TRƯỜNG ĐẠI HỌC KHOA HỌC TỰ NHIÊN

THÀNH PHỐ HỒ CHÍ MINH

□□□□□

BÁO CÁO ĐỒ ÁN 2

Môn: Khai thác dữ liệu văn bản và ứng dụng

Giáo viên hướng dẫn: Nguyễn Trần Duy Minh

Lê Thanh Tùng

Thành viên:

21127680 – Trần Văn Quyết

19127336 – La Gia Bảo

21127577 – Trịnh Hoàng An

21127056 – Lâm Thiều Huy

Mục lục

I. Bảng phân công việc

MSSV	Thành viên	Phân công
21127680	Trần Văn Quyết	Mô hình DistilBERT
19127336	La Gia Bảo	Viết báo cáo
21127577	Trịnh Hoàng An	EDA
21127056	Lâm Thiều Huy	Làm powerpoint và mô hình XLNet

II. Mức độ hoàn thành

Trần Văn Quyết: 100%. Phát triển một mô hình trả lời câu hỏi tự động dựa trên kiến trúc DistilBERT. Thiết lập một tập dữ liệu tùy chỉnh, xây dựng lớp QADataset để xử lý dữ liệu và sử dụng tokenizer của mô hình. Mô hình được huấn luyện với hàm mất mát CrossEntropyLoss và optimizer Adam. Quá trình huấn luyện diễn ra trong một số epoch nhất định và đã lưu trọng số tốt nhất của mô hình. Kết quả kiểm thử cho thấy mô hình có hiệu quả đáng kể trong việc trích xuất câu trả lời từ ngữ cảnh, dù vẫn còn không gian để cải thiện thêm.

La Gia Bảo: 100%. Tóm tắt và ghi nhận lại tất cả các bước, kết quả của những thành viên và trình bày chi tiết ra báo cáo. Hợp tác với mọi người để miêu tả chính xác thành quả những mô hình và đồng thời bộc lộ đúng theo những yêu cầu đề hỏi về phần report.

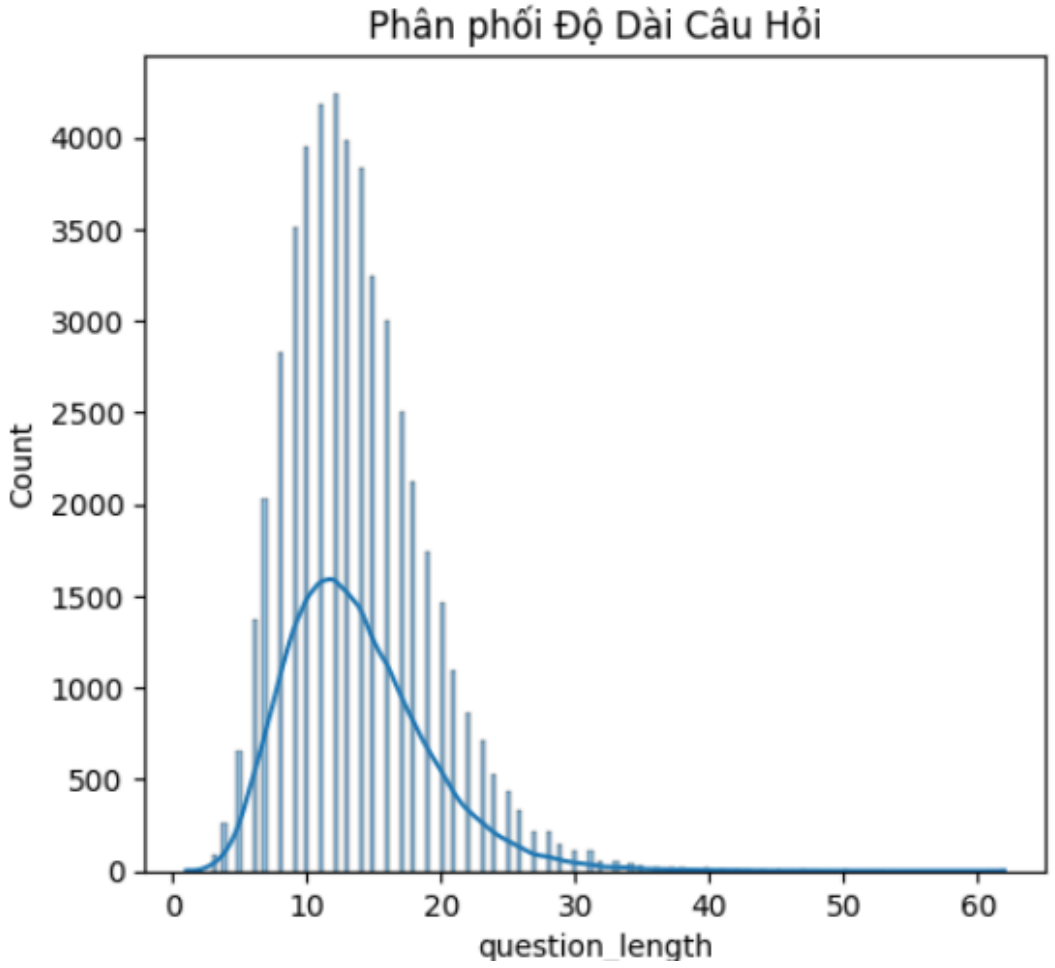
Trịnh Hoàng An: 100%. Nghiên cứu nhận thấy rằng bộ dữ liệu hỏi đáp tự động này chủ yếu gồm những câu hỏi và câu trả lời ngắn, cũng như đoạn văn cung cấp ngữ cảnh đủ rõ ràng nhưng không quá dài. Tất cả câu hỏi đều có câu trả lời, thể hiện tính ứng dụng cao trong huấn luyện mô hình trả lời tự động. Phân tích này hỗ trợ việc thiết kế và tinh chỉnh các hệ thống hỏi đáp, cũng như cung cấp cái nhìn sâu sắc về cách tối ưu hóa các mô hình học máy.

