Artificial Intelligence Final Report Assignment 問題 3 (Problem 3)

レポート解答用紙 (Report Answer Sheet)

学生証番号 (Student ID): 21520389

名前(Name): Phan Cả Phát (Group Leader)

学生証番号 (Student ID): 21520297

名前(Name): Trần Lê Anh Khoa

学生証番号 (Student ID): 21520646

名前(Name): Đào Đại Chí

問題 3 (Problem 3)のレポート

I. Introduction

1. Machine translation

Machine translation is an automatic translation from one language to another. The benefit of machine translation is that it is possible to translate a large corpus of text instantly. Given a sequence of text in a source language as an input, the output is a translated sentence in the target language. But there is no best translation of that text. This is because of the flexibility of human language. This makes the challenge of automatic machine translation difficult, maybe one of the most difficult in artificial intelligence.

There are two main types of machine translation: Statistical machine translation, or SMT for short, and Neural machine translation, or NMT for short. SMT finds the most probable translation in the target language by some of the formulas which are difficult to understand. In this work, we focus on NMT, which has been proven to be much more effective than its predecessor. NMT attempts to build and train a single, large neural network that reads a sentence and outputs a correct translation.

2. Dataset

Created by Stanford in 2015, the IWSLT 15 English-Vietnamese Sentence pairs for translation., in Multi-Lingual language. The dataset was split into 3 parts: train, test, validation. The distribution of each part is train set with 133,317

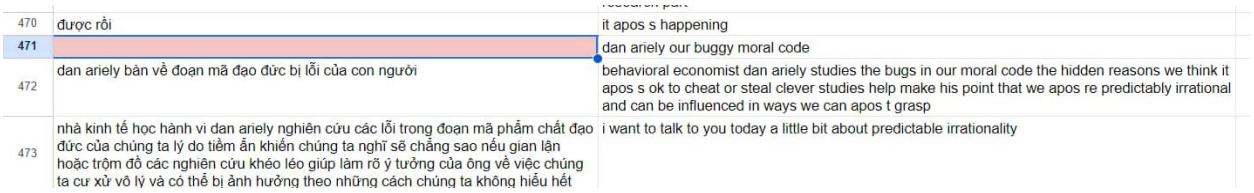
sentences, validation set with 1553 sentences, test set with 1268 sentences. Here is an example of a sentence pair for translation.

English	we blow it up and look at the pieces
Vietnamese	chúng tôi cho nó nổ và xem xét từng mảnh nhỏ

II. Method

1. Preprocessing

Before building the model, we briefly reviewed the dataset and noticed that some data were corrupted. Here is an example of this error:



470	được rồi	it apos s happening
471		dan ariely our buggy moral code
472	dan ariely bàn về đoạn mã đạo đức bị lỗi của con người	behavioral economist dan ariely studies the bugs in our moral code the hidden reasons we think it apos s ok to cheat or steal clever studies help make his point that we apos re predictably irrational and can be influenced in ways we can apos t grasp
473	nhà kinh tế học hành vi dan ariely nghiên cứu các lỗi trong đoạn mã phẩm chất đạo đức của chúng ta lý do tiềm ẩn khiến chúng ta nghĩ sẽ chẳng sao nếu gian lận hoặc trộm đồ các nghiên cứu khéo léo giúp làm rõ ý tưởng của ông về việc chúng ta cư xử vô lý và có thể bị ảnh hưởng theo những cách chúng ta không hiểu hết	i want to talk to you today a little bit about predictable irrationality

Picture 1: Example of Data Failures

This preprocessing sequence is crucial for handling two bilingual text files, train.en and train.vi. It prepares the data for training a machine translation model by ensuring that the dataset is clean and suitable for machine learning algorithms. The steps include reading the data into DataFrames, replacing invalid values with NaN, renaming columns for clarity, concatenating the two DataFrames, and converting the 'en' column to the string data type. This thorough preparation is essential for enhancing the efficiency and accuracy of the subsequent model training.

