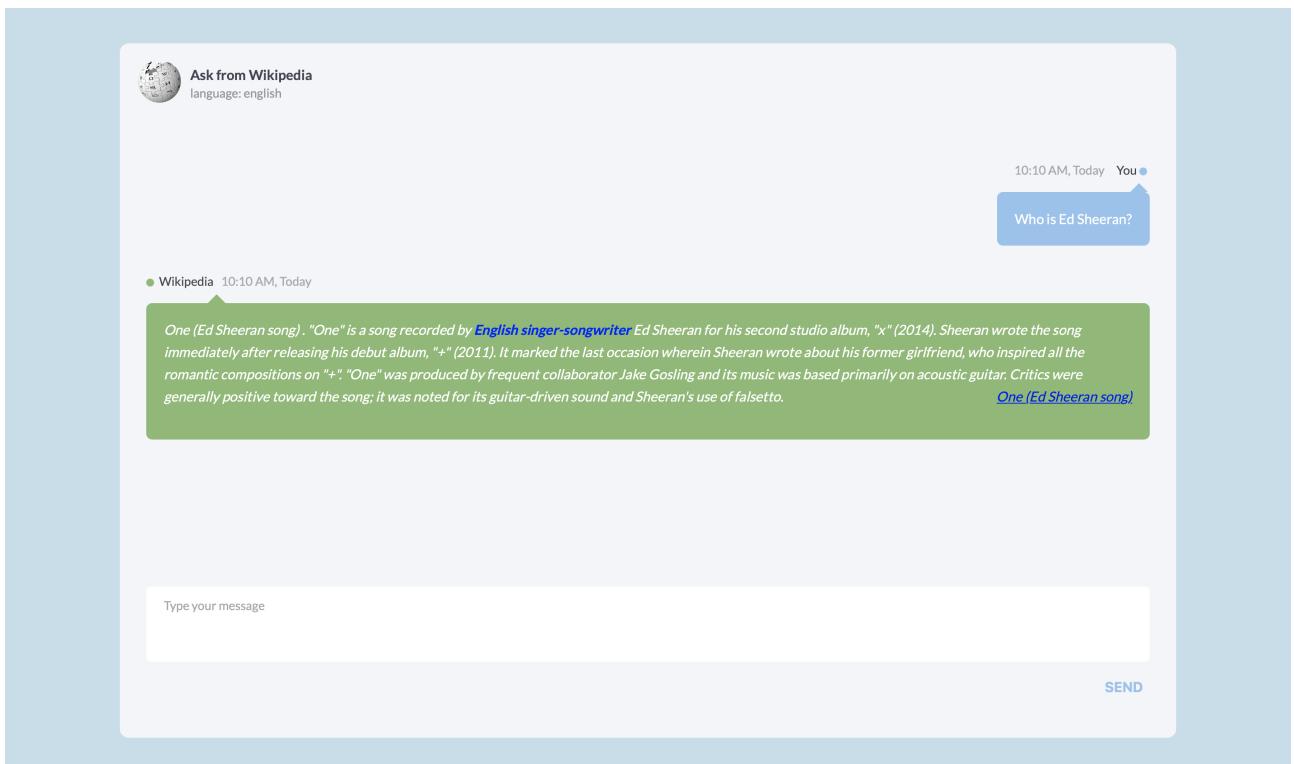
Ngoài ra, ở cuối mỗi đoạn phản hồi của hệ thống, hệ thống sẽ đưa ra link dẫn đến bài viết tương ứng với mỗi câu trả lời trên Wikipedia.

Các Hình 19, 20, và 21 dưới đây là một số câu hỏi ví dụ chúng em đã thực hiện.