Lâm Thiều Huy: 100%. Đã xây dựng và huấn luyện một mô hình XLNet trên dữ liệu văn bản để trả lời các câu hỏi. Đọc và tiền xử lý dữ liệu từ file JSON, mã hóa văn bản và câu hỏi, xác định vị trí của câu trả lời và gắn nhãn cho tính khả thi của câu trả lời. Chuẩn bị mô hình bằng cách sử dụng tokenizer và cấu hình optimizer và scheduler cho quá trình huấn luyện. Huấn luyện mô hình và đánh giá hiệu suất của nó trên tập kiểm thử, tìm ra các tham số tối ưu và lưu trữ trọng số tốt nhất. Cuối cùng, theo dõi và ghi nhận loss, thời gian huấn luyện trên tập huấn luyện và đánh giá mô hình trên tập kiểm thử để đo lường loss và accuracy, cũng như thời gian kiểm thử.

III. EDA

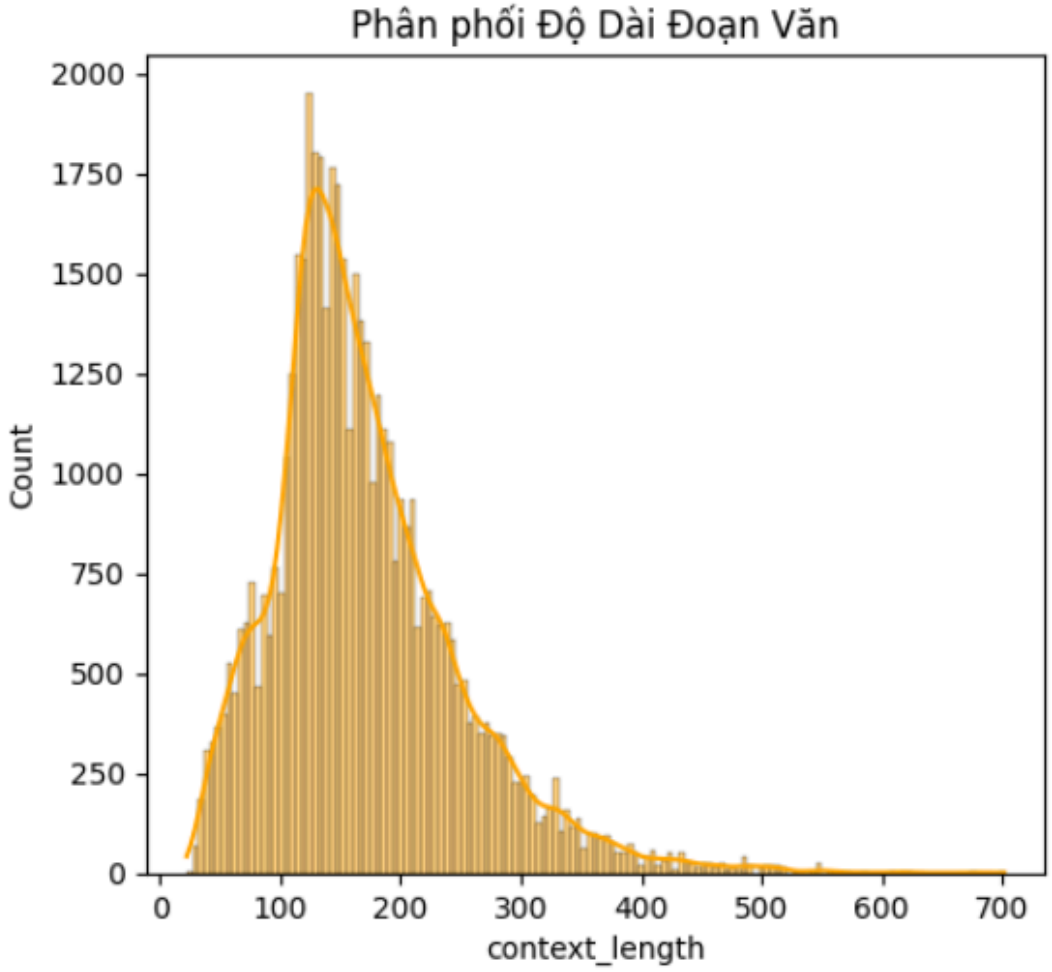
1. Biểu Đồ Phân Phối Độ Dài Câu Hỏi

- **Biểu Đồ Mô Tả**: Phân phối độ dài câu hỏi cho thấy phần lớn câu hỏi có độ dài từ khoảng 10 đến 20 từ. Có một đỉnh nổi bật xung quanh khoảng 10-15 từ, nơi đó có nhiều câu hỏi tập trung nhất. Phân phối giảm nhanh chóng sau khoảng 20 từ, cho thấy số lượng câu hỏi dài hơn giảm đáng kể.
- **Điểm Đặc Biệt**: Đỉnh của biểu đồ phân phối gần với trục y, điều này có thể chỉ ra rằng một số câu hỏi ngắn có độ dài ít từ được lặp lại nhiều lần.
- **Kết Luận**: Đa số câu hỏi trong bộ dữ liệu này ngắn gọn, có thể do chúng được thiết kế để trực tiếp và cô đọng, điều này là lý tưởng cho hệ thống hỏi đáp tự động, nơi mà thông tin cần được trích xuất nhanh chóng và chính xác.