First, the data from the two text files, train.en and train.vi, is read into two separate DataFrames, df_train_en and df_train_vi. These files contain bilingual sentences, with each sentence on a separate line. This data reading uses the pd.read_csv function from the Pandas library, with the delimiter="\t" parameter to specify that the columns are separated by tabs and header=None to skip the header row.

```
df_train_en = pd.read_csv('train. en', delimiter="\t", header=None)
df_train_vi = pd.read_csv('train.vi', delimiter="\t", header=None)
```

Picture 2: Read data from two text files.

Next, the code identifies specific rows in both DataFrames to replace invalid values with NaN. In df_train_en, the rows specified in the rtpe list are replaced with NaN. Similarly, the rows specified in the rows_to_replace list in

df_train_vi are also replaced with NaN. After replacing, the rows containing NaN are removed to ensure clean data.

```
rtpe = [121099, 121582]
for row in rtpe:
  if row < len(df_train_en):
    df_train_en.at[row, 0] = np.nan
df_train_en.dropna(inplace=True)

rows_to_replace = [469, 8695, 9762, ... ]
for row in rows_to_replace:
  if row < len(df_train_vi):
    df_train_vi.at[row, 0] = np.nan
df_train_vi.dropna(inplace=True)
```

Picture 3: identify and replace invalid values

After handling invalid values, the columns of df_train_en and df_train_vi are renamed for clarity. The single column of df_train_en is named 'en', while the single column of df_train_vi is named 'vi'. This renaming makes it easier to manipulate and manage the data in subsequent steps.

```
df_train_en.columns = ['en']
df_train_vi.columns = ['vi']
```

Picture 4: Rename for clarity.

Next, the two DataFrames, df_train_en and df_train_vi, are concatenated horizontally to create a single DataFrame, df_train. This DataFrame contains both the 'en' and 'vi' columns, representing the English and Vietnamese sentences, respectively. The concatenation uses the pd.concat function with the axis=1 parameter.

```
df_train = pd.concat([df_train_vi.reset_index(drop=True), df_train_en.reset_index(drop=True)], axis=1)
```

Picture 5: concatenated horizontally to create a single DataFrame.

Finally, the 'en' column in the df_train DataFrame is converted to the string data type. This ensures that the data in this column is correctly processed in the subsequent steps, especially during the training of the machine translation model.

```
# Change 'en' into string
df_train['en'] = df_train['en'].astype(str)
```

Picture 6: Change data type.

Here is our result before and after replacing.

```
Before replacement:
Row 469: .
Row 8695: Khán giả vỗ tay
Row 9762: Vỗ tay .
Row 10707: vỗ tay
Row 21738: Vỗ tay
Row 26408: vỗ tay
Row 29494: Vỗ tay .
Row 38600: &quot; tán thưởng &quot;
```

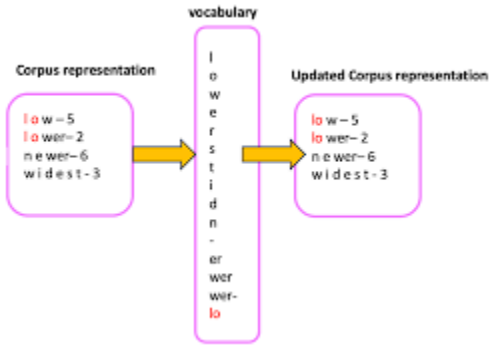
Picture 7: Data before replacement.

```
Replaced values (should be NaN):
Row 469: nan
Row 8695: nan
Row 9762: nan
Row 10707: nan
Row 21738: nan
Row 26408: nan
Row 29494: nan
Row 38600: nan
```

Picture 8: Data after replacement.

Next, Implements a text tokenizer utilizing Byte-Pair Encoding (BPE) for both Vietnamese and English languages. The tokenizer function accepts text and a language identifier (lang) to tokenize the text using pre-trained BPE models (bpemb_vi for Vietnamese and bpemb_en for English). The token_len function computes the number of tokens in a tokenized string, excluding BPE

markers. These utilities are essential for preprocessing text data efficiently in natural language processing applications.



Picture 9: Implement text tokenizer.