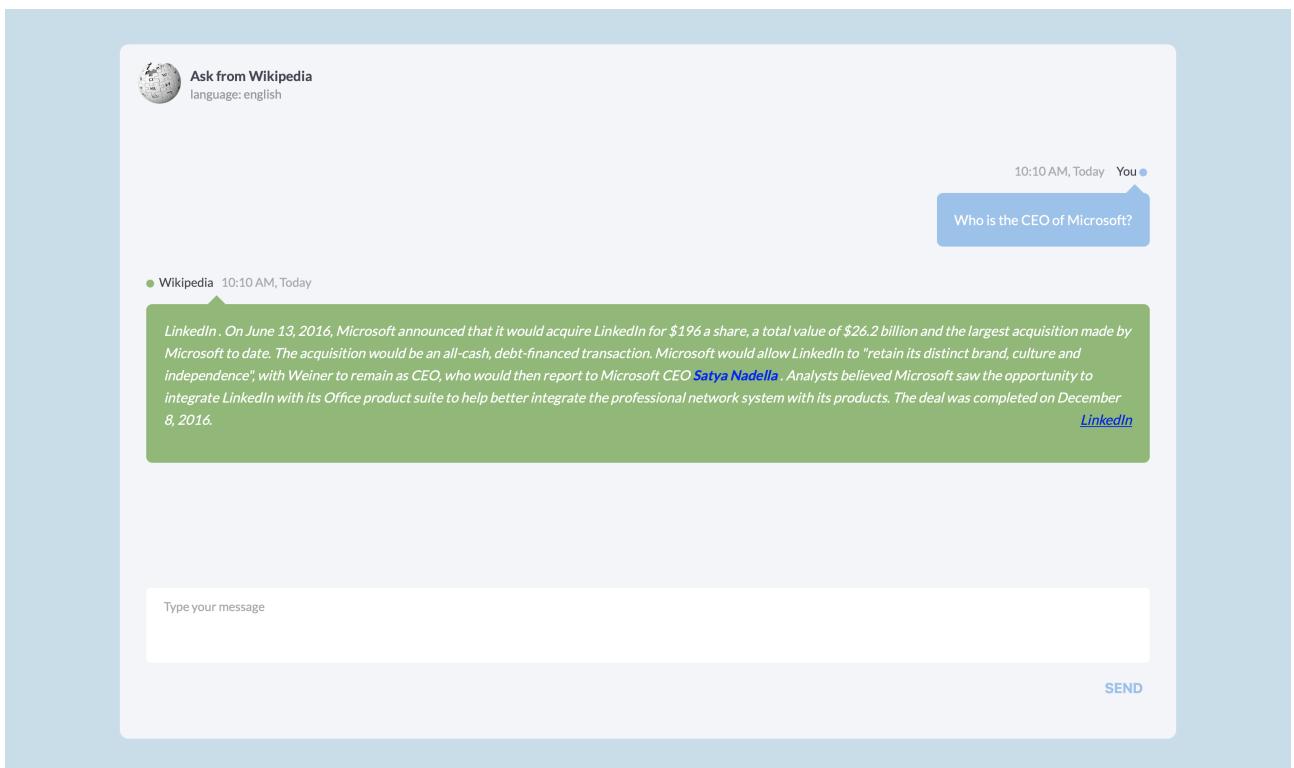
Có thể thấy câu trả lời đối với câu hỏi “Who is the incumbent US President?” ở Hình 19 là “Donald Trump”, đó là do nguồn dữ liệu chúng em thu thập được là dữ liệu trong năm 2018, nên chưa được cập nhật so với thời điểm hiện tại.



Hình 19: Câu hỏi về Tổng thống Hoa Kỳ đương nhiệm



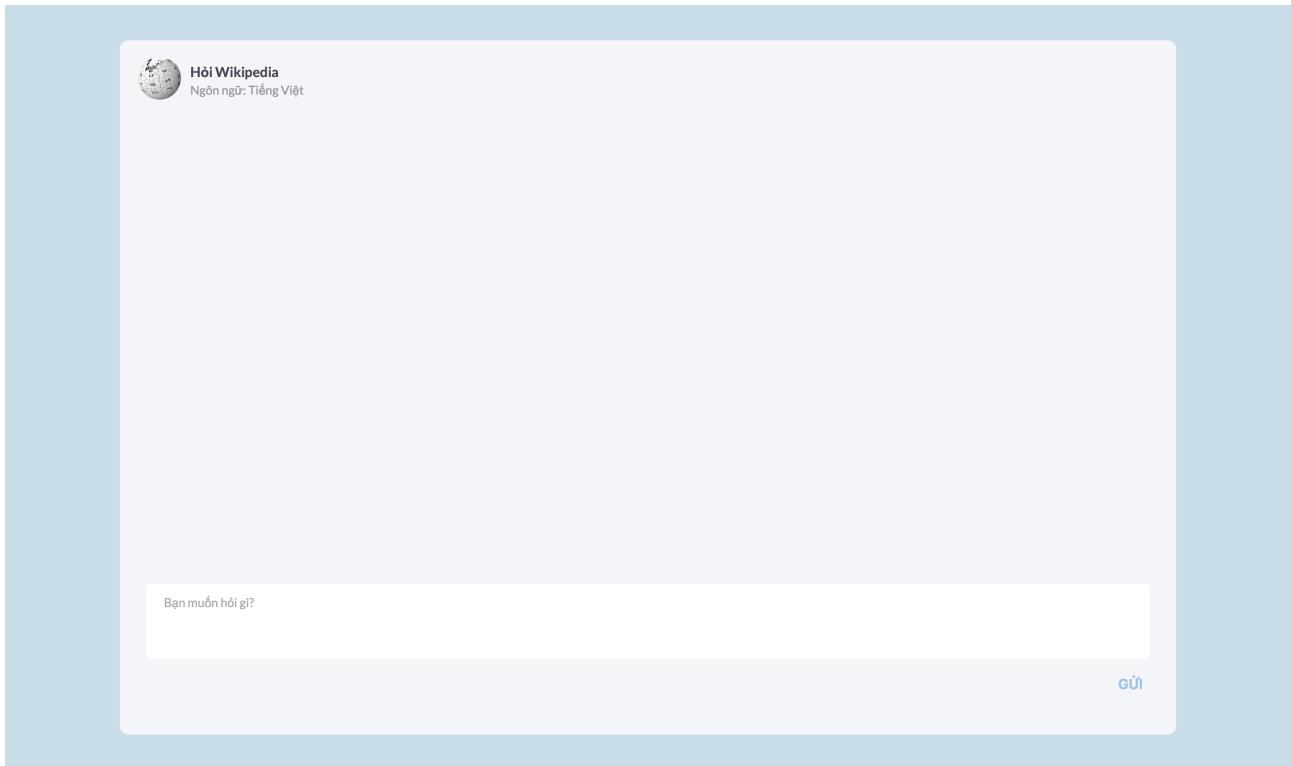
Hình 20: Câu hỏi về Ca sĩ/Nhạc sĩ Ed Sheeran



