ĐẠI HỌC QUỐC GIA TP HCM
TRƯỜNG ĐẠI HỌC KHOA HỌC TỰ NHIÊN
KHOA CÔNG NGHỆ THÔNG TIN

ĐỒ ÁN 2

KHAI THÁC DỮ LIỆU VĂN BẢN & ỨNG DỤNG

Giảng viên hướng dẫn :
Lê Thanh Tùng
Nguyễn Trần Duy Minh

Nhóm sinh viên :

- Hoàng Quang Minh- MSSV : 19120292
- Dư Thanh Huy - MSSV : 20120102
- Phạm Nguyễn Khánh Minh - MSSV : 20120117
- Trần Trọng Tấn - MSSV : 20120185

Thành phố Hồ Chí Minh, ngày 15 tháng 6 năm 2024

MỤC LỤC

I. Thông tin chung

1.Thông tin nhóm:

STT	Họ và tên	MSSV
1	Hoàng Quang Minh	19120292
2	Dư Thanh Huy	20120102
3	Phạm Nguyễn Khánh Minh	20120117
4	Trần Trọng Tấn	20120185

2. Bảng phân công công việc và đánh giá mức độ hoàn thành

2.1.Phân công công việc :

STT	Sinh viên thực hiện	MSSV	Mô tả công việc	Phần trăm đóng góp
1	Hoàng Quang Minh	1910292	Phần 2.1 + Viết báo cáo	100%
2	Dư Thanh Huy	20120102	Phần 2.1 + Web giao diện (Front end + Back end) cho Phần 2.2 + Viết báo cáo	100%
3	Phạm Nguyễn Khánh Minh	20120117	Phần 2.1 + Viết báo cáo	100%
4	Trần Trọng Tấn	20120185	Phần 2.2 (Tiền xử lí dữ liệu, train model, đánh giá,…) + Viết báo cáo	100%

2.2.Mức độ hoàn thành :

STT	Yêu cầu	Mức độ hoàn thành
1	**2.1.EDA:** Thực hiện việc phân tích và thống kê về dữ liệu. Một số phân tích : Thống kê độ dài của câu hỏi, độ dài của đoạn văn, phân bố của label,....	100%
2	**2.2.Thử nghiệm trên mô hình có sẵn** Huấn luyện cùng đánh giá hiệu năng mô hình • Sử dụng mô hình "deepset/xlm-roberta-base-squad2" • Tiến hành huấn luyện lại từ dataset đã cho • Đánh giá hiệu năng mô hình • Giao diện Web (Reactjs + Flask) cho mô hình đã huấn luyện	100%

II.Nội dung chính:

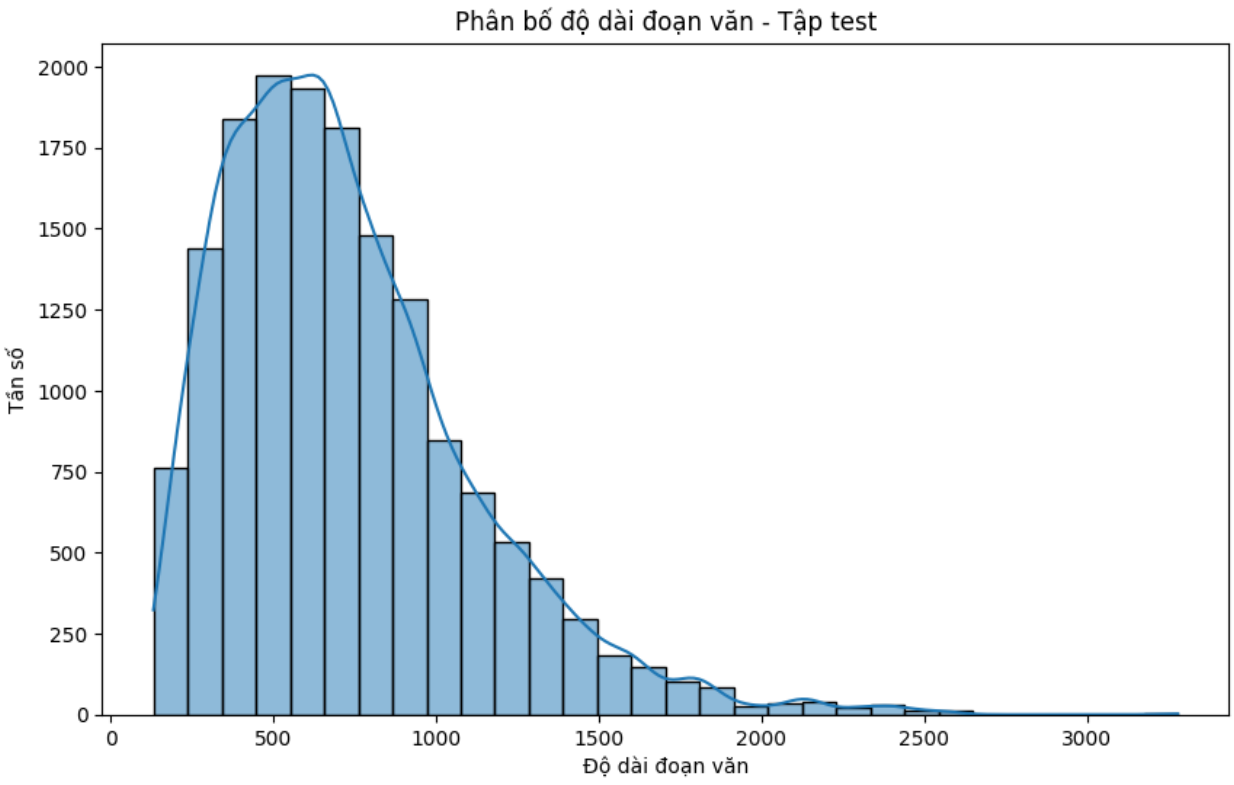
1. EDA (Exploratory Data Analysis):

- Thống kê cơ bản:

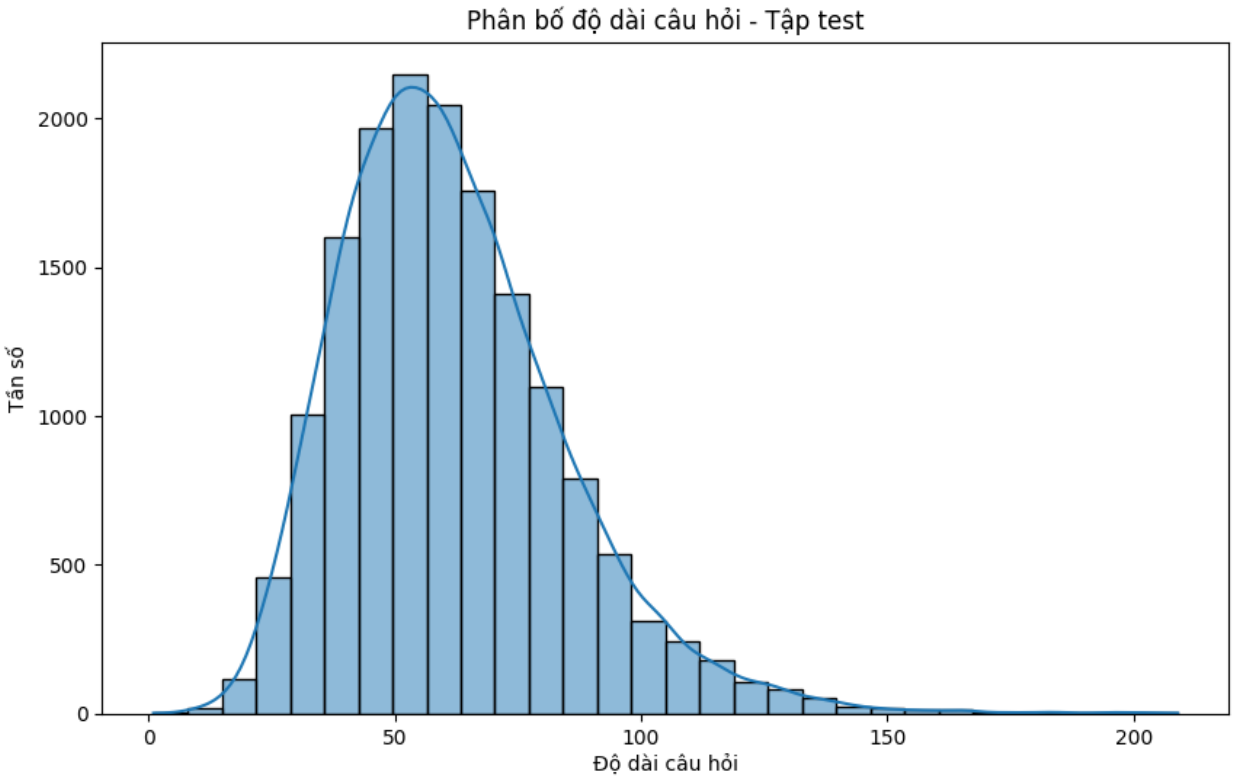
```
Số lượng câu hỏi trong tập train: 50046
Số lượng câu hỏi trong tập test: 15994
Số lượng câu hỏi "is_impossible" trong tập train: 0
Độ dài trung bình của đoạn văn: 792.469867721696
Độ dài trung bình của câu hỏi: 61.94646924829157
Độ dài trung bình của câu trả lời: 20.32030531910642
Số lượng từ trung bình trong câu hỏi: 13.718738760340488
Số lượng từ trung bình trong đoạn văn: 169.6543579906486
Số lượng từ trung bình trong câu trả lời: 4.3172880949526435
```

- Tập test:

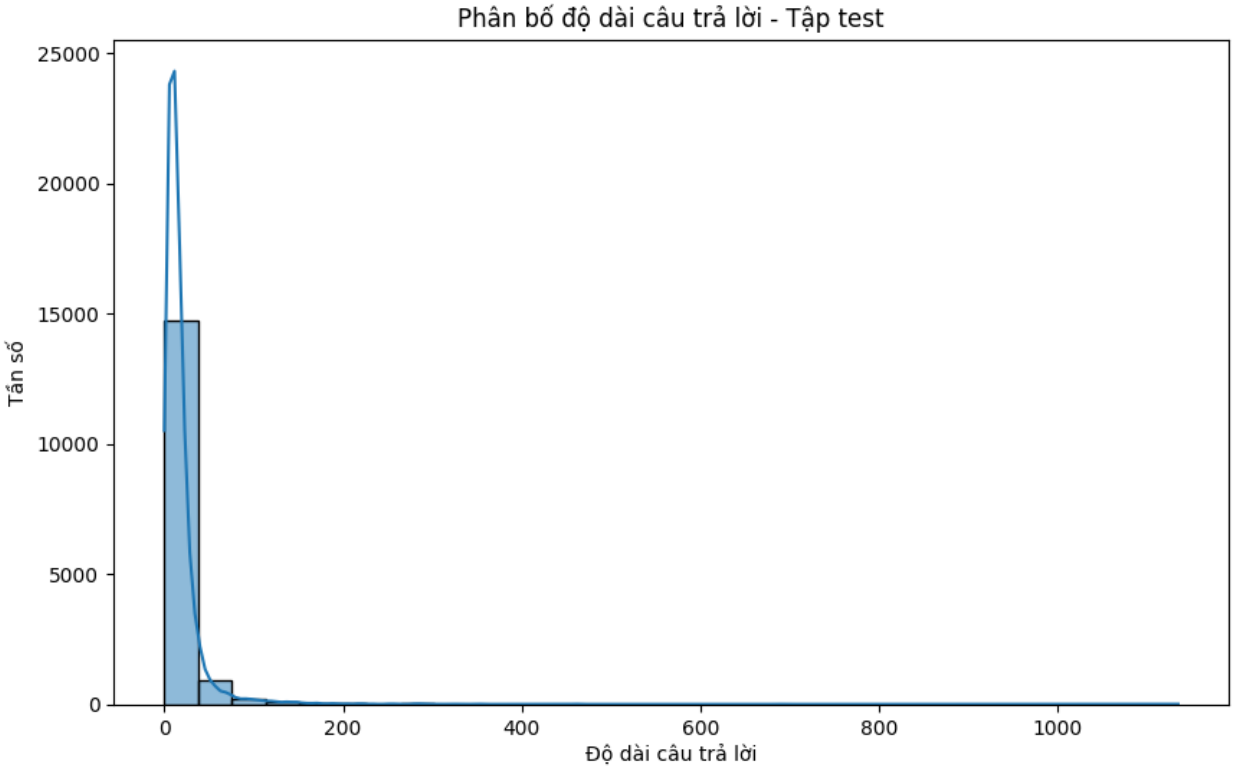
+ Phân bố độ dài đoạn văn:



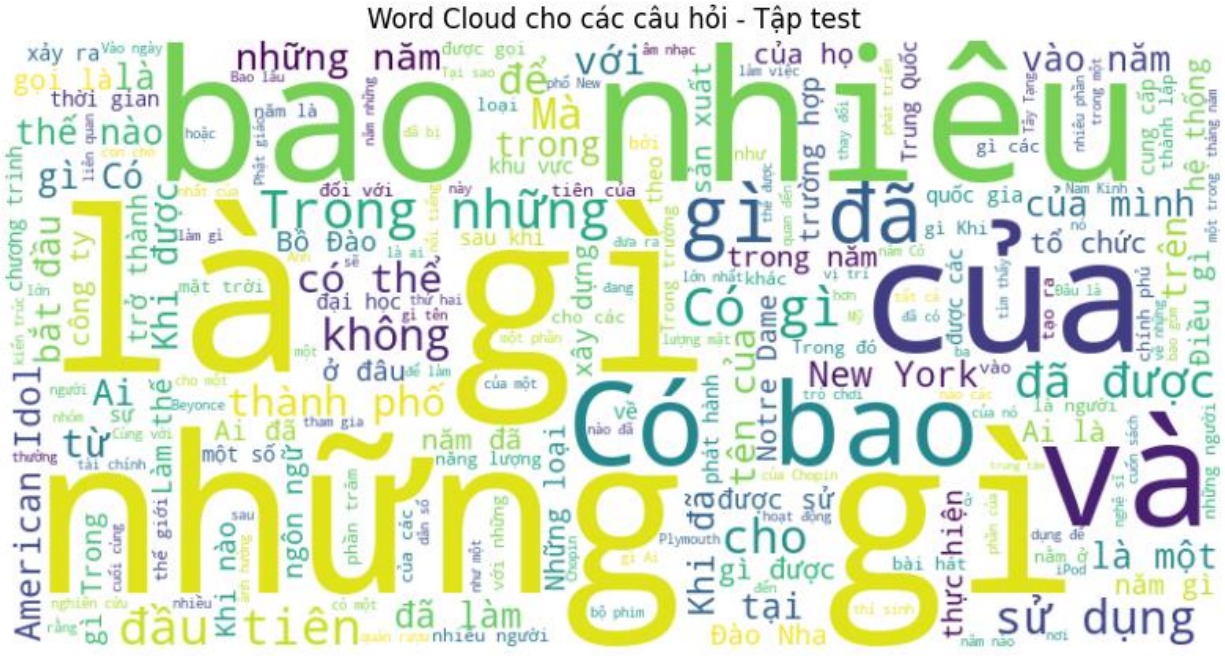
+ Phân bố độ dài câu hỏi:



+ Phân bố độ dài câu trả lời:



+ Word cloud cho các câu hỏi



Nhận xét:

- Phân bố độ dài đoạn văn:

Đa số các đoạn văn trong tập test có độ dài khoảng 300 đến 1000 ký tự, với đỉnh phân bố khoảng 500 ký tự. Một số đoạn văn rất dài (trên 2000 ký tự) là thiểu số.

- Phân bố độ dài câu hỏi:

Câu hỏi trong tập test chủ yếu có độ dài từ 30 đến 80 ký tự, với đỉnh phân bố khoảng 50 ký tự. Một số câu hỏi rất ngắn hoặc rất dài là rất ít.

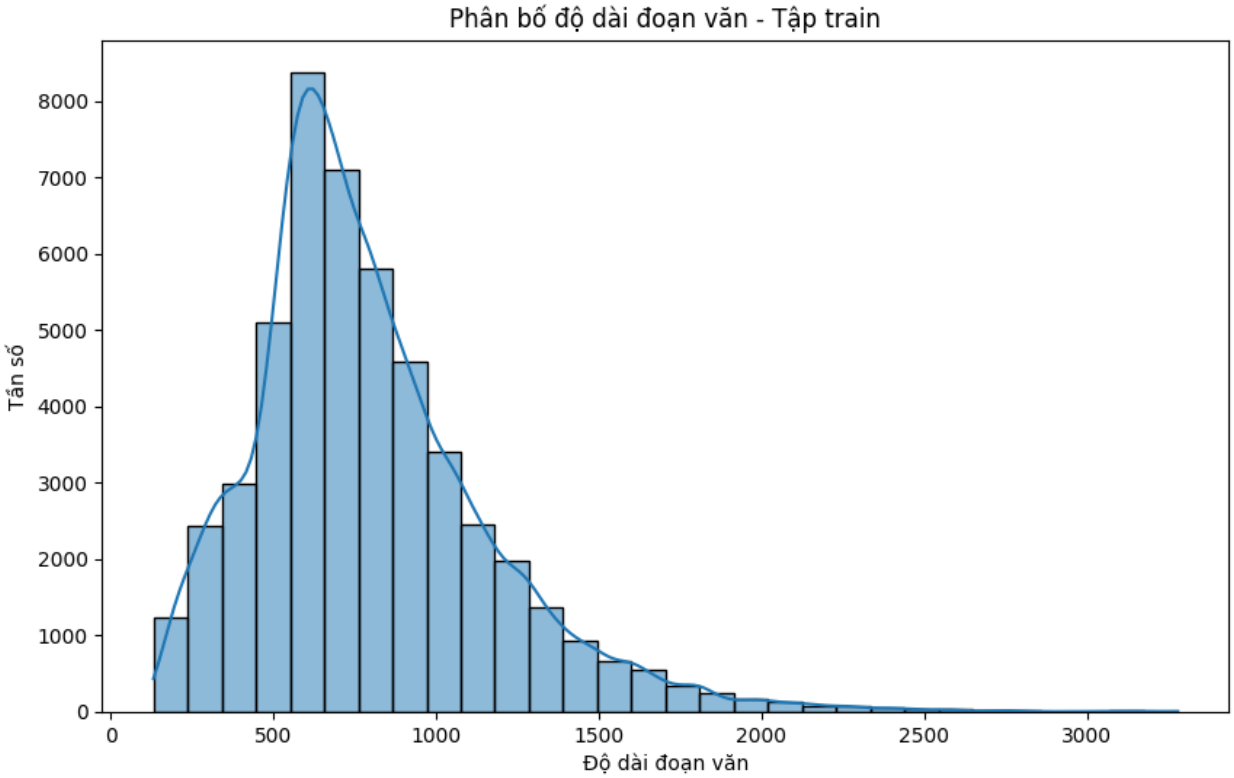
- Phân bố độ dài câu trả lời:

Phần lớn các câu trả lời trong tập test có độ dài dưới 50 ký tự, với đỉnh phân bố dưới 20 ký tự. Số lượng câu trả lời dài hơn 100 ký tự là rất ít.

