Master's Thesis

Incorporating syntactical information to improve Neural Machine Translation

Trieu Hoang An

Supervisor Nguyen Le Minh

Graduate School of Advanced Science and Technology
Japan Advanced Institute of Science and Technology
(Information Science)

02, 2023

Abstract

Recently, researches about machine translation have reached impressive results and can be applied in practical, thank to the development of deep learning model, especially the Transformer model. The Transformer model allows us to build an end-to-end translation system straight from the datasets. However, there are some issues that still exist even with the participation of the Transformer. Those issues are the lack of dataset in minor language and the loss of information while translating long sentences. There is one approach to face with this issue which is incorporating syntactical information into the Transformer model to enrich information from the small dataset in minor language and also help the model to get more information in translation step.

In this research, we proposed the methods to inject the syntactical information such as TF-IDF score for pair of words and Part of speech tags into the Transformer model. With each type of information, we have designed different type of attention mechanism to incorporate those information into Transformer effectively. Our target is to highlight the relationship between the tokens inside the input sequence, since then, enhance the quality of translation process.

To evaluate the methods, we did experiments on two benchmark dataset which are IWSLT 2015 English-Vietnamese and IWSLT 2014 English-Germany and the results is quite positive.

Keywords: Deep Learning, Machine Translation, Transformer, Syntactical information.

Acknowledgement

For all the time since I started studying as a Master student, I have received so much support and caring that I cannot describe through any words.
Firstly, I want to show the highest appreciation to my main supervisor, Associate Professor NGUYEN LE MINH. It is him who gave me the chance to study at JAIST and also always supported me as well as gave me many precious advice to help me get through many difficult moments.
Secondly, I would like to express my sincere gratefulness to Associate Professor NGUYEN VIET HA and Dr. NGUYEN THI NGOC DIEP for guiding me in researching and studying from the very first days of my master program.
Thirdly, I would like to thank Dr. NGUYEN MINH PHUONG, Mr. BUI MINH QUAN, Mr. PHAM TRUNG TIN and all other members of NGUYEN lab for their support in my research as well as in my social life at JAIST.

On this occasion, I want to thank all of my friends in JAIST and in Vietnam for all of the memories we shared, all of the fun and sorrow we have been though. All of those were the motivation pushing me to move forward on my own path.

Finally, I want to send love to my precious family, my father, my mother and my brother who always believe in me and encourage me so I can get through the hardest period of my life. You are the best family anyone can get in this life.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Machine Translation (MT) is a natural language processing (NLP) topic that aims to automatically map sentences or words from the current human language to other human languages with the minimum change in the meaning or the content of those sentences. There are three approaches to machine translation methods which are Rule-based Machine Translation (RMT), Statistical Machine Translation (SMT), and Neural Machine Translation (NMT). Rule-based Machine Translation exploits the stability in the syntax of languages to map the 2 sentences of each language together. While SMT focuses on statistical relationships between original texts and their existing human translations, NMT applies Deep Learning (DL) models to analyze the relationship between the sentences in the source language and the target language. Recently, both approaches have achieved impressive results in popular languages that are applied in many systems, services, and applications. These applications and services are used widely by people around the world.

1.1 Problem statement

SMT exploits the statistical information from the monolingual dataset or multilingual dataset to predict the sentences in the target language which are similar to the sentences in the source language about meaning. On the other hand, NMT can analyze the hidden meaning between the words in the source sentences and then map that meaning to the target language to generate the "translated sentence". With the improvement of DL architectures, such as Long-Short Term Memory (LSTM), Recurrent Neural Networks (RNN), Transformer, and so on, NMT models can analyze the more complex relationship between the tokens in a sentence, a paragraph, and

even a whole document. Thanks to that, NMT models are getting better results when compared to SMT and can be used in real life. However, there are several issues that exist in NMT models.

- The main challenge of NMT is the lack of data. The performance of the model relies on the coverage of the dataset. So, with the minority language, which is used by small groups of the population all over the world, NMT may not reach the same accuracy as the popular language. The cost for a multilingual labeled dataset in those languages is usually high, and it also takes so much time to create such a dataset.

- The second difficulty of NMT is processing long sentences. Almost all NMT models can generate short and medium sentences effectively but when processing long sentences, some NMT models usually miss a part of the information or translate the source document wrongly. Long sentences may contain too much syntactical information to resolve and the relation between the phrases and tokens in the long sentences may be more complex than in short and medium sentences.

- One of the approaches to deal with the lack of data is exploiting additional information along with the input sentences. The connections between tokens in a sentence can be found in syntactical information such as the appearance frequency of tokens, the dependency relationship between each token or the semantic role of each token. The injection of this information into NMT models helps the model to understand the sentence more clearly. However, the cost for labeling dependency data is very high and the tags generated by dependency parser usually contain errors that may cause bad effects on translation process. Finding a new type of information and suitable method to inject that information to NMT model is still promising.

1.2 Objective

The main purpose of our research is to find a method to inject syntactical information into the NMT model so that the model can have better performance while working with minority languages. As the first step, we want to check the types of information that may help to improve the performance of the model. After that, we develop the technique to inject suitable information into the model. Then, we test the model on the benchmark datasets and observe the results to adjust the current method. For further development, we aim to complete a translation system that can work well on minority languages.

1.3 Originality

In this research, we proposed new methods to inject the syntactical information into the Transformer model. For this research, we chose the TF-IDF score for pair of words and POS tags as the additional information to inject into the Transformer model. With each type of information, we designed a suitable method to inject them into the Transformer model. We did experiments with two methods on the IWSLT2015 English-Vietnamese and the IWSLT2014 German-English datasets to measure the performance of our design in the translation task. The results we achieved are promising and show the improvement in the quality of the translation task. Since then, the paths to incorporate other different kinds of information into the translation model are opened to us.

1.4 Thesis outline

The thesis is organized into 5 chapters. Chapter 1 is the introduction, and the main contents of the next 4 chapters are summarized as follows:

Chapter 2 presents some background knowledge of NMT and related works about injecting additional information into the NMT model

Chapter 3 describes the information we added to the DL model and the method we applied in our experiments.

Chapter 4 shows the dataset we used for experiments and the settings of experiments as well as the results we archived. We will also give some analysis of the experiment results in this chapter.

Chapter 5 is the discussion about the result we achieved and the effectiveness of the method as well as the further development in the future.

Chapter 2

Related Works

2.1 Machine Translation

The concept of a system that can express the same meaning in many different languages was proposed by Rene Descartes in the 17th century. However, until the 1950s, specific research about MT began to appear when Yehoshua Bar-Hillel started his research on MT at MIT in 1951. Since then, there have been many researches and ideas that have contributed to the fast development of MT. From 1951 until now, we can categorize the research of MT into 3 groups based on the architecture of systems: rule-based Machine Translation, Statistical Machine Translation, and Neural Machine Translation.

2.1.1 Rule-based Machine Translation

Rule-based Machine Translation (RMT) is the first design for MT. The main idea of this approach is that every language will have a set of symbols to represent the same meaning. Such as we can usually find a word in a language that is similar to some words in other languages. For example, the word "dog" in English has the same meaning as the word "inu" in Japanese. Thanks to this correspondence between each language we can consider the translation process as a word-replacing process in which we replace the phrases in the source sentence with the phrases of the target language that have the same meaning. This idea is simple and effective in major situations. However, it has several issues that make it unsuitable for application in real situations. RMT depends heavily on the grammar rules that it is built on. However, grammar rules are also hard to organize and there are too many syntax rules in one language. Besides, some syntax rules can conflict with each other and cause mistakes in translation progress. Therefore, rule-based machine translation ignores the context information while translating, which is the most

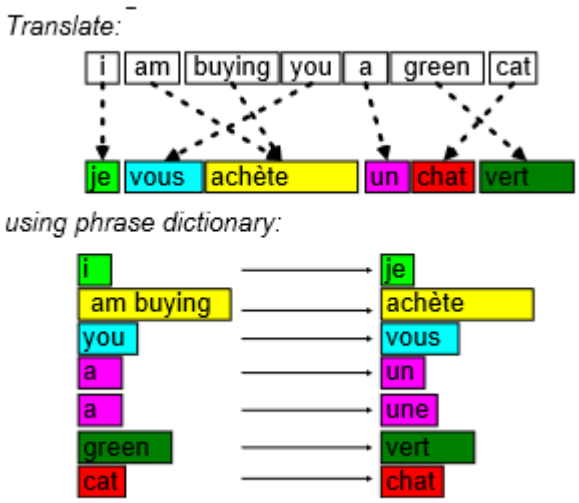

Figure 2.1: Rule-based machine translation

critical drawback. Because in different contexts, the meaning of a word or a phrase of words can be different. For example, in the sentence "He is going to the bank for a loan" the word "bank" should be understood as an organization or a building where people are supported in the financial domain. But in the sentence "He went to the bank for fishing", the meaning of the word "bank" in this case is the land alongside or sloping down to a river or lake. So in the two examples above, we should translate the word "bank" in two different ways because of the contrast in the meaning of the context. However, the system of RMT cannot cover all the situations like these examples. Even in most languages, there are strict grammar structures and rules, but the flexibility of how people use the languages and the flexibility of the languages themselves make it almost impossible to build a set of rules that can cover all of the cases. And even when the rules are organized then which rule should have higher priority might be a hard issue.

2.1.2 Statistic-based Machine Translation

Statistical Machine Translation (SMT) uses a separate approach from RMT to handle the translation task. The key idea of SMT is using statistical information between words and phrases of words to find the same meaning through a bilingual dataset. To translate a sentence, SMT also replaces the words or phrases in the source language with the corresponding words

and phrases in the target language, but, unlike RMT, SMT use statistic as the baseline instead of rules to choose suitable words or phrases in the target language to fill in the place of the old ones in the source language. This approach allows the SMT models to translate more flexibly than RMT models and also the cost for SMT seems to be lesser than RMT models. The main reason is that SMT does not depend on the rules, so to improve the performance of translation models, we just need to improve the statistic and the datasets. That makes models more simple and easier to use in real situations. However, SMT models depend heavily on the datasets that they are trained on. So if the datasets contain a considerable amount of noise, the performance of SMT models can be affected. Besides, the performance of SMT models is weaker when they have to deal with long sentences which have complex contexts.

Since then, we can see that noisy data and long sentence processing are the two major issues of SMT models. Many sub-tasks, techniques, and new SMT models have been proposed to cope with those issues. Many techniques such as pre-processing, sentence alignment, word alignment, phrase extraction, phrase feature preparation, and language model training, are proposed along with Phrased-based Statistical Machine Translation (PBSMT). The PBSMT model generates the translated sentence using the relations between phrases in the source language and phrases in the target language. These relations are not produced by any specific rules but by the statistic learned by the model from bilingual data sets. Therefore, PBSMT achieved an impressive result that overcome word-to-word translation models.

Another factor that helps SMT models outperform the RMT methods is the word embedding technique. The very first idea of word embedding is how to map a word into numbers to serve the statistical operation behind it. The numbers that represent each word must be unique and in some aspect, must reflex the meaning of the represented word. One of the first embedding methods is using the Term Frequency - Inverse Document Frequency (Tf-IDF) score of each word to represent them in calculations. However, TF-IDF did not show all the aspects of a word such as the relationship between that word and the others. So, language models are proposed to solve this problem. The target of language models is to represent the relations between the words in vocabulary and to do that, each language model creates a multi-dimensional new vector space of its own to measure the words in the vocabulary. After that, before being processed by SMT models, the input tokens will be embedded into those vector space to generate a representation vector for each token and the SMT models will do calculations of those representation vectors. The popular language models are word2vec, doc2vec, Glove, fastText, and so on.