Hình 21: Câu hỏi về CEO của Tập đoàn Microsoft

#### 4.2.2 Giao diện tiếng Việt

Ngoài ra, chúng em còn thêm một giao diện tiếng Việt cho hệ thống, khi cần hỏi những câu hỏi bằng tiếng Việt. Giao diện tiếng Việt của hệ thống sau khi truy cập như Hình 22 dưới đây.

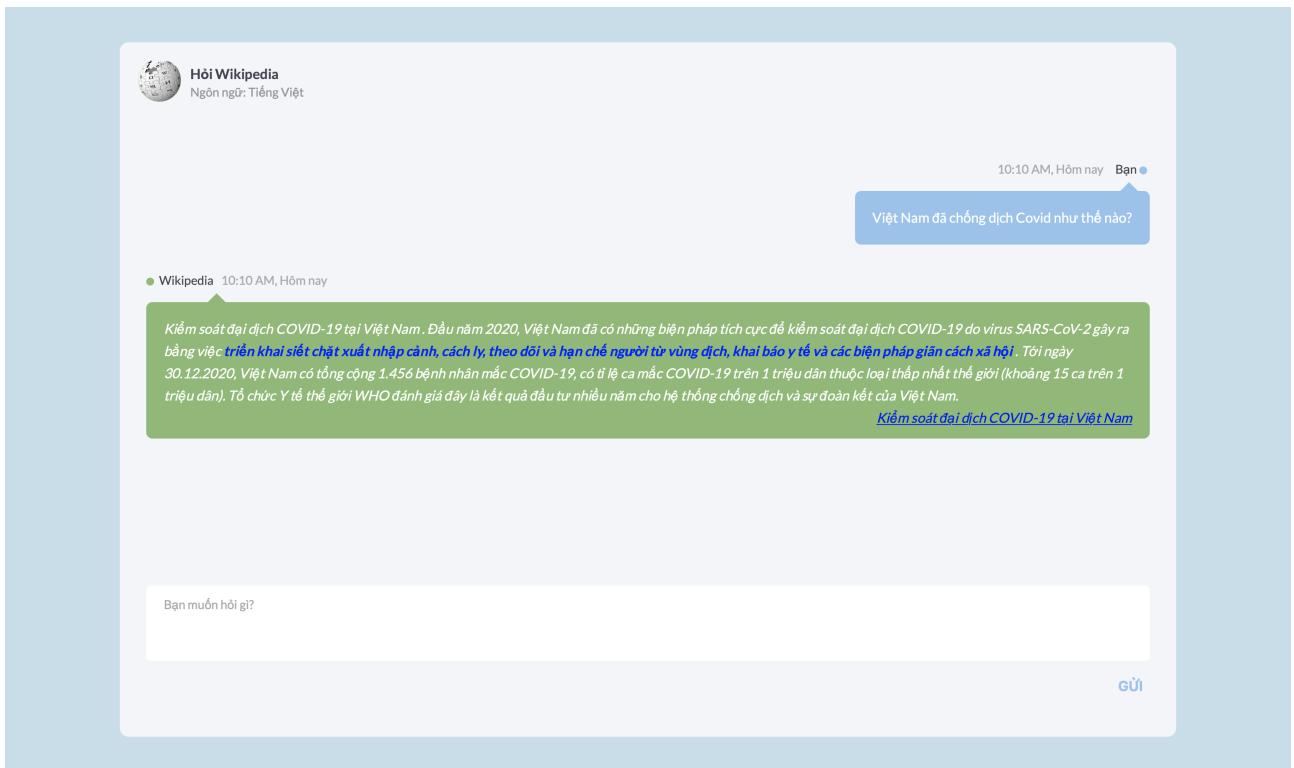


Hình 22: Giao diện tiếng Việt của hệ thống

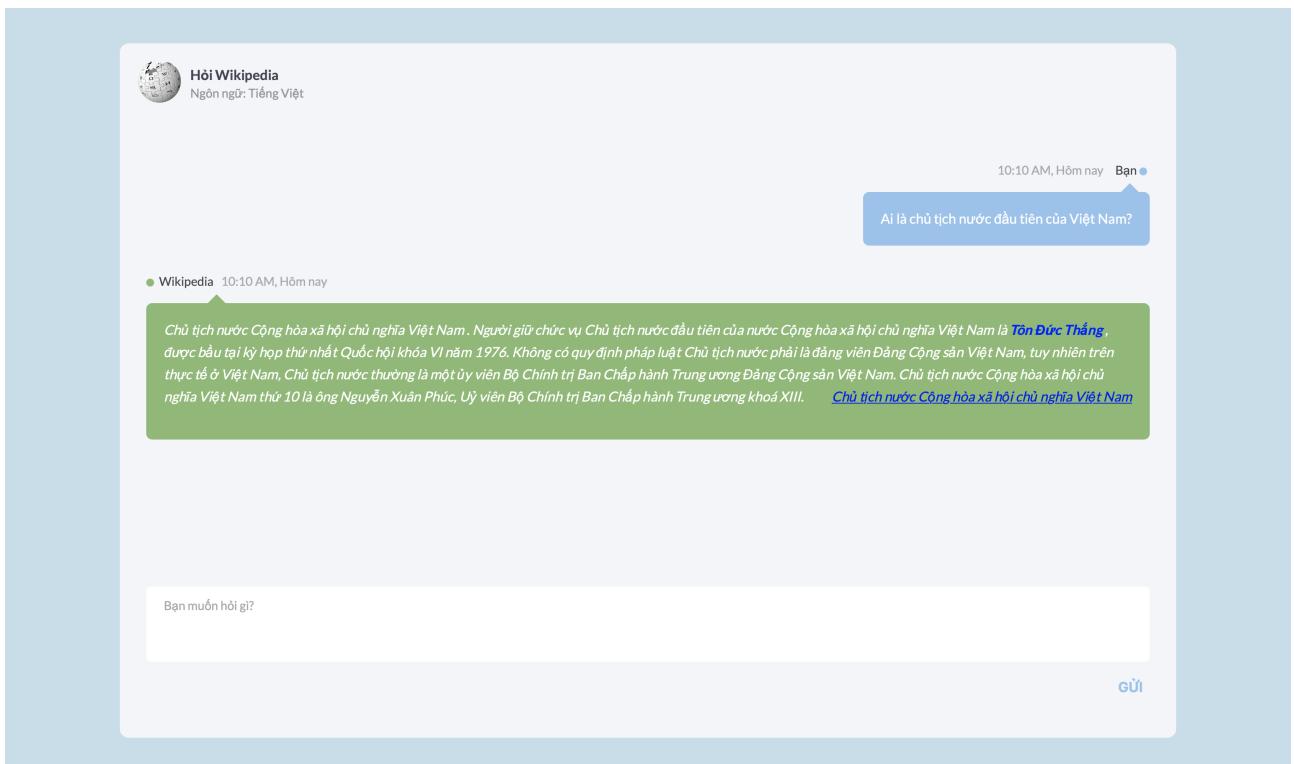