And then, Using the sacremoses library, the MosesPunctNormalizer class is employed to standardize punctuation across texts in both Vietnamese and English languages. This normalization ensures uniformity in punctuation usage, aiding in subsequent text analysis tasks such as sentiment analysis and machine translation.

	vi	en	vi_token	en_token
0	Khoa học đằng sau một tiêu đề về khí hậu	Rachel Pike : The science behind a climate hea...	_khoa _học _đằng _sau _một _tiêu _đề _về _khí ...	_rachel _pike _: _the _science _behind _a _cli...
1	Trong 4 phút , chuyên gia hoá học khí quyển Ra...	In 4 minutes , atmospheric chemist Rachel Pike...	_trong _0 _phút _, _chuyên _gia _hoá _học _khí...	_in _0 _minutes _, _atmospheric _chemist _rach...
2	Tôi muốn cho các bạn biết về sự to lớn của nhữ...	I 'd like to talk to you today about the ...	_tôi _muốn _cho _các _bạn _biết _về _sự _to _l...	_i _& ap os ; d _like _to _talk _to _you _toda...
3	Có những dòng trông như thế này khi bàn về biế...	Headlines that look like this when they have t...	_có _những _dòng _trông _như _thế _này _khi _b...	_headlines _that _look _like _this _when _they...
4	Cả hai đều là một nhánh của cùng một lĩnh vực ...	They are both two branches of the same field o...	_cả _hai _đều _là _một _nhánh _của _cùng _một ...	_they _are _both _two _branches _of _the _same...
...	...	...	...	...
132995	Cái cô cần là sự cảm thông và lòng trắc ẩn , v...	You needed my empathy and compassion , and abo...	_cái _cô _cần _là _sự _cảm _thông _và _lòng _t...	_you _needed _my _empathy _and _compassion _, ...
132996	Mà chính hệ thống y tế mà tôi góp phần , đã là...	Maybe the system , of which I was a part , was...	_mà _chính _hệ _thống _y _tế _mà _tôi _góp _ph...	_maybe _the _system _, _of _which _i _was _a _...
132997	Nếu giờ đây cô đang xem chương trình này , tôi...	If you 're watching this now , I hope you...	_nếu _giờ _đây _cô _đang _xem _chương _trình _...	_if _you _& ap os ; re _watching _this _now _,...

Picture 10:Punctuation Standardization Using sacremoses for Vietnamese and English Texts.

Converts tokenized text data into a format suitable for machine translation training. Then uses the clean-corpus-n.perl script to perform cleaning tasks such as removing empty lines, overly long words, and sentences that exceed a specified length (-max-word-length 50).
Output: Produces cleaned parallel corpora (train.clean.vi and train.clean.en) ready for further preprocessing steps in machine translation model training.

```
[ ] df_train = pd.DataFrame({"vi": data_train_vi, "en": data_train_en})
    df_train
```

	vi	en
0	_khoa _học _đằng _sau _một _tiêu _đề _về _khí ...	_rachel _pike : _the _science _behind _a _clim...
1	_trong _0 _phút _, _chuyên _gia _hoá _học _khí...	_in _0 _minutes _, _atmospheric _chemist _rach...
2	_tôi _muốn _cho _các _bạn _biết _về _sự _to _l...	_i _& ap os ; d _like _to _talk _to _you _toda...
3	_có _những _dòng _trông _như _thế _này _khi _b...	_headlines _that _look _like _this _when _they...
4	_cả _hai _đều _là _một _nhánh _của _cùng _một ...	_they _are _both _two _branches _of _the _same...
...	...	...
132989	_mà _chính _hệ _thống _y _tế _mà _tôi _góp _ph...	_maybe _the _system _, _of _which _i _was _a _...
132990	_nếu _giờ _đây _cô _đang _xem _chương _trình _...	_if _you _& ap os ; re _watching _this _now _,...
132991	_paul _ph ol eros : _làm _sao _để _bớt _nghèo ...	_paul _ph ol eros : _how _to _reduce _poverty ...
132992	_năm _0000 _, _kiến _trúc _sư _paul _ph ol ero...	_in _0000 _, _architect _paul _ph ol eros _was...
132993	_mong _muốn _xoá _nghèo _là _0 _mục _tiêu _vĩ ...	_the _idea _of _eliminating _poverty _is _a _g...

