AI VIET NAM – AIO COURSE 2024

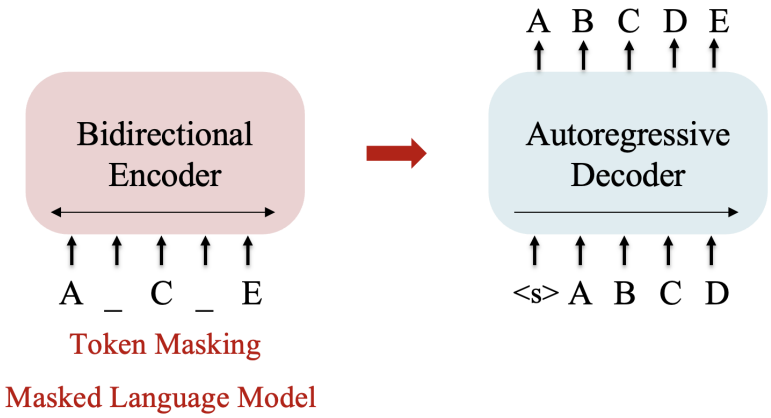
Pre-trained LMs: BART - T5

Low-Resource Neural Machine Translation

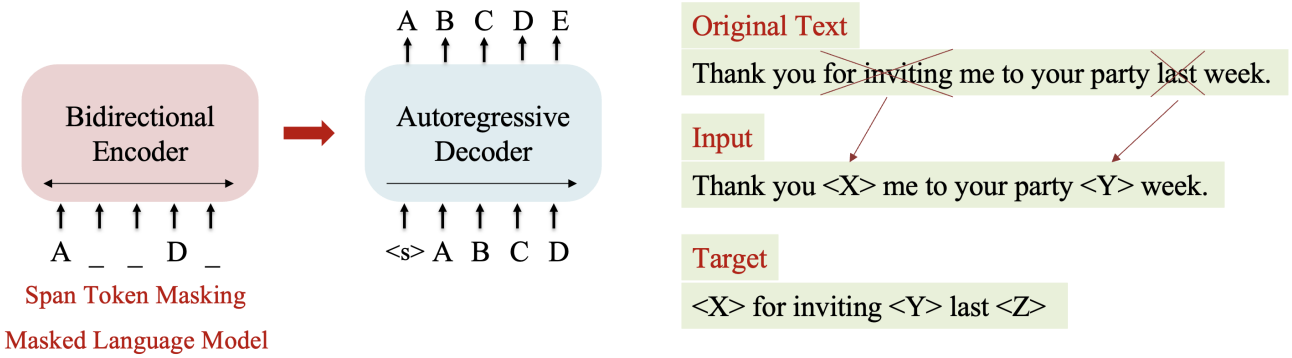
Quoc-Thai Nguyen và Quang-Vinh Dinh

Ngày 23 tháng 2 năm 2025

Phần 1. Giới thiệu



Hình 1: Mô hình BART.



Hình 2: Mô hình T5.

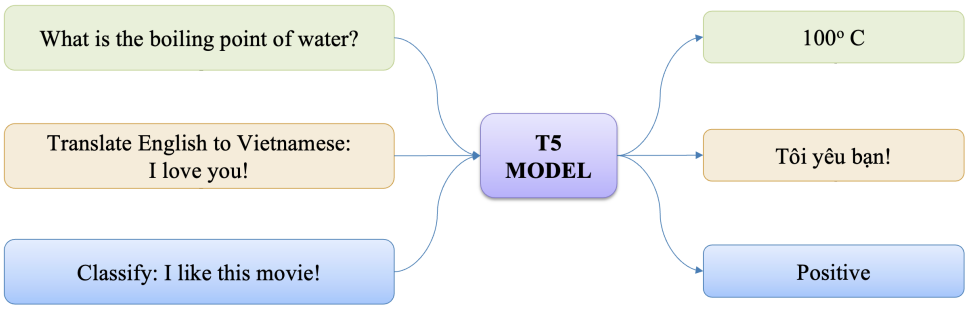
Dịch Máy (Machine Translation) với mục đích tự động dịch văn bản từ ngôn ngữ tự nhiên này sang ngôn ngữ tự nhiên khác. Một trong những thách thức lớn hiện nay của các hệ thống dịch là có ít

