UNIVERSITAT POLITÈCNICA DE CATALUNYA
BARCELONATECH

Escola Tècnica Superior d'Enginyeria
de Telecomunicació de Barcelona

Linguistic-family-specific Encoders and Decoders for Multilingual Machine Translation

A Master's Thesis

Submitted to the Faculty of the

Escola Tècnica d'Enginyeria de Telecomunicació de Barcelona

Universitat Politècnica de Catalunya

by

YINING YANG

In partial fulfilment

of the requirements for the degree of

MASTER IN TELECOMMUNICATIONS ENGINEERING

Advisor:

CARLOS ESCOLANO PEINADO

MARTA RUIZ COSTA-JUSSA

Barcelona, January 2022

Title of the thesis: Linguistic-family-specific Encoders and Decoders for Multilingual Machine Translation

Author: Yining Yang

Advisor: ESCOLANO PEINADO, CARLOS

RUIZ COSTA-JUSSA, MARTA

Abstract

Multilingual Machine Translation has been approached from different perspectives including the shared and the language-specific encoders-decoders. The shared one use a single encoder and decoder for all languages but the language-specific encoders-decoders allocate encoder and decoder for each language. Both perspectives have their own benefits and drawbacks on translation quality and resource consumption aspect. To find a balance of these two factors, this project explores a new approach that is to share the encoders and decoders for language families.

The new model was train and test on the TED2020 dataset with 21 chosen languages to form 4 language families. Comparison between the all-language shared baseline and our model shows a great improvement in BLEU score which can from 3 points to maximum 10 points according to the family pairs.

The new model also has a good performance of zero-shot translation, which outperforms that of the baseline model and the improvement follows the rule of growth concluded from the model training.

Acknowledgements

Firstly, I want to thank Marta and Carlos, who are director and co-director of this thesis, thank them for giving me chance to study such a interesting topic. I also thank their continuous help during my work.

Then I want to thank to my roommate Hu, she accompanied me throughout the entire period of thesis writing and gave me advices at language aspect.

Finally I want to thank my parents, they support me no financially and emotionally during my Master's studies.

Revision history and approval record

Revision	Date	Purpose
0	04/12/2021	Document creation
1	06/01/2022	Document revision
2	18/01/2022	Document revision

Written by:		Reviewed and approved by:	
Date	04/12/2021	Date	-
Name	Yining Yang	Name	Carlos Escolando Peinado Marta Ruiz Costa-Jussa
Position	Project Author	Position	Project Supervisor

Contents

List of Figures

List of Tables

1. Introduction

Machine translation, also called MT, is an important subfield of NLP (Natural Language Processing) which aims to translate one language to another using computer or computational resource. Nowadays many popular online translators such as google translate and some personal assistant applications such as Siri are based on technologies in this area. The development of MT helps the world become closer, the communications of people from different countries with different cultures and languages have been enhanced during recent years no matter in life, literature, or academic area. It now becomes an essential part of human life as it both reduces the time cost and labour cost.

Deep learning is a subfield of machine learning which is essentially a network that contains 3 or more layers. Techniques for this field developed very fast in recent 20 years. As what we want for MT is to get a translation as close as human translation, and deep learning also attempts to mimic the human brain through a combination of data inputs, weights, and bias. Machine translation based on deep learning has also grown at a rapid pace in the last five years and now can get a relatively accurate result.

The initial MT task was based on a translation system between two languages, researchers then discovered that the framework also could work on multiple languages. Therefore, a new system called multilingual neural machine translation appeared, which could deal with the problem of translation between several languages. This system was built based on the idea of sharing parameters between languages, it shares a common attention mechanism among all languages and also could learn a shared representations between languages according to the encoders and decoders.

Multilingual Machine Translation has been approached from different perspectives including the shared and the language-specific encoders-decoders. The language-specific encoders-decoders model, which is proposed by Firat et al[11], allocate one encoder and decoder for each language, this kind of model could maximize the performance for the languages, however, the parameters grows linearly with the number of languages, Johnson et al[10] shows that such models with large number of parameters are not required. Johnson et al[10] then proposed a model that all languages share the same embeddings, encoder, decoder, and attention mechanism, this kind of model has the maximum simplicity and the minimum parameter size, but the translation quality may decrease as the growing of language numbers. However, this kind of model have another benefit which the language-specific model not have is that it is capable of zero-shot translation, which is the translation between language pairs with no training data.

1.1. Motivation

The fully shared system has a single model for all languages, at the cost that the capacity of the model may limit the performance, however, the language-specific model grow linearly with the number of languages, which could be a hardware limitation.

As the MNMT system could encode the text from different languages in a shared representation space, studies from Kudugunta et al[9] shows that languages from the same family may have a similar representation. Using this feature, this project explores an intermediate approach which is to share the encoder and decoder for language

families, which is the trade-off where the capacity is split between several models but the required resources grow much slower than linear, another benefit for this kind of model is that it could reduce noise from different alignments or vocabularies by sharing languages that are similar.

1.2. Objective

As mentioned before, this project aims to explore a linguistic-family-specific encoder-decoder model. To do this, the following objectives should be achieved:

- Create a dataset with chosen language pairs
- Train an all-language-shard model as the baseline
- Arrange the parameters and train a linguistic family model as target
- Train two models which are the zero-shot version of the baseline and target
- Analyse the evaluation result for all models

1.3. Requirements and specification

During the experiments, some requirement should be considered, the dataset and tools are shown below:

- TED2020[15], which is the dataset where the chosen language data comes from.
- Fairseq[6], which is a sequence modeling toolkit that allows researchers and developers to train custom models for translation, summarization, language modeling and other text generation tasks.
- Pytorch, which is a Python package that provides two high-level features: tensor computation (like NumPy) with strong GPU acceleration and deep neural networks built on a tape-based autograd system
- Anaconda, which is a distribution of the Python and R programming languages for scientific computing (data science, machine learning applications, large-scale data processing, predictive analytics, etc.), that aims to simplify package management and deployment.
- Subword-nmt[5], which is a tool to segment text into subword units.
- Moses[16], which is a free software, statistical machine translation engine that can be used to train statistical models of text translation from a source language to a target language. This project uses it to pre-process the text data.