Picture 11:Data Preparation and Cleaning for Machine Translation Training.

2. Architecture

In this project we have built 2 model to support the comparison for the bleu result, one is Transformers, and the last one is mBart

2.1 Transformers

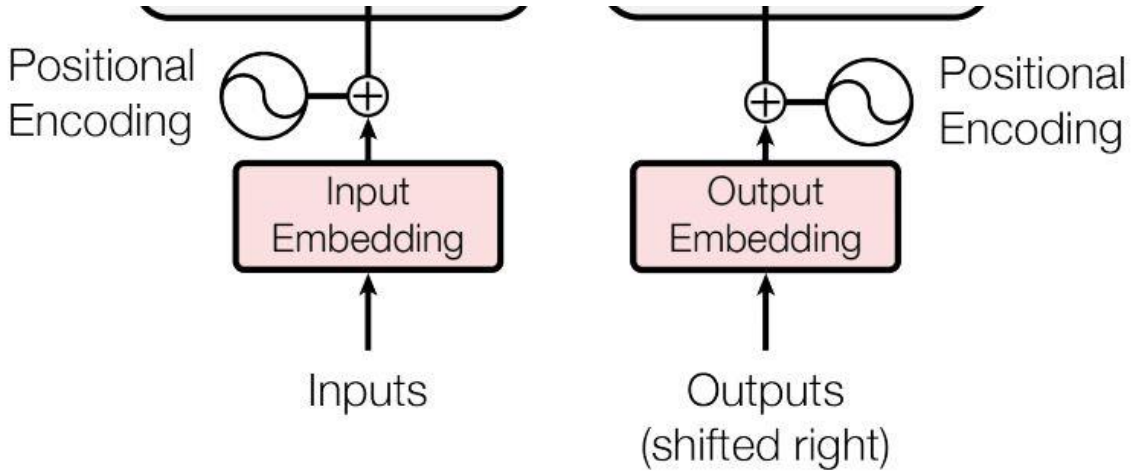
We employ a Seq2Seq model leveraging the Transformer architecture, which is particularly adept at tasks like machine translation. The self-attention mechanism at the core of the Transformer excels at capturing long-range dependencies in sequential data, making it a superior choice for sequence-to-sequence tasks by enhancing both training efficiency and model performance.

This implementation features several essential components: Positional Encoding, Token Embedding, and the Seq2Seq Transformer itself. Each component is integral to effectively transforming the input sequence into the desired output sequence, ensuring precise and context-aware translations.

- Positional Encoding.

The Positional Encoding class provides a way to incorporate the position information into the embeddings, which is essential because the Transformer model does not have any built-in notion of sequence order. The

positional encoding is added to the token embeddings, allowing the model to make use of the relative positions of tokens in the sequence.



Picture 12:Positional Encoding.

- Token Embedding

The TokenEmbedding class converts input tokens into dense vectors of a specified size. This is crucial for transforming the discrete token indices into continuous representations that can be processed by the Transformer layers.

- Seq2Seq Transformer

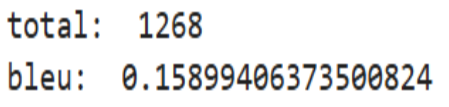
The Seq2SeqTransformer class combines the encoder and decoder of the Transformer model. It includes embedding layers for both source and target languages, positional encoding, and the Transformer itself. The forward method manages the input and output sequences, applying the appropriate masks to handle padding and ensure that the model does not look ahead in the sequence during training.

- Configuration

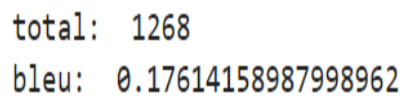
Our configuration is based on LAB 6. We adjust LSTM to Transformer and set the numbers of Transformer layers to 3. The hidden size is 512, dropout is 0.01 in both Encoder and Decoder, all of them are the same as LAB 6. We also change the dimensions of embedding 512. We decrease the batch size to 32. In addition, we have 15 epoch for training.

- Result

Here is a small comparison of BLEU scores when we apply preprocessing versus when we do not. This is the evidence.



