English-Vietnamese Neural Machine Translation

Table of contents:

Group members

Phạm Cát Vũ - 20194465

Nguyễn Tuấn Dũng - 20194427

Phạm Văn Cường - 20194421

1. Introduction

Machine Translation (MT) is described as translating a sentence or document of one language to another language. There are a variety of methods researchers have come up with over the years, ranging from rule-based to more modern neural-based approaches. In recent years, Neural Machine Translation, which models the translation as a sequence to sequence learning task, has become the dominant method to tackle this problem.

Recently, the Transformer architecture has proven to be a new SOTA model for many Natural Language Processing tasks, which adopts the self attention mechanism that makes it possible to track the relations between words across very long text sequences in both forward and reverse directions. In this project, we implements NMT models based on the Transformer architecture using the PyTorch framework. Our chosen data is the *IWSLT'15 English-Vietnamese* created by Stanford NLP groups.

Despite great advances in Machine Translation thanks to Deep Learning models, this method’s effectiveness is largely contingent on the quality as well as the size of the training corpus. To train a reliable Machine Translation model, numerous sentence pairs are needed. A naive solution is to manually annotate more data, but this approach is

often very costly, which creates a barrier for low-resource languages. Thus, there is a need to further research Data Augmentation techniques for text processing. Thus, in this project, we examine and compare various Data Augmentation techniques then analyze the advantages as well as the shortcomings of each of them. Four Data augmentation techniques used in this project are Synonym Replacement, Back Translation, English - France Back Translation and Contractions.

The rest of the report is organized as follows: Section 2 will describe our dataset. The third section will go into our preprocessing steps. Section 4 goes into our baseline Transformer model. Next, section 5 will go in-depth into all of the Data augmentation techniques included in this project. Experimental results and conclusions drawn are presented in Sections 6 and 7 respectively.

2. Dataset

To evaluate the effectiveness of each Data augmentation method, we choose the *IWSLT'15 English-Vietnamese* built by the Stanford NLP group. The dataset consists of English-Vietnamese sentence pairs from speeches of TED and TEDx talks. The statistics of the dataset are given in the table below

Dataset	Sentences
Train	133,317
Dev	1,553
Test	1,268

3. Preprocessing

3.1. Preprocessing

We first perform a few basic preprocessing steps for our sentences.

In the original dataset, many special symbols are denoted using HTML tags, like `"\""` is denoted with `"quot;"`. We will replace these HTML tags with their original symbol.

Next, we get to the tokenization step for our sentences, which is the process of splitting an input sentence into words. In this project, we will experiment with 3 tokenization techniques:

Simple Tokenization

Our first tokenization method is the simple tokenization technique by the NLTK toolkit for both languages. Tokens are split by the space character, and punctuations are counted as tokens as well.

For example, the sentence “Bạn là ai?” will be split into the following tokens: [”Bạn”, “là”, “ai’”, “?”]

Word Segmentation

Vienamese contains many multi-syllable words with spaces inbetween, like “Việt Nam” for example. Thus, applying a simple tokenization will break these words into multiple tokens, which can lose valuable information from the original word. To tackle this, we can use word segmentation tools for Vietnamese that automatically detects the presence of multi-syllable words in Vietnamese. In this project, we utilized the “*underthesea*” Vietnamese NLP framework for this task.

The English sentences will still be tokenized using the simple tokenizer provided by NLTK as above.

Byte-Pair Encoding

BPE builds a token vocabulary and a merge table. The token vocabulary is initialized with the character vocabulary, and the merge table is initialized with an empty table. First, each word is represented as a sequence of tokens plus a special end of word symbol. Then, the method iteratively counts all pairs of tokens and merges the most frequent pair into a new token. This token is added to the vocabulary, and the merge operation is added to the merge table. This is done until the desired vocabulary size is reached (4000 in our implementation).

The resulting merge table specifies which subwords have to be merged into a bigger subword, as well as the priority of the merges. In this way, it defines the segmentation procedure. First, a word is split into distinct characters plus the end of word symbol.

Then, the pair of adjacent tokens which has the highest priority is merged. This is done iteratively until no merge from the table is available.

Iteration	Sequence	Vocabulary
0	a t e /w a t /w	{a, t, e, /w}
1	at e /w at /w	{a, t, e, /w, at}
2	at e /w at/w	{a, t, e, /w, at, at/w}
3	at e/w at/w	{a, t, e, /w, at, at/w, e/w}
4	ate/w at/w	{a, t, e, /w, at, at/w, e/w, ate/w}

An example of how BPE obtains vocabulary given a raw sequence

We implemented this technique using the *youtokentome* framework. Each languages will build a token vocabulary from the training set, and new sentences will be split using this built vocabulary.

3.2. Vocabulary Building

After performing these tokenization techniques, the input sentences are now represented as arrays of tokens. We will next build two vocabularies: one for English, one for Vietnamese from the sentences of the training set, where each token in the training data is represented by an index.

Before building vocabularies, we converted all of the tokens into lowercase, so that there will be no identical tokens being added only because of different casing. However, doing this has one downside: all of the input tokens in the source and language will be in lowercase, thus our Machine Translation can not preserve the uppercase words in its translation, like personal names… To combat this problem, we will add a special token <*upper*> before each uppercase words in the sentence, and add this token to the vocabulary. This way, when the translation model outputs the token <*upper*>, it will automatically uppercase the following token in its prediction.

We also need to add some special tokens as well: <SOS>, <EOS> , <UNK>, <PAD> to pad our sentences to a fixed length.