1.4. Outline

Based on the purpose mentioned before, the thesis is divided into five chapters:

- Chapter 2 is the state of art which shows the development status of NLP and brief description of techniques applied in the project.
- Chapter 3 is the methodology used, it main explained the model architecture used in this project.
- Chapter 4 is the implementation of the project, which shows the steps that should be done during the experiment.

- Chapter 5 shows the evaluation result using BLEU for the baseline and target model and analysis based on the result.
- Chapter 6 make a conclusion about the project.

2. State of the art

2.1. Natural Language Processing

Before artificial intelligence appears, the machine could process some structured data, but in fact, in our daily life, most of the data are unstructured, for example, the image, the audio, or the text. As the main carrier of natural language, the text contains the largest amount of information among these kinds of data. The machine could not understand the information directly, so we need to process the data, then NLP was born for this purpose. As the common way to process the raw text is at token level because tokens are building blocks of natural language, the piece of text will be separated into tokens. The predefined dictionary of the model will be built according to the tokens that appear in the corpus.

NLP now is mainly divided into the following fields:

- Text searching, which is used for search text among massive data.
- Machine translation, which is to translate a language to another using computer.
- Text classification, which is to assign categories to a sentence or documents, this technique now is widely used in emotion classification, language detection, topic labeling, and so on.
- Information extraction, which is to extract the desired information from irregular text, such as Named Entity Recognition.
- Speech recognition, which enables the recognition of person sounds and translation from speech to text.
- Speech translation, which is capable to translate speech from one language to speech in another.

NLP can use machine learning and deep learning methods, for these two methods the process steps are different. The step for the machine learning method is like figure 1, and that for the deep learning method is shown in figure 2.

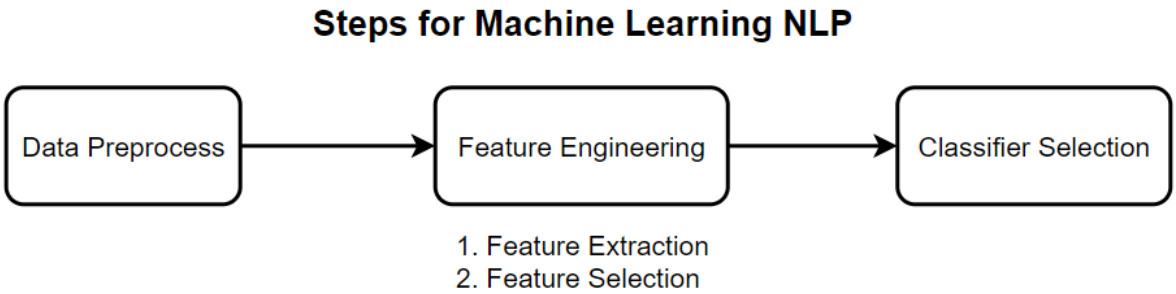

Figure 1: Machine Learning NLP steps

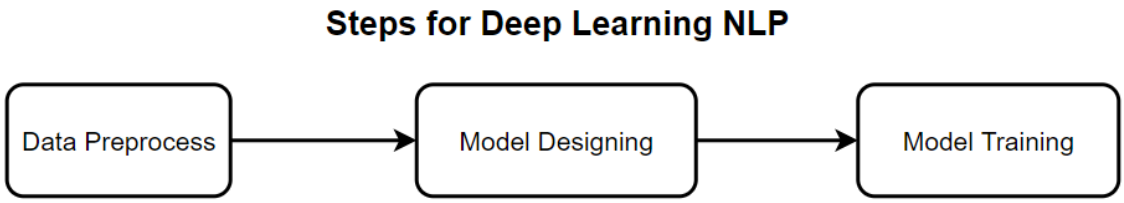

Figure 2: Deep Learning NLP steps

No matter in which method data pre-processing is the first step, for each NLP task, this step could be a little different. The feature engineering step is very important in machine learning method, it could be morphological information such as Part-of-Speech Tagging (POS) or alignment information between parallel sentences, all of these aims to achieve a higher accuracy using machine learning algorithm. Text unlike traditional data is not a fixed set of independent features, but a sequence of elements where the order and the context where they appear affects the information they provide, so in deep learning algorithm, a model with nural networks such as CNN, RNN, and Transformer is built to process long-term dependecies, encoding information from the whole sequence.

2.2. Neural Machine Translation

Machine translation is one of the main tasks of NLP, the purpose is to translate one language to another using machines. Neural machine translation is a deep-learning-based approach for MT. This kind of approach was first tried in the last century [1], but it only has made dramatic development in recent years due to poor hardware conditions at that time. Due to the neural network architecture it has, it can learn from a large amount of data and could quickly adapt to new contexts, this lead to its widespread use in many companies today.

Nowadays, the NMT model is designed for the end-to-end translation task, which could directly process the source sequence and target sequence, learning from these data and finally generate the corresponding target sequence according to the input source sequence. This is implemented using an encoder-decoder architecture proposed by Sutskever[13]. A simple encoder-decoder model structure is shown in figure 3. The encoder part reads the input source sentence and encodes it into a fixed-length vector, the decoder part reads the output vector from the encoder and finally generates the target sentence.

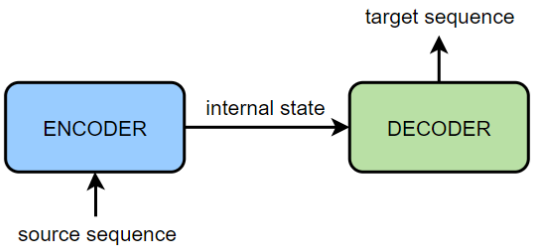

Figure 3: Encoder-Decoder Model Structure

As this kind of model could only memory short sentences, to fix this problem, the attention mechanism on NMT was first introduced by Bahdanau et al[14] in 2014. This mechanism will predict the next word by concentrating on a few relevant parts of the sequence. The encoder representations which are converted from raw text at the encoder side, the attentions and the previous generated word are used in decoder to generate a decoder representation which will be then used to generate the next target word.

2.3. Multilingual Neural Machine Translation

After the neural system could process machine translation, researchers find that this framework also could naturally incorporate with multiple languages. This task is so called multilingual NMT, which aims to translate multiple language pairs using one single model.

The common model architecture used for MNMT was Encoder-Attention-Decoder which is based on the encoder-decoder architecture with an attention mechanism that we mentioned before. There are two classical approaches for MNMT, one is minimal parameter sharing proposed by Firat et al[11], another is a complete parameter sharing model proposed by Johnson et al[10].