Theo đó, tương tự như giao diện tiếng Anh, hệ thống cũng sẽ cung cấp một khu vực để người dùng có thể điền câu hỏi vào. Sau khi được hỏi, hệ thống sẽ phản hồi lại đoạn văn chứa câu trả lời, với câu trả lời sẽ được đánh dấu **màu xanh** và **in đậm**.

Ngoài ra, ở cuối mỗi đoạn phản hồi của hệ thống, hệ thống sẽ đưa ra link dẫn đến bài viết tương ứng với mỗi câu trả lời trên Wikipedia.

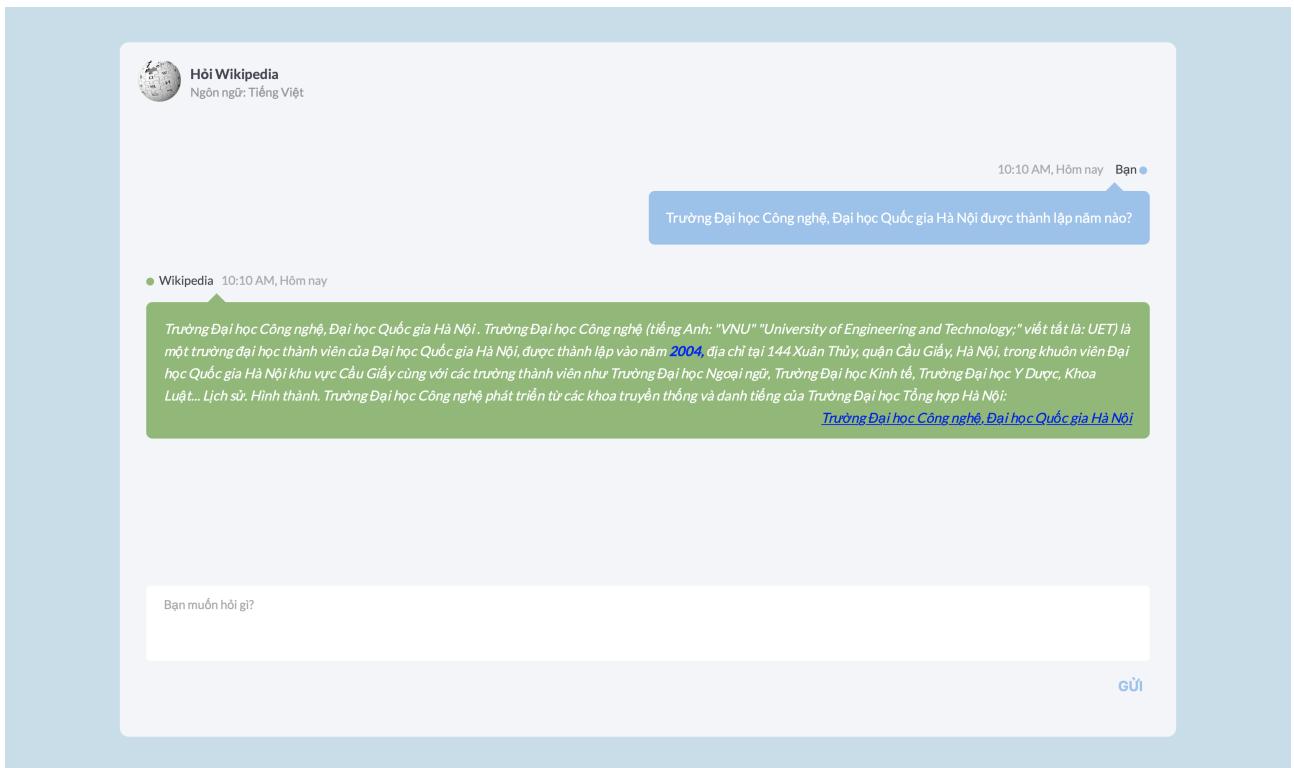
Các Hình 23, 24, 25, 26, 27, và 28 dưới đây là một số câu hỏi ví dụ chúng em đã thực hiện.