tài nguyên để huấn luyện mô hình. Vì vậy, trong phần này, chúng ta sẽ tập trung vào cải thiện các mô hình dịch với ít tài nguyên (Low-resource machine translation), số lượng các cặp dữ liệu song ngữ hạn chế, ví dụ mô hình dịch huấn luyện với chỉ 20000 cặp câu tiếng Anh và tiếng Việt. Vì vậy, để cải tiến mô hình dịch trong trường hợp có ít tài nguyên, chúng ta tập trung vào 2 hướng tiếp cận như sau:

1. Hướng 1: Sử dụng mô hình pre-trained mBART50, T5
2. Hướng 2: Sử dụng kỹ thuật để tăng cường dữ liệu như thu thập thêm dữ liệu, kỹ thuật học bán giám sát,...

Trong đó sử dụng các pre-trained models như mBART50 hoặc T5 sẽ đạt hiệu quả tốt hơn. Cả 2 mô hình đều có kiến trúc Transformer với nhiều lớp xếp chồng lên nhau.

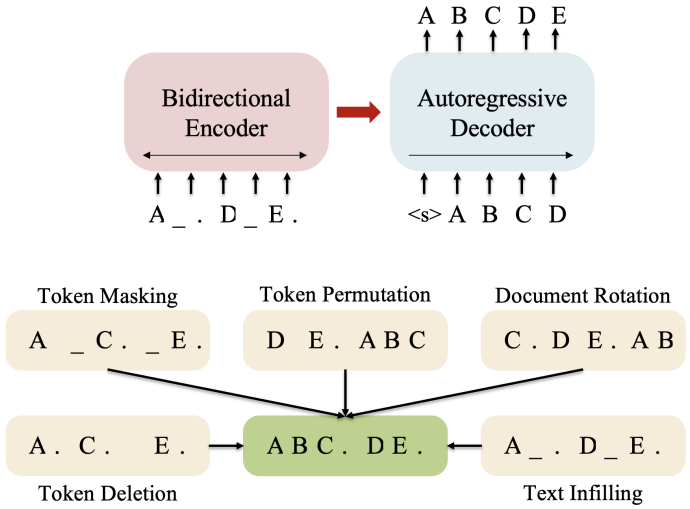
1. Mô hình T5: T5 là mô hình sử dụng cùng dạng biểu diễn chuỗi-chuỗi cho tất cả các bài toán, được mô tả như sau:



Hình 3: Biểu diễn đầu vào / đầu ra mô hình T5.

Kỹ thuật huấn luyện mô hình T5 được biểu diễn trong Hình 2.

2. Mô hình BART: Mô hình BART xây dựng để tối ưu cho các bài toán sinh chuỗi, vì vậy BART có một số hàm mục tiêu như sau:



Hình 4: Các hàm mục tiêu của mô hình BART.

