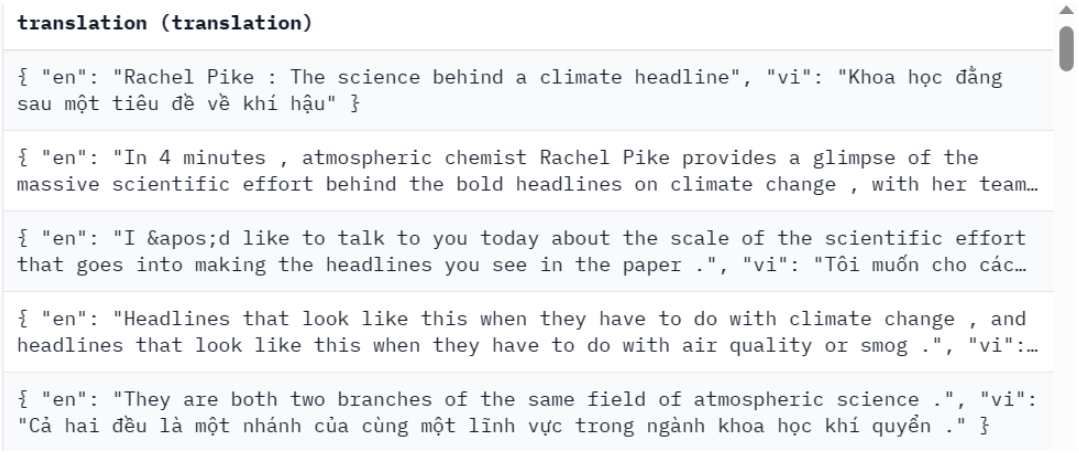
CHƯƠNG 3
XÂY DỰNG MÔ HÌNH

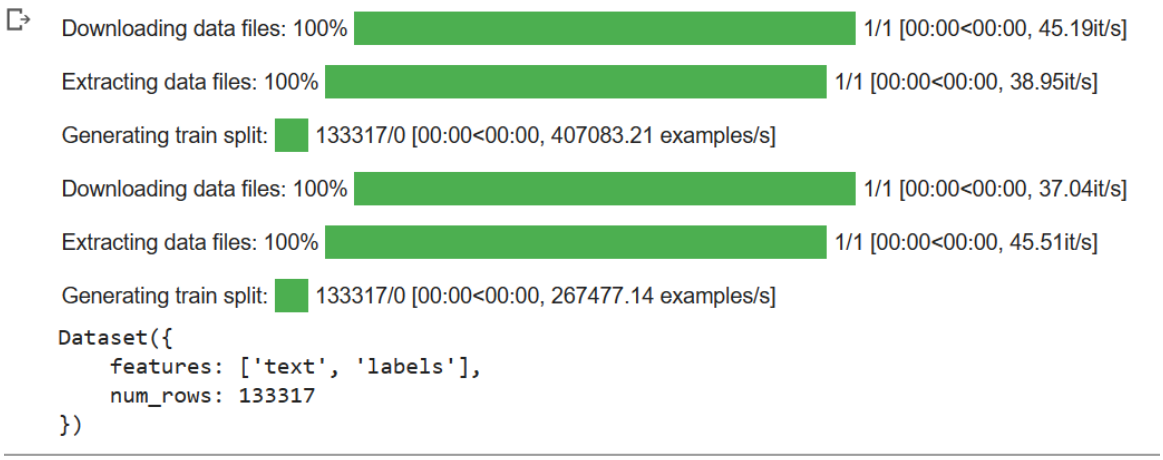
3.1 Chuẩn bị dữ liệu

Bộ dữ liệu được sử dụng ở đây bao gồm bộ data song ngữ Anh - Việt IWSLT 15 dùng để huấn luyện và bộ IWSLT 13 dùng cho giai đoạn kiểm thử thực tế. Dữ liệu có cấu trúc như sau:



Hình 3. 1. *Mô tả dữ liệu IWSLT 15*

Ta sẽ tiến hành load dữ liệu IWSLT 15 để training bằng thư viện **datasets** của **Hugging Face** để việc xử lý dữ liệu trở nên nhanh chóng hơn vì thư viện này sẽ giúp xử lý dữ liệu song song, tận dụng hết hiệu năng của phần cứng. Ta sẽ xây dựng 2 mô hình riêng biệt cho mỗi task (dịch Anh - Việt, dịch Việt - Anh) để tối ưu chi phí phần cứng cũng như đạt hiệu suất tốt nhất. Đầu tiên ta sẽ bắt đầu với mô hình dịch Anh - Việt trước.



