TRƯỜNG ĐẠI HỌC KHOA HỌC TỰ NHIÊN
KHOA CÔNG NGHỆ THÔNG TIN

Nguyễn Thành Nhân - 20120151
Huỳnh Thiết Gia - 20120070

Ứng dụng mô hình Transformer Tóm tắt văn bản

HỌC THỐNG KÊ

ĐỒ ÁN CUỐI KỲ

Tp. Hồ Chí Minh, tháng 06/2024

Mục lục

1. Yêu cầu của đồ án

Bài toán yêu cầu các bước làm việc sau:

1) Tìm một bài toán liên quan.
2) Tìm một model ứng dụng Transformer để giải bài toán.
3) Tìm một bộ dữ liệu phù hợp với bài toán.
4) Ghi chép lại quá trình tiền xử lý dữ liệu.
5) Huấn luyện model trên tập dữ liệu.
6) Tạo một ứng dụng có giao diện web để chạy model.

Bài toán mà nhóm chọn là Tóm tắt văn bản (Text Summarization), bộ dữ liệu là BillSum, mô hình được chọn là Google T5 Small, web được viết bằng Flask với SQLite.

2. Bài toán tóm tắt văn bản

Bài toán tóm tắt văn bản có mục tiêu là đưa ra được một tóm tắt chứa đựng những thông tin quan trọng nhất trong một đoạn văn bản. Tùy vào cách tạo output mà được phân chia làm 2 nhóm, Extractive ("chiết xuất") và Abstractive ("trừu tượng"), với Extractive là không thay đổi nội dung của đoạn văn và Abstractive là tóm tắt tạo sinh đoạn văn mới từ văn bản gốc.

Ví dụ Extractive Summarization và Abstractive Summarization:

- **Text:** "Peter and Elizabeth took a taxi to attend the night party in the city. While in the party, Elizabeth collapsed and was rushed to the hospital."
- **Extractive summary text:** "Peter and Elizabeth attend party city. Elizabeth rushed hospital."
- **Abstractive summary text:** "Elizabeth was hospitalized after attending a party with Peter."

Với cách cổ điển trước khi Transformer ra đời như: (Extractive) Tiếp cận bằng phân phối bằng cách đánh trọng số các câu bằng tần suất xuất hiện của các từ trong đoạn văn, (Abstractive) các mô hình cơ bản như Naive Bayes và SVM, nén đoạn văn, sử dụng mẫu có sẵn.

Mô hình của nhóm lựa chọn, sẽ được đề cập ở dưới, tóm tắt văn bản theo hướng Abstractive.

3. Mô hình dự đoán

Mô hình được sử dụng là google-t5/t5-small bởi Google. Mục tiêu của họ khi tạo ra model này là dùng nó cho nhiều tác vụ khác nhau với một model duy nhất, một metric duy nhất, một thuật toán tối ưu duy nhất nhằm đơn giản hóa quá trình so sánh giữa các bộ dữ liệu và các bài toán khác nhau.

Cấu trúc mô hình: Google T5 là một mô hình ứng dụng Transformer với input là văn bản, output là phiên bản mã hóa của văn bản, theo cấu trúc Encoder - Decoder (Khác với model Bart, cũng của Google). Tuy nhiên về cái cốt lõi của Transformer thì không có sự thay đổi đáng kể nên nhóm tác giả đã hướng người đọc tới báo cáo gốc của Transformer để hiểu rõ hơn. Và thầy cũng đã giảng về Transformer.

Nguyên văn của họ (đã dịch): "Chúng tôi đơn giản hóa những "nhúng vị trí" (position embedding), trong đó mỗi “nhúng” chỉ đơn giản là một con số được đưa vào hàm logit (ln (p/(1-p))) tương ứng mà được sử dụng để tính toán trọng số của lớp attention. Để tối ưu hiệu năng, chúng tôi cũng chia sẻ các tham số "nhúng vị trí" đó cho tất cả các lớp trong mô hình, mặc dù trong mỗi lớp, các đầu attention sẽ sử dụng tham số học được khác nhau đối với các "nhúng" đó." [...] "Ngắn gọn thì mô hình của chúng tôi gần như tương đồng với mô hình Transformer gốc được đề xuất bởi Vaswani và cộng sự (2017) với các sự khác biệt là: Loại bỏ bias Layer Norm, đặt lớp chuẩn hóa theo lớp (Layer Normalization) ngoài lớp nối tắt, và sử dụng khác phương pháp nhúng vị trí. Vì sự thay đổi cấu trúc này không ảnh hưởng tới các yếu tố mà chúng tôi đang nghiên cứu (Nghiên cứu về quá trình học chuyển giao - người dịch), sự ảnh hưởng của việc thay đổi cấu trúc cứ để tương lai rồi tính." [2]

Input – Output: Input của mô hình T5 là chuỗi mã hóa của các từ trong văn bản đầu vào (tương tự các mô hình Transformer) với độ dài tối đa 1024 ký tự. Output của mô hình là các token tương ứng với văn bản đầu ra, cần dùng tokenizer để “dịch” output ra văn bản và ngược lại, dịch văn bản đầu vào thành chuỗi các từ được mã hóa để đưa vào input (Chi tiết ở phần 5).

Giới hạn phần cứng: Chạy predict trên gpu Tesla P40 24Gb Ram với 50 câu tốn dưới 20 giây, xấp xỉ 0.4 giây cho một câu. Nhóm chưa thử nghiệm với cpu.

Pretrain trên dataset: Trước khi huấn luyện trên các tập dataset chất lượng cao dành riêng cho bài toán này, đa phần các mô hình sẽ được pre-train sẵn trên các tập dataset văn bản không gán nhãn để tập cho mô hình xuất ra được văn bản “gần giống tiếng người”.