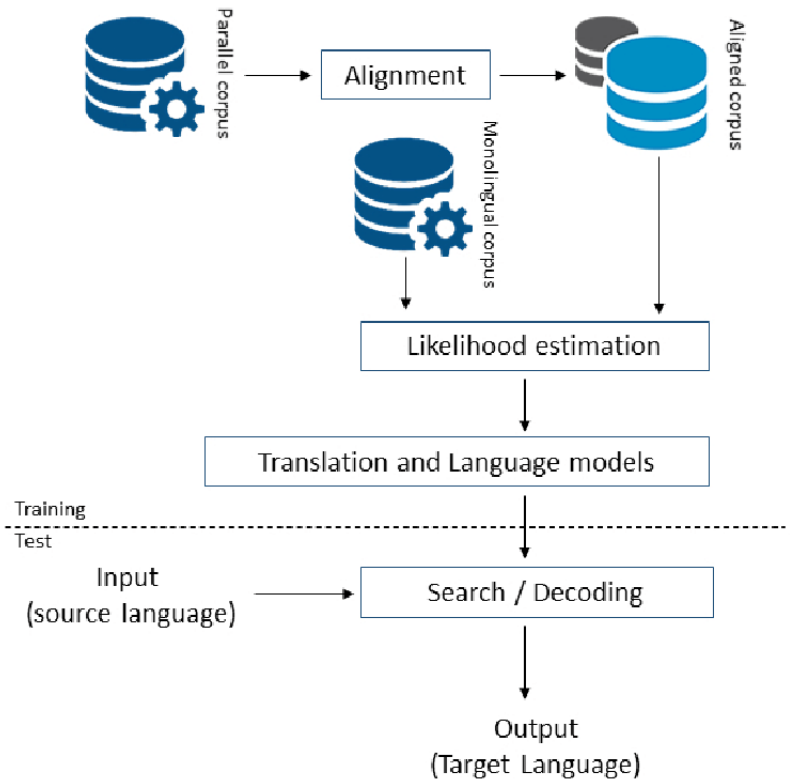

Figure 2.2: Statistical machine translation flows chart

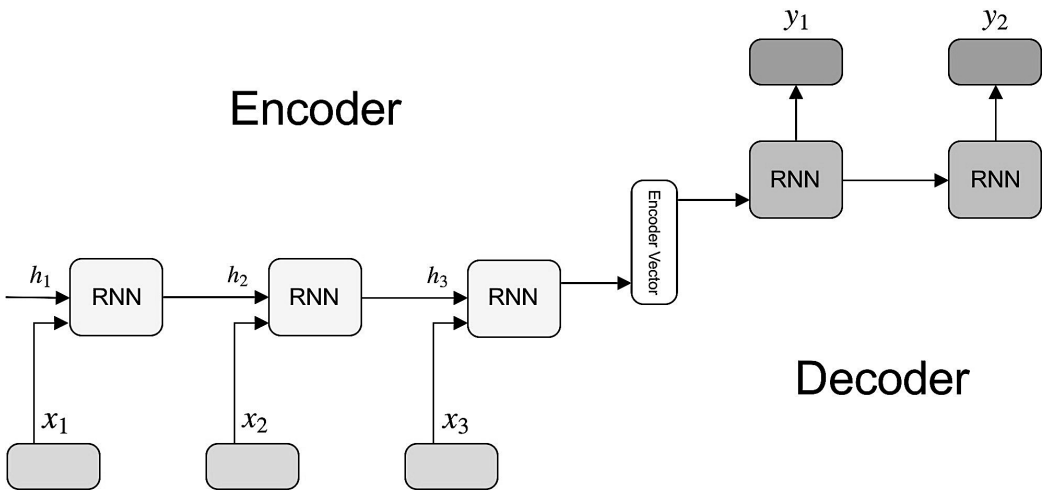

Figure 2.3: RNN structure

2.1.3 Neural Machine Translation

Even though achieved promising results, SMT could not completely understand the context of the input sentence to generate the sentence in the target language with the closest meaning. Besides, SMT models cannot handle the noise in training data totally. Therefore, many researchers proposed to apply deep learning architectures in translation tasks to face those issues. Since then, Neural Machine Translation (NMT) begin to develop and achieved many impressive results. Many variations of the deep neural networks were designed, but almost of NMT models can be categorized into 2 major architectures: Convolution Neural Network-based NMT and Recurrent Neural Network-based NMT.

Recurrent Neural Network-based NMT.

Before being applied to the translation task, Recurrent Neural Network (RNN) is well-known in other NLP topics. The structure of RNN models allows processing the input documents sequentially and reserving the order of each token inside the input. Since then, RNN models can capture the information inside input documents better than other Deep Learning models and soon become dominant in the machine translation field.

RNN models compute the hidden state representation for each step based on the input vector of the current token and the hidden state representation of the previous step. So that the final hidden representation will have the information of the whole sentence as well as the information on the order of each token inside input sentences.

There were many researchers who used the approach that applied RNN

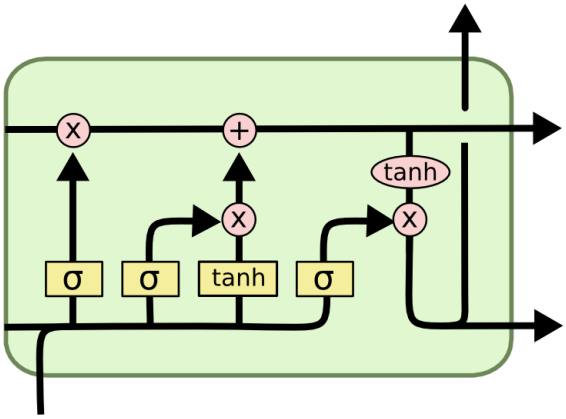

Figure 2.4: GRU cell

structure in machine translation, Sutskever et. al proposed a full RNN model for the translation task and achieved an impressive result when compared with the SMT models [2]. The model used RNN structure in both the encoding and decoding steps to generate the target sentences. With the final representation of the source sentence and the "start sentence" token, the RNN model will compute the first token of the target sentence. Then, the next hidden state representation of the target token will be computed using the currently hidden state representation and the embedded vector of the newly generated token. However, traditional RNN models have a drawback which is the loss of information occurs while translating long sentences. This happens when encoding too many tokens, the representation at the final step does not contain any information about the tokens that lie in the head of sentences. To solve this problem, Long-short Term Memory (LSTM) has been proposed by Hochreiter et. al [3]. LSTM is a variation of RNN in which, each unit is a Gated recurrent unit (GRU) that helps the model select which information needs to reserve and which information needs to forget to retain the expression of the sentence. In 2016 [5], the Fast-Forward Connection for RNN was designed by Zhou et al. based on the LSTM structure. This design allows to construct a deeper neural network for the LSTM model.

In [4], Badanau et. al proposes the attention mechanism that allows RNN models to pay attention to important information to express the meaning of the source sentence. Also, in [10], Luong et al. design an attention mechanism as well. These attention mechanisms both aim to highlight the importance of every token to the expression of the input sentences. However, there is

some slight divergence between them. The Badanau attention mechanism calculates a unique context vector at each step based on the previous decode hidden state representation h_d^{t-1} and encoder hidden state representation h_e^i. With previously hidden state decode representation h_d^{t-1}, we can get the alignment vector $e_{t,i}$ at step t by feeding forward the previous decoder hidden state representation h_d^{t-1} and the hidden state representation h_e^i into a neural network or any activation function F. Then we can calculate the attention weight $a_{t,i}$ by feeding alignment vector $e_{t,i}$ through softmax function. The context vector c_t the is computed as follow:

$$e_{t,i} = F(h_d^{t-1}, h_e^i) = F(W_d h_d^{t-1} + W_e h_e^i) \tag{2.1}$$

$$a_{t,i} = softmax(e_{t,i}) \tag{2.2}$$

$$c_t = \sum_{i=1}^{T} a_{t,i}, h_e^i \tag{2.3}$$

The context vector is then fed together with the embedded vector of the previous decoder output to the RNN unit to generate the hidden state representation for decoding the output of this step. In Luong's attention mechanism, there is a small difference in the computation of the alignment vector. There are 3 methods to compute the alignment vector according to [11] as below:

- **Dot**

  This method is the most simple of the three methods. The alignment vector is the result of dot product multiplication between the encoder hidden representation and the decoder hidden representation.

  $$e_{t,i} = h_d^{t-1} \cdot h_e^i) \tag{2.4}$$

- **General**

  In this method, we add a weight matrix to the dot product between the encoder hidden representation and the decoder hidden representation.

  $$e_{t,i} = h_d^{t-1} \cdot W_a(h_e^i) \tag{2.5}$$

- **Concatenate**

  In this method, the encoder hidden representation is concatenated with the decoder hidden representation and the combined vector is multiplied by the weight matrix W_2 to get a combined representation. The combined representation is then fed to a tanh function and is multiplied with the weight matrix W_1 to get the final alignment vector

  $$e_{t,i} = W_1 \cdot tanh(W_2([h_d^{t-1} : h_e^i])) \tag{2.6}$$

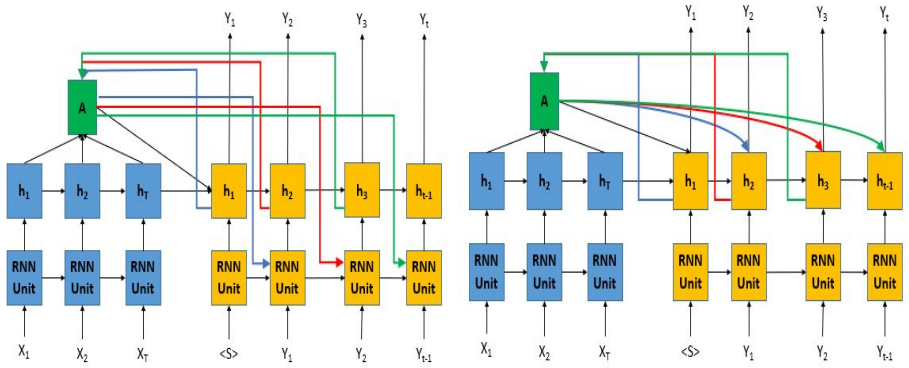

Figure 2.5: Demonstration of attention mechanisms proposed by Badanau et al. (figure on the left) and attention mechanism proposed by Luong and Manning (figure on the right). ¡S¿ is the start sentence token, h is the hidden state representation of each step, A is the attention weight compute from the hidden state representation.

In all of the methods above, the hidden encoder representation and hidden decoder representation are combined together first before being fed into a neural network or activation function, unlike Badanau's attention mechanism. This makes the hidden encoder representation and hidden decoder representation in Luong's attention mechanism share the same weight matrix instead of two separate weight matrices in Badanau's attention mechanism.

The context vector is calculated as in Badanau's attention mechanism. However, later the context vector is concatenated with the hidden decoder representation of this step and is fed forward a linear layer to get the output of the current step.

Convolution Neural Network-based NMT

Convolution Neural Network (CNN) is well-known in many computer vision topics such as object detection, document layout detection, object segmentation and so on. In the other aspect, CNN is also applied in some NLP topics especially machine translation but the performance of the CNN models in MT field seem to be overcome by RNN models for a long period. In the effort to implement CNN architecture into MT task, Kalchbrenner and Blunsom tried to use CNN as the encoder of the translation system and implemented RNN as the decoder.