Hình 3. 2. *Cấu trúc dữ liệu IWSLT 15*

Ta thấy trên tập IWSLT 15 có **133.317** dòng dữ liệu. Ta xem thử một điểm dữ liệu ở vị trí nhãn tiếng Anh để xem thử.

```
ds["labels"][2]
```

```
'I &apos;d like to talk to you today about the scale of the scientific effort that goes into making the
headlines you see in the paper .'
```

Hình 3. 3. *Mô tả một điểm dữ liệu*

Để ý sẽ thấy những chỗ như **' , "..** đây không phải lỗi mà đây là những ký hiệu dấu nháy và ngoặc được html hoá đi. Vì vậy để chuyển về dạng đúng ta sẽ viết thêm một hàm clean data trên tiếng Anh và tiếng Việt như sau:

```
def preprocess(text):
  text = text.replace('&apos;', "'")
  text = text.replace('&quot;', "'")
  text = text.replace('&#91;', "(")
  text = text.replace('&#93;', ")")
  return text

def clean_data(sample):
  sample["text"] = preprocess(sample["text"])
  sample["labels"] = preprocess(sample["labels"])
  return sample

# Cleaning
ds = ds.map(clean_data)
```

```
Map: 100% 133317/133317 [00:12<00:00, 9846.56 examples/s]
```

Hình 3. 4. *Hàm tiền xử lý dữ liệu*

Kết quả sau khi làm sạch dữ liệu (clean data):

```
ds["labels"][2]
```

```
'I 'd like to talk to you today about the scale of the scientific effort that goes into making the head lines you see in the paper .'
```

Hình 3. 5. *Dữ liệu sau khi đã tiền xử lý*

Sau khi đã làm sạch dữ liệu ta sẽ tiến hành vector hoá (tokenizer) dữ liệu này thành dạng số để có thể đưa vào mô hình training. Để tiện, ta sẽ dùng bộ tokenizer của model **facebook/bart-large** do mô hình này đã được training trước đó trên data tiếng Anh và tiếng Việt nên bộ từ điển tokenizer này đã có sẵn các từ vựng Anh Việt rồi, ta không cần tạo lại một bộ tokenizer mới nữa.

```
[ ] from transformers import AutoTokenizer, AutoModelForCausalLM

    tokenizer = AutoTokenizer.from_pretrained("facebook/bart-large")
```

Hình 3. 6. *Load bộ tokenizer của BART từ HuggingFace*

Số lượng ký tự tối đa **max_length** ở cả 2 đầu dịch thuật là **128 ký tự.** Tham số **truncation = True** để tự cắt ngắn khi tổng ký tự đoạn văn vượt qua số lượng này.

```
max_input_length = 128
max_target_length = 128

def tokeni(examples):
    #Anh -> Viet
    model_inputs = tokenizer(examples["labels"],
                             text_target=examples["text"],
                             max_length=128,
                             truncation=True)
    #Viet -> Anh
    # model_inputs = tokenizer(examples["text"],
    #                          text_target=examples["labels"],
    #                          max_length=128,
    #                          truncation=True)
    return model_inputs
```

Hình 3. 7. *Hàm vector hóa dữ liệu chuỗi qua tokenizer*

Tiến hành tokenizing tập dữ liệu.

```
] tokenized_datasets = ds.map(tokeni,batched=True)
```

Map: 100% 133317/133317 [00:33<00:00, 4455.40 examples/s]

Hình 3. 8. *Vector hoá dữ liệu (Tokenizing)*

Sau khi đã vector hóa dữ liệu xong ta có thể lưu lại bộ data đã xử lý này để có thể tiện cho việc sử dụng về sau. Ở đây ta sẽ dùng hugging face để làm nơi lưu trữ.

```
[ ] # tokenized_datasets.push_to_hub("phatjk/ds_translate_vi_en_bart")
    tokenized_datasets.push_to_hub("phatjk/ds_translate_en_vi_bart")
```