The minimal parameter sharing model separate the embeddings, encoders and decoders for each language but share the same attention mechanism. The attention score is calculated according to the specific encoders and decoders. As for each language it has its specific encoder and decoder, this model could get a maximum performance of translation, but the parameters increase linearly according to the language numbers. Also, this kind of model is very flexible because special process on individual language is possible.

The complete parameter sharing model is a model with all languages sharing the same encoder, decoder and attention mechanism. All data will be merged and at the input side every sentence will add a special tag which indicates the target language. This tag will help decoder generate the correct language although all languages share the same decoder parameters. Shatz[12] finds that training model with massive language pairs may help a poor-resource language get extra knowledge from the other languages. Johnson's work also shows that this kind of model have improvement on low-resource translation, as for the zero-shot performance, which shows the performance of translation between language pairs with no training data, it also has a good result.

2.4. BLEU

BLEU, which is called bilingual evaluation understudy, is a common evaluation method for the neural machine translation task proposed by Papineni et al[3], it's a score to compare the generation translation and the reference translation.

To calculate the score, first we assume that the perfect match result in a score 1.0 and a perfect mismatch result in a score 0.0. We usually think the higher the BLEU score, the closer the machine translation is to a human translation. But in practice, human translation may have smaller score because the two for comparison may use different vocabulary and phrasing.

The calculation is like:

$$BLEU = BP \cdot exp\left(\sum_{n=1}^{N} w_n log P_n\right)$$

Where P_n is the modified precision to calculate the sum of the clipped n-gram counts for all candidate sentences divided by the number of candidate n-grams, where the n-gram means n tokens that appear together in a sentence. *N* represent the number of n-grams, unigram, bigram, 3-gram and 4-gram are usually used, so *N* is usually 4. w_n is the weight for each n-gram. *BP* is the brevity penalty which is to avoid the case that the model generates a half but correct translation leading to a perfect match, it can be calculated by:

$$BP = \begin{cases} 1 & if\ c > r \\ e^{(1-r/c)} & if\ c \leq r \end{cases}$$

Where *c* is the length of the machine translation result and *r* is the length of the reference translation. Only the candidate translation length is the same as the reference length it could get a 1.0 BP score.

3. Methodology

3.1. Project Implementation Steps

The main purpose of the project is to explore an approach to find balance in language-specific encoder-decoder model and all-language-sharing model. The flow of the experiment is the following: dataset choosing - data pre-processing - model training - model testing.

As the most important part of the experiment, the model used in this project is based on the transformer architecture from fairseq tool, which is explained in the following part.

3.2. Transformer

This project build models based on the transformer architecture, which is a wide use model proposed by Vaswani et al[2].

This kind of model is also based on the encoder-decoder architecture but make use of attention mechanism. It first converts the input tokens into word embeddings, which are real-value multidimensional vectors for each word that encode the word meanings. As the transformer cannot capture any relative information about the position of the words in the sentence, the position encoding is needed. It's a vector that has the same dimension as the input embeddings so they can sum up. Then the encoder map the input sequence into a continuous representation, during this process, the attention mechanism will be used.

The attention mechanism is the most important part of this model. The attention used in this model is so called scaled dot-product attention, the input was queries, keys of dimension d_k and values of dimension d_v. This model computes the output of matrix in this way:

$$Attention(Q, K, V) = softmax\left(\frac{QK^T}{\sqrt{d_k}}\right)V$$

Where first compute the dot product of a set of queries with all keys, then divide the result by $\sqrt{d_k}$, finally apply a softmax function to obtain the weight of the value.

Instead of performing a single attention function, transformer performed a multi-head attention which could help the model expand the ability to focus on the information of different representation on different positions.

The decoder generates the output sequence one element at a time, and at each step the model is auto-regressive. After getting the encoder representation, the decoder generates the output sequence using the encoder representation and previous generated tokens as additional input to predict the next token until reaching the end of sentences.

The model architecture is shown in figure 4.

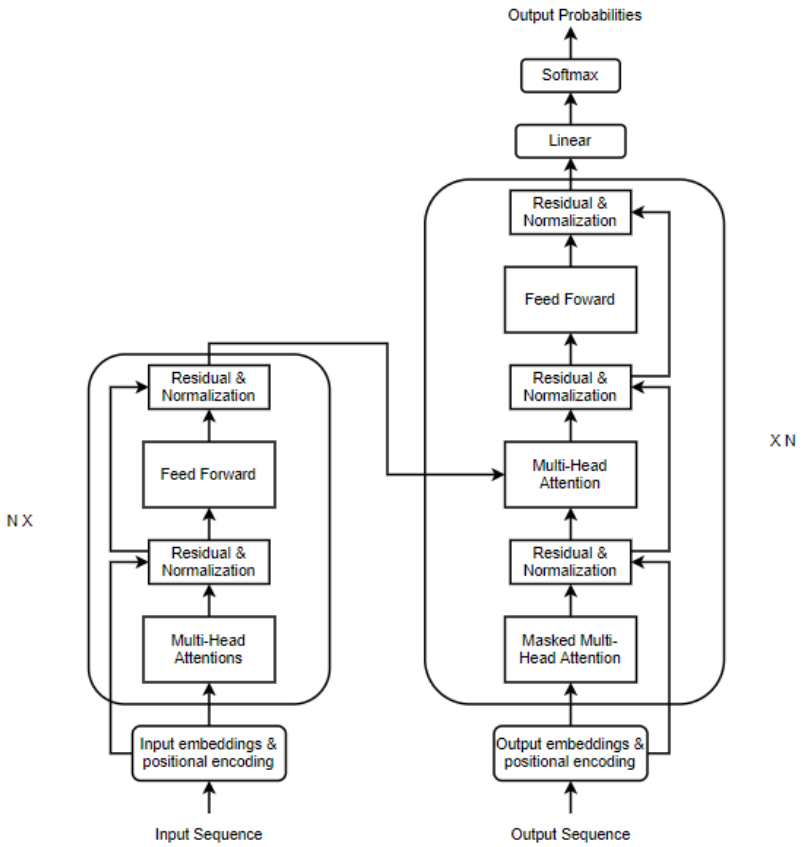

Figure 4: Transformer Architecture

The encoder part is at the left side which main has 6 identical layers, for each layer there are 2 sub-layers which are multi-head self-attention mechanisms and a position-wise fully connected feed-forward network. Between the sub-layers there is a residual connection which is followed by layer normalization.