3.3. Sentence Numericalization

From the vocabularies that we've built, each input sentence to the model will be numericalized by replacing its tokens with their corresponding index in our vocabulary (assigned to <UNK> if they're not in our vocabulary). Since the Transformer model doesn't process our data recurrently like RNNs, our input vector need to be of a fixed size. We set the number of tokens in each input sequence to be 120. Any sequences longer than that will be truncated, and shorter sentences will be added with the "<PAD>" token until they reach the length of 120.

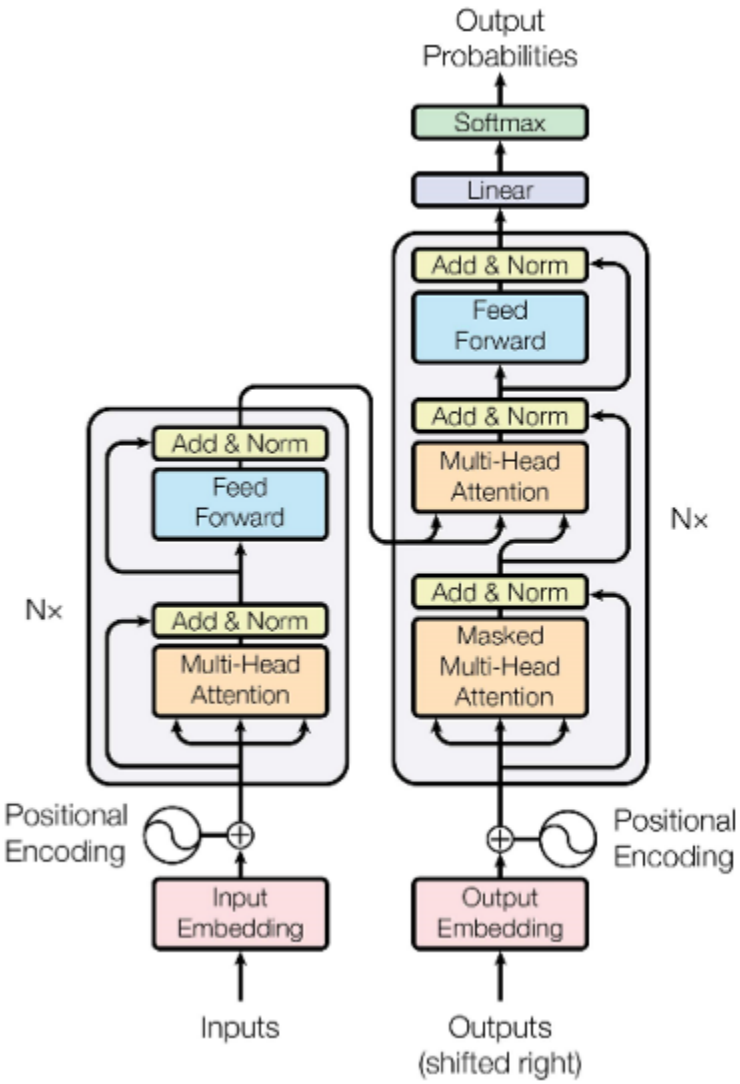
While 120 might not be a great maximum length for generating quality prediction, the Transformer models have a quadratic dependencies on the input length, a greater sequence length like 512 for example might be too computationally expensive for us to afford.

4. Main Architecture

4.1. The Transformer Architecture

For this project, we implement the Transformer architecture as the base model for our translation. The transformer architecture contains 4 main components as below:

- Word embedding and Position encoding
- Encoder
- Decoder
- Output layer



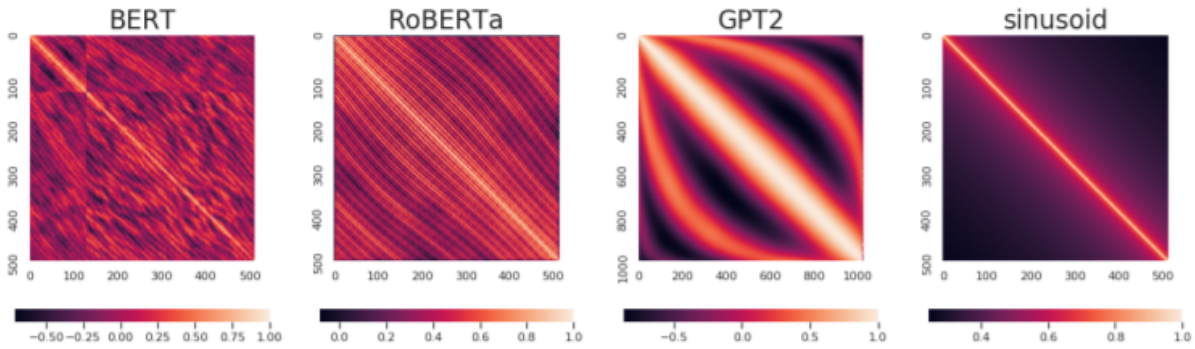
4.1.1 Word embedding and Position encoding

Firstly, our input sequences are passed through two embedding layers: a word embedding layer and a positional encoding layer.

Similar to other Sequence models, we use learned embedding to convert tokens into latent embedding vectors of dimension d_{model}. The goal is to firstly, reduce the dimension of word vectors, and secondly, it is better for representing the similarity and dissimilarity between words.

Since the model requires no recurrence and no convolution, to extract useful information regarding the order of sequence, a Position encoding layer is required. In the original paper, the author uses an established function for this layer. However, we

instead use a Learned Position encoding as used in BERT, RoBERTa, and ViT. The image below presents the cosine similarity of position embedding after pre-training of BERT, RoBERTa, GPT-2, and fixed sinusoid embedding (the method used in the original Transformer).