Pushing dataset shards to the dataset hub: 100% 1/1 [00:07<00:00

Creating parquet from Arrow format: 100% 134/134 [00:00<00:00,

Hình 3. 9. *Đưa data đã làm sạch lên cloud HuggingFace*

3.2 Thiết lập thông số mô hình

Ta sẽ tiến hành load model **facebook/bart-large** về để sử dụng. Mô hình này nặng 1,6 GB với số lượng **406.291.456 parameter**. Các siêu tham số sẽ được thiết lập như sau:

Hyper Parameters	Values
batch-size	25
learning-rate	2e-5 (0.00002)
epochs	1
weight_decay	0.01

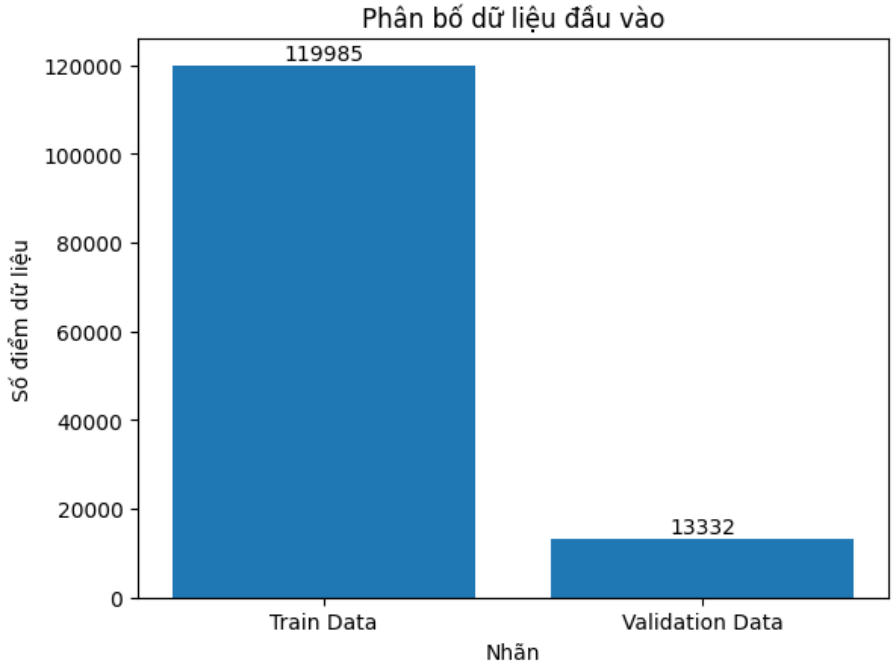
Bảng 3. 1. *Siêu tham số được cài đặt cho BART*

Với cấu hình GPU T4 của Google Colab ta sẽ đặt **batch-size = 25** để việc training tận dụng được tối đa sức mạnh phần cứng (14.5GB / 15GB GPU) cũng như khả năng hội tụ về target nhanh hơn. Tham số **learning-rate = 2e-5** theo kinh nghiệm sẽ phù hợp với

mạng này. Số lượng **epoch = 1** do thời gian huấn luyện trung bình khoảng 1 giờ/1 epoch do đó để tiện thời gian treo máy ta sẽ train theo từng đợt, tức là khi train xong sẽ save lại mô hình rồi sẽ lấy ra train tiếp tục ở một thời điểm khác. Cuối cùng tham số **weight-decay = 0.01** dùng để tránh việc mô hình quá nhạy dẫn đến overfitting.