Model đã được pre-train sẵn trên tập dataset Colossal Clean Crawled Corpus (C4), vốn là một tập dữ liệu văn bản khổng lồ được crawl từ dữ liệu khắp nơi trên internet. Nhóm Google thấy tập đó có số lượng lớn nhưng chất lượng cực thấp nên họ đã tiến hành "lọc" lại bộ dữ liệu như sau[1]:

- Chỉ giữ những câu kết thúc bằng dấu câu. (Ví dụ: "? . !")
- Loại bỏ các trang web ít hơn 3 câu, và chỉ giữ những câu dài hơn 4 chữ.
- Loại bỏ các trang có những từ "nhạy cảm".
- Nhiều trang đưa ra cảnh báo cần bật Javascript nên họ loại bỏ những câu nào có chữ "Javascript".
- Loại bỏ các trang có cụm từ "Lorem Ipsum" vì nó là văn mẫu để tạm.
- Một số trang web sẽ phải chứa code. Vì ngoặc nhọn "{" được dùng trong nhiều ngôn ngữ lập trình (như Javascript, vốn được dùng rất nhiều trên web), loại bỏ trang nào chứa ngoặc nhọn.
- Loại bỏ các ký hiệu dẫn nguồn (Như của Wikipedia là "[1], [Cần dẫn chứng],...") nên loại bỏ hết các ký hiệu đó.
- Nhiều trang có các câu thông báo về quy định lưu trữ cookie nên loại bỏ các dòng chứa các chuỗi: “terms of use”, “privacy policy”, “cookie policy”, “uses cookies”, “use of cookies”, or “use cookies”.
- Để giảm sự trùng lặp cho bộ dữ liệu, tiến hành loại bỏ các bộ 3 câu liên tiếp bị trùng trong cả bộ dữ liệu, giữ lại đúng một bộ duy nhất.

4. Bộ dữ liệu

Dataset được sử dụng là dataset BillSum[4], gồm các dự luật của quốc hội Mỹ và bang California (“US Congressional and California state bills”) và tóm tắt của chúng. Bao gồm các đặc trưng: Nội dung của Bill, tóm tắt, tiêu đề (chỉ có với dự luật quốc hội, không có với California), độ dài của nội dung, độ dài tóm tắt.

Dataset được chia sẵn thành 2 tập Train và Test với kích thước xấp xỉ tương ứng là 19000 hàng và 3000 hàng[5]. Nhóm cũng đã thử sử dụng mỗi tập train cho quá trình huấn luyện, tách tập train đó thành 90% để train và 10% để test trong quá trình train để đảm bảo chắc chắn tập subset Test 3000 hàng kia không liên quan gì đến quá trình huấn luyện, mà nó chỉ được dùng để kiểm thử độc lập, thì thấy các chỉ số không chênh lệch quá 1% (Để tránh việc nhóm vô tình huấn luyện nhầm luôn tập subset Test, đảm bảo tính chính xác của kết quả ở mục kết luận).

Lựa chọn khác cũng có thể phù hợp là CNN_Dailymail với cấu trúc cũng tương tự như vậy, chi tiết bài báo và tóm tắt của hai tạp chí: CNN và Daily Mail. Nhưng vì sự đa dạng trong văn phong của bài báo so với văn phong trong dự luật và kích thước lớn hơn so với BillSum (300.000 hàng so với 24.000) nên nhóm đã chọn BillSum.

Ảnh chụp một phần dữ liệu gốc chưa qua tiền xử lý.

5. Tiền xử lý bộ dữ liệu

```
from transformers import AutoTokenizer
tokenizer = AutoTokenizer.from_pretrained("google-t5/t5-small")
```

Dữ liệu được tokenized theo như báo cáo của mô hình T5, vì mô hình này có cấu trúc tương tự với Transformer nên quá trình tiền xử lý dataset cũng gần như tương tự báo cáo gốc, bao gồm các bước[3]:

1) Gán thêm vào đầu câu chuỗi giúp khởi tạo tác vụ, ở đây là "Summarize: ", báo cáo của Google nói rằng làm thế giúp cải thiện hiệu quả của mô hình so với không làm.
2) Mã hóa văn bản đầu vào thành vector các token, ở đây không biến đổi thông qua các mô hình Word2Vec mà chỉ ánh xạ các từ sang một giá trị thứ tự của từ đó trong file tokenizer.json, đưa kết quả vào cột "Input_Id". File tokenizer.json chứa giá trị “nhúng tọa độ” tương ứng đã được tính toán sẵn.
3) Tạo cột "Attention_Mask" để gán nhãn Attention cho mô hình, 1 là quan tâm, 0 là mặc kệ, dùng cho trường hợp mình cần thêm padding vào các câu ngắn cho độ dài đồng đều.
4) Tạo cột "Labels" chứa phiên bản đã mã hóa (Tương tự như bước 2) của câu mục tiêu, ở đây là câu tóm tắt trong cột “Summary”.

```
>>> print (Tokenized_TextDataset["test"][0]['input_ids'][0:15])
[21603, 10, 180, 3073, 9562, 1300, 13209, 7765, 13044, 11810, 4090,
>>>
>>> print (Tokenized_TextDataset["test"][1]['input_ids'][0:15])
[21603, 10, 466, 48, 1983, 164, 36, 3, 11675, 38, 8, 3, 2, 371, 15]
>>>
```

Ảnh cột input_ids, đầu mỗi câu đều gán chữ ”Summarize” như bước 1 nên mỗi câu đều có giá trị input_ids đầu tiên là 21603

```
87430        "conceived",
87431        -12.659772872924805
87432      ],
87433      [
87434        "▁summarize",
87435        -12.65977382659912
87436      ],
```

Ảnh file tokenizer.json minh họa cho bước 2,
giá trị ”summarize” này cũng nằm ở vị trí thứ 21603 trong mảng.