A full CNN model is used for the translation task performed by Łukasz Kaiser and Samy Bengio[7], they applied Extended Neural GPU on the CNN model so that it can reach the result of RNN models. Concurrently, Kalchbrenner et al. also proposed ByteNet (a kind of CNN) based NMT,

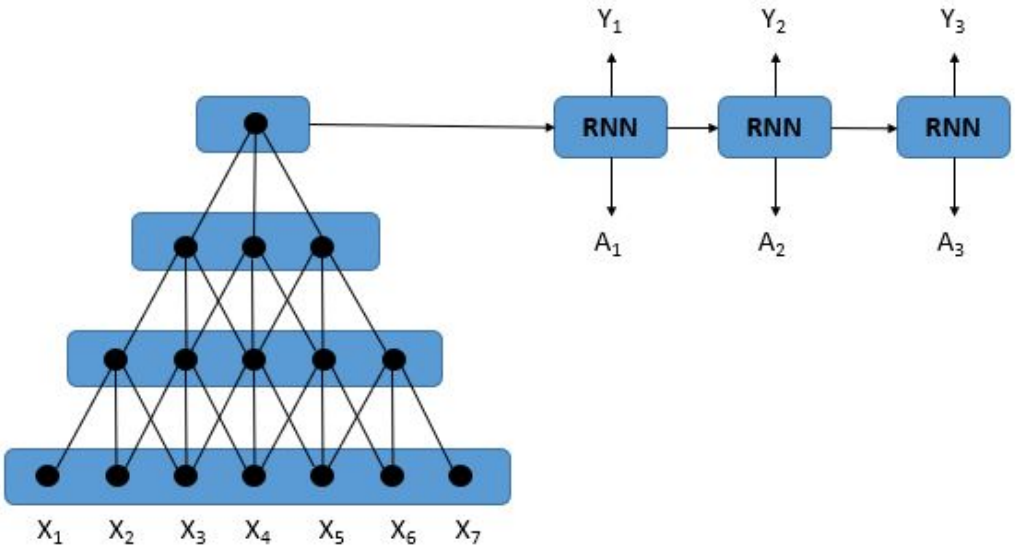

Figure 2.6: demonstration of MT model with CNN encoder and RNN decoder

which show great performance when reaching current state-of-the-art result in character-level translation at that time by the result of models in word-level translation is quite poor [8]. Generally, CNN-based NMT models have the advantage in training speed when compare with RNN-based NMT models. Because CNN models process the whole input sentence with their filters at the same time while RNN models handle tokens in input sequentially. Besides, the structure of the CNN model allows it to deal with the gradient vanishing better than RNN models. However, some critical issues still exist in the translation quality of CNN-based NMT models. First of all, the CNN models extract features of input by convolution filters, so they can only capture the relation information between tokens within the range of their filters. With the tokens that lie further than the width of their filter, CNN can only find relation information between them in the high-level convolution layer. This causes the concrete expression in translated sentences generated by CNN models and reduces the translation quality of CNN-based NMT models. Secondly, compressing a sentence to a vector may cause information loss during computation steps, this drawback exists in both RNN-based models and CNN-based models and is later solved by the attention mechanism partly.

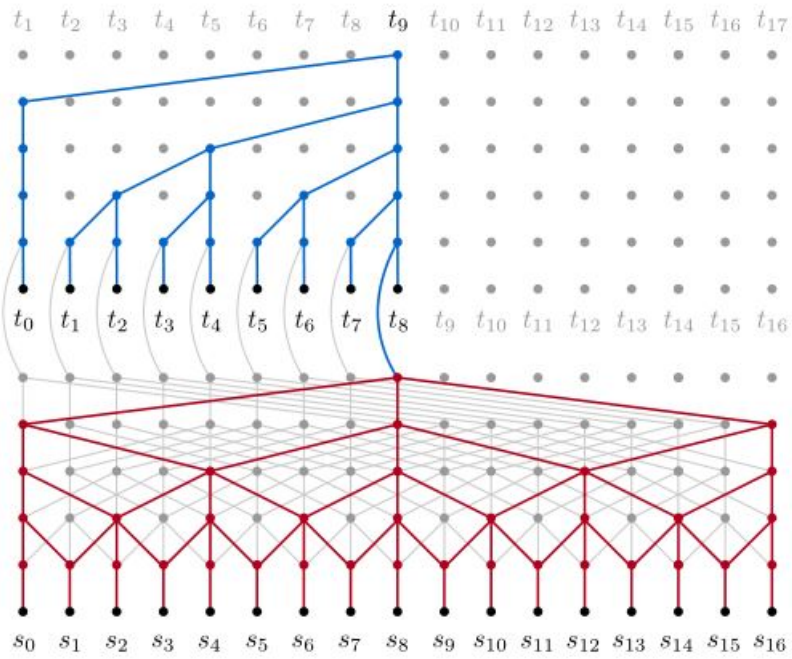

Figure 2.7: Architecture of the ByteNet model. The target decoder (blue) is stacked on top of the source encoder (red). The decoder generates the variable-length target sequence using dynamic unfolding.[8]

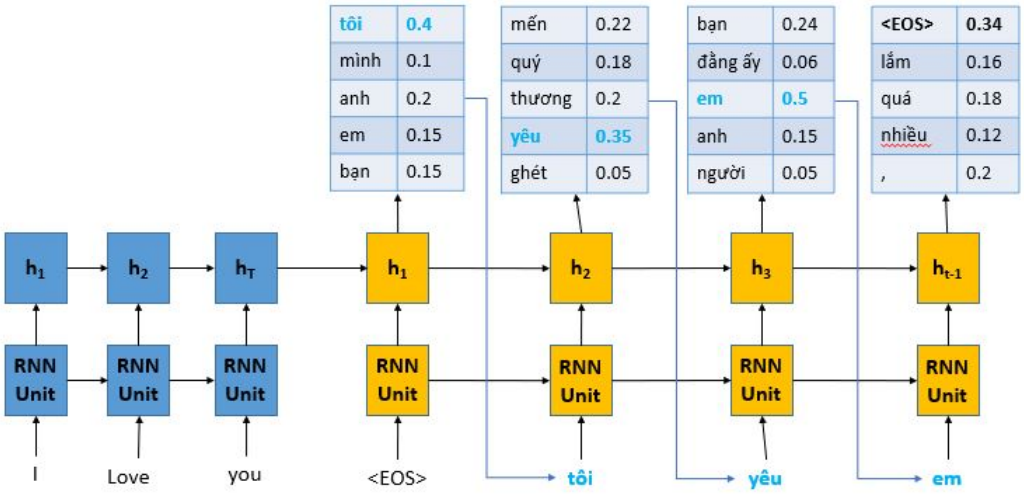

Figure 2.8: demonstrate of greedy search for NMT model with RNN-based decoder

2.1.4 Inference method

The inference is using a trained RNN-based model in practical application. In this section, the model has information on the source input sequence only. So to decode the target sentence all we have is the encoder's hidden representation. There are several methods were proposed to serve for decoding target sentences in the inference section such as Greedy search and Beam search. **Greedy search** The idea of greedy search is to choose the output word with the highest predicted probability as the next word. The step of the greedy search algorithm include:

- Step 1: The decoder starts to decode when the model receives the ¡EOS¿ (end of sequence¿ token. The decoder then starts to receive hidden encoder representation as the input to predict output words
- Step 2: The decoder predicts the probability of each word in the vocabulary to become the next word of the sentence in the target language.
- Step 3: The decoder chooses the word with the highest probability as the next word of the sentence in the target language and the input for the next time step also.
- Step 4: Continue the loop until reach ¡EOS¿ token.

Beam Search The Greedy search algorithm provides an impressive result in generating the sentence in the target language. However, the greedy search

does not give us a chance for reconsidering in case there are 2 or more words that have approximate probabilities. In this aspect, beam search has proven to be much more better than greedy search we can choose a more suitable output with beam search than greedy search. Beam search is proposed in [12] in other tasks, however, it is an effective method for MT and is usually used for the decoding steps of ML models. In the beam search algorithm, the model will keep n words with the highest probabilities and then treat all n words as the input for the next time step. For each word that is retained, we will obtain the next n words. Then each branch of the tree now will continue looping until reaching the ¡EOS¿ token. In the end, to determine the final output, the model will compute the total probability of each branch and choose the branch with the highest total probabilities as the final translation. Thank to this method, we can have more choice when decoding the output of MT models. On the other hand, this method also makes the model consume more time while translating. To prevent the model consume too much time for decoding, the number of words to be retained is often set to be a small number to reduce the number of possible results. In [13], the best performance of the beam search algorithm occurs with n lie between 5 to 10.

2.2 Transformer Architecture

Although the Deep learning models, especially RNN-based models, achieved many impressive results in the MT topic, there are some issues that still exist. In 2017, a new architecture called Transformer [9] has been proposed by Vaswani et al. and it solved almost all of the problems that remain in other deep learning models at that time. With impressive achievements, Transformer and its variations soon reach state-of-the-art results in many other NLP topics, not only machine translation. So the following, we will break down Transformer to understand the idea of the Transformer and how the Transformer works.

2.2.1 Encoder-Decoder Structure

Transformer architecture follows the Encoder-Decoder structure. The Encoder-Decoder for a long while has been accepted by most machine translation researchers as an original structure to build a translation model. The Encoder-Decoder structure was first time proposed by Kalchbrenner and Blunsom [6]. Even though there were many variations with separate details and customization added by researchers, the main structure is maintained. This