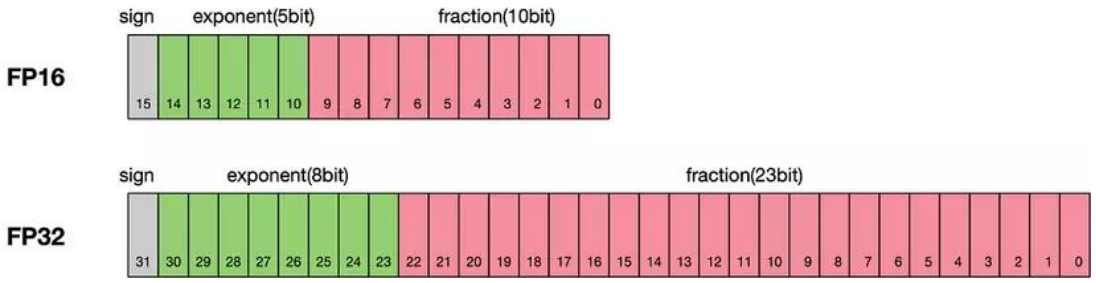
3.3 Chia dữ liệu

Tập dữ liệu trên ta sẽ cắt ra làm 2 phần với **tỉ lệ 90/10** với **90% cho huấn luyện** và **10% cho validation.** Việc chia tỉ lệ 9/1 thường được ưa chuộng trong NLP do ngôn ngữ thường có tính phức tạp, giúp tránh overfitting do dữ liệu đang có ít. Và để dễ so sánh với bài báo ở trên, ta sẽ có thêm một bước cuối là tính và đánh giá chỉ số BLEU Score trên tập IWSLT 13.



Hình 3. 10. *Mô tả phân bố dữ liệu đầu vào*

3.4 Lượng tử hoá mô hình (Model Quantization)



Hình 3. 11. *Sự khác biệt giữa 16 bit và 32 bit*

Ngoài ra để tăng hiệu năng của mô hình ta sẽ đặt **fp16=True** để trọng số được lưu trữ ở dạng **16 bit** thay vì **32 bit** theo mặc định, việc này giúp đỡ tốn không gian lưu trữ và tăng tốc độ xử lý của mô hình, tuy nhiên đồng thời cũng sẽ làm giảm độ chính xác của nó.

```
batch_size = 25
model_name = "bart-translate"
args = Seq2SeqTrainingArguments(
    f"{model_name}-finetuned",
    evaluation_strategy = "epoch",
    learning_rate=2e-5,
    per_device_train_batch_size=batch_size,
    per_device_eval_batch_size=batch_size,
    weight_decay=0.01,
    num_train_epochs=1,
    fp16=True,
    report_to="wandb",
    run_name="bart-en-vi"
)
```

Hình 3. 12. *Thiết đặt Arguments cho quá trình huấn luyện*

3.5 Thiết lập độ đo và huấn luyện mô hình

Cuối cùng để có thể ghi lại chi tiết các thông số diễn ra trong quá trình training ta sẽ set **report-to=wandb,** thì Weights & Biases (wandb) đây là một công cụ hữu ích giúp ta có thể tự động ghi nhận và trực quan số liệu thành các biểu đồ như thông số ram, GPU, CPU, loss, accuracy, … thành một bảng báo cáo hoàn chỉnh (tương tự TensorBoard), giúp ta có thể giám sát quá trình huấn luyện mô hình một cách trực quan hơn.

Đối với những task dịch thuật thì ta sẽ sử dụng chỉ số đánh giá BLEU Score để đánh giá độ tốt của đầu ra mô hình. BLEU Score càng cao sẽ càng tốt.

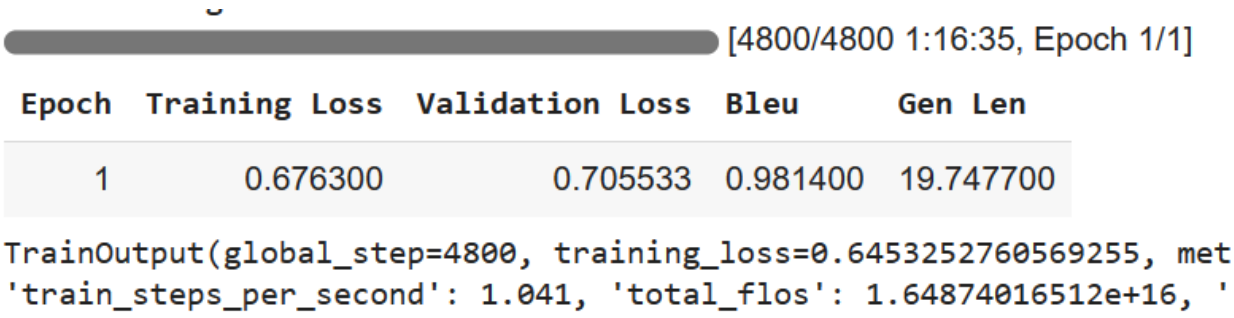
```
[ ] trainer = Seq2SeqTrainer(
        model,
        args,
        train_dataset=dataset_train,
        eval_dataset=dataset_test,
        data_collator=data_collator,
        tokenizer=tokenizer,
        compute_metrics=compute_metrics
    )
```

```
[ ] trainer.train()
```