Picture 13: Blue result before preprocessing.



Picture 14: Bleu result after preprocessing.

As you can see, the BLEU scores have improved after we applied preprocessing.

2.2 Mbart

mbart is a transformer-based model developed by Facebook AI, specifically designed for multilingual machine translation tasks. It extends the BART (Bidirectional and Auto-Regressive Transformers) architecture to handle multiple languages efficiently. By using language-specific tokens and extensive pre-training on diverse multilingual datasets, mbart can translate between a wide range of language pairs with high accuracy and fluency. It excels in scenarios requiring seamless cross-lingual communication and has become a cornerstone in multilingual NLP applications due to its robust performance and versatility.

- Setup and Configuration**:**
    - Import necessary libraries and define language codes for translation.
    - Initialize the mbart-large-50-many-to-many-mmt model and configure tokenizer based on source (vi) and target (en) languages.
- Data Preparation:
    - Create a custom dataset (CustomDataset) using tokenizer and prefix for both training (df_train) and test data (df_test).
- Training Configuration:
    - Define training parameters such as epochs, learning rate, weight decay, and batch size.
    - Set up training arguments (Seq2SeqTrainingArguments) including optimizer (adafactor), FP16 for mixed-precision training, and logging strategy.
- Training Execution:
    - Initialize Seq2SeqTrainer with the configured model, training arguments, and custom dataset.
    - Execute training using trainer.train() method and monitor training progress.

- Evaluation:

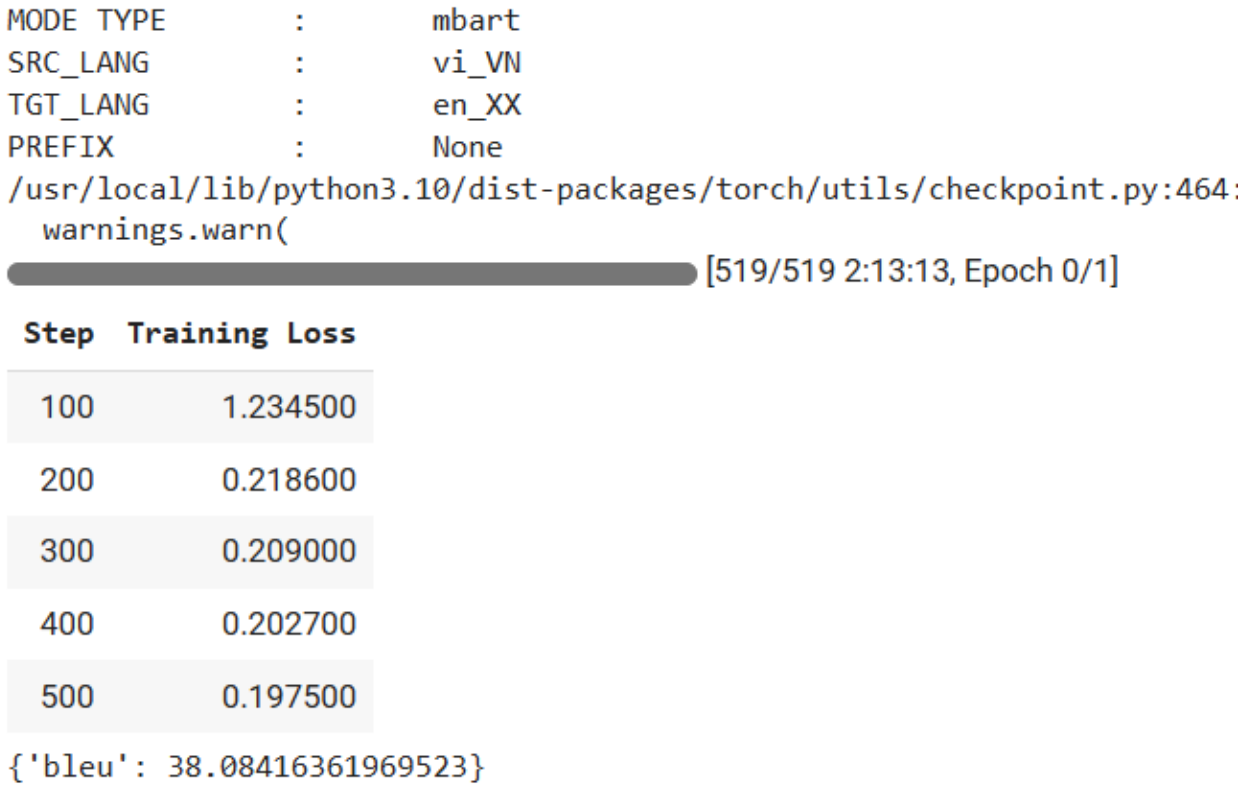
    - Evaluate model performance on the test dataset using BLEU score metric (compute_metrics function).
    - Generate translations (pred) for test inputs and compare with ground truth (label).