6. Metric

Rouge score

Là nhóm các độ đo được dùng để đánh giá chất lượng của text summarization. Cách mà Rouge score hoạt động là so sánh văn bản tóm tắt được tạo ra và văn bản tóm tắt tham chiếu.

a. Rouge-N:

Đo lường **tần suất xuất hiện của n-gram** (cụm từ n từ liên tiếp) trong tóm tắt máy so với tóm tắt tham chiếu. Trong rouge-n, n có thể là 1 hoặc 2.

Ví dụ: “I love her very much”. Rouge-1: (I) , (love), (her), (very), (much).Rouge-2: (I love), (love her), (her very), (very much)

Cách tính cụ thể của Rouge 1 và Rouge 2:

- Machine result: “the cat is on the mat”.
- Reference result: “the cat sat on the red mat”.

Rouge-1:

1) Liệt kê các unigrams:

- Machine result: ["the", "cat", "is", "on", "the", "mat"] => 6
- Reference result: ["the", "cat", "sat", "on", "the", “red”, "mat"] => 7

2) Đếm số lượng unigrams phù hợp giữ machine với reference:

["the", "cat", "on", "the", "mat"] => 5

3) Tính precision (P), recall (R) và f1-score (F1)

- P = số unigrams phù hợp / tổng số unigrams trong machine result.
- P = 5/6 = 0.833.
- R = số unigrams phù hợp / tổng số unigrams trong reference result.
- R = 5/7 = 0.714.

- F1 = (2xPxR) / (P+R).
- F1 = (2x0.833x0.714)/(0.833+0.714) = 0.769

Rouge-2:

1) Liệt kê các unigrams:

- Machine result: ["the cat", "cat is", "is on", "on the", "the mat"] => 5
- Reference result: ["the cat", "cat sat", "sat on", "on the", "the red", “red mat”] => 6

2) Đếm số lượng unigrams phù hợp giữa machine với reference:

["the cat", "on the"] => 2

3) Tính precision (P), recall (R) và f1-score (F1)

- P = số unigrams phù hợp / tổng số unigrams trong machine result.
- P = 2/5 = 0.4
- R = số unigrams phù hợp / tổng số unigrams trong reference result.
- R = 2/6 = 0.333
- F1 = (2xPxR) / (P+R).
- F1 = (2x0.4x0.333)/(0.4+0.333) = 0.364

b. Rouge-L:

Đo lường **chiều dài chuỗi con chung dài nhất** (Longest Common Subsequence - LCS) giữa tóm tắt máy và tóm tắt tham chiếu. Thay vì chỉ quan tâm đến các n-gram cụ thể, ROUGE-L xem xét các chuỗi con dài nhất mà cả hai câu cùng có và đánh giá mức độ trùng khớp về trật tự từ.

Tính 3 chỉ số: Precision (P), Recall (R), F1-score (F1)

- P: tỷ lệ độ dài của LCS với tổng số từ trong machine result.
    - P = |LCS| / length of machine result.
- R: tỷ lệ độ dài của LCS với tổng số từ trong reference result.
    - R = |LCS| / length of reference result.
- F1 = (2xPxR) / (P+R)

Ví dụ:

- Machine result: “the cute cat is on the mat”. => 7.
- Reference result: “the cute cat sat on the red mat”. => 8

1) Tìm chuỗi con giống nhau dài nhất:

“the cute cat” => len = 3.

2) Tính P, R, F1:

- P = 3/7 = 0.429
- R = 3/8 = 0.375
- F1 = (2xPxR) / (P+R) = 0.281

c. Rouge-S:

Đo **dựa trên các cặp từ** (bigram) mà không yêu cầu các từ trong cặp phải liền kề nhau nhưng vẫn giữ nguyên thứ tự xuất hiện. ROUGE-S đánh giá khả năng của tóm tắt máy trong việc bảo tồn các mối quan hệ giữa các từ trong tóm tắt tham chiếu.

Tính 3 chỉ số: P, R, F1

- Precision (P): Tỷ lệ skip-bigram phù hợp so với tổng số skip-bigram trong tóm tắt máy.
- Recall (R): Tỷ lệ skip-bigram phù hợp so với tổng số skip-bigram trong tóm tắt tham chiếu.
- F1 = (2xPxR) / (P+R)

Ví dụ:

- Machine result : "the cat is on the mat"
- Reference result: "the cat sat on the red mat"

1) Liệt kê các skip-bigrams trong machine result và reference result.

- Skip-bigrams trong machine result: "the cat", "the is", "the on","the the", "the mat", "cat is", "cat on", “cat the”, "cat mat", "is on", “is the”, "is mat", "on the", "on mat", “the mat” => 15
- Skip-bigrams trong tóm tắt tham chiếu: "the cat", "the sat", "the on", “the the”, "the red", "the mat", "cat sat", "cat on", “cat the”, "cat red", "cat mat", "sat on", “sat the”, "sat red", "sat mat", "on the", "on red", "on mat", "the red", "the mat", "red mat" => 21

2) So sánh các skip-bigrams trong tóm tắt máy với các skip-bigrams trong tóm tắt tham chiếu để tìm các cặp trùng khớp:

- "the cat", "the on", "the the", "the mat", "cat on", "cat the", "cat mat", "on the", "on mat”. => 9