4.1.2 Encoder and Decoder

Encoder: The encoder consists of N = 3 identical layers (in the original paper, the author used 6 layers instead of 3). Each of these layers has 2 components: the first is a Multi-head Self-attention layer and the second is a simple Feed forward layer. Residual connection is also applied alongside layer normalization to reduce overfitting.

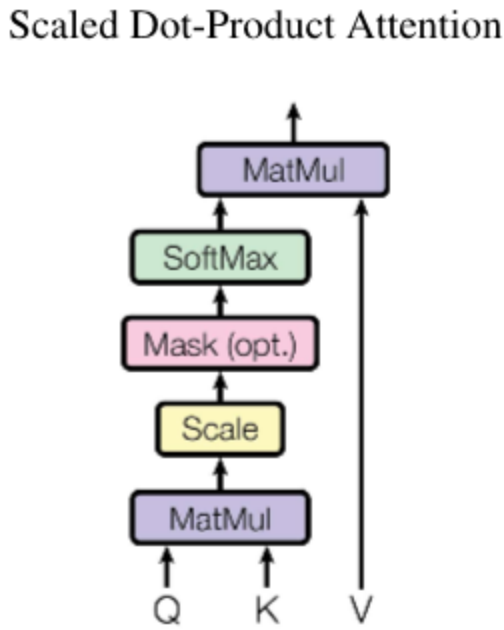
Decoder: Similar to encoder, decoder also consists of N = 3 identical layers. Other than 2 sub-layers used in each layer of the encoder, each layer of the decoder also has a third sub-layer, which performs Multi-head Attention over the output of the encoder stack. To avoid giving out information of subsequent positions, the Multi-head Self-attention sub-layer is modified using masks and the output sentence is also shifted one position to the right.

- **Scaled Dot-product Attention**

The attention mechanism used in this model is called Scaled Dot-product Attention. For each input embedding, we have 3 projections: query, key and value. All queries are packed into a matrix Q, and similarly, all keys and values are packed into 2 matrices K and V respectively. The attention matrix is computed as follows:

$$Attention(Q, K, V) = softmax(Q \cdot K^T / \sqrt{d_k}) \cdot V$$

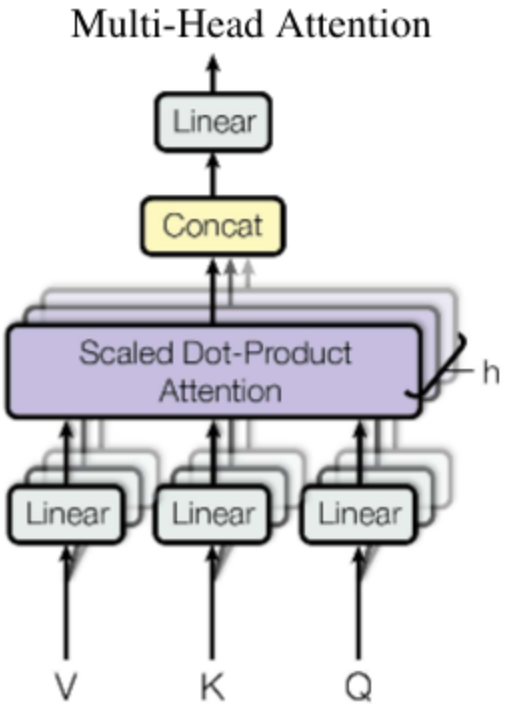
This is very similar to Dot-product Attention, except for the scaling factor $1/\sqrt{d_k}$. Given that the dimensionality of Q and K is d_k, and all elements in vectors Q and K are independent random variables with mean 0 and variance 1, then their dot product $QK^T = \sum_{i=1}^{d_k} q_i k_i$ will have mean 0 and variance d_k. In order to make the variance 1, we divide the random variable by its standard deviation $\sqrt{d_k}$.



The idea of dot product attention is analogous to information retrieval systems, where queries are matched with keys to retrieve the information value. Indeed, when Q and K are normalized (i.e. norm=1), their dot product is the same as their cosine similarity. The softmax function turns the similarity into probabilities, effectively telling how much each word needs to attend to others. The dot product will then be multiplied with the value vector V to get new embeddings.

- **Multi-head Attention**

Instead of performing attention on a single d_{model} dimension vector, the author projects key, value and queries to h different learned projections in order to utilize multiple attention heads. Later research shows that each attention head learns a different way in which words correlate to each other.



4.1.3 Output layer

Output layer takes output vector of decoder, feeds them to a simple Feed-forward layer to calculate the probability that a word V appears in position S.

Here, the output of the final feed-forward layer will return to us a vector with the size of the number of words in the target language's vocabularies, which predicts the probabilities of each words at the decoding step.

4.2. Decoding

After passing the source sentence through the Transformer network, to generate translation we implement the popular Beam Search algorithm for decoding, with a beam size $k = 5$.

4.3. Loss Function

The loss function used by us in this project is a standard Cross Entropy loss with Label smoothing.

Label Smoothing (LS) is a commonly used technique that improves the generalization ability of NMT models. At training time, the model makes a prediction over the target

side vocabulary at each time step t, conditioned on source inputs x and prefix strings $y_{1...t-1}$:

$$p_i^t \triangleq p(y_i^t|x, y_{1...t-1}), i \in [1, 2, ...V]$$

where V is the size of vocabulary. The loss is the cross-entropy between $p^t = [p_1^t, ..., p_V^t]$ and the ground-truth distribution $q^t = [q_1^t, ..., q_V^t]$:

$$L_t = -\sum_{i=1}^{V} q_i^t log(p_i^t)$$

To prevent overfitting on the hard 0-1 label distribution, LS softens the distribution via interpolation between q and a uniform distribution:

$$q' = (1 - \alpha)q + \alpha/V$$

where α is a hyperparameter set to a small value. In NMT training, α is commonly set to 0.1. (In our implementation, we also set it to 0.1)

So the final loss function at each time step in our implementation will be:

$$L_t^{'} = -\sum_{i=1}^{V} q_i^{'t} log(p_i^t)$$

5. Data Augmentation techniques

In this part, we will discuss various techniques examined in this project. In total there are 5 different techniques of generating auxiliary sentence pairs, some use Deep Pre-train Neural Network while others are rule-based.