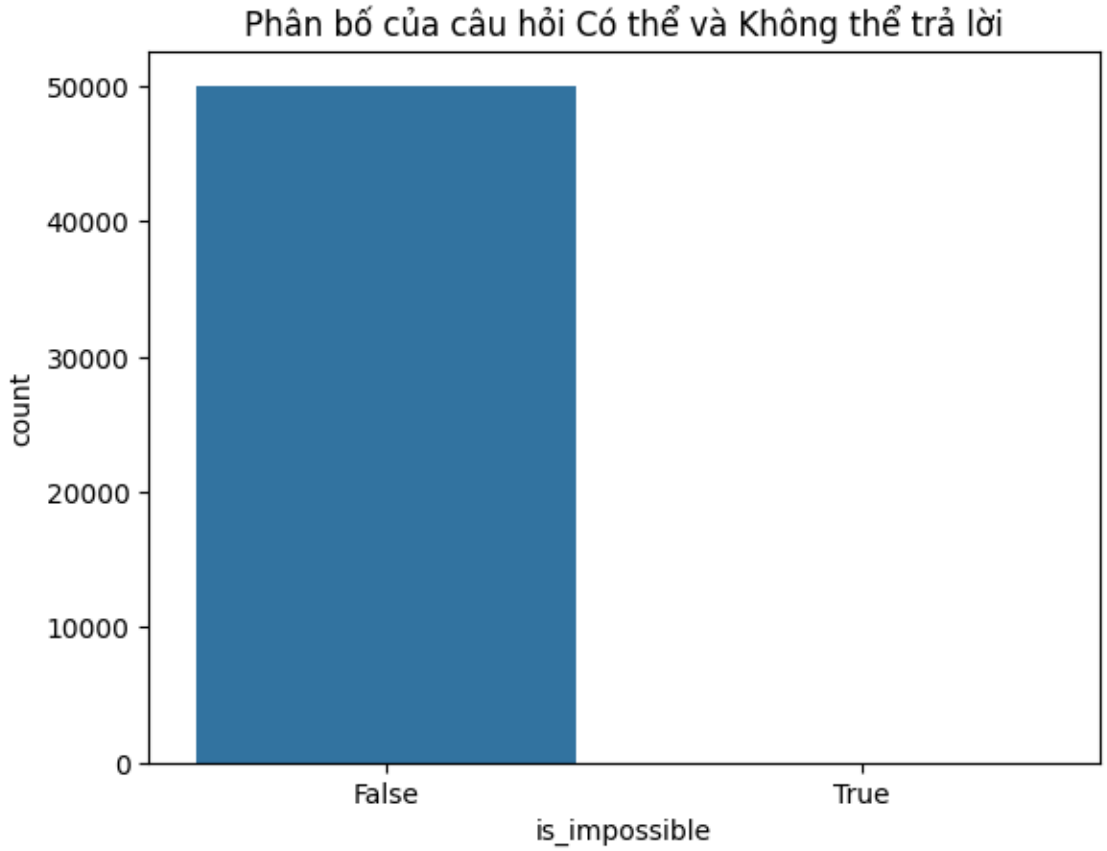
2. Biểu Đồ Phân Phối Độ Dài Đoạn Văn

- **Biểu Đồ Mô Tả**: Đối với đoạn văn, phân phối có vẻ rộng hơn và đỉnh không cao bằng phân phối của câu hỏi. Độ dài của đoạn văn chủ yếu tập trung trong khoảng từ 50 đến 250 từ. Độ dài trung bình của một đoạn văn là khoảng 100 từ.
- **Điểm Đặc Biệt**: Phân phối có đuôi dài về bên phải, nghĩa là có một số đoạn