Chinese Embedding with Radical Based Byte-Pair Encoding and Pre-trained Vision Transformer

XU YIFAN

Master's Program in Informatics
Degree Programs in Comprehensive Human Sciences
Graduate School of Comprehensive Human Sciences
University of Tsukuba
March 2023

Chinese Embedding with Radical Based Byte-Pair Encoding and Pre-trained Vision Transformer

Name: XU YIFAN

It's necessary to build a dictionary and convert words into corresponding embedding in natural language processing tasks, which leads to problems such as OOV (Out of Vocabulary). Fine-grained level methods, which decomposes OOV words into existing sub-words generated from known words has been widely used.

BPE algorithm reaches excellent performance in the processing of text in multiple languages. However, characters in Chinese have sparsity problem, so that BPE algorithm cannot effectively segment words to build sub-words. As a kind of hieroglyph, Chinese also contains rich information that cannot be ignored in the glyph. For the use of BPE algorithm in Chinese, the author proposes a method to alleviate sparsity problem and combine sub-word embedding with glyph feature vectors to further improve the performance of embedding. Particularly, the author converted Chinese characters into corresponding radicals to alleviate sparsity problem, and proposed a method to extract the glyph feature by vision transformer. Finally, a new model is proposed for combining embedding with glyph features extracted by vision transformer.

In the proposed method, I show how to convert characters into corresponding radicals according to different conversion ratios, and how to use vision transformer for extracting glyph features. In the experiments, I evaluate the performance of the combination of embedding and glyph feature vectors according to different conversion ratios. Results show that converting character at a larger scale can further alleviate the OOV problem, while reducing the performance of sub-word embedding. However, this problem can be addressed by combining embedding with glyph features extracted by vision transformer. After combination, the performance of translation results can be improved. The author concludes that converting characters into corresponding radicals at a certain conversion ratio and combining sub-word embedding with glyph features using vision transformer can improve the performance of embedding on Chinese .

Main Academic Advisor: Yohei SEKI

Secondary Academic Advisor: Kei WAKABAYASHI

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Objective

Recently, because of the rapid development of deep learning, natural language processing has achieved excellent performance in various tasks. To apply various deep learning models to the task of natural language processing, it is necessary to process various text data first. For text processing, the corresponding words in the text should be converted into corresponding embedding. Usually before model training, I build a dictionary for containing the corresponding word embedding. In Table 1.1, I show a dictionary that each word has a unique corresponding embedding. By looking up the dictionary, the text will be converted into processable data.

The capacity of the dictionary is always limited, while the number of words is unlimited. Especially considering the rapid development of the network today, new words are constantly being created, and the single method of expanding the capacity of the dictionary can no longer meet the demand. Instead of expanding the dictionary, the more commonly used method is to use sub-words to generate words. A sub-word is a unit smaller than a word, such as a suffix, prefix, root, etc. of a word. Through this method, when the OOV (Out Of Vocabulary) problem occurs, I can find the corresponding embedding in the dictionary by decomposing the word into sub-words. This approach performs well on tasks targeting English, etc., since the composition of words in English-like languages relies on inflection or inflexion changes, making it easier to segment useful sub-words from words. However When dealing with Chinese, existing algorithms such as BPE (Byte-Pair Encoding) cannot segment efficient sub-words because Chinese words expression do not heavily rely on the inflection or inflexion.

What's more, Chinese characters are also rich in information in their glyph, and the meaning

Table 1.1: Structure of Dictionary

Index	Word	768-dimensional embedding
107	range	$[0.01, -0.34, 0.05, ..., -0.15]$
108	ranger	$[0.06, -0.33, 0.07, ..., -0.01]$
109	rank	$[-0.02, -0.08, 0.1, ..., 0.3]$
.....		
4001	prob	$[-0.19, -0.22, -0.11, ..., 0.4]$
4002	probability	$[-0.3, 0.61, -0.01, ..., 0.08]$

of Chinese characters is directly or indirectly related to its glyph. If the same processing method as English is used, this part of the information will be lost, resulting in the model not being able to achieve the same excellent performance in English.