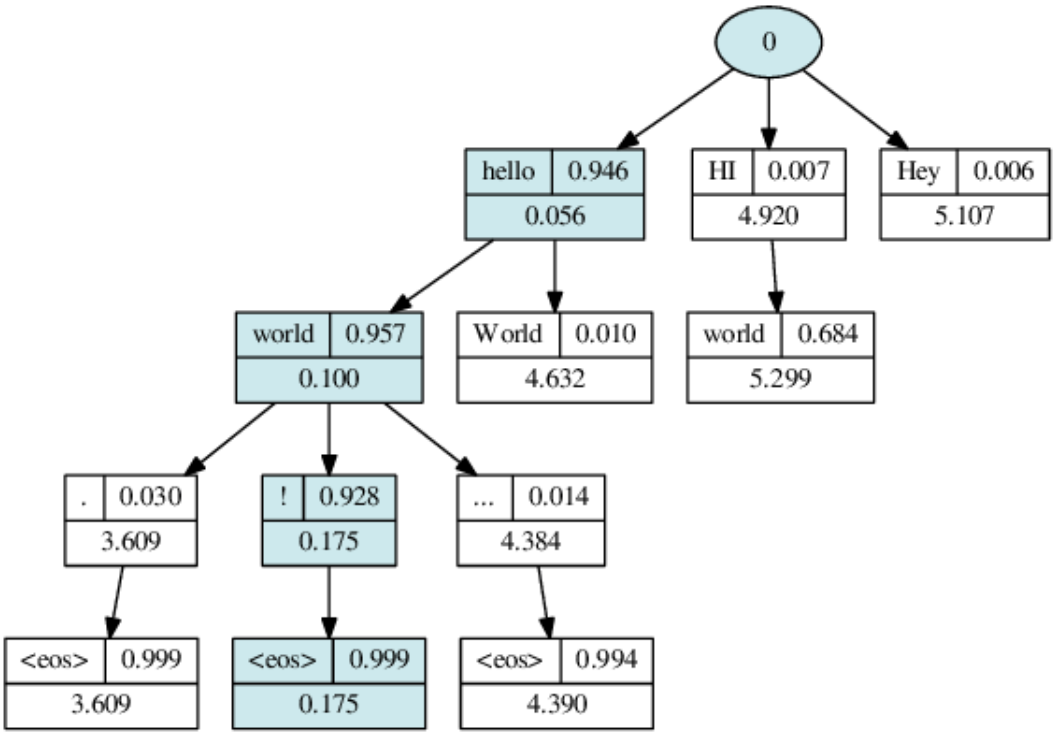

Figure 2.9: Beam search algorithm

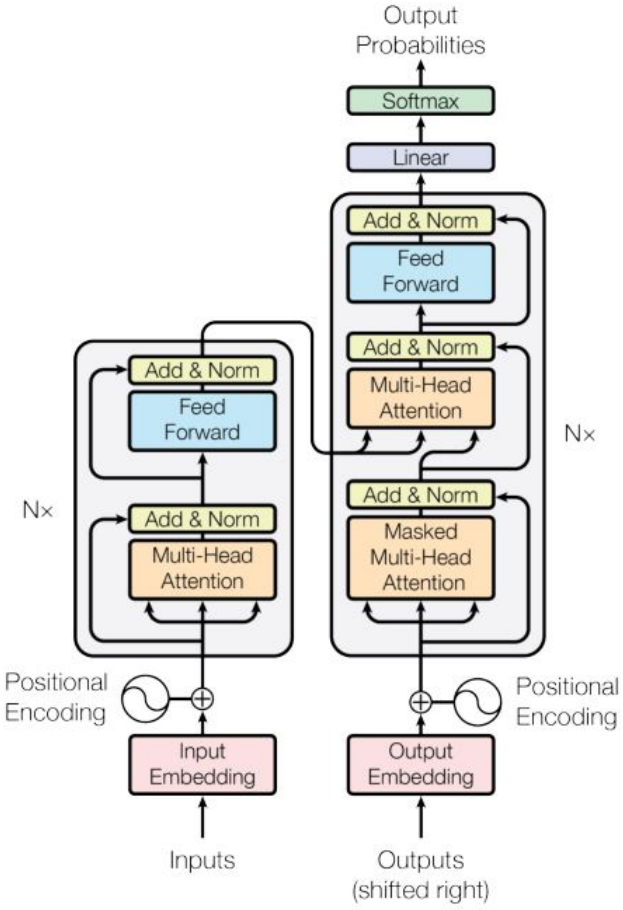

Figure 2.10: demonstration of transformer architecture

model structure involves two connected components, the Encoder, and the Decoder. The Encoder transforms the input sentence in the source language into a vector representation then the Decoder will transform the vector representation of input into the sentence in the target language. Each component of the structure is a neural network but they can be designed with different architecture, such as in the research of Kalchbrenner and Blunsom [6], the Encoder was a CNN encoder and the decoder was an RNN decoder. Basically, the concept of this structure is using the representation vector in semantic space to map two sentences in two different languages together. The middle process is hard to visualize so we can say this structure is an end-to-end translation.

In the case of the Transformer, the encoder is a stack of N identical layers (N=6 as in [9]) and the decoder also is a stack of M identical layers (M=6 as in [9]). Each encoder layer contains two sub-layers which are the multi-head self-attention layer and a fully connected feed-forward network. Following each sub-layer is a normalization layer so with the input x that layer receives, the output of layer should be *LayerNorm(x + SubLayer(x))*. The encoder will encode the input sentence under the self-attention mechanism to get a final representation that retains most of the semantic information of the input sentence.

The decoder layers have two sub-layers as the encoder layers and one additional multi-head self-attention layer for the previous output of the decoder. Especially, the multi-head self-attention sub-layers in the decoder layers have a slight divergence from the multi-head self-attention in the encoder. Before the embedded vector of the input source sentence is fed toward the encoder, they are added the positional encoding, but in the decoder, the multi-head self-attention sub-layers prevent the positional encoding to make sure that the output of encoding only gets affected by the previous output tokens.

To reserve the information of the position of each token, the Transformer model adds the positional encoding to the input embedding vectors [9]. The positional encoding has the same length as the embedding vector, representing for absolute position of each token inside the sentence. The creating of positional encoding is shown in Figure 2.11, "n" is the number of embedding dimensions. For the position i in the input sequence and dimension j (0¡=j¡=n/2), P_ij is calculated by the formula:

$$P_(i,j) = sin(\frac{i}{10000^{\frac{2j}{n}}}) \tag{2.7}$$

$$P_(i,j) = cos(\frac{i}{10000^{\frac{2j+1}{n}}}) \tag{2.8}$$

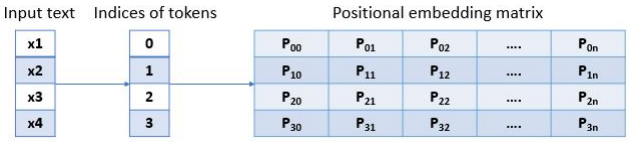

Figure 2.11: demonstration of positional encoding matrix

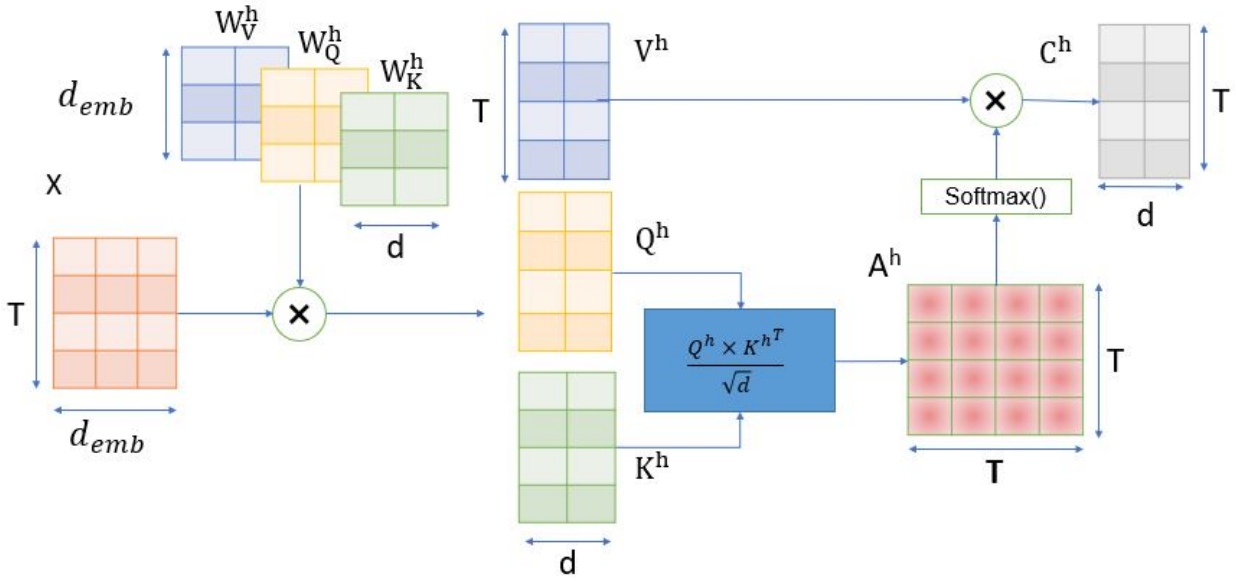

Figure 2.12: self-attention layer

2.2.2 Self-Attention mechanism

As mentioned before, every layer of the Transformer model uses the self-attention mechanism in encoding and decoding progress. This attention mechanism is totally different from the mechanism that is proposed by Badanau et al.[4] and Luong et al.[11] we mentioned before. It is considered that the self-attention mechanism is the core of Transformer architecture.

When the input vector x is fed to the self-attention layer, the input is split into three presentations which are Q,K,V matrix, and then the context vector C^h is computed as in Figure 2.12. We have:

- X: input vector
- T: length of input sequence

- d_emb: number of dimension of embedding vector
- d : number of dimension of hidden representation
- Q^h : Query matrix
- K^h : Key matrix
- V^h : Value matrix
- W_V^h, W_K^h, W_Q^h : weight matrix corresponding to V, K, Q
- softmax(): softmax function
- A^h : attention weight matrix
- C^h : context vector

The formula to compute context vector C^h at the current step is :

$$C^h = V^h softmax(\frac{Q^h \times K^{h^T}}{\sqrt{d}}) \tag{2.9}$$

Besides that, multi-head attention is also applied in the Transformer layer. The multi-head attention involves many heads of attention (in [9], number of heads is 8), each head is a different set of weight matrices of Q^h, K^h, V^h, each head will produce a different context vector. To obtain the final context vector, we concatenate all the context vectors in all heads like in Figure 2.13.

2.3 Inject additional information approach

Almost NMT models take the input that only contains the source language sequences and generate the corresponding sentence in the target language directly from the input source sequences. Even though this method achieved many impressive results, raising the quality of translation results produced by the NMT model is still a main target in the ML field. One of the approaches to this problem is providing additional information to the model so we can highlight some linguistic information that the model can learn. Many researches [19] [20] [14] [21] have proven the effectiveness of this approach.

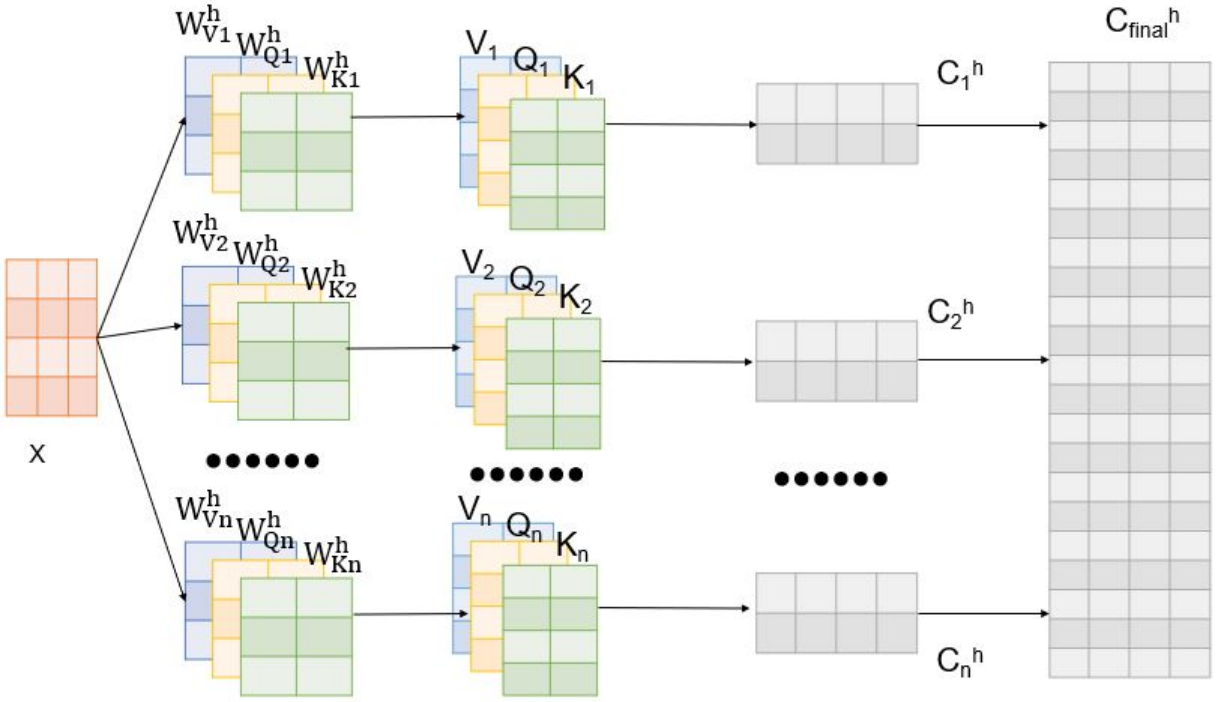

Figure 2.13: multi head self-attention

2.3.1 Syntactical information

TF-IDF

TF-IDF score of a word reflects the importance of that word to the corpus and is computed by Term Frequency (TF) and Inverse Document Frequency (IDF).