văn rất dài so với phần còn lại. Điều này là do bản chất của ngữ cảnh mà câu hỏi được đặt ra hoặc sự đa dạng trong kích thước của thông tin được cung cấp.
- **Kết Luận**: Đoạn văn bản có độ dài đủ để cung cấp ngữ cảnh cho câu hỏi, nhưng không quá dài để làm giảm hiệu quả xử lý dữ liệu.



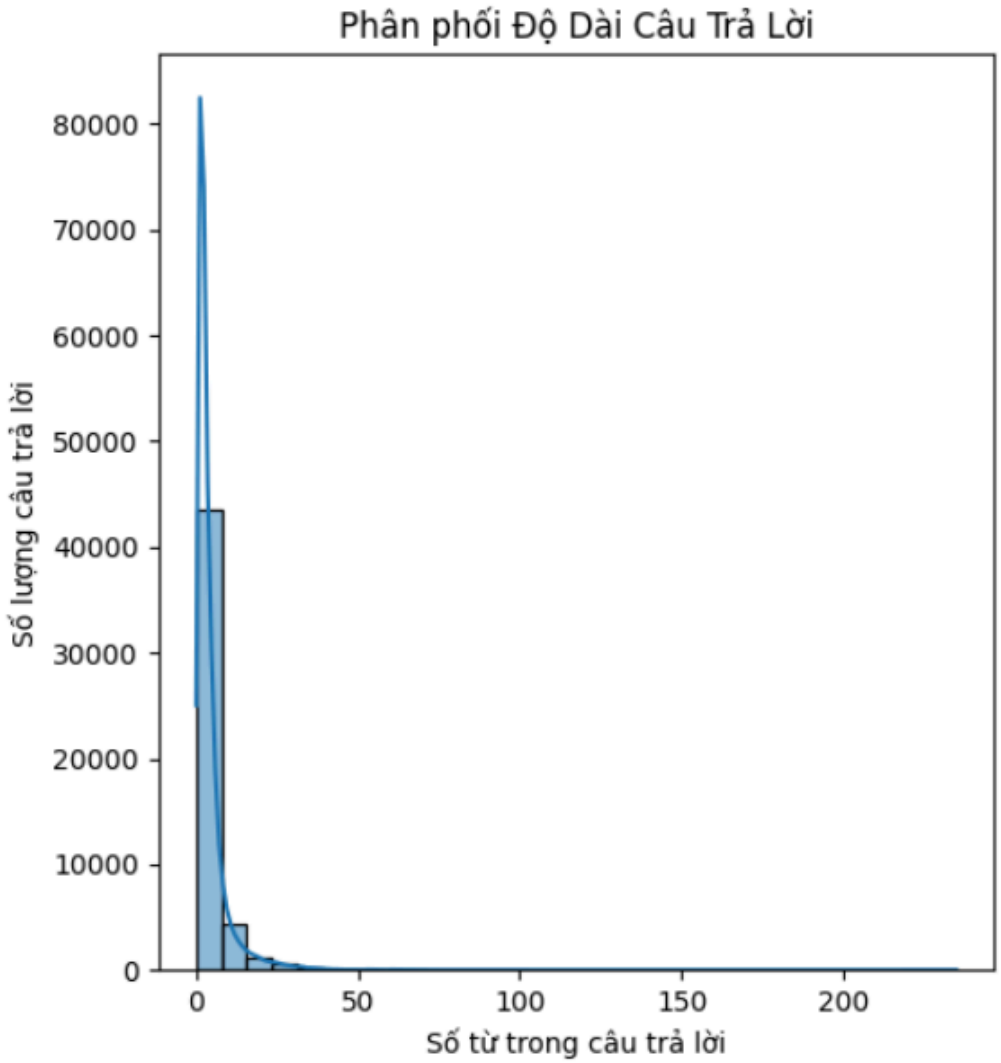
3. Biểu Đồ Phân Bố Câu Hỏi Có thể và Không thể trả lời