My goal is to propose a new processing approach for Chinese text to alleviate the problem that BPE cannot segment efficient sub-words when applied to Chinese. At the same time, I propose a method of pre-training the vision transformer, and use the trained model to extract the glyph information of Chinese characters. Finally, the sub-word embedding and character glyph feature vector are combined to generate the embedding for natural language processing task. In this way, I can improve the performance of Chinese word embedding while alleviating the OOV problem.

1.2 Motivation

In order to be able to effectively use the BPE algorithm in Chinese, it's important to preprocess the text so that the words are easier to segment. The BPE algorithm relies on the co-occurrence frequency between characters to generate sub-words (the higher the co-occurrence frequency, the more likely the symbol pairs are to form meaningful sub-words). Improving the

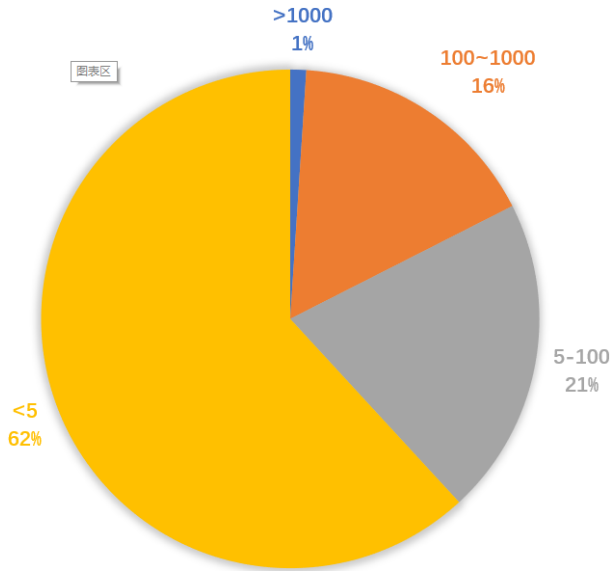

Figure 1.1: Occurrence of characters in a Chinese novel.

performance of the algorithm also means addressing the problem of sparseness in co-occurrence matrix. Since there are about 3,000 commonly used characters in Chinese, and a small number of characters such as "是" and "的" appear much more frequently than other characters, sparsity problem tends to occur. I show in Figure 1.1 the large differences in the occurrence frequency of different characters in a Chinese novel [42]. Therefore I use the radicals of Chinese characters instead of Chinese characters to convert the text to increase addressing the problem of sparseness in matrix. The radicals of Chinese characters are part of Chinese characters, and each Chinese character must have its own unique radical. The total number of radicals is about 300, which compared with Chinese characters, the problem of sparseness can been alleviated. In addition, radicals and various aspects of Chinese characters are often directly or indirectly related. For example, among the characters with "木" as the radical, many are related to trees, such as "林", "森", "植" and so on. Therefore, I think that after replacing the characters with their corresponding radicals, part of the information in the Chinese characters can still be preserved, and the sub-word method based on the co-occurrence frequency can have better performance.

Compared with English, the glyph of Chinese characters contain rich information, which is likely to be discarded or lost when Chinese is processed in the same way as English. Similarly, when using the above-mentioned method of converting characters into corresponding radicals, part of the information of the characters is also lost because a plurality of different characters may correspond to the same radical. The glyph of characters can be used as supplements for missing part of the information because of their uniqueness. Therefore, I propose a method to extract

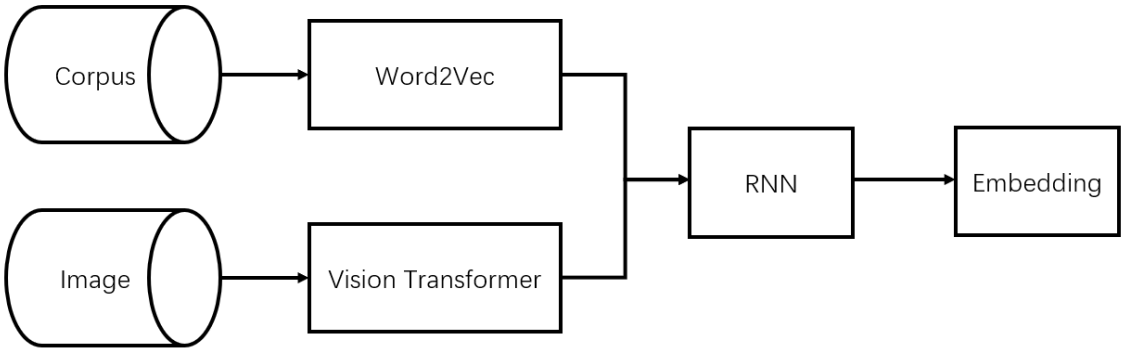

Figure 1.2: The workflow of combination of word embedding and glyph feature

Chinese character glyph features to improve the performance of Chinese word embedding.