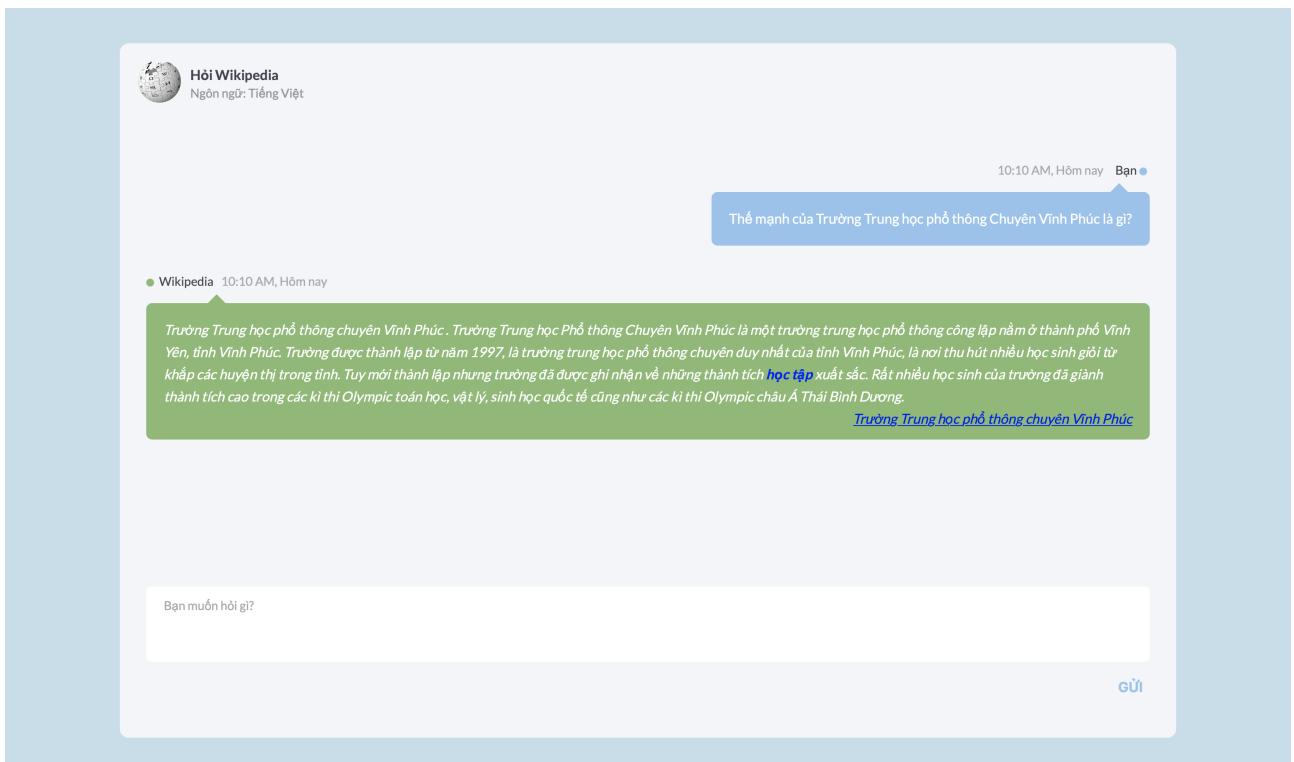
Hình 23: Câu hỏi về Cách thức Việt Nam chống dịch COVID-19