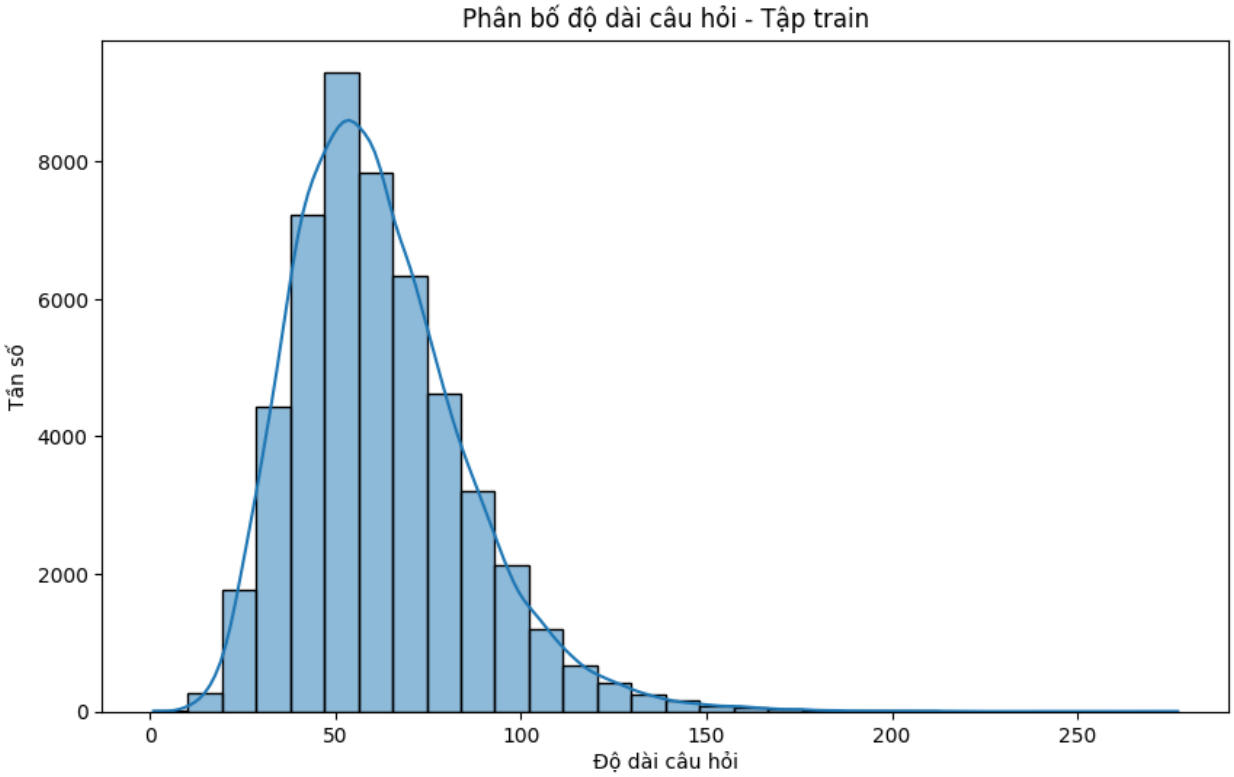
Dữ liệu trong tập test chủ yếu bao gồm các đoạn văn có độ dài vừa phải, câu hỏi và câu trả lời có độ dài ngắn. Mô hình cần tập trung vào việc trích xuất các câu trả lời ngắn và chính xác từ các đoạn văn và câu hỏi có độ dài phù hợp.

- *Tập train:*

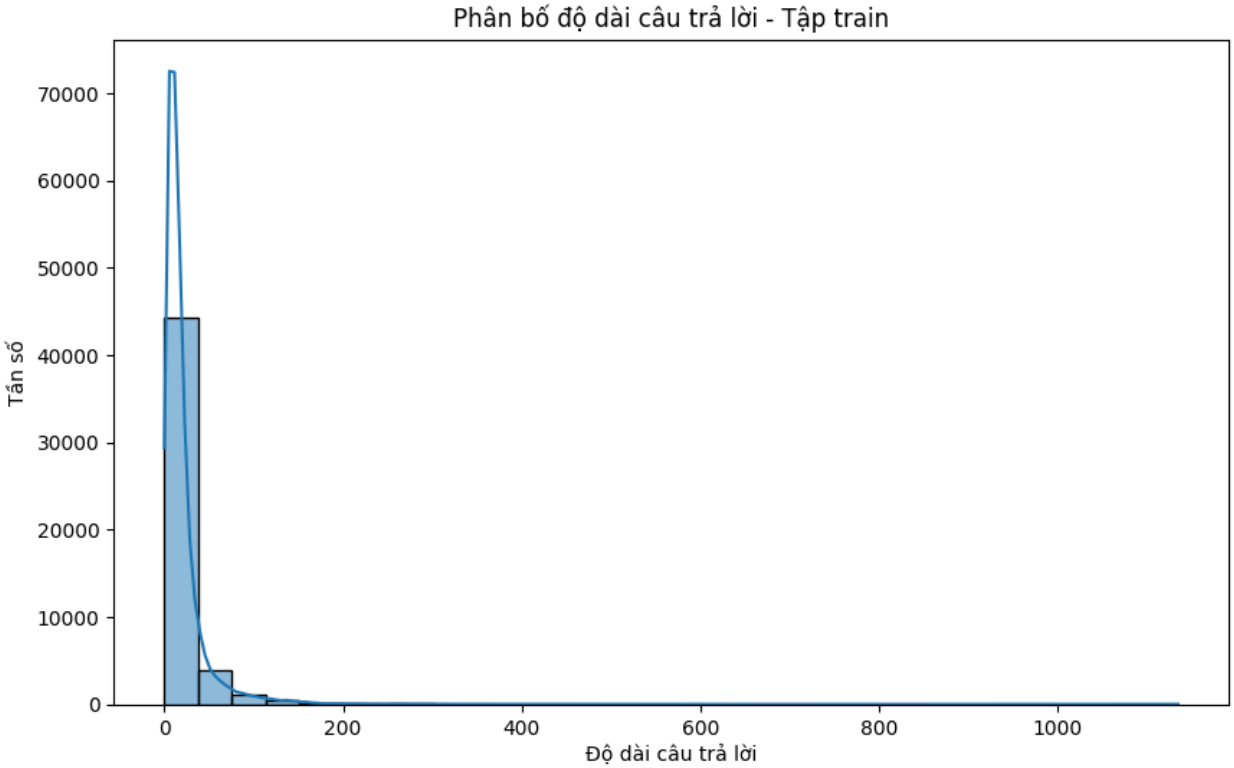
+ Phân bố độ dài đoạn văn:



+ Phân bố độ dài câu hỏi:



+ Phân bố độ dài câu trả lời:



*Nhận xét:

• Độ dài đoạn văn:

Phần lớn các đoạn văn có độ dài vừa phải (300-1000 ký tự) với đỉnh phân bố khoảng 500 ký tự. Điều này phù hợp cho việc huấn luyện mô hình hỏi đáp, giúp mô hình học và xử lý tốt các đoạn văn phổ biến.

• Độ dài câu hỏi:

Câu hỏi chủ yếu có độ dài từ 30 đến 80 ký tự với đỉnh phân bố khoảng 50 ký tự. Điều này cho thấy câu hỏi có độ dài vừa phải, không quá ngắn cũng không quá dài, giúp mô hình dễ dàng nắm bắt ngữ nghĩa và trích xuất câu trả lời.

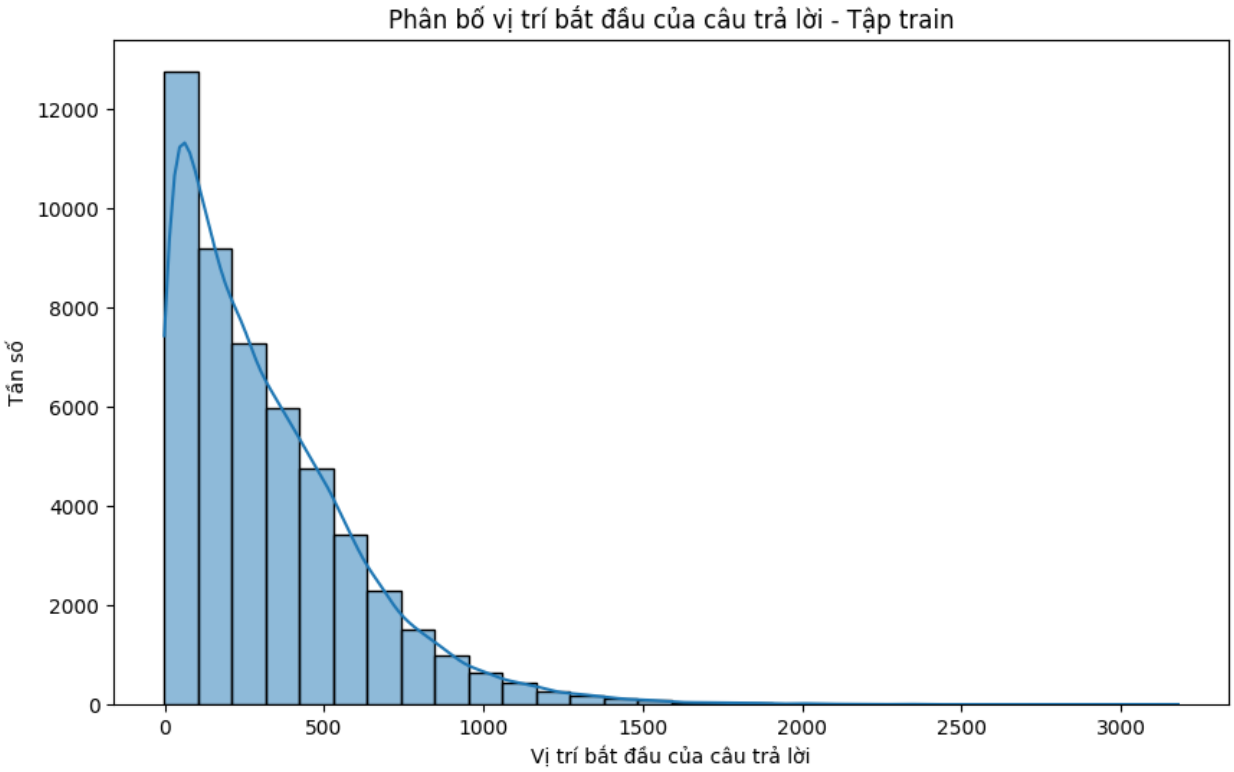
• Độ dài câu trả lời:

Câu trả lời chủ yếu rất ngắn, dưới 50 ký tự với đỉnh phân bố dưới 20 ký tự. Điều này cho thấy các câu trả lời trong tập dữ liệu thường rất ngắn gọn.