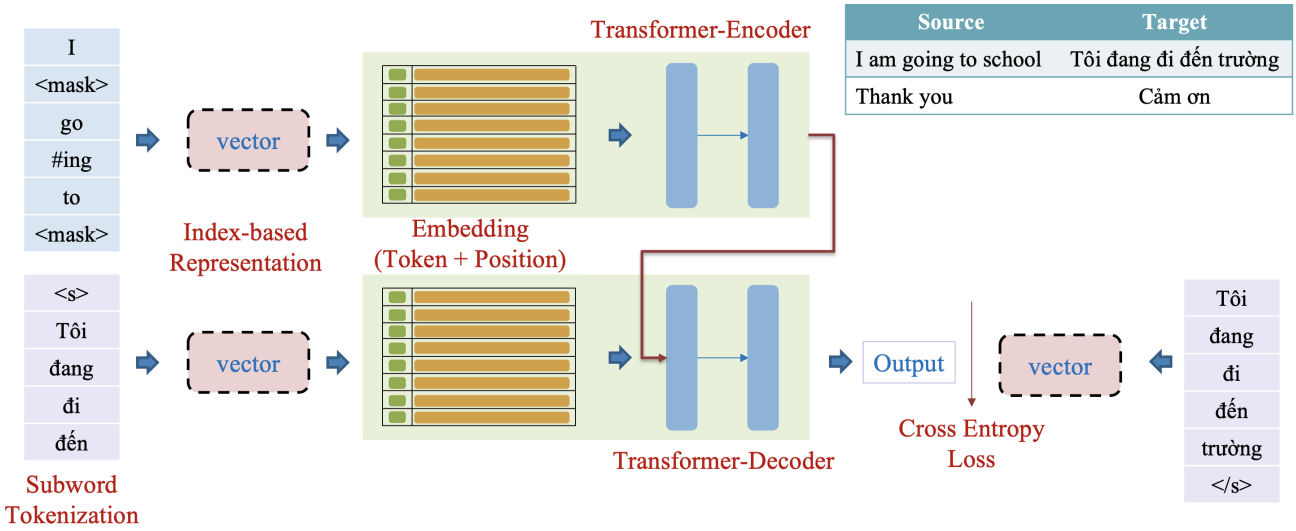
Phần 2. Fine-tuning mBART50

Để xây dựng, đánh giá hiệu suất các mô hình chúng ta sử dụng bộ dữ liệu dịch **IWSLT'15 English -Vietnamese** với số lượng mẫu cho training: 133,317 cặp câu song ngữ, tập validation: 1,553 cặp câu song ngữ và tập test: 1,269 cặp câu song ngữ. Chiều dịch sẽ từ tiếng anh sang tiếng việt.

Metric để đánh giá chúng ta sử dụng: ScacreBleu (BLEU)

Trong phần này, chúng ta sẽ sử dụng kỹ thuật fine-tuning để huấn luyện mô hình dịch theo chiều EN-VI.

Mô hình được sử dụng là **mBART50** được huấn luyện trên 50 ngôn ngữ khác nhau và phù hợp cho bài toán dịch EN-VI.



Hình 5: Kỹ thuật fine-tuning mô hình BART.

Các bước thực hiện như sau:

- Dataset: tải về bộ dữ liệu
- Tokenizer: tải về bộ mã hoá
- Encoding: chuyển văn bản thành vector
- Model: tải về mô hình mBART50
- Evaluate: định nghĩa độ đo đánh giá mô hình
- Trainer: huấn luyện mô hình
- Inference: kiểm thử kết quả
- Deployment: triển khai trên streamlit

1. Dataset

Chạy đoạn code sau đây để tải về bộ dữ liệu.

```
# install libs
!pip install -q transformers sentencepiece datasets accelerate evaluate sacrebleu

# import libs
from datasets import load_dataset

ds = load_dataset("thainq107/iwslt2015-en-vi")
```

2. Tokenizer

Chạy đoạn code dưới đây để tải vệ bộ tách từ của mô hình mBART50.

```
from transformers import AutoTokenizer

model_name = "facebook/mbart-large-50-many-to-many-mmt"
tokenizer = AutoTokenizer.from_pretrained(model_name)
```

3. Encoding

Sử dụng bộ tách từ để biểu diễn văn bản thành vector.

```
import torch

MAX_LEN = 75

def preprocess_function(examples):
    input_ids = tokenizer(
        examples["en"], padding="max_length", truncation=True, max_length=MAX_LEN
        )["input_ids"]

    labels = tokenizer(
        examples["vi"], padding="max_length", truncation=True, max_length=MAX_LEN
        )["input_ids"]
    labels = [
        [-100 if item == tokenizer.pad_token_id else item for item in label]
        for label in labels]

    return {
        "input_ids": torch.tensor(input_ids),
        "labels": torch.tensor(labels)
    }

preprocessed_ds = ds.map(preprocess_function, batched=True)
```