$$TF(t,d) = \frac{f_{t,d}}{\sum_{t' \in d} f_{t',d}} \tag{2.10}$$

$$IDF(t,D) = log\frac{N}{|d \in D : t \in d|} \tag{2.11}$$

$$TF - IDF(t,d,D) = TF(t,d) \cdot IDF(t,D) \tag{2.12}$$

$f_{t,d}$: number of times that t appear in document d
N : total number of documents in corpus D
$|d \in D : t \in d|$: number of document d that contains term t

Dependency tags

Dependency parsing has been a key NLP task in analyzing grammar structure for a long time. Dependency tags are natural and flexible representations

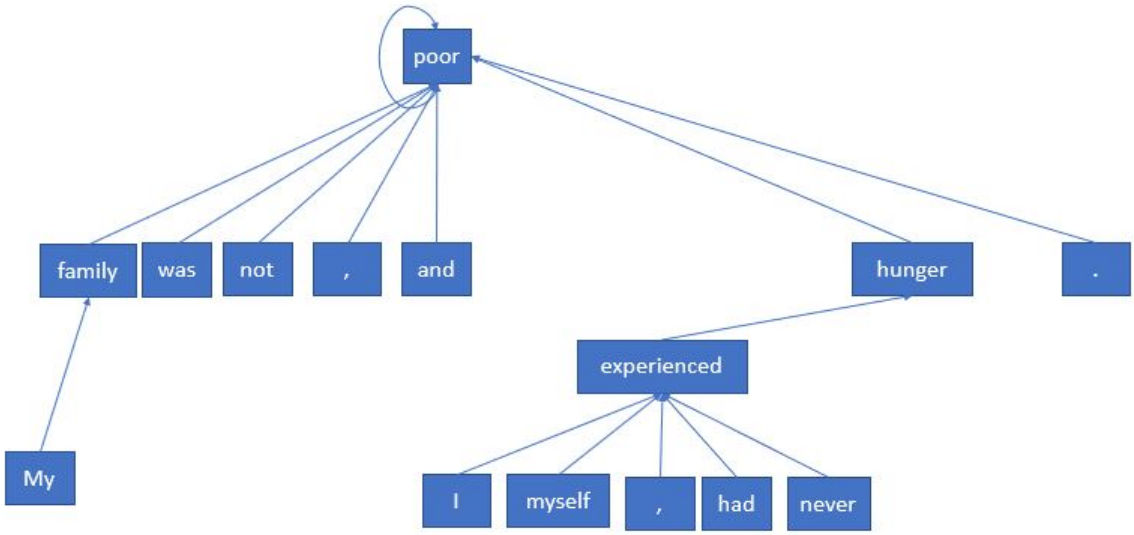

Figure 2.14: Dependency tree

of semantics. Their generality over trees, for instance, allows them to represent relational semantics while handling phenomena like co-reference and coordination [16].
For example, with the sentence "My family was not poor, and I had never experienced hunger myself." we can get a dependency tree as in Figure 2.14. This dependency tree helps us to understand more about the dependency relationship between the tokens inside the sentence, such as we can see that the word "my" depends on the word "family" in Figure 2.14. To simplify the tree structure of the dependency, we can transform the tree format to the matrix format as in Figure 2.15. In this matrix element at line i, column j will represent the probability that if token i depends on token j inside of the source sequence. The value of each element inside the matrix lies between 0 and 1. If the value is 1, it means that token i depends on token j.

Part of speech

The Part-of-speech(POS) tag of a word is the category of that word such as noun, verb, adjective, adverb, and so on. Based on the definition and context of sentences or documents, a word can have multiple POS tags. POS tags play an important role in many NLP tasks [18] such as speech recognition, information retrieval, question answering, and so on. The POS tags provide a lot of information about the words.

0	1	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	1	0	0	0	0	0	0	0	0	0	0
0	0	0	0	1	0	0	0	0	0	0	0	0	0	0
0	0	0	0	1	0	0	0	0	0	0	0	0	0	0
0	0	0	0	1	0	0	0	0	0	0	0	0	0	0
0	0	0	0	1	0	0	0	0	0	0	0	0	0	0
0	0	0	0	1	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	1	0	0
0	0	0	0	0	0	0	0	0	0	0	0	1	0	0
0	0	0	0	0	0	0	0	0	0	0	0	1	0	0
0	0	0	0	0	0	0	0	0	0	0	0	1	0	0
0	0	0	0	0	0	0	0	0	0	0	0	1	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	1	0
0	0	0	0	1	0	0	0	0	0	0	0	0	0	0
0	0	0	0	1	0	0	0	0	0	0	0	0	0	0

Figure 2.15: Dependency Matrix

2.3.2 Tree-LSTM

As mentioned before, the LSTM model encodes the input sentence following a sequential order since then reserving the information about the position of tokens inside the input sentence. However, this method of encoding also has a drawback, the gradient vanishing may occur with long sentences and make the hidden representation of the tokens at the ends of the sequence lose information about the relationship with the tokens in the head of the sequence. Besides, the LSTM model does not exploit the syntactical information inside the sequence directly and this may be a waste of information. To cope with this problem, Akiko Eriguchi et al.[14] proposed the Tree-LSTM model which encodes the input sequence following the order of the phrases inside the sentence. The phrase structure is usually constructed as a tree with the node as the phrase of the sentence and the leaves is the single words, especially, since the root will be the full sentence. The encoder's hidden representation of a node in the tree is generated bottom-up by the hidden presentation of its children. Hidden representation for the leaves will be the embedding vectors of words in those leaves. The hidden representation of a node $h_k^p hr$ is computed from hidden representation of two child nodes h_k^l and h_k^r as:

$$h_k^p hr = f_t ree(h_k^l, h_k^r) \tag{2.13}$$

f_tree is a non-linear function.
The LSTM cell is also modified a bit to fit the tree structure. More specific, we will have the memory cel c_k^phr and hidden state representation of a node will be calculated as:

$$i_k = \sigma(U_l^i h_k^l + U_r^i h_k^r + b^i) \tag{2.14}$$

$$f_k^l = \sigma(U_l^{f_l} h_k^l + U_r^{f_l} h_k^r + b^{f_l}) \tag{2.15}$$

$$f_k^r = \sigma(U_l^{f_r} h_k^l + U_r^{f_r} h_k^r + b^{f_r}) \tag{2.16}$$

$$o_k = \sigma(U_l^c h_k^l + U_r^c h_k^r + b^c) \tag{2.17}$$

$$c_k = \tanh(U_l^c h_k^l + U_r^c h_k^r + b^c) \tag{2.18}$$

$$c_k^phr = i_k \cdot c_k + f_k^l \cdot c_k^l + f_k^r \cdot c_k^r \tag{2.19}$$

$$h_k^phr = o_k \cdot tanh(c_k^phr) \tag{2.20}$$

Where i_k, f_k^l, f_k^r, o_k, c_k are the input gate, the forget gates for left and right child units, the output gate, and the state for updating the memory cell. c_k^l and c_k^r are the memory cell of the left and right child nodes. The initially hidden representation for the decoder is the hidden state representation of the root h_k^root. And because the number of hidden state representations in the encoder is not the same as the length of the sequence so the attention mechanism in this mode is also modified. The context vector at j-th step in decoding is computed as:

$$d_j = \sum_{i=1}^{n} a_j^i h_i + \sum_{i=n+1}^{2n-1} a_j^i h_i^p hr \tag{2.21}$$

2.3.3 Incorporating Dependency tags into Transformer

Parent-scaled self-attention Incorporating syntax information into Transformer to improve translation performance is a promising approach that has been proven by many research [23] [24] [21]. Almost researchers used dependency tags to inject into the Transformer model. One of the most effective methods proposed is using Parent-scaled self-attention(PACAL)[21]. They customized the attention layer to incorporate the source syntax by adding the weight matrix, which is based on the dependency relationship between each token, to the attention weights. Since then, the model now will pay more attention to the dependency relation between the tokens inside the input sequence. The distance weight between token j to a token t is computed as:

$$n_t j^h = s_t j^h d_t j^p \tag{2.22}$$

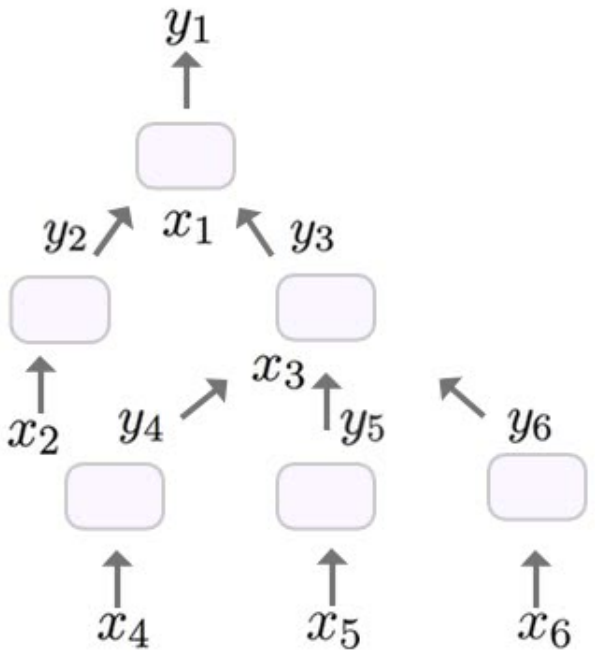

Figure 2.16: Demonstration of Tree-LSTM encoding process

$s_t j^h$ is the distance from j to t. $d_t j^p$ is the distance from j to parent of t, we have $d_t j^p = dist(p_t, j)$ and is computed as:

$$dist(p_t, j) = f_N(j|p_t, \sigma^2) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(j-p_t)^2}{2\sigma^2}} \tag{2.23}$$

Where σ^2 is a fixed variance. The distance between token j to parent of t is he value of the probability density of a normal distribution centered at p [21].

Linguistic-Information Self-Attention

Linguistically-Informed Self-Attention (LISA) is another method that incorporates the dependency tags into the Transformer model to improve the overall performance of this model [22]. However, LISA has a better mechanism to prevent the bad effects of the errors that occur in the dependency parsing step, while the PASCAL model cannot improve the dependency matrix through training steps because PASCAL does not interfere with the dependency tags generated by external tools. LISA provides a method to adjust the dependency tags to fit the translation purpose and also covers the errors inside the labels generated by external tools. LISA was designed for multi-task learning for semantic role labeling, dependency parsing, and POS tagging. So when it is applied in machine translation, it is trained for trans-

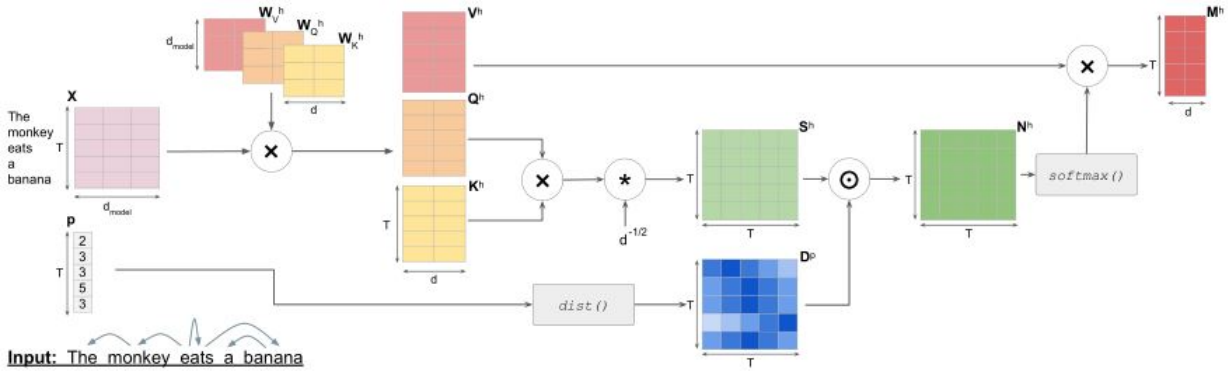

Figure 2.17: Demonstration of the pascal attention layer

lating and dependency parsing as well. The LISA model receives the parsed dependency tags from external tools as a part of the input and generates the dependency tags for itself. A deep bi-affine model U_{parse} [25] is used to replace an attention head of the encoder to predict syntactic dependencies. We have the parse attention weights A_{parse} computed as:

$$A_{parse} = Q_{parse} U_{parse} K_{parse} \tag{2.24}$$

Then the parse attention weights A_{parse} is encouraged to attend to each token's parent in a syntactical tree [22]. The probability that a token t having parent q is modeled as:

$$P(q = head(t)|X) = A_{parse}[t, q] \tag{2.25}$$

With input dependency tags X_{parse} the loss of predicting dependency tags is modeled as:

$$Loss_{parse} = log(P(head(t)|X_{parse}) \tag{2.26}$$

The total loss of the LISA model will be computed as :

$$Loss = Loss_{translation} + \lambda Loss_{parse} \tag{2.27}$$

Where λ is the penalty of dependency predicting loss.