- **Biểu Đồ Mô Tả**: Biểu đồ này chỉ hiển thị một cột cho giá trị **False**, nghĩa là tất cả câu hỏi trong bộ dữ liệu đều có thể trả lời.
- **Điểm Đặc Biệt**: Không có cột cho giá trị **True**, nghĩa là không có câu hỏi nào được đánh dấu là không thể trả lời. Điều này có thể cho thấy bộ dữ liệu được tạo ra để tập trung vào các câu hỏi có câu trả lời cụ thể.
- **Kết Luận**: Điều này chỉ ra rằng hệ thống hoặc mô hình hỏi đáp cần được huấn luyện với khả năng tìm kiếm và trả lời câu hỏi chính xác thay vì xác định câu hỏi không có câu trả lời. Điều này cũng có thể phản ánh một giả định trong việc xây dựng dataset rằng mọi câu hỏi đều liên quan và có câu trả lời có sẵn trong dữ liệu.



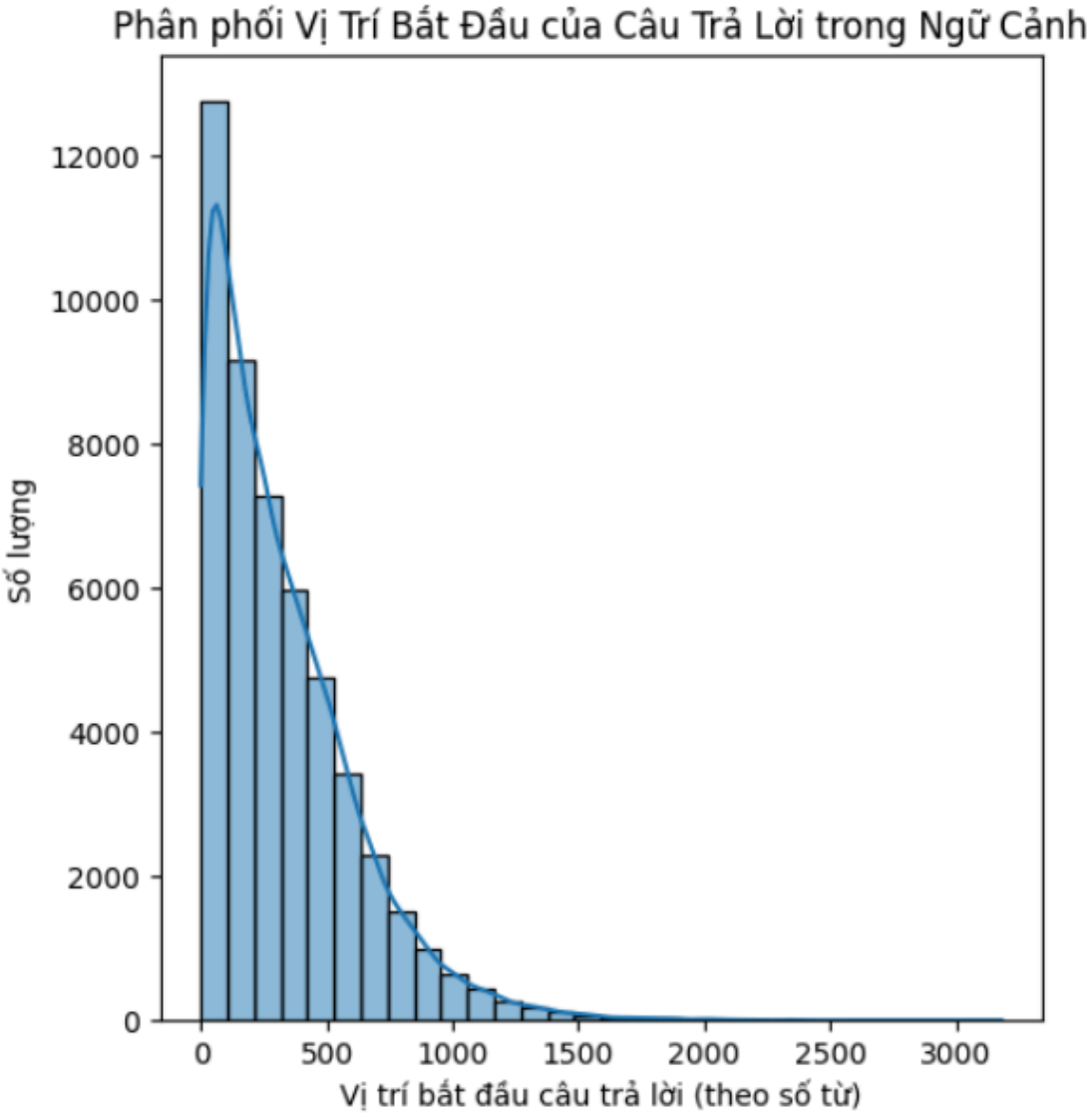
4. Biểu Đồ Phân Phối Độ Dài Câu Trả Lời

- **Đặc điểm**: Đa số câu trả lời có độ dài rất ngắn, thường ít hơn 20 từ. Điều này phản ánh việc câu trả lời chủ yếu là thông tin cô đọng, trực tiếp liên quan đến câu hỏi.
- **Phân tích**: Biểu đồ có đỉnh cao ngay tại phần đầu của trục hoành, sau đó giảm nhanh chóng, tạo thành một đuôi dài. Phần lớn câu trả lời có thể được phân loại là 'factoid' — tức là chúng trả lời cho các câu hỏi cụ thể bằng một fragment ngắn hoặc một câu đơn.
- **Kết luận**: Mô hình hỏi đáp cần phải chính xác trong việc tìm kiếm thông tin cần thiết để đưa ra câu trả lời ngắn gọn và chính xác.