5.1 Synonym replacement

Wordnet is a large English lexical database created in the Cognitive Science Laboratory of Princeton University. Words in Wordnet are grouped into sets of synonyms (synset). To augment our English dataset, we use synset to look up the synonym of a word and replace that word with its synonym in the sentence.

However, we cannot do the same for our Vietnamese because the Vietnamese Wordnet is not available to us. Therefore we propose a novel text augmentation technique that leverages two other techniques in NLP: word embedding and masked language modeling

Word Embedding:

Word embedding is a learned representation for text processing that makes words that have similar meanings also have similar representation. Before the invention of word embedding, the most prominent word representation is one-hot vectors: each word in the vocabulary is represented on a separate dimension. This method has a lot of downsides: Firstly, the vector representation of each word is of thousands or tens of thousands of dimensions, which makes the model computationally expensive, and secondly, the representation does not show that similarity or dissimilarity between words. Word embedding solves these issues by representing each word as a real-valued vector of only hundreds of dimensions, each dimension corresponds to a characteristic of words.

Masked LM Substitution:

In recent years, the popularity of pre-trained Language Models continues to rise due to the success of BERT (Bidirectional Encoder Representations from Transformers). The original paper was considered by many to be the most significant breakthrough of Deep Learning in recent years. BERT's architecture is actually the same as Transformer but with the removal of Decoder layers. The technique that separated BERT from any other model of the time was the introduction of self-supervised training objectives. BERT was trained on 2 tasks: Masked Language Modeling and Next Sentence Prediction. Masked Language Modeling task requires the model to predict the masked/removed word while Next Sentence Prediction requires the model to predict whether sentence B is the next sentence of sentence A or not. These pre-train tasks enable the model to be trained on large text corpuses such as BookCorpus or English Wikipedia. This coupled with the powerful attention module of Transformer creates a powerful pre-trained model that can be finetuned on downstream tasks.

However, BERT is an English monolingual model, which means it cannot be used for Vietnamese. In 2020, PhoBERT, the Vietnamese version of BERT, was published by VinAI Research. Due to its monolinguistic nature, PhoBERT consistently outperforms XLM-R, a multilingual version of BERT, when finetuned on downstream Vietnamese NLP tasks. In our case, we directly use PhoBERT for the Language Modeling task.

Methodology

We randomly substitute 30% of the total number of words in each English and Vietnamese sentence.

English data replacement: we use the ntlk library as the Wordnet interface to access synset. Synonyms of a word can be looked up like so:

```
>>> wn.synsets('dog')
[Synset('dog.n.01'), Synset('frump.n.01'), Synset('dog.n.03'), Synset('cad.n.01'), Synset
('frank.n.02'), Synset('pawl.n.01'), Synset('andiron.n.01'), Synset('chase.v.01')]
```

Depending on the context, a word can have a different part of speech. As a result, the meaning of the word can vary based on whether it is a noun, a verb, or an adjective, etc. If we only provide the word without any other information, the list of synonyms will contain a lot of elements, most of which do not match the semantics we are looking for. However, we can obtain results which meaning are a lot closer to the original word if we also provide the part of speech:

```
>>> wn.synsets('dog', pos=wn.VERB)
[Synset('chase.v.01')]
```

Therefore, we also extract the part of speech of the word to be replaced (using the nltk POS tagger) and pass it to synset to obtain a list of replacing candidates that have the same part of speech. Among these candidates, we randomly picked one to replace the original word.

In many cases, synset returns to us multiple words that are synonymous to the original word.

A problem with Wordnet synset is that it only gives back the lemma of the word, i.e. the base form of the word (”shelves” → “ledge” instead of “ledges”). Using the information

given by the POS tagger, we managed to retain the correct form of the word (pluralize nouns, conjugate verbs, change adjective comparative, superlative, etc)

However, we can still end up with many synonyms for one word. To pick out the best replacement, we calculate the cosine similarity of the Word2Vec candidates with the original words, and choose the word with the highest score. We will use the English Word2Vec trained on Google News data implemented by gensim.

Vietnamese data replacement: We replace the desired word with the <mask> token and pass it to PhoBERT. Then PhoBERT will give us a list of possible candidates that we can use to replace the original word.

These candidates are ranked based on the cosine similarity with the original word on the Word Vector Space. The Word2Vec representation we used is provided in . The candidate with the highest similarity score will be chosen for substitution.

Below are a few examples. In some examples. the semantic meaning is preserved, which is what we aim to do. However, in some cases, even though the new word and the original word have roughly the same meaning but after the substitution, the new sentence does not sound natural. In more severe cases, the meaning of the sentence is slightly changed.