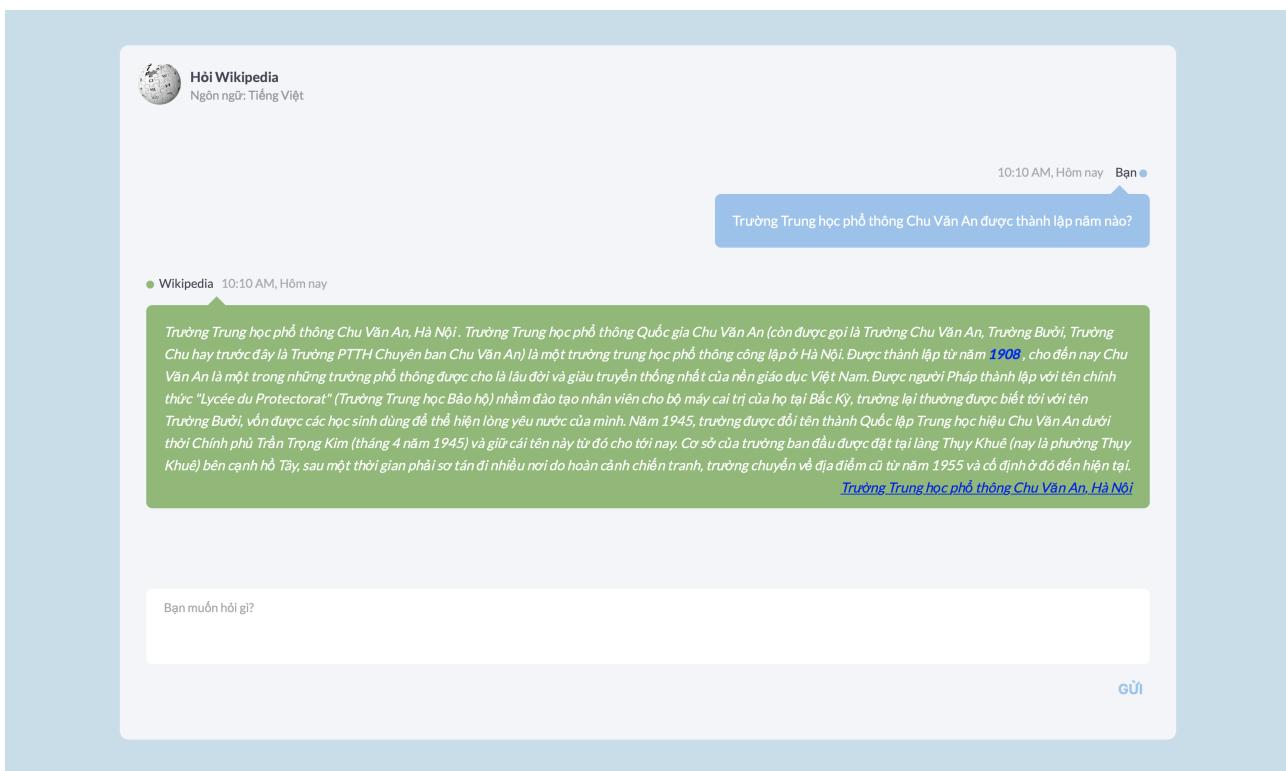
Hình 24: Câu hỏi về Chủ tịch nước đầu tiên của Việt Nam



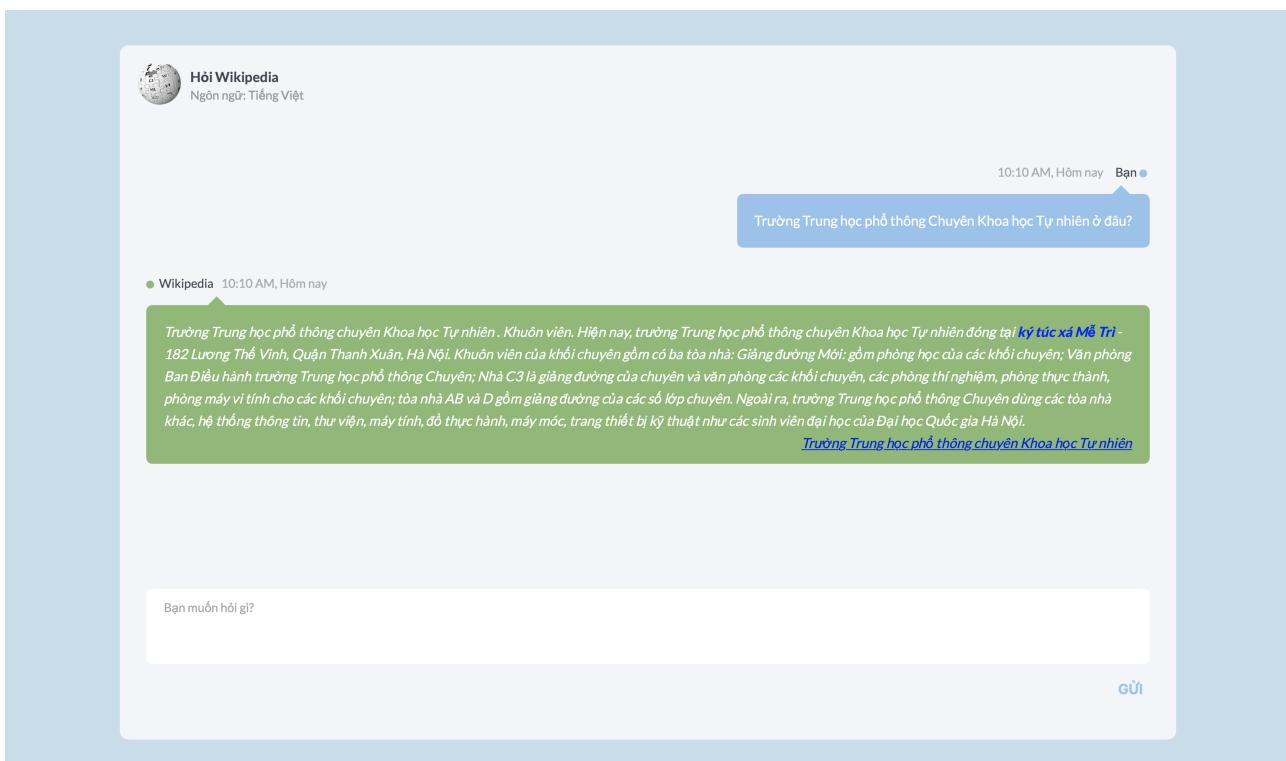
Hình 25: Câu hỏi về Trường Đại học Công nghệ, Đại học Quốc gia Hà Nội