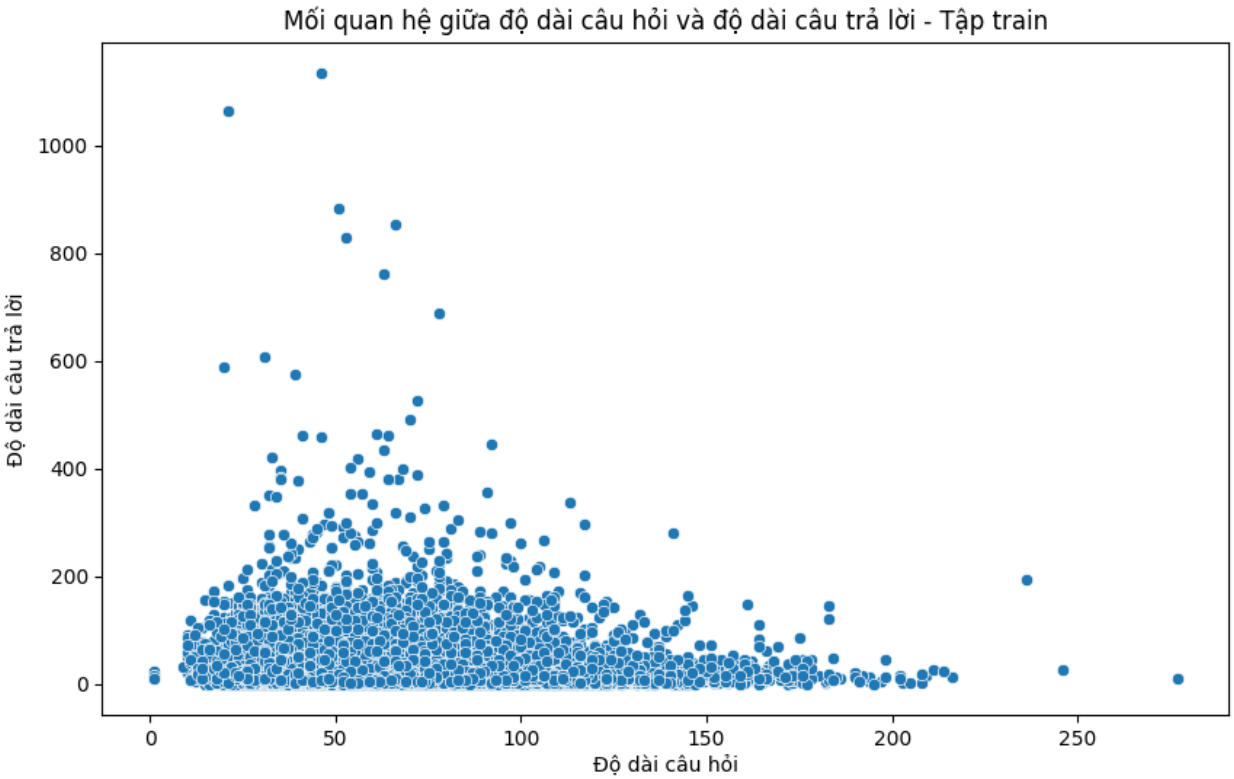
Nhận xét chung:

Dữ liệu trong cả tập train và tập test đều có sự tương đồng về phân bố độ dài đoạn văn, câu hỏi và câu trả lời. Điều này đảm bảo mô hình học được cách xử lý các đoạn văn, câu hỏi và câu trả lời có độ dài phù hợp.

+ Phân bố vị trí bắt đầu của câu trả lời:

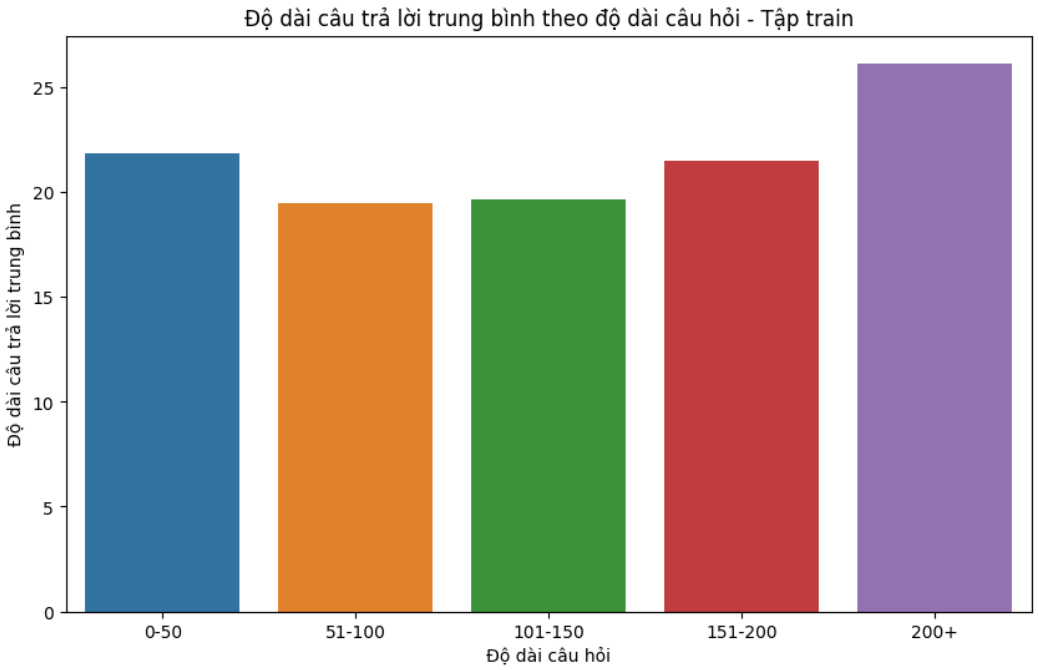


+ Khảo sát mối quan hệ giữa câu hỏi và câu trả lời:

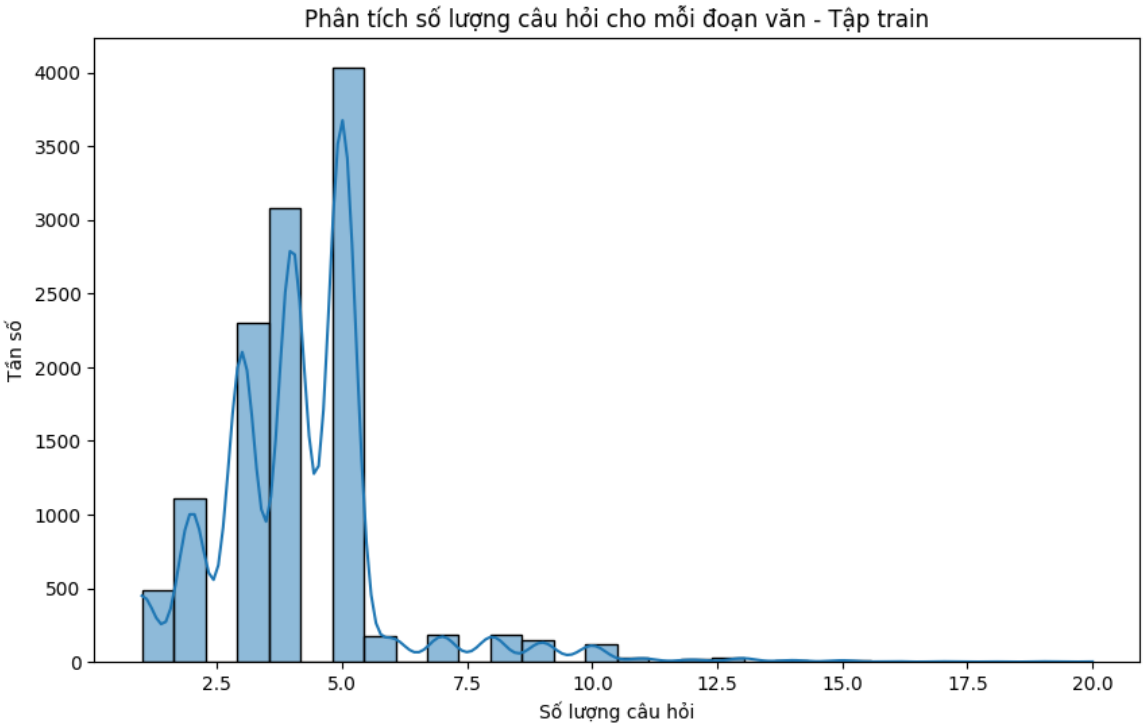


Nhận xét: Từ biểu đồ phân tán này, ta thấy:
• Phần lớn câu hỏi và câu trả lời đều có độ dài ngắn: Biểu đồ cho thấy phần lớn câu hỏi và câu trả lời đều tập trung vào các độ dài ngắn, phù hợp với các nhận xét trước đó về độ dài câu hỏi và câu trả lời.
• Độ dài câu trả lời không phụ thuộc nhiều vào độ dài câu hỏi: Không có mối quan hệ rõ ràng giữa độ dài câu hỏi và độ dài câu trả lời, điều này cho thấy các câu trả lời có thể có độ dài biến động lớn bất kể độ dài của câu hỏi.