2.4 Prepossessing

Prepossessing is a useful step before feeding the data to translation models. This step helps us to clean up the data, remove noise that exists in training or testing data, and more important, tokenize the data so that model can

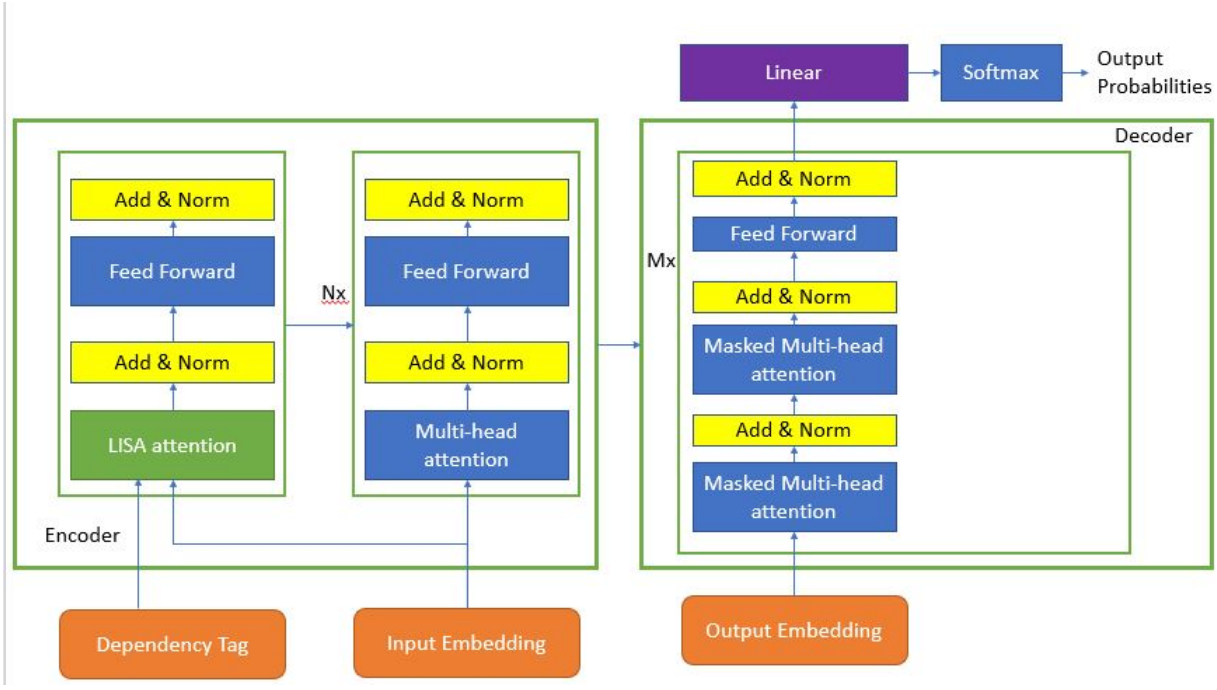

Figure 2.18: Flow of Transformer model with LISA layer

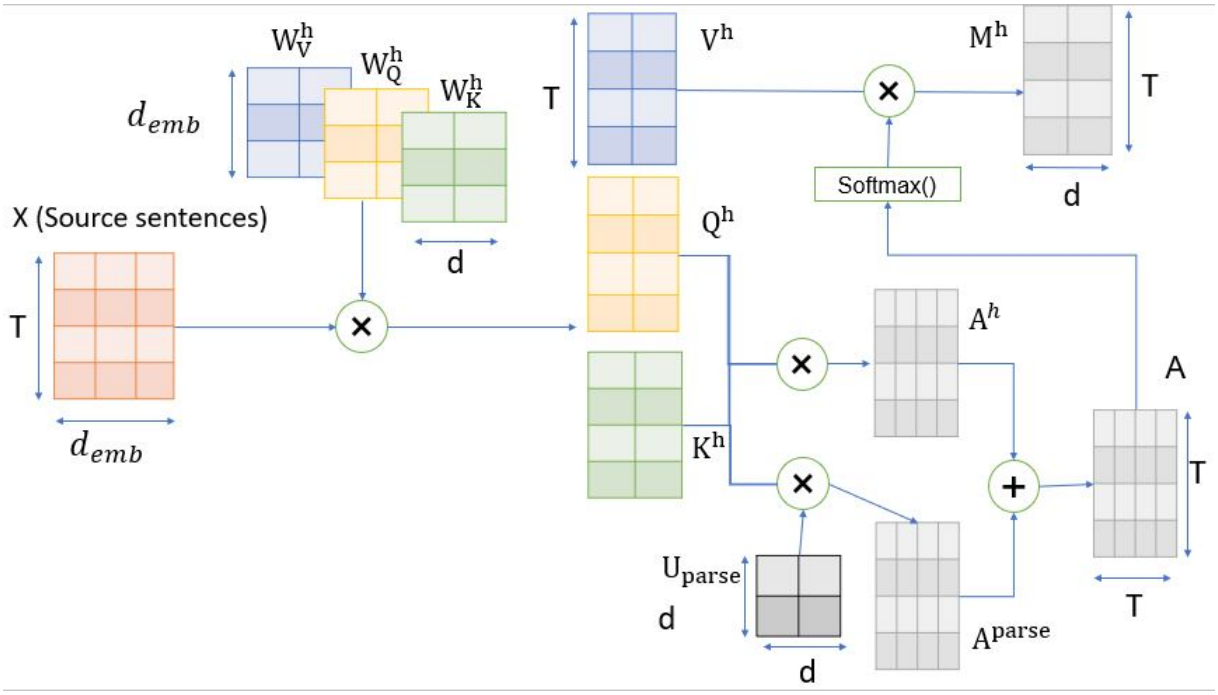

Figure 2.19: LISA layer

learn from it as much as possible. Byte Pair Encoding(BPE) is one of the prepossessing methods which are used the most that actually help to improve the quality of the NLP models. In 2016, Senrich et al. [26] proposed to apply BPE to prepossessing data before feeding it to translation models to resolve the rare word problem in the translation task. They achieved impressive results.

BPE is a data compression algorithm in which the pair of consecutive bytes of data with the highest appearance frequency will be replaced with an unknown token that did not exist in the vocabulary at the beginning. For example, given the data "gghuhhugghug", the byte pair "hu" appear with the highest frequency so we replace it with token A which did not exist in the data at the beginning. Now, we have data ggAhAggAg, we see that "B" is the next byte pair that is most common in data, so we replace "gg" with B, and the data is "BAhABAg" now. Even now, we can replace "BA" with a new token Z when "BA" appear most in the data. The final form of data should be "ZhAZg" where Z = BA, B = gg, A = hu when all the byte pairs now only appear once, and the data cannot be compressed anymore.

When this algorithm is applied to the NLP topics, it will help to break down the rare words into smaller sub-words that appear more often in the vocabulary. Maybe in some cases, the number of rare words remains but the number of common words will increase under the encoding of BPE. That is why BPE is often used in translation models to face out-of-vocabulary and rare word problems. Of course, it cannot handle the problems completely but BPE helps us to reduce the size of problems and let the model learn better with cleaner data.

Chapter 3

Proposed method

Even though incorporating the dependency tags as the source syntax to the Transformer model has achieved many impressive results, it remains a problem. Almost dependency tags are generated from external tools because the cost to label all dependency tags for translation data is really expensive. However, even though the accuracy of dependency parsing of the tool is high but the tags still contain errors that can misguide the translation model. Since that, finding other types of syntactical information to help the NMT model, in this case, the Transformer model, improve its translation quality is necessary.

3.1 Statistical information injection

In this approach, we chose to use TF-IDF to inject into the Transformer model. We use TF-IDF to measure the importance of a pair of words to the whole corpus. The TF-IDF score is computed by the term frequency and inverse document frequency for pairs of words. For example: with the sentence "I know I am wrong", we will have 8 pairs of words to calculate TF-IDF for pair instead of 5 words as in the regular method to calculate TF-IDF for single words. The pairs are: "I - know", "I – I", "I – am", "I – wrong", "know – I", "know – am","know - wrong","am - wrong". Specially, the order of each word in the pair is reserved, so that pair "I - know" and pair "know - I" are two different pairs.
The term frequency of pair p is computed below:

$$tf(p,d) = \frac{f_{p,d}}{\sum_{p}'[f_{p',d}]} \tag{3.1}$$

	I	know	I	am	wrong
I	0	TF-IDF (i-know)	TF-IDF(I-I)	TF-IDF(I-am)	TF-IDF(I-wrong)
know	TF-IDF (i-know)	0	TF-IDF (know-I)	TF-IDF(know-am)	TF-IDF(know-wrong)
I	TF-IDF(I-I)	TF-IDF(know-I)	0	TF-IDF(I-am)	TF-IDF(I-wrong)
am	TF-IDF(I-am)	TF-IDF(know-am)	TF-IDF(I-am)	0	TF-IDF(am-wrong)
wrong	TF-IDF(I-wrong)	TF-IDF(know-wrong)	TF-IDF(I-wrong)	TF-IDF(am-wrong)	0

Figure 3.1: TF-IDF matrix for the sentence "I know I am wrong"

$f_{p',d}$: raw count of p in document d

Then the inverse document frequency for pairs of words is computed as :

$$idf(p, D) = log\frac{N}{|d \in D : p \in d|} \tag{3.2}$$

N: total number of document in corpus D
After that, we have the TF-IDF score for pair of words calculated as:

$$TF - IDF = tf(p, d) \times idf(p, D) \tag{3.3}$$

To inject the TF-IDF score for the pair of words into the Transformer model, we compute the TF-IDF scores of all the available pairs all over the training dataset. Since then, we can summarize a TF-IDF matrix with size (S, S), While S is the length of the input sequence. For example, with the sentence "I know I am wrong" we will get the TF-IDF matrix as in Figure 3.1. To inject the TF-IDF score for pair into the Transformer model, we have adjusted the self-attention layer in the first encoder layer. The detail of the customized encoder layer is shown as in Fig 3.2

3.2 POS tags injection

POS tags of a sentence contain information about the role of each token in the expression of the total sentence. The words with different POS tags will have different relationships with the words that have the same relationship.