Hình 26: Câu hỏi về Trường Trung học phổ thông Chuyên Vĩnh Phúc



Hình 27: Câu hỏi về Trường Trung học phổ thông Chu Văn An



Hình 28: Câu hỏi về Trường Trung học phổ thông Chuyên Khoa học tự nhiên

## Tài liệu tham khảo

- [1] D. Chen, A. Fisch, J. Weston, and A. Bordes, “Reading wikipedia to answer open-domain questions,” 2017.
- [2] P. Rajpurkar, J. Zhang, K. Lopyrev, and P. Liang, “Squad: 100, 000+ questions for machine comprehension of text,” *CoRR*, vol. abs/1606.05250, 2016. [Online]. Available: <http://arxiv.org/abs/1606.05250>
- [3] T. Kwiatkowski, J. Palomaki, O. Redfield, M. Collins, A. Parikh, C. Alberti, D. Epstein, I. Polosukhin, M. Kelcey, J. Devlin, K. Lee, K. N. Toutanova, L. Jones, M.-W. Chang, A. Dai, J. Uszkoreit, Q. Le, and S. Petrov, “Natural questions: a benchmark for question answering research,” *Transactions of the Association of Computational Linguistics*, 2019.
- [4] E. Choi, H. He, M. Iyyer, M. Yatskar, W.-t. Yih, Y. Choi, P. Liang, and L. Zettlemoyer, “QuAC: Question answering in context,” in *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*. Brussels, Belgium: Association for Computational Linguistics, Oct.-Nov. 2018, pp. 2174–2184. [Online]. Available: <https://www.aclweb.org/anthology/D18-1241>
- [5] A. Fan, Y. Jernite, E. Perez, D. Grangier, J. Weston, and M. Auli, “Eli5: Long form question answering,” 2019.
- [6] A. Trischler, T. Wang, X. Yuan, J. Harris, A. Sordoni, P. Bachman, and K. Suleiman, “Newsqa: A machine comprehension dataset,” 2017.
- [7] D. Chen and W.-t. Yih, “Open-domain question answering,” in *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics: Tutorial Abstracts*. Online: Association for Computational Linguistics, Jul. 2020, pp. 34–37. [Online]. Available: <https://www.aclweb.org/anthology/2020.acl-tutorials.8>
- [8] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, “Bert: Pre-training of deep bidirectional transformers for language understanding,” 2018.
- [9] P. Lewis, E. Perez, A. Piktus, F. Petroni, V. Karpukhin, N. Goyal, H. Küttler, M. Lewis, W. tau Yih, T. Rocktäschel, S. Riedel, and D. Kiela, “Retrieval-augmented generation for knowledge-intensive nlp tasks,” 2021.
- [10] A. Roberts, C. Raffel, and N. Shazeer, “How much knowledge can you pack into the parameters of a language model?” 2020.
- [11] T. B. Brown, B. Mann, N. Ryder, M. Subbiah, J. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell, S. Agarwal, A. Herbert-Voss, G. Krueger, T. Henighan, R. Child, A. Ramesh, D. M. Ziegler, J. Wu, C. Winter, C. Hesse, M. Chen, E. Sigler, M. Litwin, S. Gray, B. Chess, J. Clark, C. Berner, S. McCandlish, A. Radford, I. Sutskever, and D. Amodei, “Language models are few-shot learners,” 2020.
- [12] J. Lin, X. Ma, S.-C. Lin, J.-H. Yang, R. Pradeep, and R. Nogueira, “Pyserini: An easy-to-use python toolkit to support replicable ir research with sparse and dense representations,” 2021.
- [13] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, “Attention is all you need,” 2017.