Hình 3. 13. *Thiết đặt training qua hàm Trainer*

Hàm tính thông số BLEU sẽ được tích hợp vào hàm **compute_metrics** được tích hợp từ thư viện **sacrebleu.** Vậy là chúng ta đã setup xong mô hình BART cho bài toán dịch Anh - Việt, với bài toán dịch Việt - Anh ta cũng làm tương tự chỉ cần hoán đổi đầu vào và nhãn ở dataset là được. Toàn bộ mã nguồn sẽ được đính kèm cuối tiểu luận.

Tiến hành train mô hình



Hình 3. 14. *Tiến hành huấn luyện (training)*

CHƯƠNG 4
THỰC NGHIỆM VÀ ĐÁNH GIÁ

4.1 Kết quả thực nghiệm

Ta thấy trung bình mô hình sẽ training 1 tiếng/1 epoch nên để tối ưu thời gian, chiến lược sẽ là ta sẽ train mỗi epoch rồi lưu lại sau đó sẽ load ra train tiếp và sẽ ghi nhận kết quả lại từng đợt train. Ta sẽ có kết quả như sau. Lưu ý chỉ số **BLEU (validation)** dưới đây được tính trên tập **validation** được cắt ra từ 10% của bộ dữ liệu, nên không có giá trị so sánh với paper trên vì các paper đo chỉ số BLEU trên tập IWSLT 13 tức **BLEU (IWSLT 13)**.

Mô hình dịch Anh - Việt:

Chỉ số	Epoch 1	Epoch 2	Epoch 3
Training Loss	0.929000	0.756300	0.676300
Validation Loss	0.919379	0.765422	0.705533
BLEU (validation)	0.777300	0.911900	0.981400
BLEU (IWSLT 13)	-	-	16.25

Bảng 4. 1. *Tóm tắt kết quả mô hình BART Anh - Việt*

Nhận xét: Qua quá trình training ta thấy sai số (loss) trên cả 2 tập training và validation giảm đều qua 3 epochs nên có thể kết luận là mô hình không bị học quá khớp overfitting. Tuy nhiên chỉ số BLEU trên tập validation lại khá nhỏ so với BLEU trên tập IWSLT 13.

Mô hình dịch Việt - Anh:

Chỉ số	Epoch 1	Epoch 2	Epoch 3
Training Loss	1.948300	1.803700	1.704700
Validation Loss	1.966773	1.850982	1.820606
BLEU (validation)	11.39620	12.822500	13.328700
BLEU (IWSLT 13)	-	-	25.78

Bảng 4. 2. *Tóm tắt kết quả mô hình BART Việt – Anh*

Nhận xét: Tương tự ở mô hình dịch chiều ngược lại sai số (loss) trên cả 2 tập training và validation giảm đều qua 3 epochs nên mô hình cũng không bị học quá khớp overfitting. Tuy nhiên chỉ số BLEU trên tập validation có sự cải thiện đáng kể là **13.3** cũng như BLEU trên tập IWSLT 13 cũng tăng lên tận **25.78.** Ta thấy mô hình này nếu dùng để dịch Việt sang Anh sẽ hiệu quả hơn dịch từ Anh sang Việt.

4.2 So sánh với paper và đánh giá kết quả

Model	# Params	Pretrained	Finetuned		En-Vi	Vi-En
			Dataset	# pairs		
M2M100	1.2B	-	CCMatrix + CCAligned	7.5B	35.83	31.15
Google Translate	-	-	-		39.86	35.76
Bing Translator	-	-	-		40.37	35.74
Transformer-base	65M	-	PhoMT	3M	42.12	37.19
Transformer-big	213M	-	PhoMT	3M	42.94	37.83
mBART†	448M	CC25	PhoMT	3M	43.46	39.78
EnViT5-base	275M	CC100	MTet	4.2M	43.87	39.57
			MTet + PhoMT	6.2M	45.47	40.57
Our BART	406M	-	IWSLT 15	133K	**16.25**	**25.78**