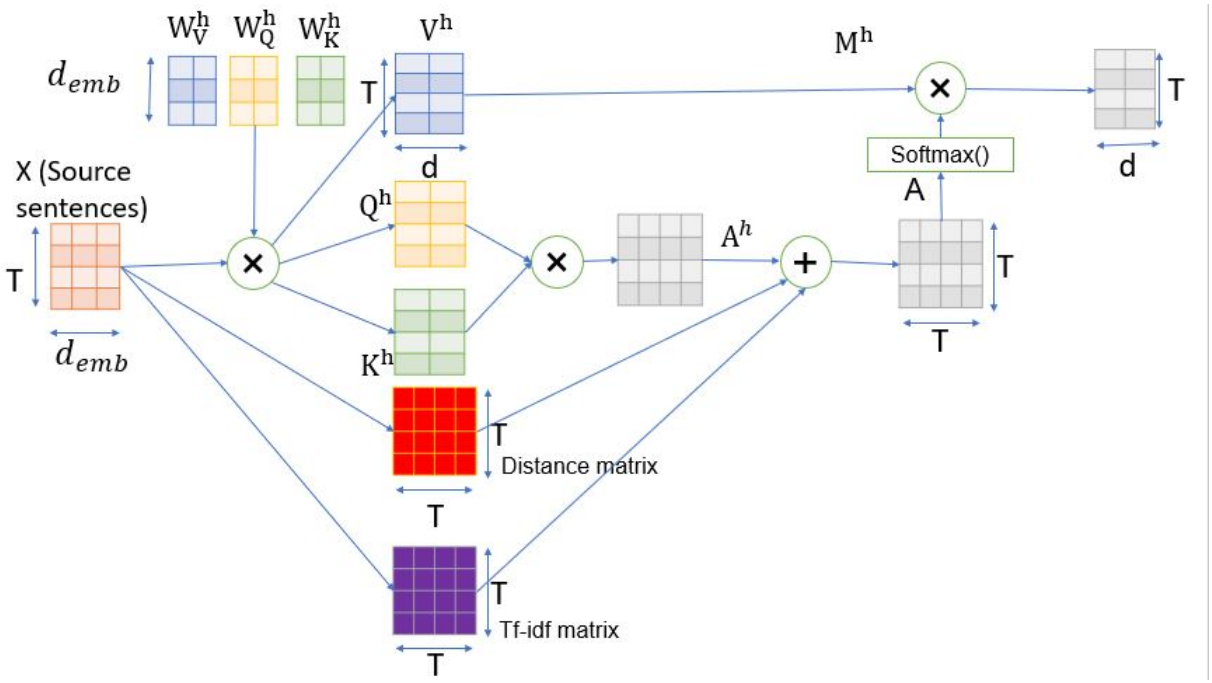

Figure 3.2: Self-attention layer with injection of TF-IDF matrix

For example, with the sentence "My family was not poor, and myself, I had never experienced hunger ." we will have the POS tags as: "PRP NN VB RB JJ , CC PRP , PRP VB RB JJ NN .". So we can see that with the POS tag, we know the relationship between the pair of words "family" and "was" is different between the pair of words "my" and "family". Because the relationship between "family" and "was" is the relationship between "NN" (Noun) and "VB" (verb) but the relationship between "my" and "family" is the relationship between "PRN"(Personal Pronoun) and "NN" (Noun). Knowing this kind of relationship may help the model to adjust the self-attention among the tokens more suitably.

To inject the POS tags into the Transformer model, first, we collect the POS tags of the input sentence by using the CoreNLP POS tagger. Then we built a separate embedding layer just for the input POS tags that come along with the input sequences. After the embedding process of the POS tags, we also add the Positional Encoding to the embedded vectors of the POS tags indices. The Query matrix Q^P and Key matrix K^P of the POS tags input will then be computed by multiplying the input with two separated weight matrices that are $W_Q^P and W_K^P$. The attention score of POS tags is computed

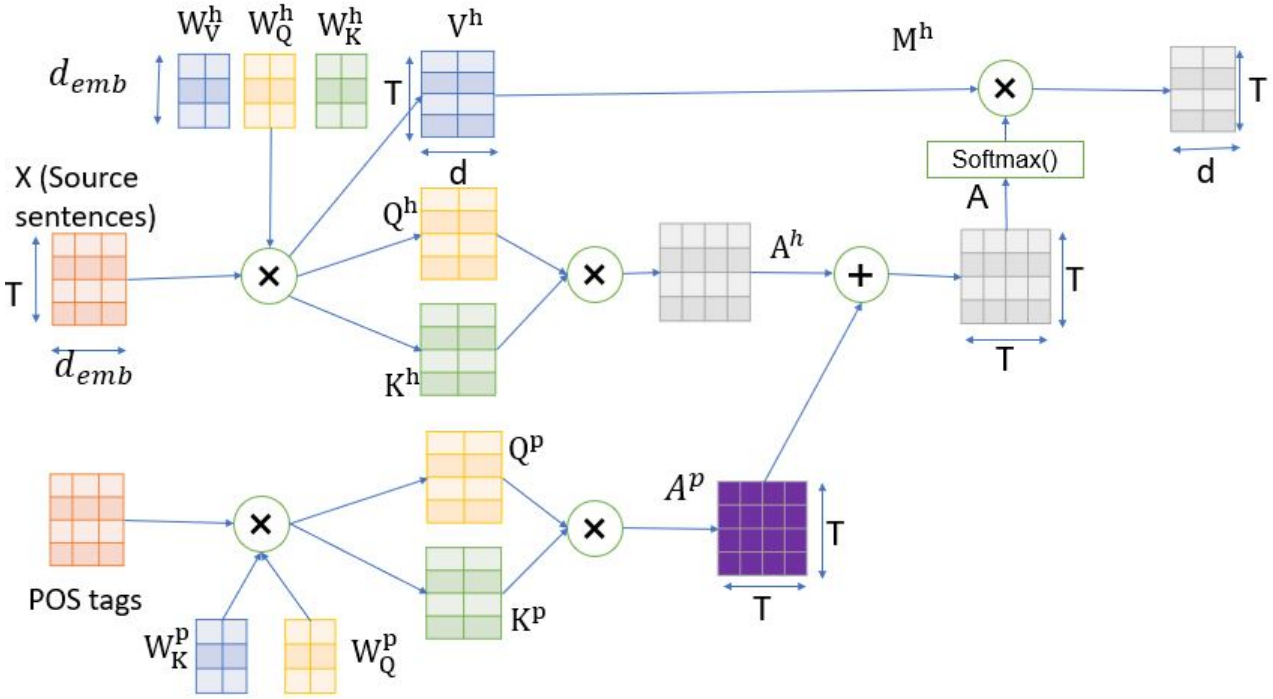

Figure 3.3: Self-attention layer with injection of POS tags

as:

$$A^P = \frac{Q^P \cdot {K^P}^T}{\sqrt{d}} \tag{3.4}$$

d is the number of hidden dimension of Q and K
The final attention score of the input will be computed as :

$$A = A^h + A^P \tag{3.5}$$

Then the attention is feed forward to a softmax function and multiplied with the Value matrix (V) to get the attention weight. The detail of the customized self-attention layer is shown as in Figure 3.3.

Chapter 4

Experimentation and Results

To examine the performance of our method, We used Transformer as the baseline model and do experiments on 2 datasets, IWSLT2014 Germany-English and IWSLT2015 Vietnamese - English. We used the BLEU score to evaluate the results. The parameters of the dataset are shown in table 4.1.

To collect the POS tags of German and English data, we used the CoreNLP POS parsers and Stanza parser for Vietnamese. During the experiments period, we tried to inject the TF-IDF on the first encoder layer of the Transformer model, 3 first encoder layers, 3 last encoder layers, and finally all of the encoder layers to measure the effect of TF-IDF on the Transformer model.
With the POS tags injection, we implemented a filtering step to control the number of relations that will occur between 2 tokens. For more clarity, we observed the appearance frequency of each tag in the total valid POS tags. We see that there are many tags that have less appearance than other tags but their role is similar to other tags. So we decided to change those tags to the tags with higher covering according to the POS tags structure of the Penn tree bank [29] for English and Negra corpus for Germany. For example, words with the tag "VBG" will be changed to "VB" because "VBG" is a sub-tag of VB in Penn Treebank. We tried different thresholds to set the filter.

We used the Bilingual Evaluation Understudy (BLEU) to evaluate the results of our methods and compare them with related works. BLEU was proposed by Kishore et al.[17] to measure the performance of the translation systems. It is easy to implement and independent of language but most importantly, the BLEU score has a high correlation with the evaluation of humans. That is the reason why the BLEU score is widely adopted in many

dataset	IWSLT2014 de - en	IWSLT2015 vi - en
Training sentences	160239	133317
Validating sentences	7283	1553
Testing sentences	6750	1268

Table 4.1: Components of dataset.

	IWSLT2015 vi - en	IWSLT2015 en - vi	IWSLT2014 en-de	IWSLT2014 de-en
Transformer	30.06	31.61	29.53	36.13
TF-IDF injection	**30.41**	**32.12**	29.66	36.07
POS tags injection	30.24	31.75	**29.61**	**36.31**

Table 4.2: Results of the Transformer model, Tranformer model with the injecttion of TF-IDF and Transformer model that incorporate POS tags.

	IWSLT2015 vi - en	IWSLT2015 en - vi	IWSLT2014 en-de	IWSLT2014 de-en
Firs encoder layer	**30.41**	**32.12**	29.66	36.07
3 first encoder layers	30.21	31.92	29.39	35.7
3 last encoder layers	30.5	31.7	29.42	35.92
all encoder layers	30.67	31.73	29.2	35.56

Table 4.3: Result of Transformer model with the different number of pair TF-IDF injected layers.

	IWSLT2015 vi - en	IWSLT2015 en - vi	IWSLT2014 en-de	IWSLT2014 de-en
non-filter	**30.24**	**31.75**	29.61	36.31
filter tags which have less than 0.03 probability appear	30.61	31.82	29.46	36.29
filter tags which have less than 0.02 probability appear	29.76	31.75	29.61	36.22
filter tags which have less than 0.01 probability appear	29.75	31.73	29.42	35.94

Table 4.4: Result of POS tags injected Transformer model with the different filter of POS tags.

researches for comparing the candidate between translation systems.

By adding TF-IDF into the Transformer model, we boosted the performance of the model in Vietnamese-English translation, but it seems this method did not improve too much in the German-English translation task. On the other hand, embedding POS tags along source sentence improve the performance of the model in all dataset but the improvement is not strong as the TF-IDF injecting method.

This contrast may be caused by the difference in word organization in Vietnamese and Germany. In Vietnamese, a word does not have any transformation while in Germany, transformations of a word are many, and combining single words to create a new word often happens, so TF-IDF shows less effect in Germany.

However, both languages have clear grammar structures, and their structures have many common so the POS tag embedding method can improve the related information in both languages.

To see more about the effects of our methods on the translation quality, we will analyze some random translations from English to Vietnamese. For short sentences, we get the example:

- source sentence: "*Because the final step in the domestic violence pattern is kill her .*"
- target sentence: "*Bởi vì bước cuối cùng trong* ***kịch bản****(pattern) bạo hành gia đình* ***là giết****(is kill) chết cô ta .*"
- Transformer translation: "*Bởi vì bước cuối cùng trong* ***mô hình****(model) bạo lực gia đình* ***đang giết****(is killing) cô ấy*"
- Translation with injection of pair TD-IDF: "*Bởi vì bước cuối cùng trong* ***quy luật****(rules) bạo lực gia đình* ***đang giết****(is killing) chết cô ấy .*"
- Translation with POS tags injection: "*Bởi vì bước cuối cùng trong* ***mẫu****(pattern) bạo lực gia đình* ***là giết****(is kill) cô ấy .*"
- POS tags: "Because(IN) the(DT) final(JJ) step(NN) in(IN) the(DT) domestic(FW) violence(JJ) pattern(NN) is(VBZ) kill(VB) her(PRP) ."