The decoder part is at the right side which is similar as the encoder parts, it also has 6 identical layers, for each layer it also has the two sub-layers that also contained in the encoder part, beside that there is another sub-layer which performs multi-head attention which also pay attention to the representations generated by encoders. Finally, there is a linear transformation and softmax layer which can convert to decoder output to next target token probability.

In this architecture, the multi-head attention is used in three parts. The first part is the self-attention layer in the encoder, in this layer, the query, key and value are from the same place where is the previous layer of the encoder. The second part is the self-attention layer at the decoder which is similar as the one in encoder, to ensure the auto-regressive, all values in the input of the softmax which correspond to illegal connections are masked out when calculate the scaled dot product attention. The last part is in the encoder, which is a cross-attention layer where the queries are from the previous layer of decoder but the keys and values are from the output of the encoder.

The output of the decoder will be finally fed into a linear layer followed by a softmax layer. These two layers will help the target vector convert into the tokens.

In our project, we also implemented beam search at the final part, it will allow the model search for predetermined number of best results, it could help the model backtrack some errors.

3.3. Model architectures

This project is based on the idea of fully-shared model and language-specific model, also called the complete parameter shared model and minimal parameter shared model that mentioned in section 2.3. Both of the models will use the architecture of transformer shown in previous section.

The architecture of the fully-shared model is more or less the same as the transformer architecture shown in figure 4 because it only has one encoder and one decoder. As all language shared the encoder and decoder in this model, it is important to add a tag at the beginning of the source sentence to indicate the target language.

The language-specific model proposed by Escolando[8] makes some modifications on the transformer architecture as it has multiple encoders and decoders. A simple example of this kind of model is shown as figure 5.

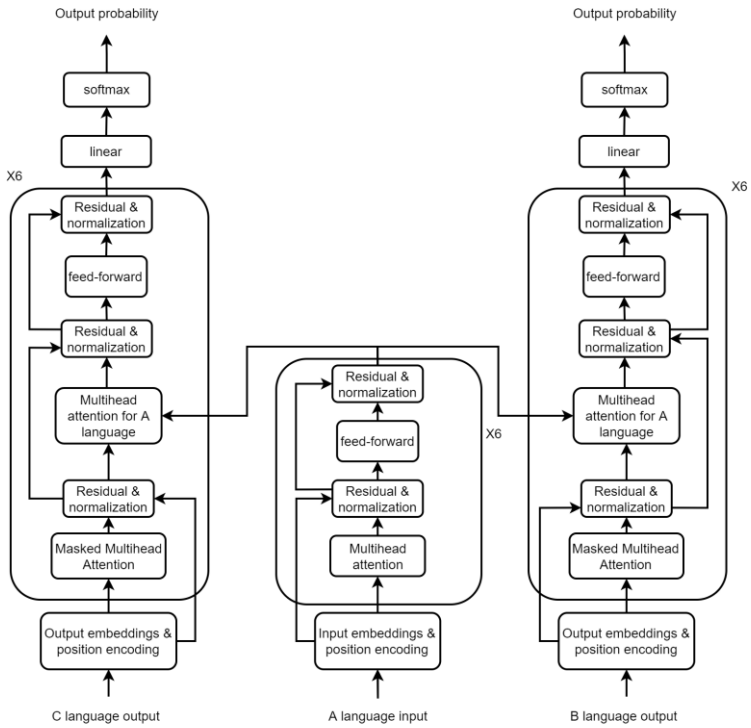

Figure 5: Architecture for language-specific model

Figure 5 shows a simple architecture of one source language and two target language, each language has its own encoder and decoder. To add new language pairs that contain new source or target language, the model could add its specific encoder and decoder, and also the corresponding cross-attention layer at the target decoder side.

The linguistic-family-specific model architecture is based on the language-specific one, but it uses language family instead of single language. Also, for this model every encoder and decoder will be shared among several language that belongs to the same family, the tag that indicate the target language is needed in the source sentence.

4. Experiments

4.1. Dataset

As this project aims to test the performance of linguistic family shared encoder decoder model, the TED2020 dataset which contains totally 108 languages was chosen as the training database, we choose 21 languages and divide them into four families according to the distance calculated by Gamallo et al[4].

The four group and their languages are:

A group	Catalan, Spanish, French, Galician, Romanian, Italian, Portuguese
B group	Bosnian, Russian, Slovenian, Czech, Slovak, Polish, Croatian
C group	Danish, Dutch, German, English, Swedish
D group	Traditional Chinese, Simplified Chinese

Table 4.1: Language groups according to the distance

We could see that the languages of group A are mostly from the romance languages, Most languages from group B are from Slavic languages, most language from group C are from Germanic languages, and the last group D, it contains two languages, but they are totally the same language but write in a different way.

TED2020 is a parallel corpus which means the training data have sentence pairs for source and target languages that are corresponding. As the project aims to explore the influence of sharing encoder and decoder of linguistic family, the training data are merged into translation scripts between language groups. The sentences numbers contained in each family pair are shown in table 4.2.

language groups	A	B	C	D
A	-	6.3 M	5.7 M	3.97 M
B	6.3 M	-	4.3 M	2.5 M
C	5.7 M	4.3 M	-	2.34 M
D	3.97 M	2.5 M	2.34 M	-

Table 4.2: Sentences number of family pairs

The dataset sizes for languages are not balanced, table 4.3 shows the sentences number percentage for each language, for the all-language shared model each language should be considered as a part of total data, but for linguistic family model, each language could be considered as a part of its language family, it is important to note data from some languages are much smaller than others, such as Bosnian, Slovak and Galician, which will lead to a worse result compared with other languages, this will be analysed in detail in Chapter 4.