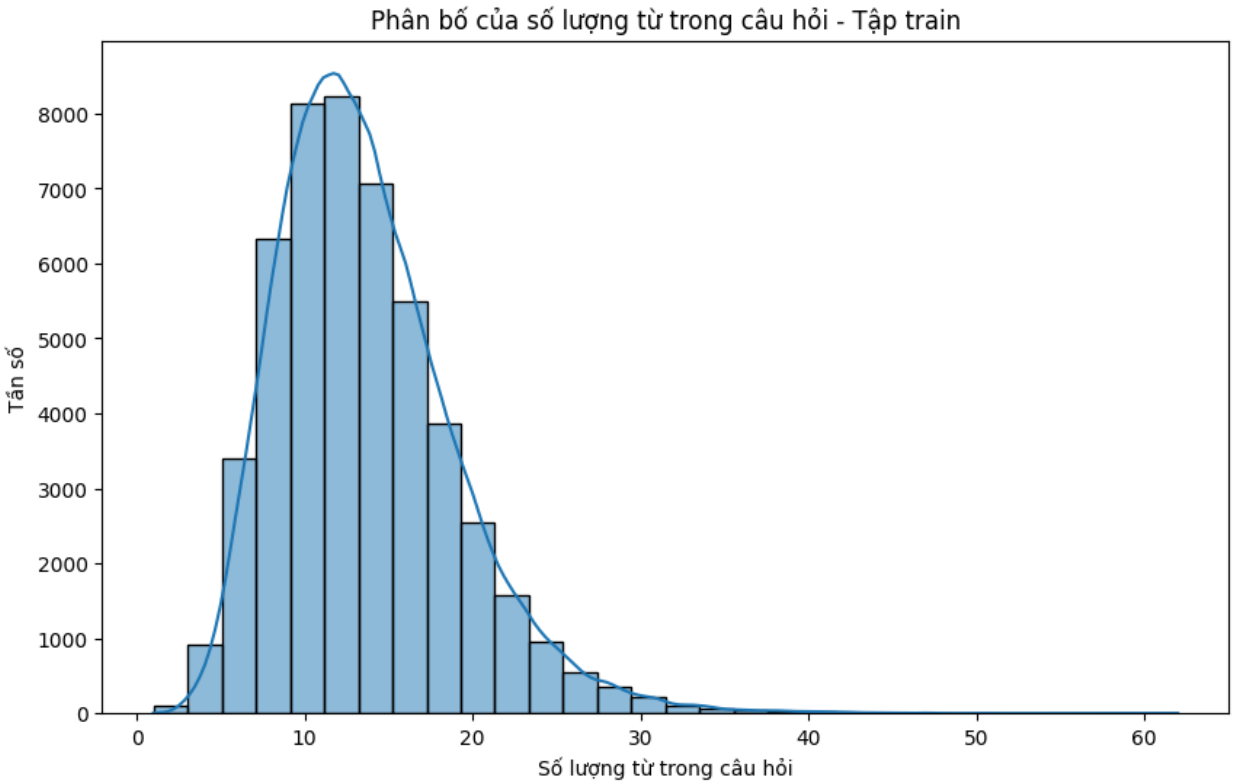
+ Độ dài câu trả lời trung bình theo độ dài câu hỏi:



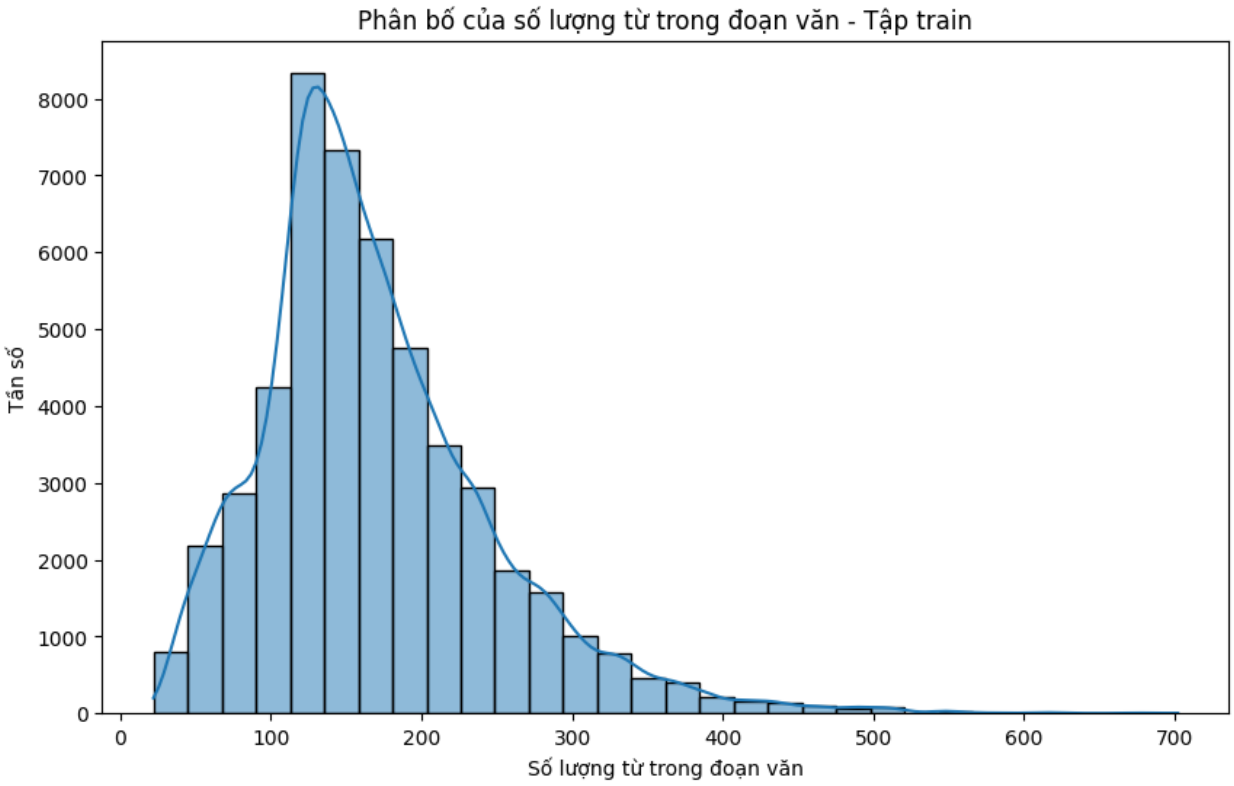
+ Phân tích số lượng câu hỏi cho mỗi đoạn văn:



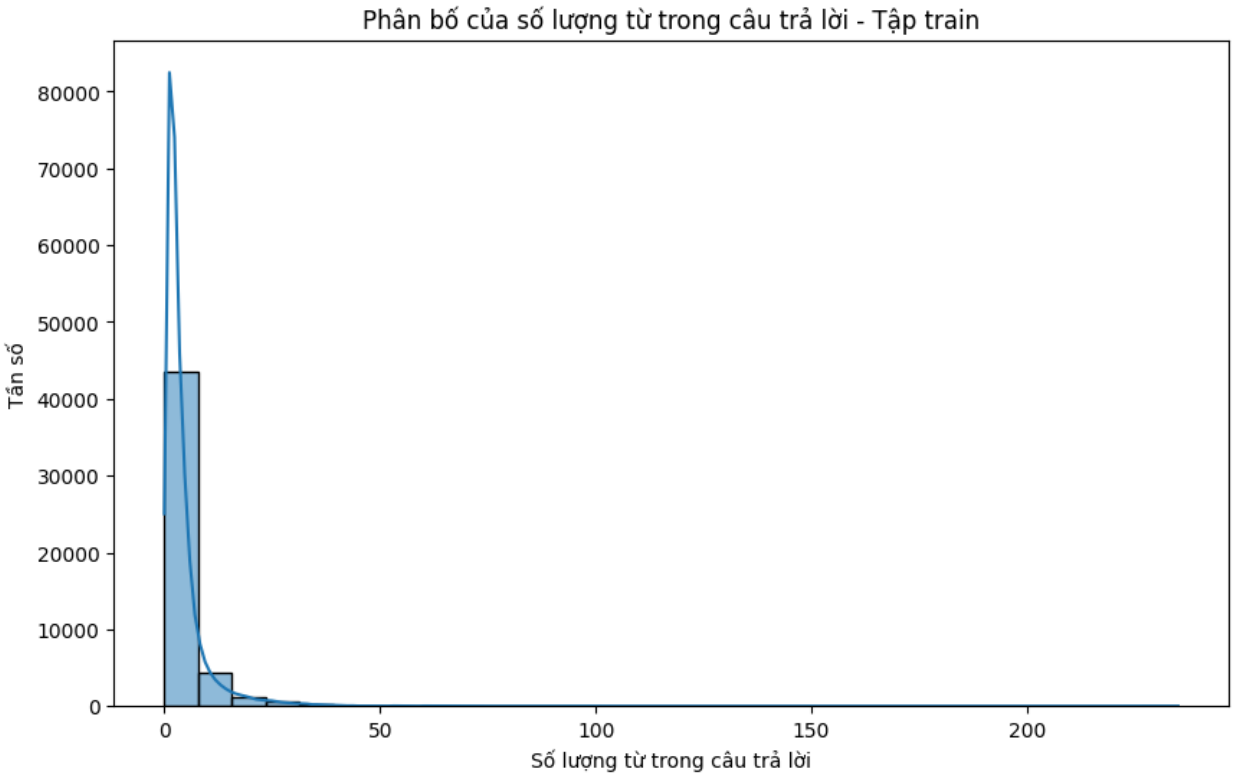
+ Phân tích sự phân bố của số lượng từ trong câu hỏi:



+ Phân tích sự phân bố của số lượng từ trong đoạn văn:



+ Phân tích sự phân bố của số lượng từ trong trả lời:



+ Word cloud cho các câu hỏi

- Các thông tin thống kê như số lượng (count), giá trị trung bình (mean), độ lệch chuẩn (std), giá trị nhỏ nhất (min), các giá trị phần tư (25%, 50%, 75%) và giá trị lớn nhất (max) của các đặc trưng trong tập dữ liệu.