- [14] J. Alammar, Jun 2018. [Online]. Available: <http://jalammar.github.io/illustrated-transformer/>
- [15] Y. Wu, M. Schuster, Z. Chen, Q. V. Le, M. Norouzi, W. Macherey, M. Krikun, Y. Cao, Q. Gao, K. Macherey, J. Klingner, A. Shah, M. Johnson, X. Liu, Lukasz Kaiser, S. Gouws, Y. Kato, T. Kudo, H. Kazawa, K. Stevens, G. Kurian, N. Patil, W. Wang, C. Young, J. Smith, J. Riesa, A. Rudnick, O. Vinyals, G. Corrado, M. Hughes, and J. Dean, “Google’s neural machine translation system: Bridging the gap between human and machine translation,” 2016.
- [16] R. Sennrich, B. Haddow, and A. Birch, “Neural machine translation of rare words with subword units,” 2016.
- [17] W. Yang, Y. Xie, A. Lin, X. Li, L. Tan, K. Xiong, M. Li, and J. Lin, “End-to-end open-domain question answering with,” *Proceedings of the 2019 Conference of the North*, 2019. [Online]. Available: <http://dx.doi.org/10.18653/v1/N19-4013>
- [18] Z. Wang, P. Ng, X. Ma, R. Nallapati, and B. Xiang, “Multi-passage bert: A globally normalized bert model for open-domain question answering,” 2019.
- [19] A. Asai, K. Hashimoto, H. Hajishirzi, R. Socher, and C. Xiong, “Learning to retrieve reasoning paths over wikipedia graph for question answering,” in *International Conference on Learning Representations*, 2020.
- [20] V. Sanh, L. Debut, J. Chaumond, and T. Wolf, “Distilbert, a distilled version of bert: smaller, faster, cheaper and lighter,” 2020.
- [21] Z. Sun, H. Yu, X. Song, R. Liu, Y. Yang, and D. Zhou, “Mobilebert: a compact task-agnostic bert for resource-limited devices,” 2020.
- [22] Z. Lan, M. Chen, S. Goodman, K. Gimpel, P. Sharma, and R. Soricut, “Albert: A lite bert for self-supervised learning of language representations,” 2020.
- [23] T. Wolf, L. Debut, V. Sanh, J. Chaumond, C. Delangue, A. Moi, P. Cistac, T. Rault, R. Louf, M. Funtowicz, J. Davison, S. Shleifer, P. von Platen, C. Ma, Y. Jernite, J. Plu, C. Xu, T. L. Scao, S. Gugger, M. Drame, Q. Lhoest, and A. M. Rush, “Huggingface’s transformers: State-of-the-art natural language processing,” 2020.
- [24] N. Nguyen, “Chatbot application for answering questions related to rules and regulations of university of information technology,” *Thesis, University of Information Technology, Ho Chi Minh City*, 2018.
- [25] M. Artetxe, S. Ruder, and D. Yogatama, “On the cross-lingual transferability of monolingual representations,” *CoRR*, vol. abs/1910.11856, 2019.
- [26] P. Lewis, B. Oğuz, R. Rinott, S. Riedel, and H. Schwenk, “Mlqa: Evaluating cross-lingual extractive question answering,” *arXiv preprint arXiv:1910.07475*, 2019.
- [27] K. Nguyen, V. Nguyen, A. Nguyen, and N. Nguyen, “A Vietnamese dataset for evaluating machine reading comprehension,” in *Proceedings of the 28th International Conference on Computational Linguistics*. Barcelona, Spain (Online): International Committee on Computational Linguistics, Dec. 2020, pp. 2595–2605. [Online]. Available: <https://www.aclweb.org/anthology/2020.coling-main.233>

- [28] G. Attardi, “Wikiextractor,” <https://github.com/attardi/wikiextractor>, 2015.
- [29] D. Q. Nguyen and A. T. Nguyen, “PhoBERT: Pre-trained language models for Vietnamese,” in *Findings of the Association for Computational Linguistics: EMNLP 2020*, 2020, pp. 1037–1042.
- [30] T. Vu, D. Q. Nguyen, D. Q. Nguyen, M. Dras, and M. Johnson, “VnCoreNLP: A Vietnamese natural language processing toolkit,” in *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Demonstrations*. New Orleans, Louisiana: Association for Computational Linguistics, Jun. 2018, pp. 56–60. [Online]. Available: <https://www.aclweb.org/anthology/N18-5012>
- [31] A. Conneau, K. Khandelwal, N. Goyal, V. Chaudhary, G. Wenzek, F. Guzmán, E. Grave, M. Ott, L. Zettlemoyer, and V. Stoyanov, “Unsupervised cross-lingual representation learning at scale,” 2020.
- [32] G. Wenzek, M.-A. Lachaux, A. Conneau, V. Chaudhary, F. Guzmán, A. Joulin, and E. Grave, “Ccnet: Extracting high quality monolingual datasets from web crawl data,” 2019.
- [33] P. N.-T. Do, N. D. Nguyen, T. V. Huynh, K. V. Nguyen, A. G.-T. Nguyen, and N. L.-T. Nguyen, “Sentence extraction-based machine reading comprehension for vietnamese,” 2021.
- [34] P. S. H. Lewis, P. Stenetorp, and S. Riedel, “Question and answer test-train overlap in open-domain question answering datasets,” *CoRR*, vol. abs/2008.02637, 2020. [Online]. Available: <https://arxiv.org/abs/2008.02637>
- [35] P. Lewis, P. Stenetorp, and S. Riedel, “Question and answer test-train overlap in open-domain question answering datasets,” 2020.