Original	Augmented
Another thing that helped me as a little kid is , boy , in the ' 50s , you were taught manners	Another matter that aided me as a petty nipper is , boy , in the ' 50s , you were taught manners .
It makes me to stand here , the fame , the money I got out of it .	It gains me to bear hither , the renown , the money I caught out of it .
And actually I could never raise awareness by myself , no matter what I 'd achieved .	And really I could never kindle consciousness by myself , no topic what I 'd accomplished .
Cả hai đều là một nhánh của cùng một lĩnh vực trong ngành khoa học khí quyển .	Cả hai đều là một nhánh của cùng một lĩnh vực trong lĩnh vực khoa học vũ trụ .
Những bộ phim được biên đạo và bối cảnh hoá .	Những bộ phim được chỉnh sửa để bối cảnh hoá .
Nó tồn tại . Nó ở đó , và bạn biết không ? Không sao !	chúng ta tồn tại . Nó còn đó , và tôi yêu nó ? Không sao !

Sự tác động của vòng đời cây cải này thật sự to lớn	những tác động của vòng đời cây cải là vô cùng to lớn
Muốn món gì đó thì chỉ đơn giản là mua nó	thích món nào đó thì chỉ đơn giản tìm mua nó
I was burned very badly .	I was burned very severely .
One of the new ones is the automobile .	One of the new we is the machine .
I got a lot of support from all around the world .	I acquired a stack of reinforcement from all around the world .

5.2 Back Translation

Obtaining labeled data for Machine Translation is not a simple task, especially for low resource language pairs. In contrast, monolingual data is much more available, much more numerous and can be easily collected. Back Translation is a data augmentation technique that can make use of monolingual data to generate sentence pairs.

For the augmenting data, we use 2 datasets. First EVBNews, built by Ngô Quốc Hưng, consists of 1000 word-aligned parallel news articles (we only take the Vietnamese text). The second one is the FPT Open Speech Dataset, consisting of 26k Vietnamese sentences as transcripts for recorded speeches. After merging these 2 datasets and removing sentences that contain more than 70 words for better augmentation performance, we end up with a dataset of 70k Vietnamese sentences.

- EVBNews: https://github.com/qhungngo/EVBCorpus
- FPT open speech dataset: https://data.mendeley.com/datasets/k9sxg2twv4/4

We used a pretrained T5 Vietnamese to English translation model by NLP HUST group available on HuggingFace. We will take 65k pairs out of this augmented dataset.

While we don’t have any official way to evaluate our translation quality, after taking a look through some of the augmented data, the results look quit great enough to us

Original Vietnamese	English
Quân đội mỹ muốn chế tạo những máy tự động hoàn toàn vì những chiếc mà họ đang có vẫn cần được điều khiển bởi con người	American troops want to build complete automated machines because the machines they have still need to be controlled by humans.
Theo Freddie Mac , cơ quan cho vay mua nhà của	According to Freddie Mac, the U.S.

chính phủ Mỹ , có vài tin tốt lành cho thị trường nhà đất , lãi suất bình quân cho các khoản thế chấp 30 năm giảm xuống mức thấp kỷ lục 4,15% trong tuần trước .	government 's housing lending agency, there are some good news for the housing market, with the average interest rate for 30 - year mortgages falling to record lows of 4. 15 % in the previous week
Hôm thứ năm ủy ban bầu cử thái lan cho rằng đảng của người thái yêu người thái đã phạm hai luật trong chiến dịch tranh cử vừa qua	On Thursday, the Thai electoral commission stated that the party of the Thai patriots had violated two laws in the recent campaign.
đầu gối có bị sưng đỏ hoặc cả hai	knee has red swelling or both

We can see that this method generally yields good quality English-Vietnamese pairs, with the English translation being both quite close in meaning compared with the Vietnamese sentences, and being quite grammatically correct. With this in mind, we expect this method to have a great performance.

5.3 English - France Back Translation

The main idea of this approach is as follows:

- Take a set of English sentences l_{en} (65k in our particular case) from our original English corpus, and translate them into French sentences l_{fr} .
- Translate those French sentences l_{fr} into English back translations l_{backtr}.
- These English back translations l_{backtr} serve as augmented data on the target side for our Machine Translation task, with the matching source sentences corresponding to the original English sentences that it originates from.

To perform this method, we use two pre-trained Transformer models by the NLP group at The University of Helsinki*,* through the *HuggingFace Transformers library.* We use an EN - FR model to generate French translations of our chosen English sentences, and an FR- EN translation model to translate the French sentences back to English.

Reference for EN-FR model: Helsinki-NLP/opus-mt-en-fr · Hugging Face

Reference for FR-EN model: Helsinki-NLP/opus-mt-fr-en · Hugging Face

The MarianMT model is a Transformer with 6 Encoder and 6 Decoder layers, with Sinusoidal Positional Encoding with an embedding dimension of 512. Also, we will set

the maximum sequence length the model accepts 512, and truncate the input sentences whose length exceeds this threshold.

Since the two models have already displayed great performances, we will only choose beam size equals to 1, this also helps reduce the inference time. After augmenting, the back translation corpus achieves a BLEU score of 57.13 compared to the original English sentences that they originate from.

By applying this method, we hope we will be able to generate augmented data that's close to the meaning of the original sentence with slight adjustments. Here's a few examples example:

Original	Back translations
When I was a kid , I wanted to be a man .	When I was a child, I wanted to be a man.
Of course not ! You've got to practice 24 / 7 .	You must practice 24 / 7.
An area that was edgy-artsy is now starting to become much cooler and engage a lot more people .	A zone that was edgy-arty is now beginning to become much cooler and engage many more people.
And most people draw something like this .	And most people draw something like that.
Nobody's ever done it before , so I'm going to go do it .	Nobody's ever done it before, so I'm will go do it.

After taking a look at the translations, we can see that the augmented data retains most of the meaning from the original sentences and are grammatically correct (with some exceptions).

Since our augmented data are relatively close in meaning to the original sentences without a lot of noise, we expect our model to perform well when trained with this augmented data.