Language Family		shared	family		shared	family
A	ca	1%	3.20%	it	6%	20%
	es	6%	20%	pt	5%	15.68%
	fr	6%	20%	ro	6%	18.86%
	gl	0.60%	2%			
B	bs	0.20%	0.8%	ru	8%	29.60%
	cs	4%	16%	sk	2.20%	8.60%
	hr	4.40%	16.90%	sl	1%	3.76%
	pl	6%	24%			
C	da	1.60%	6.40%	de	6.40%	25.70%
	en	7.80%	31.50%	nl	6.60%	26.64%
	sv	2.30%	9.50%			
D	zh_cn	8.60%	50%	zh_tw	8.60%	50%

Table 4.3: data proportion of each language for shared model and family model

Another problem for the TED2020 dataset is that it doesn't have the valid and test set which is necessary for training the model. To solve this problem, the test and valid dataset were built by random selecting from the training data and remove them in the training data. The size of these two datasets was depend on the size of language pair data, if the size is big enough, we choose 2000 sentence for test and 2000 sentence for validation, if not, the size is determined according to 60% (train data): 20% (validation data): 20% (test data).

4.2. Data Pre-process

As the data now is in raw text, before training the model, first thing is to pre-process the data. The process is the same among all the language we selected except Chinese. The common steps to pre-process the data are tokenization, cleaning, truecasing and finally convert the text file into binary format in order to feed in the training model.

4.2.1. Tokenization

Tokenization is the first step to pre-process the data. The purpose to do separate the words and punctuations to avoid ambiguity of the same word. For Chinese and other language, the difference in data pre-process is mainly in this step.

Tokenization is to generate text with tokens according to the raw text sentences. Tokens are usually words and punctuations. For languages such as English this step is very easy because the words are already separated by space, the only thing should be done is to add the space between words and punctuations. But for languages such as Chinese, tokens cannot be generated directly because the words of Chinese are not separate. Before the token generation for Chinese the first thing is to do text segmentation.

Jieba[1] is a popular tool to do text segmentation for Chinese. After installing it, it could be easy using by call just one line command. One text segmentation result is shown in table 4.4.

Text segmentation on Chinese using Jieba tool	
Raw Text	我来到北京清华大学
Segmentation	我 来到 北京 清华大学

Table 4.4: Text segmentation example on Chinese using Jieba tool

4.2.2. Cleaning

Cleaning is a step to set a threshold for the sentences, for this project the threshold was set to the default value 1-50, which limits the minimum and maximum number of tokens in one input sentences, it could ensure training sentences with a similar size. One thing should be aware of in this step is that we both clean the data for the source language side and target language side to ensure the correspondence of sentences.

This step is only for the training data, for the valid and test data it will be skipped because we want to test the generalization capabilities of our models for actual data including very long sentences.

4.2.3. Truecasing

Truecasing is a step to convert the words in raw text to their probable case, this could transfer the words into lowercase and also check the correctness of some words and rectify them. But this should first train a small truecase model using the train data, and finally using this model both on train, valid and test set to finish the truecasing. This step helps the model reduce the data sparsity.

Also, this step is not need for languages such as Chinese.

4.2.4. BPE Generation

The models usually could train with a limited number of vocabulary, however, the total vocabulary size of multiple language may be really large and some of rare word would be out of the limit vocabularies and the system would transfer it to a unknown token. This will lead to the decrease of the translation quality because the model couldn't learn the exact word during training.

To avoid this problem, our project will apply Byte Pair Encoding (BPE), which is evolution of tokenization which is first present by Sennrich et al in 2015[5]. The main purpose for this step is to reduce the vocabulary size while being able to represent the words in the sentences, even when are not seen during training.

The algorithm of the BPE is first count frequency of the words that appears in one corpus, and then add a stop token at the end of each word, then split the word into characters and count all possible consecutive character pair of each word, so after that we will have a dictionary of all words in the corpus and their corresponding consecutive character

[1] https://github.com/fxsjy/jieba

pairs, then we count the frequency of the consecutive character pairs and merge the most frequent character pair. Next keep iterating the step mentioned before until the set token limit.

To do this, we will use a tool called subword-nmt which is also proposed by Sennrich[5], the machine first learn a code from train data for each language, then apply it on both train, validation and test data.

4.2.5. Preprocess

This project experimented two kinds of model which are all language shared model and family shared model, that means the encoder and decoder will process multiple language, to ensure that the decoder knows which the target language is, a tag that indicates the target language is necessary to be added at the beginning of each sentence for source language files.

Different models need different data, according to the model all language files are merge into larger files which is suitable for the encoder and decoder.

The final step before training is to convert the text file into binary format in order to increase the reading efficiency as it is much faster to read binary data than reading sentences from disk, this can be done by using the preprocess tool of fairseq. The final formats for the data files are a bin file and an idx file with a dictionary for each encoder and decoder. As the dictionary size was set to a limit number and it could not contain all tokens in the corpus, some tokens out of the dictionary were replaced by unknown token <unk>.

4.3. Train

4.3.1. Model choosed

The aim for this project is to compare the performance for an all-language shared model and a linguistic family shared model.

The all-language shared model has only one encoder and one decoder, which could encode and decode all languages, so the model only needs one source file and one target file and it could translate all directions for the language pairs. Also, the model is smaller than the other one, which could both save training time and save the memory.

The linguistic family model has the encoders and decoders according to the linguistic family. The data for this model are in family pairs, each encoder and decoder could only process the data of its specific linguistic family. One drawback to use the fairseq tool is that during training it will only read data from the first direction of one family pair no matter if there is data of another direction. At the encoder side tags were added to indicate the target language, so data for different direction shouldn't be the same. This problem means that the model could not do bidirectional translation. Due to the multiple encoders and decoders, this model was about three times the size of the all-language shared one, which means that it needs more time to train. But when training and translating one language pair, as the model focus on just one encoder-decoder pair, these two models uses the same parameters.

4.3.2. Train parameters

The train step will use the script from fairseq tool, the parameters description could be found in the fairseq documentation, the main things to change is the working and saving direction, the task and the language pair which shows the encoder-decoder pair. For the all-language-shared model the language pair should be set to src-tgt, for the linguistic family model this should be set to the family pairs.

4.4. Evaluation

For the project the final thing is to test the performance of the trained model. To do this, a script called generate from fairseq tool was used, which could compute the BLEU score to show the result.

This process needs the dictionaries of each encoder and decoder that were generated in the data pre-process part. The models were trained with the merged file before, now for the evaluation step, data should be processed but not merged using the dictionaries and finally be feed into the model to generate BLEU score for each language pair.

The evaluation result is shown in Chapter 5.