3) Tính Precision, Recall và F1-Score

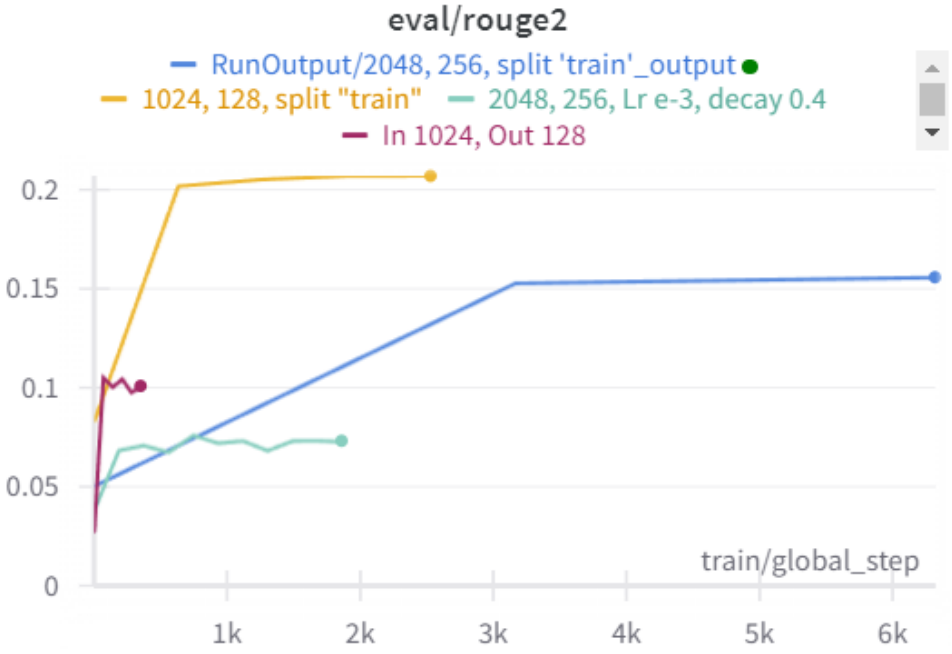
- P = 9/15 = 0.6
- R = 9/21 = 0.429
- F1 = (2x0.6x0.429) / (0.6+0.429) = 0.5

7. Quá trình huấn luyện

Cấu hình máy tính: Cpu Ryzen 5 3600, 64Gb Ram, Gpu Nvidia Tesla P40 24Gb, dataset được đặt trong ssd.

Cấu hình môi trường: Sử dụng Python 3.11, Pytorch Cuda để chạy trên gpu, thư viện Transformer của HuggingFace giúp quá trình huấn luyện và kiểm thử mượt mà hơn, thư viện rouge_score để tính các phép đo Rouge, wandb để ghi chép lại quá trình huấn luyện.

Đối với dataset BillSum, với một số siêu tham số cần thử nghiệm nhiều, nhóm đã sử dụng subset "Test" nhỏ hơn gồm 3270 dòng để giảm thời gian huấn luyện nhằm tìm ra khoảng các siêu tham số tối ưu để tinh chỉnh trên toàn bộ tập dữ liệu, tránh mất thời gian chờ mô hình huấn luyện. (Subset 3270 dòng đấy lại được chia làm 80% để train và 20% để test).



Ảnh kết quả huấn luyện với 2 cấu hình mô hình khác nhau trên 2 tập dữ liệu khác nhau, màu đỏ và cam là Input 1024 Output 128, hai màu xanh dương đều là Input 2048 Output 256. Hai đường dài cam và xanh dương đậm tương ứng 2 cấu trúc mô hình khác nhau, dài hơn vì chúng được huấn luyện trên tập dữ liệu lớn hơn.

Có thể thấy việc sử dụng một tập con của tập chính BillSum vẫn cho ra được sự chênh lệch giữa hai mô hình dù Rouge Score của hai mô hình đó có sự khác nhau vì kích thước tập dữ liệu, nhưng khi xét về tỉ lệ thì ở cả hai kích thước tập dữ liệu, mô hình Input 1024 Output 128 đều có mức cải thiện 32% ở tập lớn và 37% ở tập nhỏ so với mô hình Input 2048 Output 256.

Để dễ trình bày thì một số thử nghiệm nhóm chỉ chụp ảnh metric Rouge 2 để làm biểu đồ đại diện vì trong mọi bài thử thì các chỉ số Rouge khác cũng có sự tương quan lẫn nhau. Đồng thời mọi lần huấn luyện đều chạy kiểm thử một lần trước khi bắt đầu huấn luyện để có mốc so sánh để xem sự cải thiện của mô hình. Sau đây là các thử nghiệm để tìm các siêu tham số tối ưu:

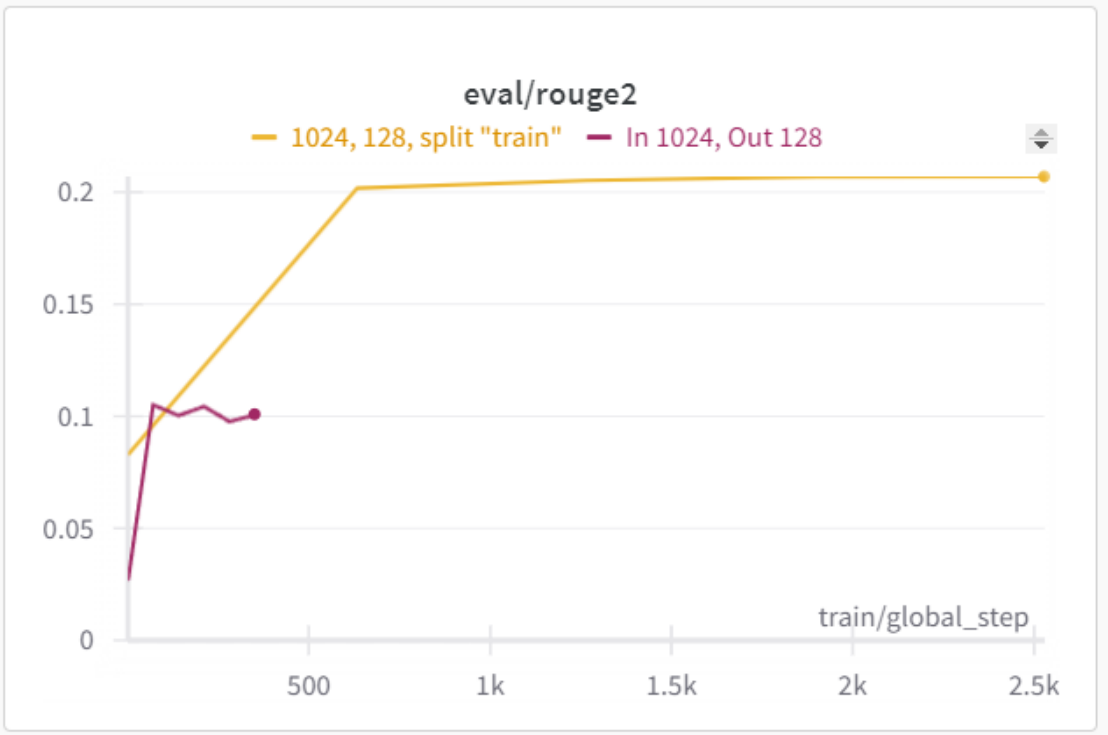
1) Thử nghiệm về số Epoch.