5.4 Contractions

- This approach is a modified version of "Paraphrases generation using regular expressions" in the paper "Text Data Augmentation Made Simple By Leveraging NLP Cloud APIs" by Claude COULOMBE.

- For English, the paper goes into details about using regular expression to transform a verbal form into a contracted form (contraction) as a means of paraphrasing the original sentence:

“The first type of transformation that comes to mind is surface transformations. A surface transformation is a transformation that ignores syntax and is done with simple pattern matching rules. Regular expressions (regex) are powerful tools to manage transformations based on pattern matching.

The surface transformations that can be produced with regular expressions are to be preferred because they are simple and very efficient in terms of computation.” ...

“For example, in English, the transformation of a verbal form into a contracted form (contraction) and its inverse (expansion) is a semantically invariant transformation provided that any ambiguity that can lead to a potential misinterpretation is not resolved. Obviously, this happens in the context of mechanical and local transformations where there is no way to resolve ambiguities

Examples of a text surface transformation
The transition to a contracted verbal form and its inverse is a semantically invariant transformation provided that the ambiguities are preserved.
`I am => I'm, you are => you're, he is => he's, it is => it's, she is => she's, we are => we're, they are => they're,, I have => I've, you have => you've, we have => we've, they have => they've, he has => he's, it has => it's, she has => she's, I will => I'll, you will => you'll, he will => he'll, are not => aren't, is not => isn't, was not => wasn't, ..., I'm => I am, I'll => I will, you'll => you will, he'll => he will, aren't => are not, isn't => is not, wasn't => was not, weren't => were not, couldn't => could not, don't => do not, doesn't => does not, didn't => did not, mustn't => must not, shouldn't => should not, can't => can not, can't => cannot, won't => will not, shan't => shall not`

In order to preserve the semantic invariance, it is allowed to introduce ambiguities but it is forbidden to resolve ambiguities that could lead to misinterpretation. For example the transformations “he is” to “he’s” and “he has” to “he’s” will be allowed even if they introduce ambiguous sentences. But the inverse transformations from “he’s” to “he is” and “he’s” to “he has” are forbidden because they could introduce a misinterpretation by solving an ambiguity without justification.”

The original approach uses regular expressions to contract the text; as mentioned, this will save computing power but at the cost of sacrificing the ability to apply contractions grammatically correctly and effectively. However, we decided to improve on how

contractions should be applied based on the Cambridge Dictionary's webpage of Contractions. The summary of the webpage forms the base on which we would build our approach:

1. Use auxiliary verb contractions with nouns, pronouns.
2. Use "not" contractions with auxiliary verbs (don't use for pronouns).
3. Don't use more than one contraction.
4. Use negative contractions at the end of clauses and we do commonly use contractions in tag questions.

We use a part of the Pycontractions library for the list of contractions and the regular expression implementation.

In our implementation, we used the pre-trained pipeline **en_core_web_sm** from the spaCy library for dependency parsing.

In order to help the dependency parser, we'll change the special characters ("apos;", "quot;", etc.) to their original characters (', ", etc.). After augmenting, we'll change them back.

We will generalize our approach here but the exact implementation will be in the Notebook "Contractions Augmentation":

- Assume that all input texts are grammatically correct.
- Perform these steps for each line:
    1. Contractions of auxiliary verbs after pronouns:
        a. Use **en_core_web_sm** to dependency parse and extract pronouns with its root dependency tag to be "nominal subject" or "passive nominal subject".
        b. Contract the auxiliary verb if it appears after the pronoun and if it is applicable to do so.
    2. Contractions of auxiliary verbs after demonstratives, modal verbs (bar "can"), and question words: Check each demonstrative, question word or modal verb in the sentence and see whether the word after it is an auxiliary verb; If true, contracts the auxiliary verb.
    3. Contractions of not: Use regular expression to replace accordingly.

4. Contractions of auxiliary verbs after other nouns: Same as 1.; instead of pronouns, we contract other nouns.

After augmenting, 46466 lines had been contracted. This amount will also be used to augment the data.

We use the dependency parser to ensure that the current noun associated with the auxiliary verb is the subject of the sentence. This will solve the 1st, 2nd points above. The 4th point will be solved if we restrict the 1., 2., 4. step from contracting auxiliary verb if there is punctuation after the auxiliary verb. E.g: *"That's just the way it is."* not *"That's just the way it 's.".*

The 3rd point is hard to resolve due to the inconsistencies between the training data of the dependency parser and our data.

- E.g. From our data: “It doesn 't have feelings ... yet .” With this sentence, the dependency parser identifies “t” as a noun which it is not.
- We will restrict contracting if the noun has any apostrophe in its last word.

Original	After Contractions (before turning back special characters)
And one of the molecules I study is called isoprene , which is here .	And one of the molecules I study is called isoprene , which 's here .
Indeed , it is hard to find a subject that film has yet to tackle .	Indeed , it 's hard to find a subject that film 's yet to tackle .
But it is not part of metabolism itself .	But it 's not part of metabolism itself .
And I have come to the belief -- this is my 12th year doing this research -- that vulnerability is our most accurate measurement of courage -- to be vulnerable , to let ourselves be seen , to be honest .	And I 've come to the belief -- this 's my 12th year doing this research -- that vulnerability 's our most accurate measurement of courage -- to be vulnerable , to let ourselves be seen , to be honest .
And that 's what I 'm going to show you , because since 1960 what has happened in the world up to 2010 is that a staggering four billion people have been added to the world population .	And that 's what I 'm going to show you , because since 1960 what 's happened in the world up to 2010 is that a staggering four billion people 've been added to the world population .