+ Train:

Train Data Description:

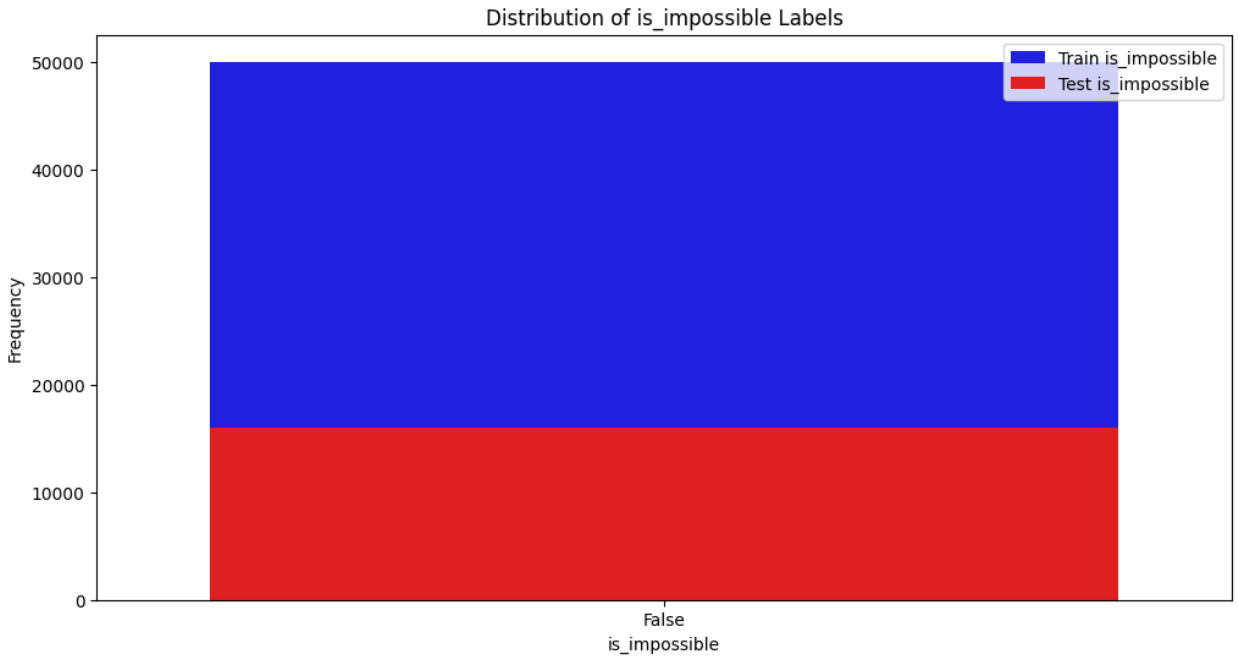
	answer_start	context_length	question_length	answer_length
count	50046.000000	50046.000000	50046.000000	50046.000000
mean	325.467470	792.469868	61.946469	20.320305
std	289.410978	360.048254	23.613835	28.352451
min	-1.000000	131.000000	1.000000	0.000000
25%	103.000000	561.000000	45.000000	7.000000
50%	253.000000	724.000000	59.000000	13.000000
75%	473.000000	969.000000	75.000000	23.000000
max	3180.000000	3280.000000	277.000000	1135.000000

+ Test:

Test Data Description:

	answer_start	context_length	question_length	answer_length
count	15994.000000	15994.000000	15994.000000	15994.000000
mean	292.538577	727.811617	61.776666	17.793235
std	280.477693	387.056042	22.841887	25.897956
min	-1.000000	131.000000	1.000000	0.000000
25%	87.000000	443.000000	45.000000	7.000000
50%	214.000000	658.000000	59.000000	12.000000
75%	415.000000	924.000000	75.000000	20.000000
max	3180.000000	3280.000000	209.000000	1135.000000

- Phân phối nhãn is_impossible:



Nhận xét: Cả hai tập train và test đều chứa các câu hỏi có câu trả lời hợp lệ, điều này cho thấy mô hình cần tập trung vào việc học cách trích xuất câu trả lời từ ngữ cảnh mà không cần phải xử lý các câu hỏi không có câu trả lời.

2. Thử nghiệm trên mô hình có sẵn:

Nhóm sử dụng mô hình "***deepset/xlm-roberta-base-squad2***".

2.1. Tổng quan về mô hình:

➢ Mô hình "deepset/xlm-roberta-base-squad2" được công bố vào năm 2020.

Nó là một mô hình được huấn luyện trên tập dữ liệu SQuAD 2.0 (Stanford Question Answering Dataset).

➢ Mô hình này được xây dựng trên nền tảng XLM-RoBERTa, một biến thể của RoBERTa với khả năng xử lý nhiều ngôn ngữ.

➢ XLM-RoBERTa kết hợp các yếu tố tốt nhất từ cả RoBERTa và XLM. Đây là một mô hình transformer đa ngôn ngữ, được huấn luyện trên hơn 2.5 terabyte văn bản từ 100 ngôn ngữ.

a) Kiến trúc mô hình:

Mô hình XLM-RoBERTa sử dụng kiến trúc transformer với encoder bidirectional. Các encoder này có thể xử lý đầu vào là một chuỗi token và trả về các vector ngữ nghĩa đại diện cho các token đó. Mô hình "deepset/xlm-roberta-base-squad2" được xây dựng trên nền tảng XLM-RoBERTa với cấu hình như sau:

- Số lớp (layers): 12
- Số đầu chú ý (attention heads): 12
- Kích thước ẩn (hidden size): 768
- Số tham số: 125 triệu

b) Huấn luyện mô hình:

Mô hình được huấn luyện theo hai giai đoạn chính:

- Pre-training: XLM-RoBERTa được huấn luyện trước trên một tập dữ liệu đa ngôn ngữ lớn. Quá trình này giúp mô hình học được các đặc điểm ngôn ngữ chung từ nhiều ngôn ngữ khác nhau.
- Fine-tuning: Mô hình sau đó được tinh chỉnh trên tập dữ liệu SQuAD 2.0 để tối ưu hóa cho nhiệm vụ trả lời câu hỏi. Quá trình này bao gồm việc huấn luyện mô hình để dự đoán vị trí bắt đầu và kết thúc của câu trả lời trong đoạn văn bản.

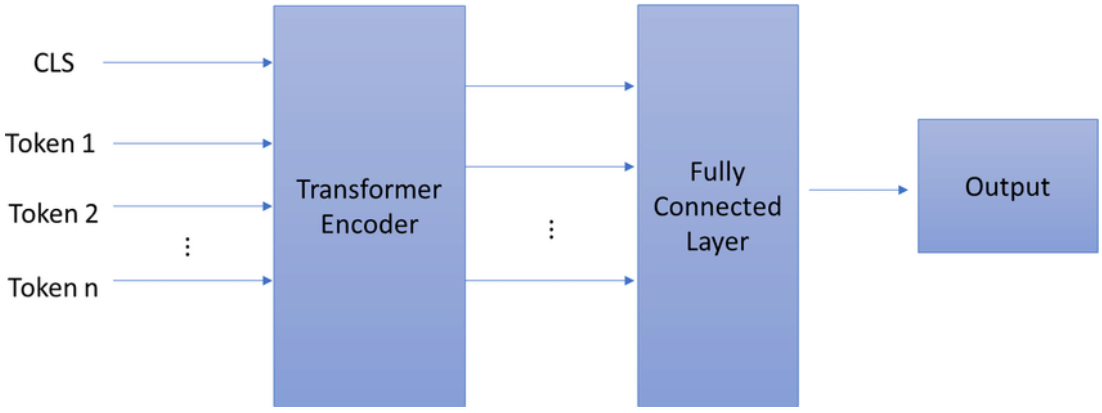
c) Cách thức hoạt động của mô hình:

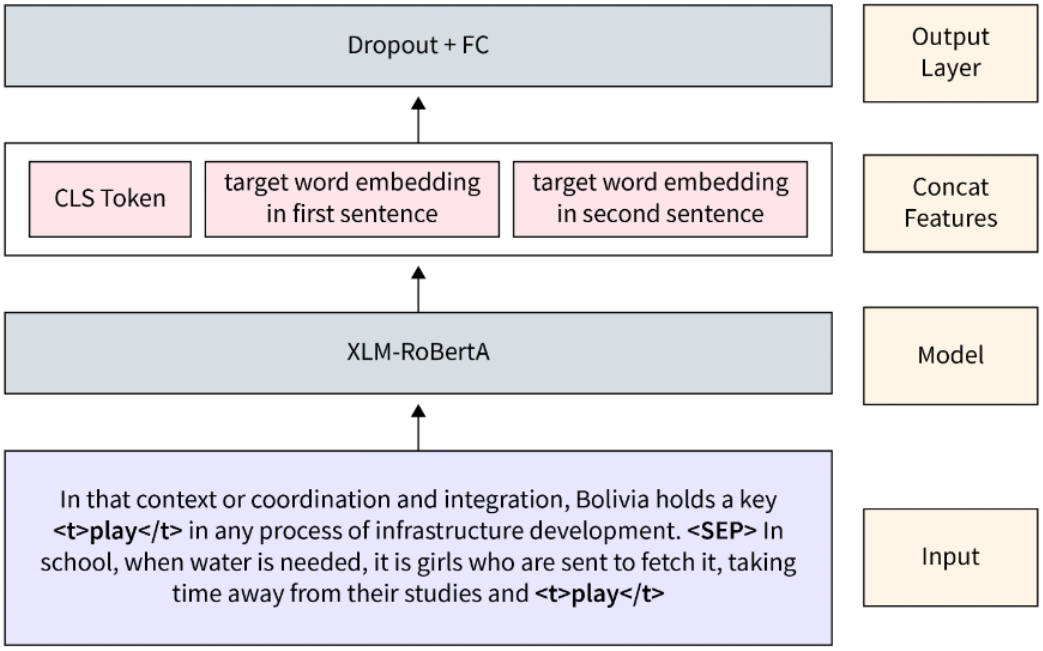
• Input tokenization: Đoạn văn bản và câu hỏi được token hóa thành các chuỗi token sử dụng từ điển của XLM-RoBERTa.

• Embedding: Các token được chuyển đổi thành các vector nhúng (embedding vectors).

• Encoding: Các vector nhúng được đưa qua các lớp transformer để mã hóa thành các vector ngữ nghĩa.

• Answer prediction: Mô hình sử dụng các vector ngữ nghĩa để dự đoán vị trí bắt đầu và kết thúc của câu trả lời. Nếu câu trả lời không tồn tại trong đoạn văn bản, mô hình sẽ trả về một câu trả lời trống.





d) Điểm mạnh của mô hình:

• Hiệu suất cao trong nhiệm vụ trả lời câu hỏi: Mô hình có khả năng xác định chính xác vị trí của câu trả lời hoặc nhận biết khi nào không có câu trả lời.

• Đa ngôn ngữ: Khả năng xử lý tốt các ngôn ngữ khác nhau, giúp mô hình có thể áp dụng trong các ngữ cảnh đa ngôn ngữ.

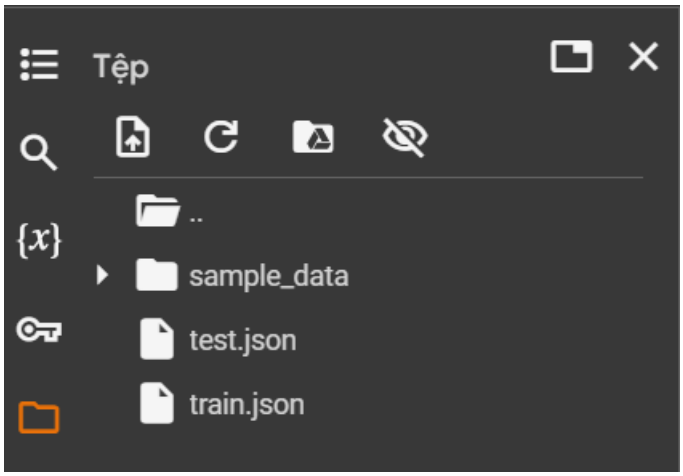
• Hiểu ngữ cảnh sâu: Nhờ vào kiến trúc transformer và quá trình huấn luyện trên tập dữ liệu lớn, mô hình có khả năng nắm bắt ngữ nghĩa và ngữ cảnh sâu sắc.

2.2. Thử nghiệm mô hình trên dataset:

Các bước chính:

1.Tải dữ liệu và chuyển dữ liệu sang Dataset của Hugging Face

Tải file lên Google colab:




```
# Load the dataset
def load_data(file_path):
    with open(file_path, 'r', encoding='utf-8') as f:
        data = json.load(f)
    return data

train_data = load_data('/content/train.json')
test_data = load_data('/content/test.json')
```

```
[ ]  # Transform the data into HuggingFace Dataset format
     def transform_data(data):
         contexts = []
         questions = []
         answers = []

         for item in data:
             context = item['context']
             for qas in item['qas']:
                 question = qas['question']
                 for answer in qas['answers']:
                     contexts.append(context)
                     questions.append(question)
                     answers.append({
                         'text': answer['text'],
                         'answer_start': answer['answer_start']
                     })

         return Dataset.from_dict({
             'context': contexts,
             'question': questions,
             'answers': answers
         })

     train_dataset = transform_data(train_data)
     test_dataset = transform_data(test_data)
```

2. Sử dụng mô hình trên đã được tinh chỉnh cho bài toán hỏi đáp. Chuẩn bị tokenizer từ mô hình đã chọn để mã hóa dữ liệu. Xử lý dữ liệu để phù hợp với định dạng đầu vào của mô hình.

```
# Tokenizer and model
model_checkpoint =  "deepset/xlm-roberta-base-squad2"
tokenizer = AutoTokenizer.from_pretrained(model_checkpoint)
# Load the model
model = AutoModelForQuestionAnswering.from_pretrained(model_checkpoint)
```

Một phần đoạn mã tiền xử lí dữ liệu (Chi tiết trong file .ipynb)

```python
# Tokenize the dataset
def prepare_train_features(examples):
    tokenized_examples = tokenizer(
        examples["question"],
        examples["context"],
        truncation="only_second",
        max_length=384,
        stride=128,
        return_overflowing_tokens=True,
        return_offsets_mapping=True,
        padding="max_length",
    )

    sample_mapping = tokenized_examples.pop("overflow_to_sample_mapping")
    offset_mapping = tokenized_examples.pop("offset_mapping")

    tokenized_examples["start_positions"] = []
    tokenized_examples["end_positions"] = []

    for i, offsets in enumerate(offset_mapping):
        input_ids = tokenized_examples["input_ids"][i]
        cls_index = input_ids.index(tokenizer.cls_token_id)

        sequence_ids = tokenized_examples.sequence_ids(i)
        sample_index = sample_mapping[i]
        answers = examples["answers"][sample_index]

        if len(answers["text"]) == 0:
            tokenized_examples["start_positions"].append(cls_index)
```

Ý tưởng chính:

➢ Mã hóa văn bản: Sử dụng tokenizer để mã hóa các cặp câu hỏi và ngữ cảnh, cắt ngắn (truncation) để đảm bảo độ dài tối đa là 384 token và thêm padding để đồng nhất độ dài.

➢ Quản lý overflow: Lưu trữ thông tin về các đoạn văn bản bị cắt ngắn và offset để xác định vị trí của các token trong ngữ cảnh gốc.

➢ Xác định vị trí câu trả lời:

- Khởi tạo danh sách để lưu trữ vị trí bắt đầu và kết thúc của câu trả lời trong các token đã mã hóa.
- Duyệt qua từng phần tử trong danh sách offset và xác định vị trí bắt đầu và kết thúc của câu trả lời dựa trên các chỉ số trong offset.
- Nếu câu trả lời không tồn tại hoặc không khớp, vị trí bắt đầu và kết thúc sẽ được đặt là vị trí của token [CLS].

➢ Trả về kết quả: Danh sách các token đã mã hóa cùng với các vị trí bắt đầu và kết thúc của câu trả lời

- tokenizer(...): Mã hóa các cặp câu hỏi và ngữ cảnh, với các tham số kiểm soát việc cắt ngắn, padding và xử lý overflow.
- sample_mapping và offset_mapping: Lưu trữ thông tin về các đoạn văn bản bị cắt ngắn và vị trí của các token trong ngữ cảnh gốc.
- Vị trí câu trả lời: Tính toán và lưu trữ vị trí bắt đầu và kết thúc của câu trả lời trong các token đã mã hóa.

3. Huấn luyện mô hình:
Thiết lập các tham số huấn luyện bao gồm đường dẫn lưu kết quả, kích thước batch, số epoch,....

```
# Training arguments
training_args = TrainingArguments(
    output_dir="./results",
    evaluation_strategy="epoch",
    learning_rate=2e-5,
    per_device_train_batch_size=8,
    per_device_eval_batch_size=8,
    gradient_accumulation_steps=16,
    num_train_epochs=3,
    weight_decay=0.01,
    fp16=True,  # Sử dụng FP16
)

# Trainer
trainer = Trainer(
    model=model,
    args=training_args,
    train_dataset=train_dataset,
    eval_dataset=test_dataset,
    tokenizer=tokenizer,
)
```

Sử dụng Trainer của Hugging Face để huấn luyện mô hình với dữ liệu đã chuẩn bị.

```
# Train the model
trainer.train()
```

[1254/1254 1:22:44, Epoch 2/3]

Epoch	Training Loss	Validation Loss
0	No log	1.072912
1	1.455800	0.904642
2	1.133500	0.850248

```
TrainOutput(global_step=1254, training_loss=1.2444229247657497,
metrics={
'train_runtime': 4969.5611,
'train_samples_per_second': 32.362,
'train_steps_per_second': 0.252,
'total_flos': 3.1455948762906624e+16,
'train_loss': 1.2444229247657497,
'epoch': 2.994179973138338})
```

4. Đánh giá mô hình:

```
# Evaluate the model
metrics = trainer.evaluate()
print(metrics)
```

```
{'eval_loss': 0.8502483367919922,
'eval_runtime': 144.3028,
'eval_samples_per_second': 118.709,
'eval_steps_per_second': 14.844,
'epoch': 2.994179973138338}
```

- Nhận xét:

+ Mô hình có mức eval_loss là 0.8502, điều này cho thấy mô hình đã học được và tổng quát hóa tốt trên tập dữ liệu đánh giá. Mức độ lỗi này là chấp nhận được và biểu thị hiệu suất mô hình khá tốt.