4. Model

Tải về mô hình mBART50.

```
from transformers import AutoModelForSeq2SeqLM

model_name = "facebook/mbart-large-50-many-to-many-mmt"
model = AutoModelForSeq2SeqLM.from_pretrained(model_name)
```

5. Evaluation

Sử dụng độ đo BLEU để đánh giá.

```
import numpy as np
import evaluate
metric = evaluate.load("sacrebleu")

def postprocess_text(preds, labels):
    preds = [pred.strip() for pred in preds]
    labels = [[label.strip()] for label in labels]

    return preds, labels

def compute_metrics(eval_preds):
    preds, labels = eval_preds
    if isinstance(preds, tuple):
        preds = preds[0]

    preds= np.where(preds != -100, preds, tokenizer.pad_token_id)
    decoded_preds = tokenizer.batch_decode(
        preds, skip_special_tokens=True, clean_up_tokenization_spaces=True
        )

    labels= np.where(labels != -100, labels, tokenizer.pad_token_id)
    decoded_labels = tokenizer.batch_decode(
        labels, skip_special_tokens=True, clean_up_tokenization_spaces=True
        )

    decoded_preds, decoded_labels = postprocess_text(
        decoded_preds, decoded_labels
    )

    result = metric.compute(predictions=decoded_preds, references=decoded_labels)
    result = {"bleu": result["score"]}

    return result
```

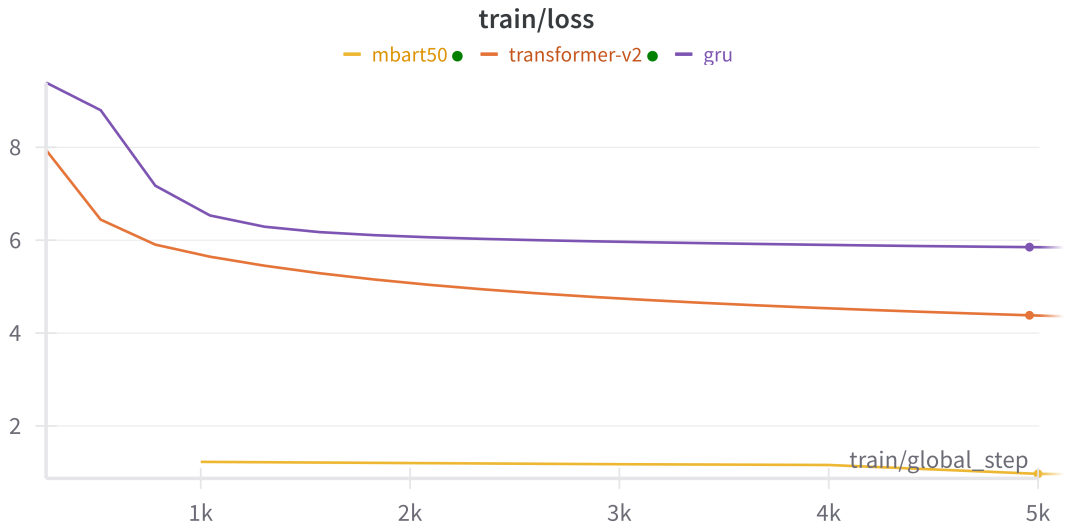
6. Trainer

Định nghĩa các tham số trong quá trình huấn luyện để tối ưu mô hình và giúp mô hình học được nhiều đặc trưng hơn. Sau khi huấn luyện có thể đẩy lên huggingface thông qua tài khoản cá nhân để sử dụng.