Many studies have carried out common sense in the extraction of glyph information of characters. For example, Bi et al. [4] uses CNN as an encoder to extract the features of character images. Tao et al. [34] uses the extracted feature information to replace traditional word embedding and is used in natural language processing tasks. While, with the widespread use of vision transformer [8] in the image field, outperforming CNN in various tasks, there is still a lack of research on applying vision transformer to Chinese character glyph feature extraction. How to introduce well-performed transfer learning methods in various tasks of deep learning into the extraction of Chinese character glyph features is still worthy of new attempts. Therefore, I apply the vision transformer to the feature extraction of Chinese character glyph, and use the transfer learning method to directly apply the vision transformer pre-trained in the Chinese character image reconstruction task to generate the corresponding vectors of the Chinese character images.

My research can be divided into two parts. The first part is the pre-process of Chinese text. In this part, I convert the Chinese characters in the Chinese text into corresponding radicals and perform word segmentation. The words after segmentation will be decomposed into sub-words according to the established dictionary, and the corresponding sub-word embedding will be obtained using Skip-gram model. In another part, the original Chinese character image corresponding to the sub-word will be used as input into the vision transformer model to obtain the glyph features. At last, the outputs of these two parts will be Combined to generate the final word vector. In these two tasks, I address the problem that the effect of BPE algorithm decreases in Chinese processing and the loss of Chinese character information in the process of converting into embedding. I show the entire workflow in Figure 1.2.

1.3 Contributions

The contributions of this paper can be summarized as follows.

1. To effectively apply BPE algorithm on Chinese, a new approach is proposed for processing Chinese text. Chinese characters are converted into radicals to alleviate the sparsity problem in this approach.

2. A method of applying the vision transformer as an autoencoder to the glyph feature extraction of Chinese characters is proposed. The vision transformer pre-trained in the Chinese character image reconstruction task is introduced into the extraction of glyph features using transfer learning.

3. A model for combining sub-word embedding with glyph feature vectors is proposed. In machine translation tasks, the combination of embedding and glyph feature vectors outperforms original embedding.

4. Results show that converting Chinese characters into radicals at a certain ratio can greatly reduce the number of OOV words while improving the performance of sub-word embedding.

1.4 Organization

The rest of this paper is organized as follows. Chapter 2 summarizes some related works, which includes the word embedding models, meaning of OOV words, fine-grained level word segmentation algorithm, and models used for extracting glyph features. Chapter 3 describes the specific methods used in the study, including how I convert Chinese characters into corresponding radicals, the specific details of using the BPE algorithm to generate dictionaries and vision transformer transfer learning, and how to combine the two as embedding. Chapter 4 describes some experimental settings and the results of experiments. In Chapter 5, I provide a detailed analysis

of the results and discuss the shortcomings. Finally, I conclude my remarks and discuss future works in Chapter 6.

Chapter 2

Related Work

In this chapter, I introduce the concepts of word embedding and two kinds of implementation approaches. In particular, I introduced the Word2Vec model in detail. After that, I describe the OOV problem and different approaches for addressing OOV problem. Finally, I introduce the research related to extracting the glyph feature of character, and methods of combining word embedding with glyph feature vector.

2.1 Word Embedding Approaches

Word embedding is the basis of all natural language processing tasks. Unlike data such as sound and images, there is no optimal solution for how to effectively convert text into corresponding word embedding. As a result, there are continuous attempts by researchers in text processing, and with the rise of deep learning, the scope of research has been further broadened.