5. Evaluation Result

The evaluation results are shown as BLEU score, which is compare the similarity between the machine translation result and human translation script. The comparison is mainly divided into two parts, the first parts is to compare the BLEU scores of the two models mentioned in 4.3.1 according to the family pairs, the second part is to compare the performance of zero-shot for these two models.

5.1. BLEU scores comparisons

5.1.1. Family A to B

This part shows the result for language from A group translate to language from B group, the BLEU scores for each language pair are shown in table 5.1, the columns show the languages from family B and the rows show the languages from family A.

		bs	cs	hr	pl	ru	sk	sl
shared model	ca	12.73	13.17	13.92	11.51	12.85	13.19	11.23
family model		15.15	16.97	17.75	15.21	16.83	16.15	14.20
shared model	es	14.69	**14.75**	**17.24**	**14.33**	**16.47**	14.68	12.46
family model		17.41	**19.26**	**21.84**	**19.63**	**21.96**	18.79	15.47
shared model	fr	13.09	**14.18**	**16.31**	**12.66**	**15.16**	14.51	12.09
family model		15.59	**18.63**	**21.12**	**16.78**	**19.83**	18.45	14.96
shared model	gl	10.84	12.80	14.18	12.55	13.79	12.74	10.51
family model		13.62	16.54	18.31	16.36	18.05	16.58	13.64
shared model	it	13.46	14.52	15.93	13.51	16.40	14.03	12.08
family model		15.99	18.39	19.95	17.89	21.18	17.99	14.84
shared model	pt	14.33	**14.12**	**15.93**	**13.11**	**14.61**	13.99	12.2
family model		16.72	**18.61**	**21.24**	**17.81**	**19.40**	17.79	15.2
shared model	ro	12.93	14.15	15.67	12.99	15.38	13.48	11.96
family model		15.20	18.69	19.98	17.23	20.19	17.59	14.68

Table 5.1: BLEU scores for family pair AB

From the table it's obvious that the results for the linguistic family model are overall better than that for the all-shared model. The average improvement of BLEU score for this family pair is about 3.8 and the average distance between these two families is 130 according to Gamallo's result in [4]. For the language pairs that both source and target have smaller dataset such as ca-bs and gl-bs, the improvement of BLUE is smaller than 3. For the language pairs that contain both larger source dataset and larger target dataset the improvement can reach to a maximum 5.5. Only Polish in language family B doesn't comply to this, according to figure 5 Polish has the largest distance to other languages in its language family, this could be the reason cause this case.

Although the size of dataset for language pairs are not balanced, the models still get results that are very close, which shows that the training the languages for the families can be mutually beneficial, especially for those languages with similar data size and smaller distance, such as Spanish, France and Italian, the result are much closer.

5.1.2. Family A to C

This part shows the results of languages from family A translate to languages from family B, the scores are in table 5.2, the columns show the languages from family C and the rows show the languages from family A.

		da	de	en	nl	sv
shared model	ca	19.61	17.10	**27.76**	18.07	19.09
family model		26.31	22.84	**33.81**	23.08	24.94
shared model	es	22.44	20.06	**37.01**	23.27	21.95
family model		29.65	27.60	**44.84**	30.18	29.37
shared model	fr	21.59	19.37	**33.28**	22.91	21.47
family model		27.63	26.07	**39.44**	28.60	27.54
shared model	gl	18.51	17.68	**30.86**	20.32	18.77
family model		24.67	23.45	**37.19**	26.00	25.36
shared model	it	20.41	18.52	**32.87**	21.71	20.95
family model		27.29	25.02	**39.37**	27.71	27.38
shared model	pt	21.14	19.00	**34.81**	22.45	21.43
family model		28.06	26.61	**43.22**	29.24	28.62
shared model	ro	19.93	18.65	**30.57**	19.71	18.97
family model		25.69	25.34	**38.25**	25.81	26.32

Table 5.2: BLEU scores for family pair AC

In general, the results for this pair are much better than that for family pair AB, one result could be the language family C has only 5 languages which means in linguistic family model for each language the data proportion increase greater. Another factor is that distance between language pairs in this group is overall much closer than that in group AB, which could be the reason lead to a higher baseline BLEU score.

The average BLEU score improvement for this family pair is about 6.6, according to Gamallo's result, the average distance between these two families is 33.8, it shows that these two language families are closer than that of A and B, this could lead to a better BLEU score for the language pairs belongs to this family pair. The improvements of this family pair are much more balanced, almost all language pair has an improvement between 6 and 7. The language pairs for English get the best results, that is because English is the nearest language to family A in family C and it also has the largest dataset among all dataset for family C. The rest four languages are closer to each other, so their results appear more similarity.

5.1.3. Family pair A to D

This part shows the results of languages from family A translate to languages from family D, the scores are in table 5.3, the columns show the languages from family D and the rows show the languages from family A.

	zh_cn		zh_tw	
	shared model	family model	shared model	family model
ca	10.4	17.96	9.34	16.02
es	**12.24**	**21.5**	**12.21**	**20.19**
fr	12.72	19.89	11.68	18.68
gl	11.56	18.65	11.14	18.72
it	12.83	20.81	10.97	18.25
pt	13.05	21.37	12.4	20.69
ro	11.72	19.95	10.57	18.06

Table 5.3: BLEU scores for family pair AD

Even though Chinese accounts for a large proportion of the data, the results for language pair AD show the lowest BLEU score among all family pairs that contain family A, that is because Chinese is a language totally different from languages of family A, so that the distance between these two families is very large. As the traditional Chinese and simplified Chinese are in fact the same language, they show a result that is very close. The result for simplified Chinese is a little better than that of traditional Chinese, this is because the traditional Chinese has some character that is also contained in simplified Chinese, but the rest so called traditional characters are not.

The improvement of BLEU score for this family pair is also remarkable. The best result is shown as the language pair of Spanish and Chinese. The average improvement of this family pair is about 7.7, which is the highest among the family pairs contained family A. This means that to define a specific encoder and decoder for Chinese but not share embeddings with other language will notably benefit for the language pairs that contains Chinese. That is because it will use a family specific dictionary in such method. Chinese uses a different script with several thousand different characters, having whole dictionary to represent the language is beneficial, giving more representation capacity.