+ Với tốc độ xử lý là 118.71 mẫu/giây và 14.84 bước/giây, mô hình có khả năng xử lý nhanh và hiệu quả trên tập dữ liệu đánh giá. Điều này là quan trọng khi triển khai mô hình trong môi trường thực tế để đảm bảo tính hiệu quả và khả năng đáp ứng.

→ Mô hình đã đạt được kết quả tốt với mức eval_loss chấp nhận được, tốc độ xử lý nhanh và thời gian chạy hợp lý. Kết quả này cho thấy mô hình có tiềm năng áp dụng trong các bài toán hỏi đáp tương tự trên các tập dữ liệu thực tế.

5. Hàm dự đoán cùng với thử nghiệm testcase:

```python
# Prediction function
def predict(context, question):
    inputs = tokenizer(question, context, return_tensors='pt', truncation=True, max_length=512)

    # Chuyển inputs sang GPU nếu có
    device = torch.device('cuda' if torch.cuda.is_available() else 'cpu')
    inputs = {key: value.to(device) for key, value in inputs.items()}

    # Chuyển mô hình sang GPU nếu có
    model.to(device)

    outputs = model(**inputs)
    answer_start = torch.argmax(outputs.start_logits)
    answer_end = torch.argmax(outputs.end_logits) + 1
    answer = tokenizer.convert_tokens_to_string(tokenizer.convert_ids_to_tokens(inputs['input_ids'][0][answer_start:answer_end]))
    return answer
```

Testcase:

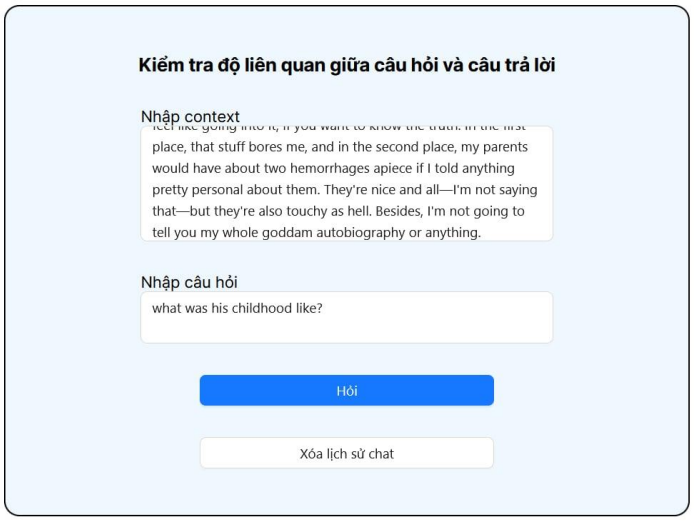
```python
test_cases = [
    {
        "context": "Albert Einstein là một nhà vật lý lý thuyết người Đức đã phát triển thuyết tương đối, một trong hai trụ cột của vật lý hiện đại.",
        "question": "Albert Einstein đã phát triển lý thuyết gì?"
    },
    {
        "context": "Amazon là một công ty công nghệ đa quốc gia của Mỹ có trụ sở tại Seattle, Washington, được thành lập bởi Jeff Bezos vào năm 1994.",
        "question": "Amazon được thành lập vào năm nào?"
    },
    {
        "context": "Công ty Apple Inc. đã được thành lập bởi Steve Jobs, Steve Wozniak và Ronald Wayne vào ngày 1 tháng 4 năm 1976.",
        "question": "Ai là những người sáng lập Apple Inc.?"
    },
    {
        "context": "Đại học Harvard là một trong những trường đại học danh tiếng nhất trên thế giới, được thành lập vào năm 1636 tại Cambridge, Massachusetts.",
        "question": "Đại học Harvard được thành lập vào năm nào?"
    },
    {
        "context": "Python là một ngôn ngữ lập trình bậc cao, thông dịch và hướng đối tượng, được tạo ra bởi Guido van Rossum và ra mắt lần đầu tiên vào năm 1991.",
        "question": "Ngôn ngữ lập trình Python được tạo ra bởi ai?"
    },
    {
        "context": "Điện thoại thông minh đầu tiên, được phát triển bởi IBM, đã được giới thiệu vào năm 1992 và được gọi là Simon Personal Communicator.",
        "question": "Điện thoại thông minh đầu tiên được phát triển bởi công ty nào?"
    },
```

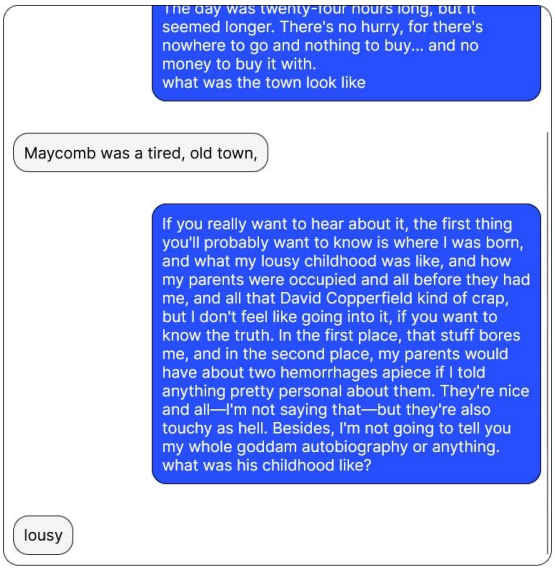
Output:

```
Test case 1:
Question: Albert Einstein đã phát triển lý thuyết gì?
Predicted answer: thuyết tương đối,

Test case 2:
Question: Amazon được thành lập vào năm nào?
Predicted answer: 1994.

Test case 3:
Question: Ai là những người sáng lập Apple Inc.?
Predicted answer: Steve Jobs, Steve Wozniak và Ronald Wayne

Test case 4:
Question: Đại học Harvard được thành lập vào năm nào?
Predicted answer: 1636

Test case 5:
Question: Ngôn ngữ lập trình Python được tạo ra bởi ai?
Predicted answer: Guido van Rossum

Test case 6:
Question: Điện thoại thông minh đầu tiên được phát triển bởi công ty nào?
Predicted answer: IBM,
```

2.3. Giao diện ứng dụng web cho mô hình:

a) Giao diện web:



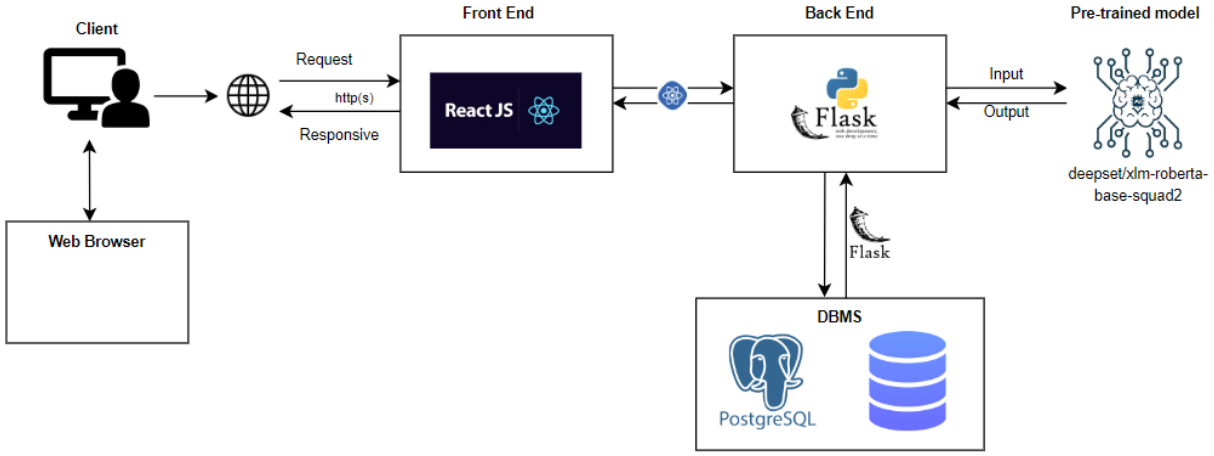


b) Phân tích usecase:

- User nhập dữ liệu vào 2 ô input lần lượt là Nhập câu hỏi và Nhập văn bản (ngữ cảnh)
- User nhấn nút gởi để gởi dữ liệu về server xử lí
- Trang web disable nút gởi để tránh tình trạng user gởi liên tục về server trong khi server đang xử lí input trước
- Trang web hiện chat bubble của user là các thông tin user nhập
- Trang web hiện chat bubble của hệ thống sau một thời gian ngắn (dưới 1 giây), hiển thị ‘Loading...’ để báo hệ thống đang xử lí
- Trang web gởi dữ liệu về server sau một khoảng thời gian ngắn
- Server nhận dữ liệu, đưa dữ liệu vào model xử lí để cho ra kết quả
- Model xử lí xong kết quả, trả về server
- Server gởi kết quả cho web
- Trang web thu hồi lại chat bubble ‘Loading...’ và hiện lên chat buble của hệ thống chứa kết quả trả về từ model
- Trang web enable lại nút gởi

c) Công nghệ sử dụng:

- Front End: Reactjs
- Back End: Flask



☙ TÀI LIỆU THAM KHẢO ❧

[1] https://huggingface.co/deepset/xlm-roberta-base-squad2

[2] https://amitnikhade.medium.com/question-answering-in-association-with-roberta-a11518e70507