The easiest way called one-hot embedding is to build a dictionary for looking up, and use a sparse discrete vector to convert words into the form of embedding. Further, researchers consider representing words in a continuous space and introduce the relationship between words into vectors. Word embedding are generally considered to be a set of methods that can be used to map high-dimensional features of words in text into a low-dimensional vector space.This concept was firstly introduced by Hinton [13]. Following this idea, I can represent the vectors of Tsukuba and University in the above example as $[0.12, -0.02, ..., 0.2]^N$ and $[0.2, 0.4, ..., -0.2]^N$, and N is a hyperparameter that determines the dimension of word embedding. To achieve the above goal, many method has been proposed to generate word embedding, and Li et al. [21] classified these methods into two type, one relies on neural network and the other is based on the occurrence of

Table 2.1: Word-document occurrence matrix

Document \ Word	I	is	work	...	NLP
1	15	102	20	...	0
2	20	98	11	...	1
3	3	90	7	...	2
4	13	122	6	...	0

each word.

2.1.1 Co-occurrence Based Method

In order to use the occurrence frequency of word, a word-document matrix is introduced. Each entry in the matrix counts the occurrence of word in documents, The corresponding embedding can be generated from the matrix via method like matrix decomposition. Compared with the one-hot vector, the co-occurrence matrix can reflect the relationship between different words, avoiding the problem of zero similarity that may occur when using the one-hot vector. In Table 2.1, I show a word-document occurrence matrix. Each entry in the matrix represents the number of co-occurrences of two words. In practice, due to the sparsity of the co-occurrence matrix, the method of dimensionality reduction is often used to make the co-occurrence matrix more dense.

Although frequency-based methods have shown promising performance, with the rise of deep learning, higher requirements are placed on word embedding, and the relevance of words and context has been emphasized.

2.1.2 Neural Network Based Approaches

Due to the influence of context on the meaning of words, it is generally inclined to think that two words have similar meanings if they share a similar context. Driven by this idea, Bengio et al. [3] proposed the Neural Networks Language Model (NNLM), in which the probability of a sentence can be calculated by the following equation according the Bayes rule

$$P(S) = \prod_{1}^{N} P(w_t|w_1, w_2, ..., w_{t-1}) = \prod_{1}^{N} P(w_t|h_t)$$

(2.1)

Where $P(S)$ is the joint probability of sentence S, h_t denotes the context of word w_t. Objective is to evaluate the probability of word w_t appears in the context. However, in the actual practice, the number of words is large, which can form a huge number of different combinations, resulting in the problem that all contexts of words cannot be effectively counted. So the n-gram model is usually used to replace all the contexts of words. In the n-gram model, only last n - 1 words before the word are counted at a time. So the conditional probability of word w_t becomes:

$$p(w_t|h_t) = P(w_t|w_{t-n+1}, ..., w_{t-1})$$

(2.2)

Word2Vec

Different neural networks can be used to build NNLM models, including feed forward layer [28], CNN [4], RNN [27], BERT [7], etc. Most of them are unsupervised models, and among these unsupervised models, Word2Vec model is widely used and has achieved excellent performance.

This model is divided into two cases. In [28, 26], Miklov et al. specifically expressed these two learning models. One learning model is the Skip-gram model, in which a word w and the context c are given, and there are some parameters θ to be learned in the model. The conditional probability of the context c determined by the word w is denoted as $p(c|w;\theta)$, and the task of the model is to adjust the parameter θ to maximize the probability of the corpus:

$$argmax_\theta \prod_{(w,c)\in D} p(c|w;\theta)$$

(2.3)

Here, D is all word w and context pairs can be extracted from text. Nowadays Skip-gram model are usually implemented using neural networks, in which soft-max is used to model the conditional probabilities:

$$p(c|w;\theta) = \frac{e^{V_c V_w}}{\sum_{c'\in C} e^{V_{c'} V_w}}$$

(2.4)

Here, C denotes all available contexts, and V_c, $V_w \in R^d$ are vector representations for word w and context c respectively, where d is hyper-parameter representing the dimension. The parameter θ are V_{c_i} and V_{w_i}. For word $w \in V$ and context $c \in C$, $i \in 1, ..., d$, where V is dictionary containing all words. In practice, Skip-gram model can be constructed with two feed forward layer. And in order to reduce the computational complexity, hierarchical soft-max and negative sampling are usually introduced [11]. I show how Skip-gram model is constructed with two feed forward layer in Figure 2.1.