5. Biểu Đồ Phân Phối Vị Trí Bắt Đầu của Câu Trả Lời trong Ngữ Cảnh

- **Đặc điểm**: Phần lớn câu trả lời bắt đầu ở vị trí đầu tiên của đoạn văn, với số lượng giảm dần theo vị trí bắt đầu.
- **Phân tích**: Có thể thấy có một số lượng lớn câu trả lời bắt đầu ngay tại đầu đoạn văn, điều này có thể phản ánh việc đặt thông tin quan trọng hoặc câu trả lời mong đợi ngay tại phần mở đầu.
- **Kết luận**: Điều này cung cấp manh mối cho việc phát triển mô hình hỏi đáp tự động; ví dụ, một mô hình có thể ưu tiên xem xét thông tin ở đầu đoạn văn khi tìm kiếm câu trả lời.



6. Tổng kết EDA

- Dựa trên phân tích của 3 biểu đồ ban đầu, bộ dữ liệu dường như được cấu trúc tốt cho việc huấn luyện và đánh giá các mô hình hỏi đáp tự động, với các câu hỏi ngắn gọn và đoạn văn có độ dài vừa phải. Sự thiếu vắng của các câu hỏi không thể trả lời có thể là một giới hạn và cần được xem xét khi sử dụng bộ dữ liệu này để kiểm thử hệ thống hỏi đáp tự động.
- Cả hai phân tích của hai biểu đồ sau đều rất hữu ích trong việc thiết kế và tinh chỉnh các hệ thống trích xuất thông tin và hỏi đáp tự động. Kết quả từ các phân tích này có thể hướng dẫn cách chúng ta tiếp cận việc xử lý ngôn ngữ tự nhiên và tối ưu hóa các mô hình học máy cho các tác vụ cụ thể.