Với tập con “Test” của dataset BillSum, chỉ cần 5 epoch thì kết quả mô hình gần chấp nhận được, 6 epoch thì chênh lệch Rouge Score rất ít so với 40 epoch (cải thiện được 1.5% khi xét Rouge L, 6% khi xét Rouge 2). Xét về giới hạn phần cứng và thời gian, nhóm đi đến kết luận dừng huấn luyện mô hình sau 6 epoch.

Ảnh chụp kết quả huấn luyện trên dataset BillSum, subset “test”,

Kết quả tại epoch thứ 6 được khoanh trong ảnh.

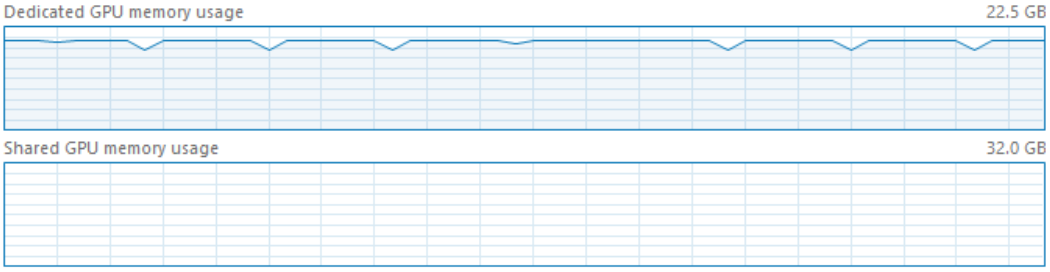
Đối với toàn bộ dataset BillSum, vì kích thước của nó lớn gấp 7 lần so với lấy mỗi test set nên quá trình huấn luyện chỉ cần 1 epoch là mô hình đã gần đạt giới hạn, những epoch sau không cải thiện rouge score nhiều.



Ảnh chụp kết quả huấn luyện.
Đường màu cam được huấn luyện trên toàn bộ dataset BillSum.
Đường màu đỏ là được huấn luyện trên subset "Test".

2) Lựa chọn Batch Size.

Khi chạy mô hình cần lựa chọn batch size đủ lớn để tận dụng khả năng tính toán song song của gpu, batch size quá nhỏ thì gpu dành nhiều thời gian rảnh hơn nên không tối ưu được thời gian chạy.



Ảnh chụp task manager báo thông số sử dụng ram của Gpu

Nhưng nếu batch size quá lớn sẽ dẫn đến việc tràn ram Gpu (Dedicated Gpu memory usage kéo hết 100%), lúc đó gpu sẽ chuyển qua dùng thêm ram máy tính (Shared Gpu memory usage) khiến cho thời gian huấn luyện chậm khủng khiếp.

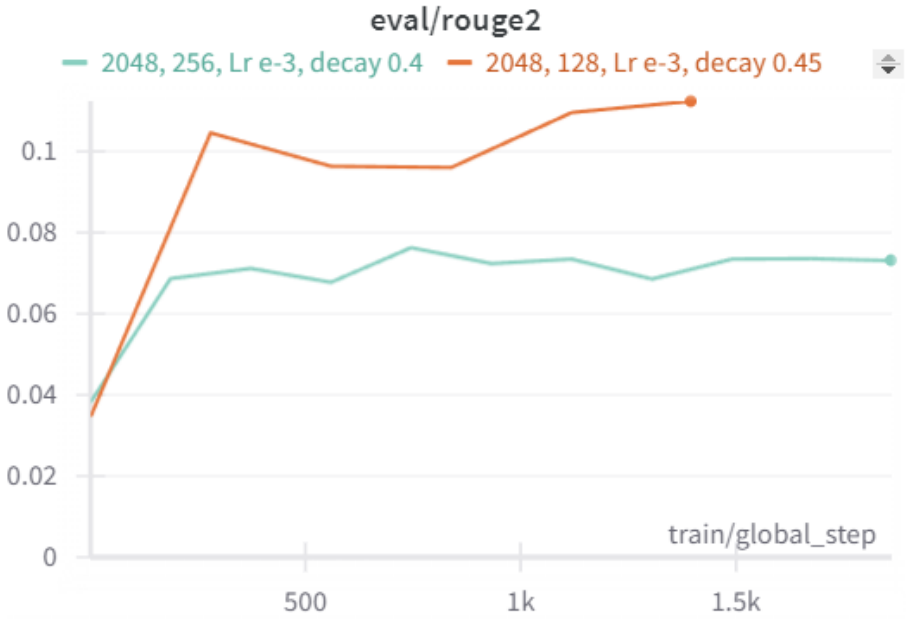
3) Thử nghiệm độ dài Input - Output.

Nhóm cố định kích thước output tối đa 128 token và thử nghiệm với kích thước input khác nhau giữa 512, 1024, 2048, riêng với input 2048 phải hạ batch_size vì tràn vram gpu, các tham số khác vẫn giữ nguyên.