In the example above, almost all of the models translated the wrong words in the first highlighted word, Transformer with the injection of POS tags

translated right as the word "pattern"in English but the model chose the wrong word in Vietnamese for that meaning. With the second highlighted word, only Transformer with POS tag injection gives the matching translation with the target sentence.

For another short sentence translation:

- source sentence: "*We were scared , but still , school was where we wanted to be .*"
- target sentence: "*Chúng tôi đã rất sợ , **nhưng dù vậy**(but still) , chúng tôi vẫn muốn tới trường .*"
- Transformer translation: "*Chúng tôi sợ hãi , **nhưng vẫn**(but still) , trường học là nơi chúng tôi muốn đến .*"
- Translation with injection of pair TD-IDF: "*Chúng tôi rất sợ , **nhưng vẫn còn trường học**(but stil there is school) là nơi chúng tôi muốn đến .*"
- Translation with POS tags injection: "*Chúng tôi vẫn sợ , **nhưng** , trường học **vẫn**(still) là nơi chúng tôi muốn đến .*"
- POS tags: "We(PRP) were(VBD) scared(VBN) , but(CC) still(RB) , school(NN) was(VBD) where(WRB) we(PRP) wanted(VBD) to(TO) be(VB) ."

In the example above, the Transformer model gave an almost matched translation with the target sentence. The TF-IDF injection version generated a sentence that has a slightly different meaning than the target sentence. The POS tags injection model shuffled some words in the translated sentence. So we see that in a short sentence, POS tags injection helps the Transformer model focus on some difficult words and chose the suitable word in the target language to generate suitable translated sentences. However, there is some noise they can give to the Transformer model but it seems does not affect too much on the overall expression of the sentence.

For the long sentence, the effects of TF-IDF injection and POS tags injection may be seen more clearly. So, we will analyze some long sentence translation examples below:

- source sentence: "*Many have been tricked by false promises of a good education , a better job , only to find that they 're forced to work without pay under the threat of violence , and they can not walk away .*"

- target sentence: "*Rất nhiều người bị lừa bởi những lời hứa điêu ngoa về giáo dục tốt , công việc tốt hơn , chỉ để thấy mình bị bắt làm việc không công dưới ách bạo lực , và không thoát ra được .*"

- Transformer translation: "*Nhiều người đã bị đánh lừa bởi những lời hứa sai lầm về một nền giáo dục tốt , một công việc tốt hơn , chỉ để thấy rằng họ bị ép buộc phải làm việc mà* ***không phải chịu đựng sự đe doạ của bạo lực(without being under the threat of violent)*** *, và họ không thể bỏ đi .*"

- Translation with injection of pair TD-IDF: "*Nhiều người đã bị ám ảnh bởi những lời hứa sai lầm về một nền giáo dục tốt , một công việc tốt hơn , chỉ để tìm ra rằng họ bị buộc phải làm việc mà* ***không phải chịu sự đe doạ bạo lực(without being under the threat of violent)*** *, và họ không thể đi lại được .*"

- Translation with POS tags injection: "*Nhiều người đã bị lừa bịp bởi những lời hứa sai lầm về giáo dục tốt , một công việc tốt hơn , chỉ để nhận ra rằng họ bị buộc phải làm việc mà* ***không phải trả tiền cho mối đe doạ bạo lực(without paying for the threat violent)*** *, và họ không thể bỏ đi .*"

- POS tags: "Many(JJ) have(VBP) been(VBN) tricked(JJ) by(IN) false(JJ) promises(NNS) of(IN) a(DT) good(JJ) education(NN) , a(DT) better(JJR) job(NN) , only(RB) to(TO) find(VB) that(IN) they're(PRP) forced(VBN) to(TO) work(VB) without(IN) pay(NN) under(IN) the(DT) threat(NN) of(IN) violence(NN) , and(CC) they(PRP) cannot(MD) walk(VB) away(RB) ."

In the example above, both the Transformer model and TF-IDF injected model missed the word "pay". Hence, the sentence only mentions "without being under threat, "which leads to a very different expression. The POS-tag injected model did pay attention to the word "pay". The POS tags let the Transformer model know that there is a noun(NN) between 2 preposition (IN) tokens which are "without"and "under"so the model focus on the word pay more than normal Transformer model. But the relation is not very clear so the meaning is different too.

So we can see that both injection help model to pay attention to keywords that have an important role in expressing the meaning of the source sentence, however, the pair TF-IDF score seems to work better than the POS tags in the cases of short sentences and the POS-tags work better in the cases of long sentences.

Chapter 5

Conclusion

We have demonstrated the method to incorporate syntactical information, in this case, TF-IDF score for pair of words and POS tags, into the Transformer model to improve the translation quality. Our methods have achieved some promising results against the Transformer model. The methods helped the Transformer model to pay attention to suitable tokens by highlighting the relation between the tokens inside the input sequences. This has proved that we can improve the performance of the translation task when improving the power of the model in representing the relation between each word in the input sentence.

However, our method is not very stable when compared with the Transformer model. The relations are highlighted in the source sentence thanks to the syntactical information of source sentences but during the translation process, the highlighted relations are not mapped exactly to the relations in the generated sentences in the target language. So, in the future, we plan to also inject syntactical information in the decoder to map the relation of the source sentence to the target sentence to generate better translation.

Bibliography

[1] Shuoheng Yang, Yuxin Wang, Xiaowen Chu. 2020. "A Survey of Deep Learning Techniques for Neural Machine Translation", arXiv preprint arXiv:2002.07526.

[2] Ilya Sutskever, Oriol Vinyals and Quoc V. Le. 2014. "Sequence to Sequence Learning with Neural Networks", *In Advances in neural information processing systems*, pp. 3104–3112.

[3] S. Hochreiter and J. Schmidhuber. 1997. "Long short-term memory", *Neural Computation*, vol. 9, no. 8, pp. 1735–1780.

[4] D. Bahdanau, K. Cho, and Y. Bengio. 2015. "Neural machine translation by jointly learning to align and translate", in ICLR.

[5] Zhou, J., Cao, Y., Wang, X., Li, P., and Xu, W. 2015. "Deep recurrent models with fast-forward connections for neural machine translation", Transactions of the Association for Computational Linguistics, 4, 371-383.

[6] Kalchbrenner, N.and Blunsom, P. 2013. "Recurrent continuous translation models", In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing* (pp. 1700-1709).

[7] Lukasz Kaiser and Samy Bengio. 2016. "Can Active Memory Replace Attention?", In Neural Information Processing Systems (pp. 3781-3789).

[8] Nal Kalchbrenner, Lasse Espeholt, Karen Simonyan, Aäron van den Oord, Alex Graves and Koray Kavukcuoglu. 2016. "Neural Machine Translation in Linear Time", arXiv preprint arXiv:1610.10099.

[9] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., ... and Polosukhin, I. 2017. "Attention is all you need", In Advances in Neural Information Processing Systems(pp. 5998-6008).

[10] Minh-Thang Luong and Christopher Manning.2015. “Stanford Neural Machine Translation Systems for Spoken Language Domains”, In *Proceedings of the 12th International Workshop on Spoken Language Translation: Evaluation Campaign* , pages 76–79, Da Nang, Vietnam.

[11] Thang Luong, Hieu Pham and Christopher D. Manning.2015. “Effective Approaches to Attention-based Neural Machine Translation”, In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing,*, pages 1412–1421, Lisbon, Portugal. Association for Computational Linguistics.

[12] Graves, A. (2013). ”Generating sequences with recurrent neural networks”. arXiv preprint arXiv:1308.0850.

[13] Britz, D., Goldie, A., Luong, M. T., and Le, Q. (2017). ”Massive exploration of neural machine translation architectures”.arXiv preprint arXiv:1703.03906.

[14] Akiko Eriguchi, Kazuma Hashimoto, and Yoshimasa Tsuruoka. 2016. ”Tree-to-Sequence Attentional Neural Machine Translation”. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics* (Volume 1: Long Papers), pages 823–833, Berlin, Germany. Association for Computational Linguistics.

[15] Akiko Eriguchi, Kazuma Hashimoto, and Yoshimasa Tsuruoka. 2019. ”Incorporating Source-Side Phrase Structures into Neural Machine Translation”. Computational Linguistics, 45(2):267–292.

[16] Hao Peng, Sam Thomson and Noah A. Smith. 2017. ”Deep Multitask Learning for Semantic Dependency Parsing”. In ACL.

[17] Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. ”BLEU: A Method for Automatic Evaluation of Machine Translation”. In Proceedings of the 40th Annual Meeting on Association for Computational Linguistics, pages 311–318.

[18] Suvarna G Kanakaraddi and Suvarna S Nandyal. 2018. Survey on Parts of Speech Tagger Techniques. In 2018 International Conference on Current Trends towards Converging Technologies (ICCTCT). IEEE. Coimbatore, India.

[19] Rico Sennrich and Barry Haddow. 2016. Linguistic Input Features Improve Neural Machine Translation. In Proceedings of the First Conference on Machine Translation: Volume 1, Research Papers, pages 83–91, Berlin, Germany. Association for Computational Linguistics.

[20] Huadong Chen, Shujian Huang, David Chiang, and Jiajun Chen. 2017. Improved neural machine translation with a syntax-aware encoder and decoder. In Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pages 1936–1945. Association for Computational Linguistics.

[21] Emanuele Bugliarello and Naoaki Okazaki. 2020. Enhancing Machine Translation with Dependency-Aware Self-Attention. In Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, pages 1618–1627, Online. Association for Computational Linguistics.

[22] Emma Strubell, Patrick Verga, Daniel Andor, David Weiss, and Andrew McCallum. 2018. Linguistically-Informed Self-Attention for Semantic Role Labeling. In Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, pages 5027–5038, Brussels, Belgium. Association for Computational Linguistics.

[23] Shuangzhi Wu, Dongdong Zhang, Zhirui Zhang, Nan Yang, MuLi, andMingZhou.2018. Dependency-to-dependency neural machine translation. IEEE/ACM Transactions on Audio, Speech, and Language Processing, pages 2132–2141.

[24] Anna Currey and Kenneth Heafield. 2019. Incorporating source syntax into transformer-based neural machine translation. In Proceedings of the Fourth Conference on Machine Translation (Volume 1: Research Papers), pages 24–33,

[25] Florence, Italy. Association for Computational Linguistics. Timothy Dozat and Christopher D. Manning. 2017. Deep biaffine attention for neural dependency parsing. In ICLR.

[26] Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016. Neural Machine Translation of Rare Words with Subword Units. In Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pages 1715–1725, Berlin, Germany. Association for Computational Linguistics.

[27] Michael Roth and Mirella Lapata. 2016. Neural semantic role labeling with dependency path embeddings. In Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (ACL), pages 1192–1202.

[28] Yonatan Bisk and Ke Tran. 2018. Inducing Grammars with and for Neural Machine Translation. In Proceedings of the 2nd Workshop on Neural Machine Translation and Generation, pages 25–35, Melbourne, Australia. Association for Computational Linguistics.

[29] Mitchell P. Marcus, Mary Ann Marcinkiewicz, and Beatrice Santorini. 1993. Building a large annotated corpus of English: The Penn TreeBank. Computational Linguistics – Special issue on using large corpora: II, 19(2):313–330.