6. Experimentations and Results

6.1. Experiments

In this Project, we conducted 2 experiments surrounding English-Vietnamese Machine Translation. The methods being compared in each experiments will be trained with the same model under the same setup. Their performance will be evaluated using the BLEU score using the library *sacreBLEU.*

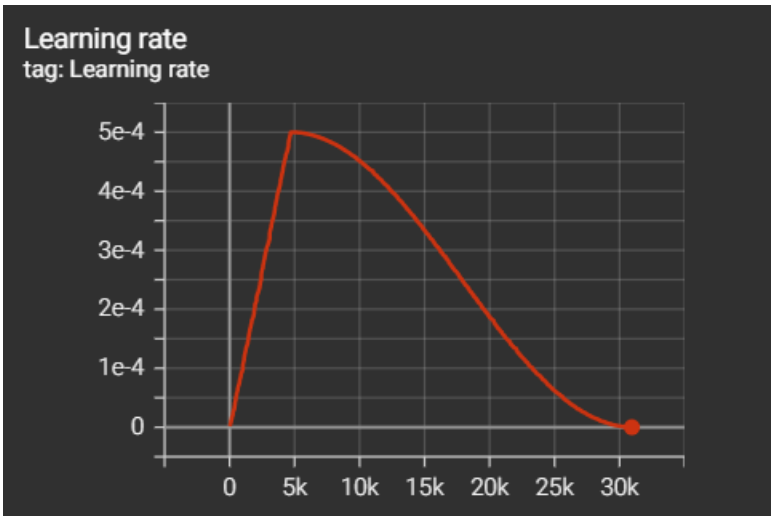
Experiment 1 : Tokenization methods

Like we've stated above, we will experiment with these 3 tokenization methods by training Transformer models with them as input. These models will have the same architectural hyper-parameters and trained under the same settings to ensure a fair evaluation.

The architecture used is a Transformer with 3 Encoder and Decoder layers, with 8 Attention heads, an embedding dimension of 512, and the dimension of the feed-forward layer is 4 implemented with PyTorch.

The models are trained for 15 epochs with a learning rate of 0.0005 and batch size 64 on Google Colab using GPU runtime. The optimizer used is the standard Adam optimizer with β_1 = 0.9, β_2 = 0.999 and ϵ = 1e-08 .

The learning rate scheduler is the Cosine scheduler with Warm-up, with the number of warm-up steps being 15% of the total steps required for the whole training process.



The model evaluates the training loss and the evaluation loss on the Dev set every 250 steps, and the model's current state is saved if the evaluation loss achieves the best performance.

We use the Beam Search algorithm, with beam size k = 5 for the inference phase.

Experiment 2 : Data augmentation Techniques

For each method mentioned above, we generated 65k sentence pairs (if possible), which is roughly 1/2 of the size of the original training set, then concatenate them with the original training set. This leaves us with datasets containing 198317 pairs of sentences for each augmentation method. Similar to experiment 1, we train each of these augmented datasets, alongside the original unaugmented data, from scratch with a Transformer model with the same architectural and training phase hyperparameters. After conducting experiment 1, we found word segmentation to be the one with the best performance. All of the models trained on the augmented datasets will have word segmentation as its tokenization technique.

The architecture used is a Transformer with 3 Encoder and Decoder layers, with 8 Attention heads, an embedding dimension of 512, and the dimension of the feed-forward layer is 4 implemented with PyTorch.

The models are trained for 20 epochs with a learning rate of 0.001 and batch size 64 on Google Colab using GPU runtime. The optimizer used is the standard Adam optimizer with β_1= 0.9, β_2 = 0.999 and ϵ = 1e-08 .

The learning rate scheduler is the Cosine scheduler with Warm-up, with the number of warm-up steps being 15% of the total steps required for the whole training process.

The model evaluates the training loss and the evaluation loss on the Dev set every 250 steps, and the model's current state is saved if the evaluation loss achieves the best performance.

We use the Beam Search algorithm, with beam size k = 5 for the inference phase.

6.2 Experimental Results

Experiment 1 : Tokenization methods

Tokenization technique	BLEU (uncased)	BLEU (cased)
Simple tokenization	25.19	24.37
Word Segmentation	**25.83**	24.69
BPE	25.56	**24.79**

The table above denote our Transformer model's performance with different tokenization technique as input. For uncased BLEU, word segmentation had the best result with a score of 25.56, while for uncased BPE performed the best with BLEU being 24.79. This is to be expected, since BPE ensures that the most common words are represented in the vocabulary as a single token while the rare words are broken down into two or more subword token, striking a balance between the issues of word-based tokenization (very large vocabulary size, large number of OOV tokens, and different meaning of very similar words) and character-based tokenization (very long sequences and less meaningful individual tokens). And for word segmentation, it helps preserve the structure and meaning of multi-syllables Vietnamese words, which is plenty in the Vietnamese vocabulary, thus it comes as no surprise that it can help us to generate better translation compared to the vanilla tokenization.

Experiment 2 : Data augmentation Techniques

Augmentation technique	BLEU (uncased)	BLEU (cased)	Able to generate 65k pairs?
None (Word Segmentation)	25.83	24.69	——
Contractions	26.84	25.71	No

English-French Translation	27.53	26.34	Yes
Synonym replacement	27.56	26.39	Yes
Back Translation	**29.01**	**27.69**	Yes

As we can see from Table 2, all 4 augmentation techniques improved the overall performance of our baseline model significantly by at least 1.01 for uncased BLEU and 1.02 for cased BLEU. Back Translation proved to be the best performing augmentation method, with a 3.18 improvement for uncased.