5.1.4. Family pair B to C

This part shows the results of languages from family B translate to languages from family C, the scores are in table 5.4, the columns show the languages from family C and the rows show the languages from family B.

		da	de	en	nl	sv
shared model	bs	17.51	16.59	28.86	18.91	19.76
family model		23.61	22.23	34.95	23.90	25.07
shared model	cs	18.62	16.75	26.05	18.26	17.37
family model		24.21	22.84	32.64	24.33	24.15
shared model	hr	**20.60**	**18.13**	**30.52**	**19.63**	**20.14**
family model		**27.37**	**25.19**	**37.86**	**26.62**	**26.60**
shared model	pl	14.72	14.91	21.97	16.62	15.00
family model		19.47	20.02	26.80	21.75	20.21
shared model	ru	15.99	14.91	24.91	17.08	16.32
family model		20.98	20.46	30.08	22.56	21.85
shared model	sk	17.33	16.78	25.48	18.20	17.15
family model		23.90	23.5	32.03	23.86	23.29
shared model	sl	15.13	11.11	20.02	14.85	15.04
family model		20.40	17.73	25.07	19.75	20.28

Table 5.4: BLEU scores for family pair BC

The average improvement of BLEU score in this family pair is about 5.83, and the average distance between these two families is about 57.4 according to Gamallo's result, it shows a distance closer than family pair AB but farer than family pair AC. The BLEU scores for this language pair are better than that of family pair AB but worse than family pair AC. This shows a relation between the distance of two language families and BLEU scores, higher BLEU scores may be obtained by closer language families.

Some results for specific language pairs are the same as what we have discussed in previous part, but this time the languages from family B are shown in the encoder side.

In this family pair, the language pairs that contain Polish and Russian show an improvement lower than 5.83 in average. The highest improvement of BLEU was appeared in language pairs contained Croatian.

5.1.5. Family pair B to D

This part shows the results of languages from family B translate to languages from family D, the scores are in table 5.5, the columns show the languages from family D and the rows show the languages from family B.

	zh_cn		zh_tw	
	shared model	family model	shared model	family model
bs	**11.1**	**18.6**	**9.15**	**16.01**
cs	**11.17**	**18.98**	**9.7**	**17.14**
hr	**11.36**	**19.21**	**10.80**	**17.85**
pl	10.37	16.82	8.89	15.19
ru	10.59	16.83	10.46	15.83
sk	**9.69**	**17.47**	**9.4**	**16.89**
sl	8.55	13.41	7.17	11.35

Table 5.5: BLEU scores for family pair BD

The distance between family B and family D is also very large, so the BLEU scores for the baseline are very low compared to that of other family pairs. In this family pair, the BLEU score increases 6.65 in average.

Due to the low BLEU score of this family pair, the differences language pairs are not remarkable. From the table we can see that the results for Bosnian-Chinese, Czech-Chinese and Croatian-Chinese are closer than that of other language pairs no matter in the translation or the improvement which is about 7 points. Although Bosnian only contains a little part of data, it still can get the same result as the language with larger data size. This may because the Bosnian is very close to Croatian, and the training for Croatian also help the training for Bosnian. The same thing happens on the language pairs that contained Slovak, as it's close to Czech, it also gets a great improvement of 7 points.

5.1.6. Family pair C to D

This part shows the results of languages from family C translate to languages from family D, the scores are in table 5.6, the columns show the languages from family D and the rows show the languages from family C.

	zh_cn		zh_tw	
	shared model	family model	shared model	family model
da	11.34	21.43	9.49	18.5
de	11.6	20.84	9.68	18.97
en	15.99	25.28	15.03	23.98
nl	12.32	20.6	11.25	19.33
sv	11.62	21.41	10.83	19.69

Table 5.6: BLEU scores for family pair CD

This family pair shows the highest improvement of BLEU score which can reach to 10. The average improvement of this pair is about 9. This increase is undoubtedly significant.

As we have talked before, although the data size for some of the language pairs are small, it could also be helped by the language pair that contain its close language. In this family pair, the results for Danish-Chinese and Swedish-Chinese show this rule.

5.2. Comparison of performance of zero-shot

As mentioned before, zero-shot translation is to translate a language pair that have never been trained in the system.

To test the performance of zero-shot of these two models, two other models which are the zero-shot version were trained. To do this, data for one language pair in each family pair was removed and keep the other data the same.

After training, Only the language pairs that haven't been trained were tested, The result was shown in table 5.7.

family pair	language pair	shared	family
AB	ro-ru	14.92	19.14
AC	it-de	18.09	24.25
AD	fr-zh_cn	**12.52**	**19.76**
BC	cs-nl	18.14	23.45
BD	pl-zh_tw	**8.89**	**14.61**
CD	sv-zh_tw	**10.41**	**18.13**

Table 5.7: BLEU scores for zero-shot models

From the table we can see that although these pairs have never appeared in the training data, it can still get a good result as the previous models that contain them.

As we have discussed before, the linguistic family model outperforms the all-language-shared model, and this is also reflected in the zero-shot performance. After calculating the improvement of BLEU scores, the increase follows the pattern of our previous calculations in 5.1. It's obvious that the improvement for the language pairs that contains Chinese have higher values which also shows that to have specific encoder and decoder will help languages such as Chinese get a much better result.

5.3. Comparison of consumption

Usually in practice we do not simply consider the performance as the only decision criteria. The time consumption and memory consumption are also two important part to make the final decision.

In terms of the time consumption, these models were trained for 32 days, the detail of this part is shown in table 5.8.

	epoch number	training days	average time consumption for each epoch
shared model	55	32 days	13.96 hours
family model	41	32 days	18.73 hours

Table 5.8: Time Consumption

The average time consumption for each epoch is very long as the data size is very large. The time consumption of the linguistic family model is approximately 1.34 times as that of all-language-shared model. This is not a big difference if the dataset and the network are not too big. But if we need to get more accurate results a large amount of data is essential. In our project the data size is already a large number, and it seems that this time consumption is acceptable.

As for the memory consumption part, the details are in table 5.9.

	shared model	family model
Memory Consumption	1358.56M	2934.60M

Table 5.9: Memory Consumption

This table shows the size of network that we have trained, the linguistic family model is 2.16 times as the all-language-shared model. This is because the shared model only has one encoder and one decoder which are multiple in the family model. If only consider the performance of translation, to allocate each language one encoder and decoder would be the best model, but that means the multiple encoders and decoders will result in a large consumption of memory, which is not allowed in the machine that we have used for training. Also, the size of model is a factor that could influence the time consumption.