Kết quả cho thấy đối với output giới hạn tối đa 128 token thì input 1024 là điểm phù hợp nhất vì 2048 không giúp cải thiện các chỉ số Rouge 1, 2, L, Lsum đáng kể mà model tốn thời gian train hơn hẳn, và input 1024 có chỉ số Rouge tốt hơn input 512.

Kết quả huấn luyện giữa các kích thước input khác nhau, màu nâu là 512, đỏ là 1024, tím là 2048.

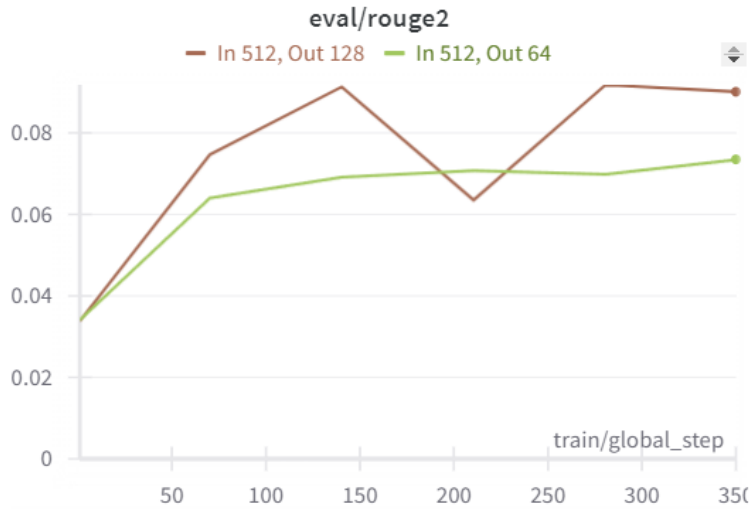
Nhóm cũng đã thử nghiệm với kích thước output 256 với các kích thước Input khác nhau như 1024, 2048, 3072 nhưng vì giới hạn phần cứng và thời gian chạy nên đã không thử với toàn bộ dataset BillSum mà chỉ huấn luyện trên subset “Test”, kết quả cũng rất kém so với output 128 (Các chỉ số Rouge của Input 2048 Output 256 chỉ bằng xấp xỉ 60% so với chỉ số Rouge của Input 2048 Output 128, đẩy Input lên 3072 cũng không cải thiện được chút nào). Nhóm nghi ngờ là do giới hạn của mô hình T5-Small hoặc kích thước bộ dữ liệu.



Ảnh so sánh giữa Output 256 Token màu xanh và Output 128 token màu đỏ, với kích thước input cố định.

Kết luận hiện tại là: Đối với Output 128 token thì Input phù hợp nhất là 1024 token, đối với Output 256 token thì không có kích thước input nào phù hợp để giúp chỉ số Rouge lên mức chấp nhận được (trên 0.1).

Vậy đối với Output ít hơn như 64 token thì sao? Nhóm cũng đã thử nghiệm và thấy kết quả Rouge Score cũng thấp không kém thử nghiệm Output 256 Token ở trên, có lẽ vì không gian Output bị giới hạn nên không thể trình bày đủ tóm tắt.



Ảnh so sánh giữa Output 64 token màu xanh và Output 128 token màu đỏ, với kích thước Input cố định.

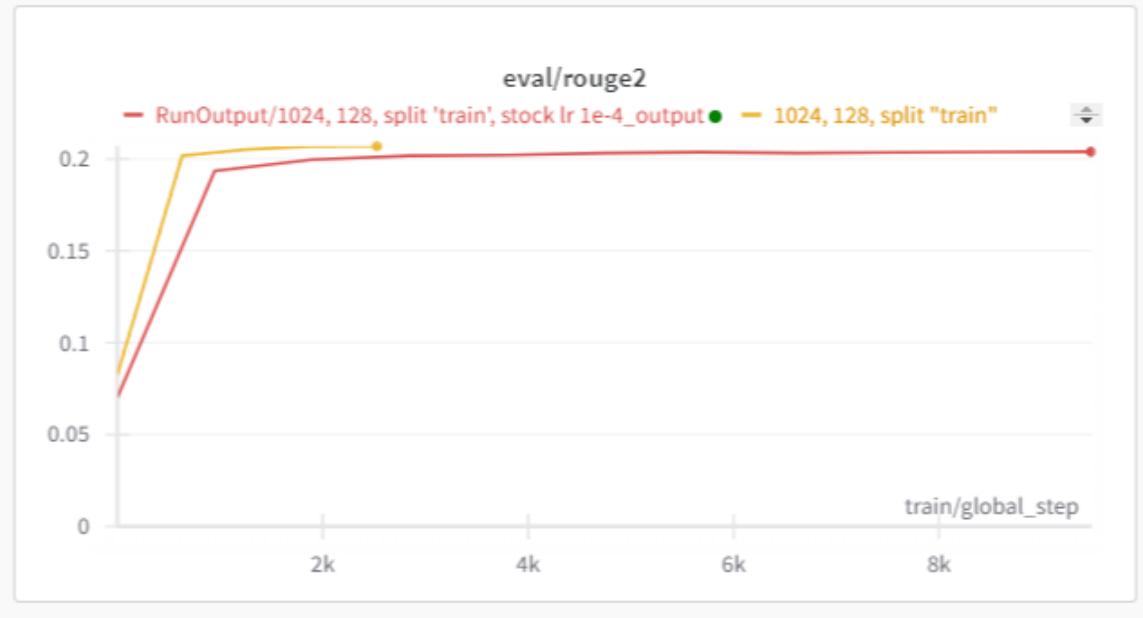
Nhóm chốt lựa chọn mô hình có kích thước Input 1024 và Output 128 để có sự cân bằng giữa Rouge Score và thời gian huấn luyện.