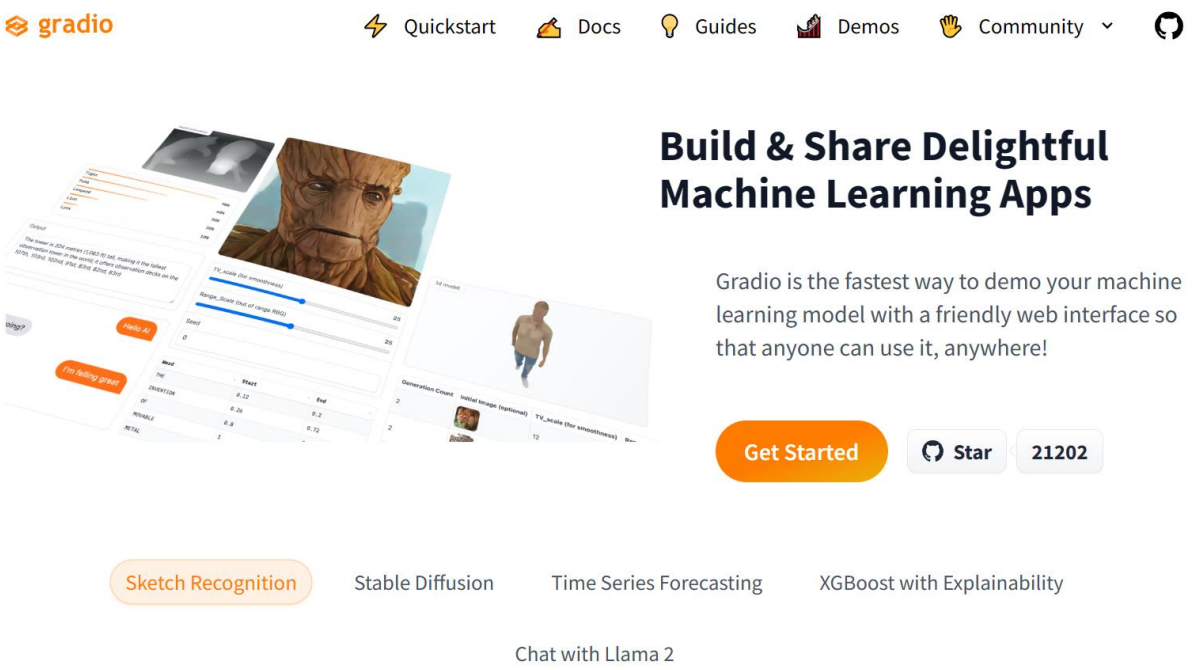
Bảng 4. 3. *Results on PhoMT English-Vietnamese Translation Test Set [4]*

Đánh giá: Ta thấy rằng chỉ số BLEU của mô hình BART chúng ta đào tạo qua 3 epochs trên model En-Vi đạt **16.25** và model Vi-En đạt **25.78.** Rõ ràng kém xa hơn so với những mô hình trên mà bài báo đề cập cũng như mô hình cốt lõi EnViT5-base mà bài báo giới thiệu. Lý do có thể kể đến đó là dữ liệu huấn luyện quá ít và số lượng epoch được train chỉ có 3 ở mỗi mô hình. Tổng cộng thời gian huấn luyện của cả 2 mô hình từ 6-7 giờ cho tất cả 6 epochs. Vì chi phí tính toán phần cứng và thời gian có hạn nên đề tài chưa thể đạt được số điểm như kỳ vọng. Tuy nhiên nếu được đầu tư phát triển có thể sẽ đạt được kết quả tốt hơn trong tương lai.