IV. XLNet

1. Hướng dẫn chạy code

- Tải file ipynb của mô hình XLNet được cung cấp
- Thêm vào 2 file **train.json** và **test.json** từ kaggle
- Tải những thư viện cần thiết qua dòng lệnh sau:
    a. pip install transformer
    b. pip install torch
    c. pip install pytorch
    d. pip install numpy
- Ấn **"Run All"** để chạy toàn bộ chương trình

2. Các bước được dùng để huấn luyện và đánh giá mô hình

a. Chuẩn bị dữ liệu

- **Đọc dữ liệu:**
    - Sử dụng Python và thư viện json để tải dữ liệu từ các tệp train.json và test.json.
    - Định nghĩa hàm **read_json** để đọc và trả về dữ liệu từ tệp JSON.
- **Kiểm tra cấu trúc dữ liệu:**
    - Lưu trữ và hiển thị cấu trúc của dữ liệu, bao gồm số lượng ví dụ trong tập huấn luyện và kiểm thử, chi tiết của ví dụ đầu tiên, độ dài của context, số lượng câu hỏi và chi tiết câu hỏi đầu tiên.

b. Tiền xử lý Dữ liệu

- **Tokenization và Định dạng Dữ liệu:**
  - **XLNetTokenizerFast.from_pretrained:** Khởi tạo tokenizer để chuyển đổi văn bản thành dạng có thể xử lý được bởi mô hình XLNet.
  - **encode_data:** Mã hóa câu hỏi và ngữ cảnh thành input IDs, attention masks, và token type IDs.
  - Tính toán vị trí bắt đầu và kết thúc của câu trả lời trong các token và xử lý câu hỏi không có câu trả lời khả thi.
- **Tạo DataLoader:**
  - Chuyển danh sách các input tensors thành **TensorDataset.**
  - Sử dụng **DataLoader** để phân batch dữ liệu, cung cấp chế độ xáo trộn cho tập huấn luyện và không xáo trộn cho tập kiểm thử, định nghĩa kích thước batch.

c. Chuẩn bị Mô Hình Huấn Luyện

- **Khởi tạo mô hình và tokenizer:**
  - Tạo thực thể của **XLNetForQuestionAnsweringSimple** từ pretrained model.
  - Khởi tạo tokenizer với cấu hình **xlnet-base-cased.**
- **Cấu hình Optimizer:**
  - Thiết lập **AdamW** làm optimizer với learning rate đã chỉ định.

d. Chuẩn bị chu kỳ huấn luyện

- **Cài đặt Lập lịch cho Optimizer:**
  - Sử dụng **get_linear_schedule_with_warmup** để thiết lập lịch trình giảm dần learning rate, dựa trên số bước tổng cộng tính theo số epochs và kích thước batch.

e. Huấn Luyện mô hình

- **Thiết lập môi trường huấn luyện:**
  - Xác định và sử dụng thiết bị phù hợp (GPU hoặc CPU).
  - Batch size cho tập dữ liệu huấn luyện là 8
  - Di chuyển mô hình lên thiết bị được chọn để tối ưu hóa tốc độ huấn luyện.
- **Thực thi vòng lặp huấn luyện:**
  - **model.train():** Đặt mô hình vào chế độ huấn luyện, cho phép tính toán gradient và cập nhật tham số.
  - Thực hiện qua các epoch, tại mỗi batch cập nhật trọng số bằng cách tính toán gradient và áp dụng backpropagation.

- **loss.backward():** Thực hiện lan truyền ngược để tính toán gradient.
- **optimizer.step(), scheduler.step():** Cập nhật các tham số của mô hình dựa trên gradient và điều chỉnh tỷ lệ học tập.
- **torch.nn.utils.clip_grad_norm_:** Hạn chế gradient để tránh vấn đề bùng nổ gradient trong quá trình huấn luyện.

f. Đánh giá mô hình

- **Chuyển sang chế độ đánh giá:**
  - Chuyển mô hình sang chế độ eval để ngừng theo dõi gradient.
  - Batch size cho tập dữ liệu kiểm tra là 6
  - **model.eval():** Đặt mô hình vào chế độ đánh giá, vô hiệu hóa tính toán gradient để tối ưu hóa hiệu suất.
- **Đánh giá mô hình trên tập kiểm thử:**
  - **torch.no_grad():** Khối lệnh này đảm bảo rằng gradient không được tính toán hoặc lưu trong quá trình đánh giá mô hình, giảm bớt tài nguyên tính toán.
  - Tính toán độ chính xác và loss trung bình dựa trên dự đoán vị trí bắt đầu và kết thúc của câu trả lời so với giá trị thực tế.