- Post-processing and Evaluation:

    - Post-process predicted translations (decoded_preds) and labels (decoded_labels) to remove special tokens and whitespace.
    - Compute BLEU score (metric.compute()) between predicted and reference translations to evaluate translation quality.

- Results:

    - Output and print the final BLEU score (result) indicating the translation quality achieved by the model with SacreBLEU.



Picture 15: Bleu result using mbart

III. Experience

The process of completing this project was both challenging and rewarding, encompassing several critical stages. We started by selecting the IWSLT 15 English-Vietnamese dataset, created by Stanford in 2015, which contains over 133,317 training sentences, making it ideal for machine translation tasks. Preprocessing was crucial due to some errors in the provided data. We read the bilingual text files into DataFrames, replaced invalid values with NaN, and removed these rows to ensure a clean dataset. We also renamed columns for clarity and concatenated the DataFrames to form a single dataset with both English and Vietnamese sentences, converting the 'en' column to the string data type to facilitate further processing.

In terms of model architecture, we developed and compared two models: a Transformer-based Seq2Seq model and the mbart model.

For the Transformer-based Seq2Seq model, we leveraged the Transformer architecture, renowned for its ability to handle long-range dependencies in sequential data due to its self-attention mechanism. Key components included Positional Encoding to incorporate position information into the embeddings, Token Embedding to convert input tokens into dense vectors, and the Seq2Seq Transformer, which combines the encoder and decoder components. Our configuration was based on LAB 6, with adjustments to use a Transformer instead of an LSTM. We set the number of Transformer layers to three, used a hidden size of 512, and a dropout rate of 0.01 for both the encoder and decoder. We also adjusted the embedding dimensions to 512 and reduced the batch size to 32, training the model over 15 epochs. This systematic approach allowed us to achieve precise and context-aware translations, demonstrating the effectiveness of the Transformer architecture in machine translation tasks.

For the mbart model, we used the mbart-large-50-many-to-many-mmt model, specifically designed for multilingual machine translation tasks. We prepared the dataset by tokenizing the text using the mbart tokenizer and adding language-specific tokens. Training involved setting parameters such as the number of epochs, learning rate, weight decay, and batch size. We executed the training using the Seq2SeqTrainer and evaluated the model’s performance using the BLEU score metric. The mbart model, with its extensive pre-training on diverse multilingual datasets, demonstrated high accuracy and fluency in translations, showcasing its robustness and versatility in multilingual NLP applications.

IV. Conclusions Future Work

This project has been a challenging yet rewarding journey, beginning with dataset selection and culminating in the successful implementation of both a Transformer-based Seq2Seq model and an mbart model for English-Vietnamese translation. Through meticulous preprocessing and model configuration, we demonstrated significant improvements in translation quality.

Moving forward, our focus will be on deepening our understanding of fundamental neural machine translation (NMT) principles and refining our models to enhance performance. We plan to explore advanced techniques such as improved attention mechanisms and enhancements in transformer technology. Additionally, we aim to experiment with other multilingual models and incorporate more diverse datasets to broaden the applicability of our solutions.

Our ultimate goal is to push the boundaries of natural language processing (NLP), leveraging our knowledge and experience to tackle increasingly complex challenges in machine translation and other NLP tasks. By continuously improving our models and methodologies, we aspire to contribute to the advancement of machine translation technology and its real-world applications.