4) Thử nghiệm Learning Rate.

Được thử nghiệm trên toàn bộ bộ dữ liệu BillSum, vì chỉ cần một epoch là mô hình đã đạt được hiệu quả tiệm cận giới hạn nên nhóm muốn thử nghiệm so sánh 2 cách học khác nhau:

1) Learning rate 0.0001 và weight_decay 0.01 như code gốc trên HuggingFace, huấn luyện với 10 epoch.
2) Learning rate 0.001 và weight_decay 0.4 với ý tưởng là khởi đầu nhanh với learning rate cao và mau chóng ổn định trọng số với weight decay [6], huấn luyện với 3 epoch.

Kết quả thực nghiệm cho thấy chỉnh learning rate cao và weight_decay cao cho ra kết quả khả quan hơn, cách 1 dù cho huấn luyện tới 10 epochs nhưng Rouge score chỉ cải thiện 0.9% so với 3 epochs và không vượt được cách 2.



Ảnh chụp so sánh kết quả huấn luyện với learning rate khác nhau. Màu đỏ là learning rate gần như cố định (cách 1), màu vàng là learning rate giảm nhanh (cách 2).

Kết luận cuối cùng: Nhóm chọn mô hình T5-Small với kích thước đầu vào tối đa là 1024, đầu ra tối đa 128, huấn luyện trên toàn bộ dataset BillSum với Learning rate 0.001 và Weight Decay 0.4 và 2 epoch.

8. Kết quả huấn luyện - Kết luận

Nhóm đã đạt kết quả khả quan trên tập Test của dataset BillSum với Rouge-1 0.4927, Rouge-2 0.3231, Rouge-L 0.3880, chỉ số này rất phù hợp với giới hạn của mô hình vì trong báo cáo của Google đã huấn luyện nó trên dataset CNN_DailyMail và đạt được chỉ số Rouge tương tự (Rouge 1, 2, L xấp xỉ 0.4, 0.2, 0.4) [1].

Thử nghiệm huấn luyện trên mỗi subset "Test" cũng cho ra mô hình có kết quả tương đồng với những cá nhân khác mà đã chạy T5 trên subset "Test" giống nhóm (Rouge xấp xỉ 0.2). Nhóm cũng đã thử tách tập Train ra 90% để train và 10% để test, chỉ dùng tập "Test" để kiểm thử cũng cho Rouge score tương đồng. Cho thấy phương pháp chạy của nhóm không gặp vấn đề gì và kết quả đáng tin cậy.

Mô hình cũng có tốc độ dự đoán khá nhanh với thời gian trung bình cho một câu là 0.4 giây trên gpu. Thời gian huấn luyện (với cấu trúc được lựa chọn ở trên) là 20 phút cho 1 epoch, kích thước input lớn hơn sẽ tốn thời gian và bộ nhớ hơn.

```
Number of sample:  3269 , Batch size:  40
model.generate Total time taken:  1249.8160018920898 , Average time for 1 sample:  0.38232364695383597
Average ROUGE-1: 0.4927
Average ROUGE-2: 0.3231
Average ROUGE-L: 0.3880
Total time taken:  1271.3403809070587
```

Ảnh Rouge Score sau khi kiểm thử trên tập Test

Để minh họa output của mô hình trong tập test, ở dưới là câu input, câu mà mô hình dự đoán và câu đáp án của một điểm dữ liệu trong dataset:

Input text:

SECTION 1. SHORT TITLE. This Act may be cited as the Special Agent Scott K. Carey Public Safety Officer Benefits Enhancement Act''. TITLE I--EDUCATIONAL ASSISTANCE TO OFFICERS DISABLED IN THE LINE OF DUTY SEC. 101. BASIC ELIGIBILITY. Section 1212(a)(1) of the Omnibus Crime Control and Safe Streets Act of 1968 (42 U.S.C. 3796d-1(a)(1)) is amended--

Predicted summary:

Special Agent Scott K. Carey Public Safety Officer Benefits Enhancement Act - Amends the Omnibus Crime Control and Safe Streets Act of 1968 to revise the definition of "eligible dependent" to include any public safety officer who dies in the line of duty on or after January 1, 1978, or becomes permanently and totally disabled as the direct result of a catastrophic injury sustained in the line of duty on or after January 1, 1978. Requires the Bureau of Investigation (BIA) to pay an annual pension to one or more survivors of the deceased public safety officer. Requires

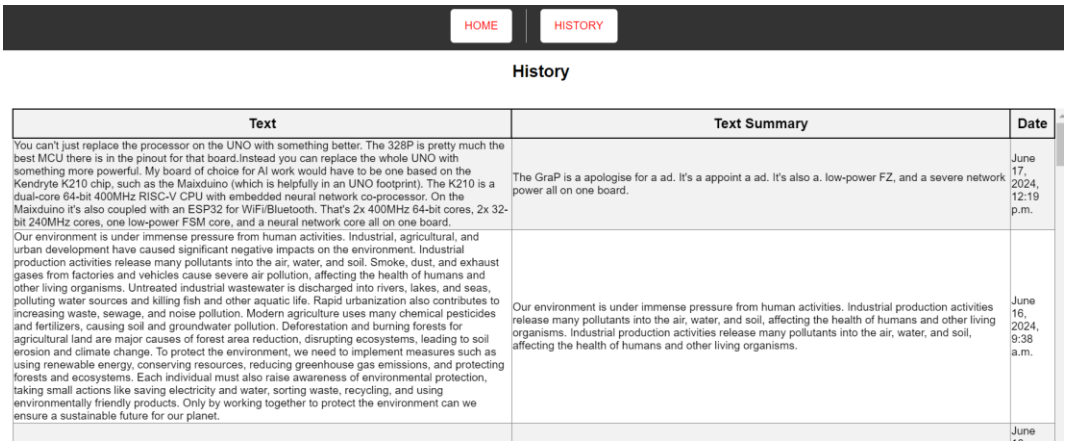
True summary:

Special Agent Scott K. Carey Public Safety Officer Benefits Enhancement Act - Amends the Omnibus Crime Control and Safe Streets Act of 1968 to extend: (1) educational benefits to public safety officers who become permanently and totally disabled in the line of duty and to their spouses and children; (2) allow payment of retroactive benefits to dependents of such disabled officers; and (3) establish a program of pension payments for certain survivors of deceased public safety officers. Authorizes the Secretary of Education to: (1) award a Public Safety Officer scholarship to disabled public safety officers, their spouses, and their children; and (2)

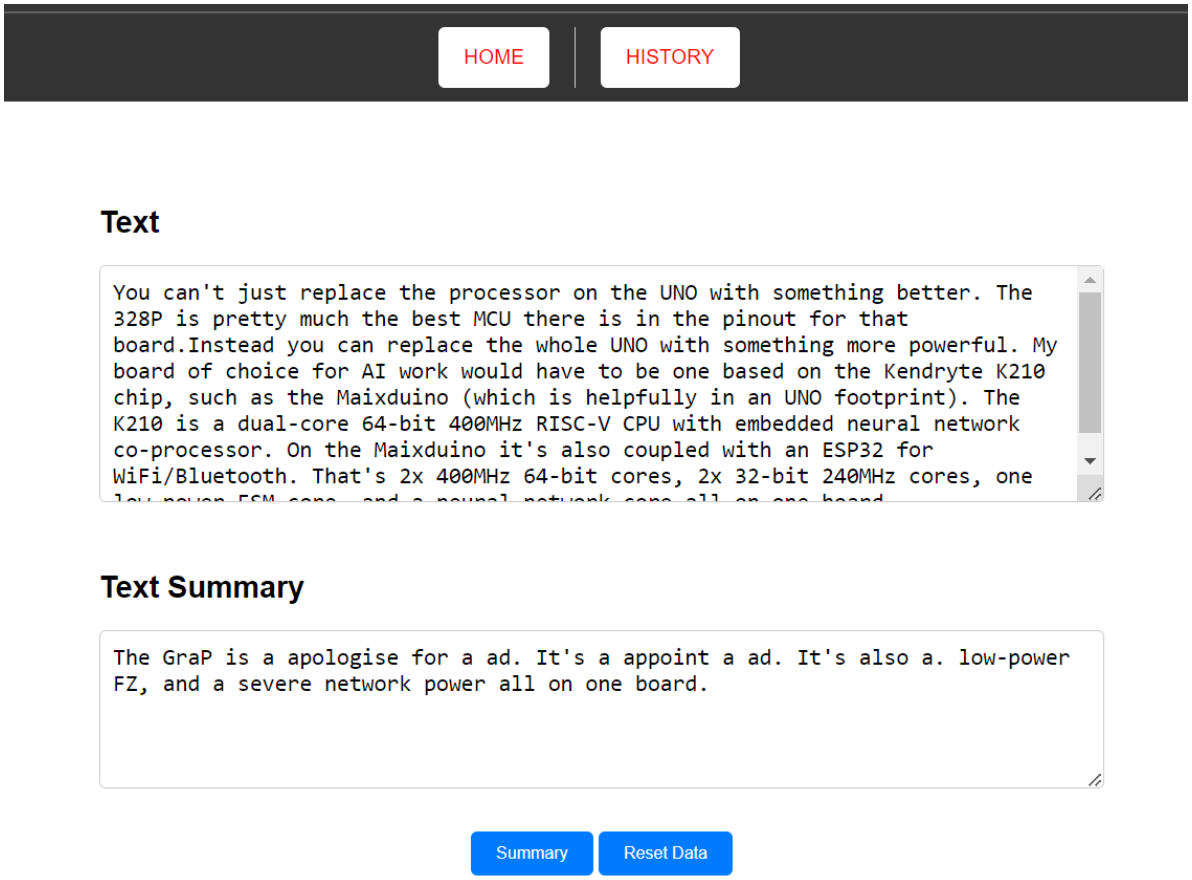
9. Minh họa chương trình

Hướng dẫn cài đặt: Có trong file readme.md của thư mục code.

Lịch sử sử dụng, có lưu lại kết quả.



Đưa 1 câu vào chương trình, ấn nút "Summary" sẽ trả về kết quả summary của câu đó, nút "Reset data" giúp làm mới các ô.



10. Tài liệu tham khảo

Code của mô hình đã sử dụng thư viện HuggingFace và cũng được tham khảo từ trang HuggingFace của mô hình[3].

1) Raffel, C., Shazeer, N., Roberts, A., Lee, K., Narang, S., Matena, M., Zhou, Y., Li, W., & Liu, P. J. (2019). Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer. ArXiv.org. https://arxiv.org/abs/1910.10683

2) Roberts, A., Raffel, C. (2020, February 24). Exploring transfer learning with T5: The text-to-text transfer transformer. *Google AI Blog*. https://research.google/blog/exploring-transfer-learning-with-t5-the-text-to-text-transfer-transformer/

3) Hugging Face. (n.d.). T5. *Transformers Documentation*. Retrieved June 19, 2024, from https://huggingface.co/docs/transformers/en/model_doc/t5

4) Kornilova, A., & Eidelman, V. (2019). BillSum: A Corpus for Automatic Summarization of US Legislation. ArXiv.org. https://doi.org/10.18653/v1/D19-5406

5) Hugging Face. (n.d.). FiscalNote/billsum. *Hugging Face Datasets*. Retrieved June 19, 2024, from https://huggingface.co/datasets/FiscalNote/billsum

6) Loshchilov, I., & Hutter, F. (2017). Decoupled Weight Decay Regularization. Arxiv.org. https://arxiv.org/abs/1711.05101