Back Translation proved to be the most effective method, outperforming all the others with an uncased BLEU of 29.01 and 27.69 in cased BLEU. We hypothesized this is due to the fact that this method helped us synthesize great sentence pairs from a monolingual data given a good back translation model.

6.3 Augmentation Commentaries

To compare Augmentation approaches to a further extent, we will briefly go over the availability of each approach as well as give our thoughts on what could be done to improve the performance:

- Synonym replacement: The strength of this approach is that it can be used to create a large number of sentences without the need for large datasets, meaning, it can be used for both high as well as low-resource languages. Despite the problems we mentioned in previous sections, it is still one of the best approaches in terms of performance. As stated, there are 2 main problems: slight changes in semantic and unnaturalness. We observe that the more substitutions made in one sentence, the more unnatural it sounds. To tackle the issue of unnaturalness, we think that the longer the sentence, the lower the percentage of words replaced should be. Moreover, we should also employ a Sentence BERT model which is proven to perform very well regarding the embedding of a whole sentence, i.e sentences that have the same meanings also have similar embedding. After replacing words with synonyms, if the semantic is not preserved, we will discard the new sentence or generate another sentence.
- Back Translation: The Back translation method showed the most improvement in BLEU score compared to the model trained on the original unaugmented dataset. As we can see from the augmented data above, the quality of the translation is very high thanks to the pretrained model. Another point to consider is that new

information are also introduced into the model via data from the augmenting datasets. This method is especially useful when you have access to limited bilingual data since the method only requires monolingual data of the source language, which is much more abundant compared to monolingual data. However, we should prioritize choosing augmented data in the same domain as the original training data to generate better translations.

- English - French Back Translation: This method showed least improvement compared to the previous 2 methods. This could be partially attributed to the great quality of the pretrained models available for the EN-FR and FR-EN sentence pairs. The back translations are quite close in meaning to the original data, most sentences are grammatically correct. However, this also means that the diversity obtained in the previous 2 methods are not found here. Another downside of this approach is that it's the most time and resource consuming method. This is mostly due to the fact that we have to generate the translations twice, from English to French and vice versa. Translating from English to French took around about 3 hours, while translating from French back into English took about 2.5 hours for the 65k sentences. Also, applying multiple back translations for multiple language pairs of high resource languages might be a good idea as well (EN - GR, EN - ZH,...)
- Contractions: Time-wise, this approach is very effective since we're only dependency parsing and modifying the sentences (40 mins. which can be further optimized). We can see small improvement in BLEU scores but caution is still needed because this approach doesn't add any meaning to the sentence, it only hides information in a grammatically correct way (which also generates less noise than other approaches); secondly, the amount of sentences generated is limited by the original data; the format of the dependency parser's training data also matters as well. As for improvements, there might still be additional grammar rules for contractions that we have not implemented. But overall, an easy to apply approach for English to Low Resource Languages translation with datasets of informal speech.

7. Conclusion

In conclusion, in this project we explored the task of Machine Translation by implementing a NMT system with the Transformer model. We also tried out different tokenization algorithms, and evaluated their performance on our Machine Translation system. Both word segmentation and BPE proved to be better compared to the simple tokenization technique.

We also researched and implemented four different data augmentation methods for Machine Translation, which includes Synonym Replacement, Back Translation, English - French Back Translation and Contractions. All methods proved to be effective in improving our baseline model, with improvements of at least 1.01 BLEU score. Back Translation ended up outperforming all the other methods

We also discuss the advantages and drawbacks of each approach as well as propose ways to improve on these methods. For future work, we would like to implement and examine the effectiveness of these proposals.

References

Sennrich, Rico, Barry Haddow, and Alexandra Birch. "Neural machine translation of rare words with subword units." *arXiv preprint arXiv:1508.07909* (2015).

Vaswani, Ashish, et al. "Attention is all you need." *Advances in neural information processing systems*. 2017.

Wordnet

Thanh Vu, Dat Quoc Nguyen, Dai Quoc Nguyen, Mark Dras and Mark Johnson. 2018. VnCoreNLP: A Vietnamese Natural Language Processing Toolkit. In Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Demonstrations, NAACL 2018, pages 56-60. bibtext, github

Mikolov, Tomas, et al. "Efficient estimation of word representations in vector space." *arXiv preprint arXiv:1301.3781* (2013).

Devlin, Jacob, et al. "Bert: Pre-training of deep bidirectional transformers for language understanding." *arXiv preprint arXiv:1810.04805* (2018).

Nguyen, Dat Quoc, and Anh Tuan Nguyen. "PhoBERT: Pre-trained language models for Vietnamese." *arXiv preprint arXiv:2003.00744* (2020).

Junczys-Dowmunt, Marcin, et al. "Marian: Fast neural machine translation in C++." *arXiv preprint arXiv:1804.00344* (2018).

Coulombe, Claude. "Text data augmentation made simple by leveraging nlp cloud apis." *arXiv preprint arXiv:1812.04718* (2018).

NLTK toolkit

EVB Corpus

Definition of contraction, according to Cambridge dictionary

pycontractions

spaCy NLP toolkit

Pre-trained EN-FR Transformer model

Pre-trained FR-EN Transformer model

YouTokenToMe

sacreBLEU

Member Assignment

Phạm Văn Cường:

- Implement code for Transformer model
- Implement train function for model
- EN-FR Translation augmentation

Nguyễn Tuấn Dũng:

- 50 % Preprocessing
- Implement Beam Search
- Synonym Replacement augmentation

Phạm Cát Vũ

- 50 % Preprocessing
- Contractions augmentation
- Back Translation augmentation