4.3 Xây dựng giao diện sử dụng

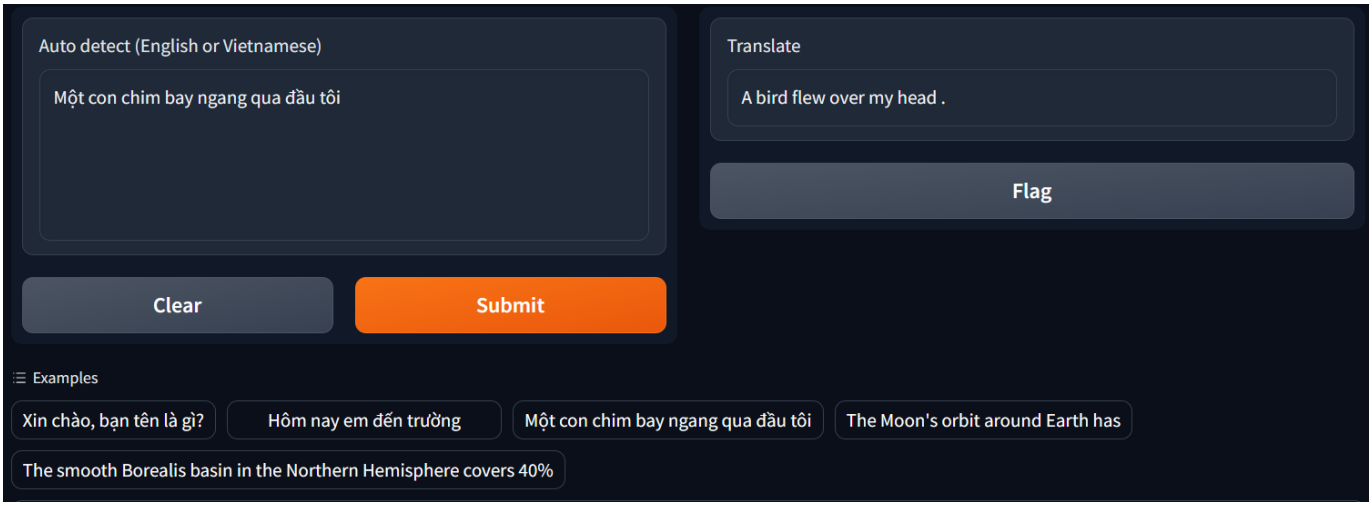
Để có thể triển khai mô hình lên giao diện web và public link cho mọi người sử dụng phục vụ việc dịch thuật ta sẽ tiến hành như sau:

Ta sẽ tận dụng **GPU T4** miễn phí của Google Colab làm máy chủ chính và sẽ port local url từ server nội bộ ra internet bằng một đường link có thể truy cập được từ bên ngoài. Thông thường ta sẽ dùng **Flask** làm backend rồi port **API** ra thông qua **ngrok.** Tuy nhiên việc này sẽ tốn nhiều thời gian và gây phức tạp một cách không cần thiết, vì vậy ta sẽ sử dụng một thư viện mang tên **Gradio** để xử lý vấn đề này. Đây là một thư viện hữu ích giúp tạo nhanh một giao diện sử dụng cho hầu hết các task phổ biến trong học máy và học sâu. Đồng thời nó cũng hỗ trợ port public link, giúp ai cũng có thể truy cập vào và sử dụng mô hình thông qua Internet.



***Hình 4. 1**. Thư viện Gradio*

Ta sẽ tạo một file Jupyter Notebook mới có tên **TranslateUI.ipynb** để thực hiện. Tương tự ta cũng sẽ load 2 model từ url kiểu Hugging Face mà đã lưu trước đó rồi load một số ví dụ câu mẫu để người dùng có thể trải nghiệm thử. Ngoài ra, để giao diện thông minh hơn ta sẽ dùng thêm thư viện **langdetect** để giao diện có thể tự động nhận diện ngôn ngữ đang nhập là Tiếng Anh hay Tiếng Việt để đưa vào đúng mô hình cần dịch, người dùng chỉ cần nhập câu rồi submit, tự động sẽ ra được kết quả mình mong muốn mà không cần phải chỉnh thủ công. Ta sẽ có giao diện như sau:



Hình 4. 2. *Giao diện website dịch Anh - Việt, Việt - Anh*

Đồng thời khi chạy, Gradio cũng sẽ xuất ra một đường link public để ta có thể gửi nó cho bạn bè, người dùng hoặc tester để họ trải nghiệm thử, sẽ được hiển thị như sau:

```
Colab notebook detected. To show errors in colab notebook, set debug=True in launch()
Running on public URL: https://3f486607a209a79fdb.gradio.live

This share link expires in 72 hours. For free permanent hosting and GPU upgrades, run
```

Hình 4. 3. *Mô phỏng public link*

Lưu ý là đường link này sẽ thay đổi liên tục và chỉ hoạt động trong thời gian bạn chạy Colab. Vì vậy khi tắt đi thì đường link cũng sẽ hết hiệu lực, ta cần chạy lại để lấy đường link truy cập mới.