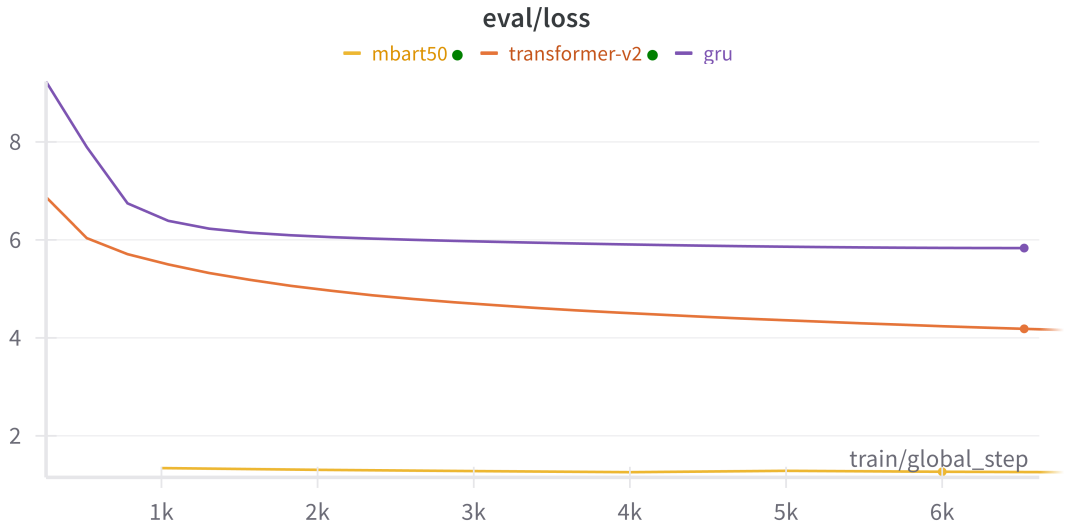
```python
# Disable wandb
import os
os.environ["WANDB_DISABLED"] = "true"

from transformers import Seq2SeqTrainingArguments, DataCollatorForSeq2Seq, Seq2SeqTrainer
training_args = Seq2SeqTrainingArguments(
    output_dir="./en-vi-mbart50",
    logging_dir="logs",
    logging_steps=1000,
    predict_with_generate=True,
    eval_strategy="steps",
    eval_steps=1000,
    save_strategy="steps",
    save_steps=1000,
    per_device_train_batch_size=32,
    per_device_eval_batch_size=32,
    save_total_limit=1,
    num_train_epochs=3,
    load_best_model_at_end=True,
)
data_collator = DataCollatorForSeq2Seq(tokenizer, model=model)
trainer = Seq2SeqTrainer(
    model,
    training_args,
    train_dataset=preprocessed_ds["train"],
    eval_dataset=preprocessed_ds["validation"],
    data_collator=data_collator,
    processing_class=tokenizer,
    compute_metrics=compute_metrics
)
```

Kết quả hàm loss so sánh với mô hình transformer và seq2seq-gru hình sau:



Hình 6: Kết quả train loss.



Hình 7: Kết quả eval loss.

Kết quả đánh giá trên tập validation thu được như sau:

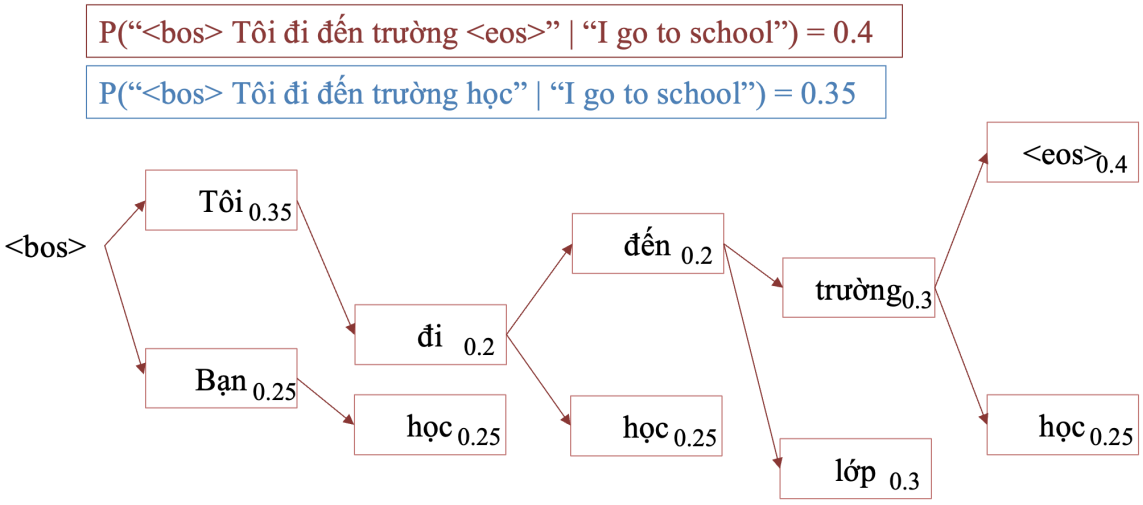
Training Loss	Epoch	Step	Bleu	Validation Loss
1.228	0.2400	1000	33.4010	1.3419
1.2022	0.4800	2000	34.0752	1.3063
1.1771	0.7199	3000	34.1612	1.2806
1.1607	0.9599	4000	34.3856	1.2582
0.9698	1.1999	5000	34.3075	1.2860
0.9298	1.4399	6000	34.4419	1.2671
0.9282	1.6799	7000	34.7962	1.2552
0.9174	1.9198	8000	34.7904	1.2516
0.8538	2.1598	9000	1.3000	34.3462
0.822	2.3998	10000	1.2953	34.3928
0.8206	2.6398	11000	1.2834	34.4372
0.8177	2.8798	12000	1.2825	34.6635

Hình 8: Kết quả cho từng bước và kết quả bleu.

7. Inference

Sau khi train model có thể sử dụng model từ huggingface để dự đoán và đánh giá trên tập test. Sử dụng greedy search (luôn chọn token dự đoán với giá trị xác suất lớn nhất) và beam search (chọn top k các token với giá trị xác suất lớn nhất) để đánh giá.

Thuật toán của beam search được mô tả hình sau với beam size = 2:



Hình 9: Thuật toán beam search.

```python
# download model
from transformers import pipeline
translator = pipeline(model="thainq107/en-vi-mbart50")

# test a sample with beam search
translated_text = translator("I go to school", num_beams=2)
translated_text

# greedy search for test set
pred_sentences = translator(ds["test"]["en"], batch_size=32, num_beams=1,
    do_sample=False)

# beam search for test set
pred_sentences = translator(ds["test"]["en"], batch_size=32, num_beams=5)

# evaluate
import sacrebleu

bleu_score = sacrebleu.corpus_bleu(pred_sentences, [ds["test"]["vi"]], force=True
    )
bleu_score
```

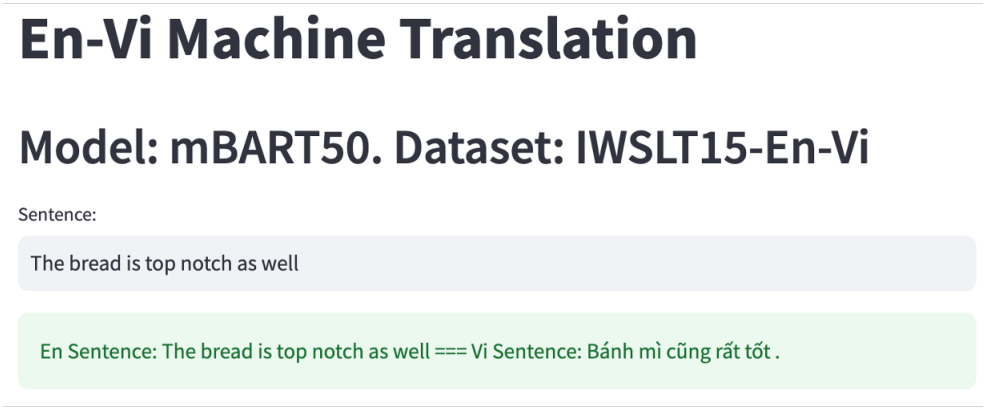
Kết quả đánh giá trên tập test set của các mô hình như sau:

Experiment	Model	ScareBLEU
#1	Standard Transformer (Greedy Search)	24.66
#2	mBART50 (Greedy Search)	33.50
#3	mBART50 (Beam size = 5)	34.17

Hình 10: Kết quả BLEU trên tập test.

7. Deployment

Trong phần này, chúng ta triển khai mô hình trên streamlit. Tham khảo về code và demo.



Hình 11: Triển khai ứng dụng trên Streamlit.

Phần 3. Câu hỏi trắc nghiệm

Câu hỏi 1 Mô hình BART có kiến trúc là gì?
a) Transformer-Encoder
b) Transformer-Decoder
c) Transformer-Encoder-Decoder
d) BiLSTM

Câu hỏi 2 Hàm mục tiêu nào sau đây không phải của BART?
a) Token Masking
b) Token Deletion
c) Sentence Permutation
d) Next Sentence Prediction

Câu hỏi 3 Mô hình BART-Base có bao nhiêu lớp encoder?
a) 6
b) 12
c) 24
d) 48

Câu hỏi 4 Mô hình BART-Large có bao nhiêu lớp encoder?
a) 6
b) 12
c) 24
d) 48

Câu hỏi 5 mBART50 được huấn luyện trên bao nhiêu ngôn ngữ?
a) 5
b) 25
c) 50
d) 100

Câu hỏi 6 mBART50 là mô hình tiền huấn luyện đa ngữ cho bài toán nào?
a) Text Summarization
b) Text Classification
c) Question-Answering
d) Machine Translarion

Câu hỏi 7 Tập dữ liệu dịch máy tiếng việt tiếng anh là?
a) PhoMT
b) IMDB-Review

c) C4
d) ROOT

Câu hỏi 8 Mô hình nào sau đây có kiến trúc tương tự mô hình BART?
a) BERT
b) GPT
c) RoBERTa
d) T5

Câu hỏi 9 Mô hình T5 sử dụng tiền tố nào cho bài toán dịch?
a) translate English to Vietnamese
b) translate to Vietnamses from English
c) Cả 2 đáp án đều đúng
d) Cả 2 đáp án đều sai

Câu hỏi 10 Độ đo đánh giá được sử dụng trong phần thực nghiệm là?
a) F1
b) SacreBLEU
c) Accuracy
d) ROUGE

Phần 4. Phụ lục

1. **Hint**: Dựa vào file tải về Fine-tuning mBART50 for En-Vi Machine Translation để hoàn thiện các đoạn code.

2. **Solution**: Các file code cài đặt hoàn chỉnh và phần trả lời nội dung trắc nghiệm có thể được tải về tại đây (Lưu ý: Sáng thứ 3 khi hết deadline phần project, admin mới copy các nội dung bài giải nêu trên vào đường dẫn).

3. **Rubric**:

Phần	Kiến Thức	Đánh Giá
1	- Hiểu rõ bài toán dịch máy sử dụng mBART50 - Hiểu rõ thách thức bài toán dịch với ít tài nguyên	- Xây dựng mô hình dịch máy sử dụng mBART50
2.	- Hiểu rõ kỹ thuật để sinh bản dịch như greedy search hay beam search	- Sử dụng thư viện transformers và thử nghiệm để đánh giá độ đo BLEU trên các phương pháp giải mã khác nhau.

- Hết -