6. Conclusions

This project aims to develop a translation model with linguistic family shared encoder and decoder. To compare with this model, we use the model that only have one encoder and one decoder that shared for all selected languages as the baseline.

The selected languages are divided into four language family according to the distance between them, the family A include Spanish, Catalan, French, Galician, Italian, Portuguese, and Romanian, the family B includes Bosnian, Croatian, Slovenian, Slovak, Polish, Czech and Russian, the family C includes Danish, Dutch, German, English and Swedish, the last family D includes only Chinese, but in traditional and simplified forms.

The all-language-shared model train with the merged data, and the linguistic family model train with the family pair data. The result was shown as BLEU score, which compute the similarity between machine translation and human translation.

In general, the model with language family specific encoder and decoder outperforms that of all-shared model. The improvement of BLEU score is from 3 points to 10 points according to the distance between language families. This may because having all languages in a single system may lead to some noise for each language pair training as languages from different families have different morphologies and alignments. To specify encoder and decoder for each language family could reduce this noise and result in a higher translation quality. Also, comparing the language families that only contains European languages, it is shown that the closer the families the better the translation performance, and there is also a correlation between the average distance between families according to Gamallo's result and the improvement of BLEU score.

It is worth noting that training with close language could help the training for some languages with few data.

For the language pairs that contain Chinese, the improvement is the highest among all language pairs. This means that it is much better to use a specific encoder and decoder for Chinese than shared with other European languages as it doesn't share the dictionary with other languages.

As for the zero-shot performance, both models are capable of zero-shot translation. The result is similar as it has train the data directly, also the linguistic family model outperforms the all-language-shared model in this aspect.

In summary, to share the encoder and decoder for language families is a good choice considering both translation quality and resource consumptions. Some future work could be delineating a more precise language family which the contained languages are closer. Also, we could explore how much languages in one family to train can get a better result.

Bibliography

[1] Allen, R. B. (1987, June). Several studies on natural language and back-propagation. In *Proceedings of the IEEE First International Conference on Neural Networks* (Vol. 2, No. 5, pp. 335-341). IEEE Piscataway, NJ.

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

[3] Papineni, K., Roukos, S., Ward, T., & Zhu, W. J. (2002, July). Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th annual meeting of the Association for Computational Linguistics* (pp. 311-318).

[4] Gamallo, P., Pichel, J. R., & Alegria, I. (2017). From language identification to language distance. *Physica A: Statistical Mechanics and its Applications*, 484, 152-162.

[5] Sennrich, R., Haddow, B., & Birch, A. (2015). Neural machine translation of rare words with subword units. *arXiv preprint arXiv*:1508.07909.

[6] Ott, M., Edunov, S., Baevski, A., Fan, A., Gross, S., Ng, N., ... & Auli, M. (2019). fairseq: A fast, extensible toolkit for sequence modeling. *arXiv preprint arXiv*:1904.01038.

[7] Arivazhagan, N., Bapna, A., Firat, O., Lepikhin, D., Johnson, M., Krikun, M., ... & Wu, Y. (2019). Massively multilingual neural machine translation in the wild: Findings and challenges. *arXiv preprint arXiv*:1907.05019.

[8] Escolano, C., Costa-jussà, M. R., Fonollosa, J. A., & Artetxe, M. (2020). Multilingual machine translation: Closing the gap between shared and language-specific encoder-decoders. *arXiv preprint arXiv*:2004.06575.

[9] Kudugunta, S. R., Bapna, A., Caswell, I., Arivazhagan, N., & Firat, O. (2019). Investigating multilingual NMT representations at scale. *arXiv preprint arXiv*:1909.02197.

[10] Johnson, M., Schuster, M., Le, Q. V., Krikun, M., Wu, Y., Chen, Z., ... & Dean, J. (2017). Google's multilingual neural machine translation system: Enabling zero-shot translation. *Transactions of the Association for Computational Linguistics*, 5, 339-351.

[11] Orhan Firat, Kyunghyun Cho, and Yoshua Bengio. 2016. Multi-way, multilingual neural machine translation with a shared attention mechanism. In *Proceedings of the Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, 866–875.

[12] Itamar Shatz. 2016. Native language influence during second language acquisition: A large-scale learner corpus analysis. In *Proceedings of the Pacific Second Language Research Forum (PacSLRF'16)*. 175–180.

[13] Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. 2014. Sequence to sequence learning with neural networks. In *Proceedings of the 27th International Conference on Neural*

Information Processing Systems - Volume 2 (NIPS'14). MIT Press, Cambridge, MA, USA, 3104–3112.

[14] Bahdanau, D., Cho, K., & Bengio, Y. (2014). Neural machine translation by jointly learning to align and translate. arXiv, [1409.0473].

[15] Reimers, N. and Gurevych, I., "Making Monolingual Sentence Embeddings Multilingual using Knowledge Distillation", *arXiv e-prints*, 2020.

[16] Koehn P, Hoang H, Birch A, Callison-Burch C, Federico M, Bertoldi N et al. Moses: Open Source Toolkit for Statistical Machine Translation. In *Proceedings of the 45th Annual Meeting of the Association for Computational Linguistics Companion Volume Proceedings of the Demo and Poster Sessions*. Prague, Czech Republic: Association for Computational Linguistics. 2007. p. 177-180

Appendices

Parameters to train the model

--arch	interlingua_transformer
--optimizer	adam
--adam-betas	(0.9, 0.98)
--clip-norm	0
--lr-scheduler	inverse_sqrt
--warmup-init-lr	1.00E-07
--warmup-updates	4000
--lr	0.001
--min-lr	1.00E-09
--dropout	0.1
--weight-decay	0
--criterion	label_smoothed_cross_entropy
--label-smoothing	0.1
--max-tokens	1000
--update-freq	24
--save-interval-updates	5000
--task	interlingua_nodistance_translation
--encoder_embed_dim	512
--encoder_ffn_embed_dim	2048
--encoder_attention_heads	8
--encoder_layers	6
--decoder_embed_dim	512
--decoder_ffn_embed_dim	2048
--decoder_attention_heads	8
--decoder_layers	6