Vậy là chúng ta đã xây dựng xong một pipeline từ khi xử lý dữ liệu, xây dựng model cho đến có một giao diện sử dụng hoàn chỉnh.

Mã nguồn đề tài:

Link	https://bit.ly/47FT9ml
QR Code	

CHƯƠNG 5
KẾT LUẬN – HƯỚNG PHÁT TRIỂN

5.1. Kết luận

Trong bối cảnh toàn cầu hóa và giao tiếp quốc tế ngày càng thúc đẩy, phát triển hệ thống máy dịch đa ngôn ngữ trở nên cấp thiết. Đề tài "Xây dựng hệ thống dịch Anh - Việt bằng mô hình BART" đã cung cấp những thông tin quý báu về hoạt động tự học ngôn ngữ và ứng dụng thực tế trong dịch thuật. Chúng tôi đã thành công trong việc xây dựng hệ thống dịch Anh - Việt dựa trên mô hình BART, cải thiện khả năng truyền đạt giữa hai ngôn ngữ. Hệ thống này đã cho thấy khả năng dịch tương đối chính xác và tự nhiên, giúp cải thiện hiệu quả trong việc truyền đạt thông tin giữa hai ngôn ngữ khác nhau.

Tuy nhiên, mô hình còn tồn tại một số hạn chế và thách thức mà chúng tôi đã gặp phải trong quá trình nghiên cứu. Mô hình BART, mặc dù có khả năng tạo ra các bản dịch đạt chất lượng, vẫn còn mắc phải vấn đề hiểu biết sai lệch về ngữ nghĩa và cấu trúc ngôn ngữ phức tạp. Điều này tạo ra những bản dịch không chính xác hoặc khó hiểu đối với một số ngữ cảnh. Hơn nữa, việc xây dựng và fine-tune mô hình BART cần sự đầu tư về thời gian và tài nguyên đáng kể.

5.2. Hướng phát triển

Trong thời gian tiếp theo, chúng tôi sẽ nghiên cứu và mở rộng thêm ngôn ngữ, đa dạng hóa ngữ cảnh cho các câu dịch từ ngôn ngữ gốc sang ngôn ngữ đích.

Nghiên cứu và áp dụng các kỹ thuật như quantization và pruning để giảm kích thước mô hình mà vẫn giữ được chất lượng dịch tốt.

Triển khai, phân tích nhu cầu của người dùng thực tế, chỉnh sửa giao diện sao cho thu hút và thuận tiện cho người dùng nhất.

TÀI LIỆU THAM KHẢO

[1] C. Ngo et al., "MTet: Multi-domain Translation for English and Vietnamese." Accessed: Aug. 23, 2023. [Online]. Available: https://arxiv.org/pdf/2210.05610.pdf

[2] M. Lewis et al., "BART: Denoising Sequence-to-Sequence Pre-training for Natural Language Generation, Translation, and Comprehension," Oct. 2019. Available: https://arxiv.org/pdf/1910.13461.pdf

[3] Y. Liu et al., "RoBERTa: A Robustly Optimized BERT Pretraining Approach," 2019. Available: https://arxiv.org/pdf/1907.11692.pdf

[4] L. Doan, L. T. Nguyen, N. L. Tran, T. Hoang, and D. Q. Nguyen, "PhoMT: A High-Quality and Large-Scale Benchmark Dataset for Vietnamese-English Machine Translation," arXiv.org, Oct. 23, 2021. https://arxiv.org/abs/2110.12199 (accessed Aug. 27, 2023).

[5] M. Joshi, D. Chen, Y. Liu, D. S. Weld, L. Zettlemoyer, and O. Levy, "SpanBERT: Improving Pre-training by Representing and Predicting Spans," arXiv:1907.10529 [cs], Jan. 2020, Available: https://arxiv.org/abs/1907.10529.

[6] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding," Proceedings of the 2019 Conference of the North, vol. 1, 2019, doi: https://doi.org/10.18653/v1/n19-1423.

[7] Alec, Karthik, Tim Salimans, and Ilya Sutskever, "Improving Language Understanding by Generative Pre-Training," 2018. Available: https://s3-us-west-2.amazonaws.com/openai-assets/research-covers/language-unsupervised/language_understanding_paper.pdf