3. Kết Quả

a. Kết quả huấn luyện

Loss và Thời Gian Huấn Luyện

- Epoch 1: Mô hình bắt đầu với loss là 2.62 và mất khoảng 10 giờ 47 phút để hoàn thành. Điều này cho thấy quá trình huấn luyện khởi đầu với việc mô hình còn nhiều phải học từ dữ liệu.
- Epoch 2: Loss đã giảm xuống còn 1.94, và thời gian huấn luyện giảm xuống còn 10 giờ 45 phút. Sự cải thiện về mặt loss và thời gian cho thấy mô hình đã hiệu quả hơn trong việc học từ dữ liệu và tối ưu hóa các trọng số.

b. Kết quả kiểm thử

Độ Chính Xác và Loss

- Mô hình đạt được độ chính xác là 61% trên tập kiểm thử với loss là 1.54. Điều này cho thấy mô hình đã khái quát tốt trên dữ liệu mới mà không bị quá khớp, dù rằng độ chính xác vẫn còn dư địa để cải thiện.
- Thời gian cần thiết để đánh giá mô hình là 2 giờ 2 phút 6 giây, phản ánh mức độ phức tạp của mô hình và tính toán nặng nề liên quan đến XLNet.

c. Biến Tốt Nhất Trong Mô Hình

- File "best_params.txt" được cung cấp.
- Ở batch 4103 của epoch 2 với loss khoảng 0.277.

4. Phân Tích và Nhận Xét

a. Hiệu Suất Mô Hình

- Mô hình XLNet đã cho thấy khả năng học từ dữ liệu và cải thiện qua từng epoch. Tuy nhiên, độ chính xác 61% cho thấy mô hình vẫn cần được tinh chỉnh thêm để đạt hiệu quả cao hơn, đặc biệt là trong các trường hợp dữ liệu kiểm thử phức tạp hơn.

b. Đề Xuất Cải Thiện

- **Thiết lập tham số tối ưu hơn:** Điều chỉnh learning rate, số lượng epochs, và thử nghiệm các kỹ thuật như dropout có thể giúp mô hình đạt được độ chính xác cao hơn.
- **Phân tích sâu hơn về lỗi:** Xem xét các trường hợp cụ thể mà mô hình dự đoán sai để hiểu rõ hơn về những hạn chế và điều chỉnh chiến lược huấn luyện.
- **Cải thiện phần cứng:** Sử dụng GPU mạnh mẽ hơn hoặc huấn luyện mô hình trên nhiều GPU có thể giảm đáng kể thời gian cần thiết cho huấn luyện và kiểm thử.

V. DistilBERT

1. HuggingFace model

Load model

```
from transformers import AutoTokenizer, AutoModelForQuestionAnswering
from torch.utils.data import DataLoader, Dataset
import torch

tokenizer = AutoTokenizer.from_pretrained("distilbert/distilbert-base-cased-distilled-squad")
model = AutoModelForQuestionAnswering.from_pretrained("distilbert/distilbert-base-cased-distilled-squad")
```

- **Custom Dataset:**
  - **def spilt_samples:** dùng để chia các sample, bao gồm context, question, answer và is_impossible
  - **Class QADataset:** dùng tokenize của model để return về input_ids, attention_mask, start_position của answer, end_postition
- Chia bộ train dataset thành bộ train và validation, validation dùng bộ này để lưu lại bộ best weight của model trong quá trình training.

2. Training model

- Sử dụng Adam để optimize model với mức learning_rate = 5e-e.
- Sử dụng hàm loss CrossEntropyLoss.
- Vì bộ nhớ có giới hạn nên sử dụng batch_size = 64, mức average loss ban đầu rất cao khoảng 5.5 nên cần nhiều epoch để train model này. Cũng chính vì vấn đề memory nên sử dụng 10 epoch để train tuy nhiên nó đạt được mức loss chấp nhận được, khoảng 3.6 => loss này còn rất cao để đánh giá.
- Best weight được lưu lại ở **'best_model_weights.pth'**.

3. Testing model

- Dựa vào start_position và end_position của output để so sánh và đưa ra kết quả.
- Model được độ acc khoảng 92.4%.

VI. Nguồn tài liệu tham khảo

- https://github.com/pallavrajsahoo/Question-Answering-System-with-SQuAD-dataset
- https://www.analyticsvidhya.com/blog/2020/04/beginners-guide-exploratory-data-analysis-text-data/
- https://www.borealisai.com/research-blogs/understanding-xlnet/
- https://paperswithcode.com/paper/xlnet-generalized-autoregressive-pretraining#code
- https://huggingface.co/docs/transformers/model_doc/xlnet
- https://huggingface.co/docs/transformers/model_doc/distilbert