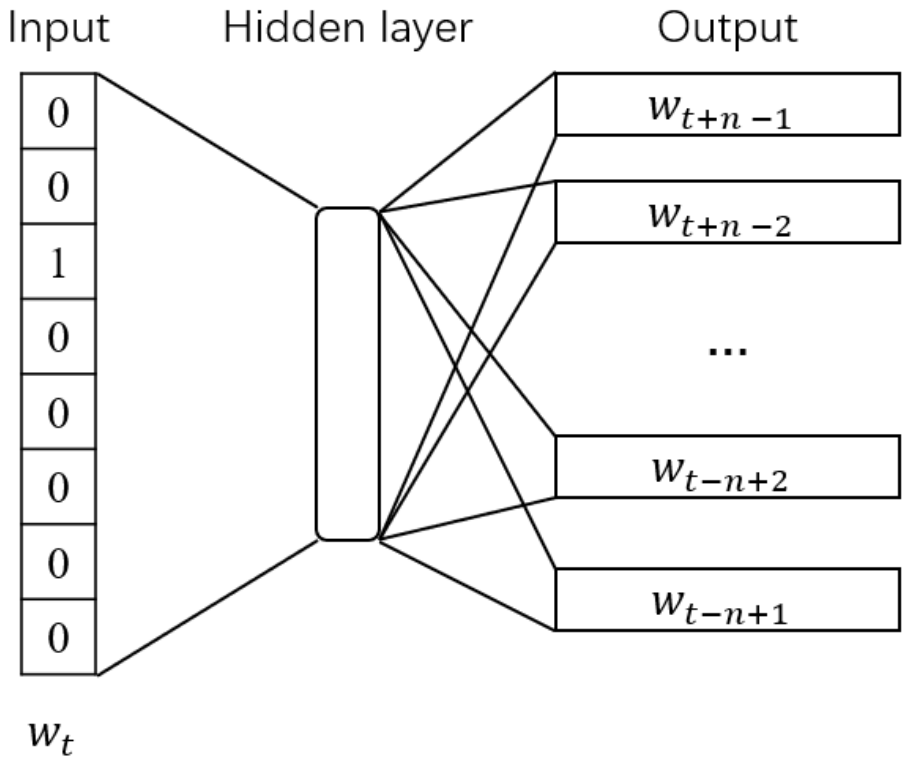

Figure 2.1: Structure of Skip-gram model

The other way to perform Word2Vec is CBOW model. In CBOW model, given the context c and word w, the conditional probability of the word w is denoted as $p(w|c;\theta)$, where θ is parameters to be learned during model training. Like Skip-gram model, the objective of CBOW is to adjust parameter θ to maximize the probability of the corpus:

$$argmax_\theta \prod_{w,c)\in D} p(w|c;\theta)$$

(2.5)

Compared with the CBOW model, the Skip-gram model requires K times training and tuning when each word is used as the central word, and K is the width of the context. This makes Skip-gram perform better when there are many rare words. Due to the sparsity of Chinese, I choose the Skip-gram model to generate word embedding in this study.

BERT

BERT (Bidirectional Encoder Representations from Transformers) [7] is a deep learning model consisting of multi-layer transformer encoders. It is another method that is widely used to generate word embedding. The input of the BERT model is the index number of the word in the dictionary, and the output is the embedding of the word. Compared with the Word2Vec model, the word embedding generated by BERT is no longer fixed, but changes according to the context of the word. That is to say, BERT is a context-based approach. At the same time, BERT also takes into account the relative positional relationship between words in the input. However, while the BERT model is more complex, it has higher requirements on the amount of data used for training and number of training steps, and since the text processing method in this paper is proposed for the first time, there is no suitable pre-trained BERT model available. Therefore, in the current research, BERT has not been applied into training word embedding.

2.2 Out of Vocabulary Words

In all the exiting word embedding approach, it is obvious that a dictionary needs to be established, and the embedding is corresponding to the word through the dictionary, so as to train the model or apply it to other natural language processing tasks. When actually used, there is a limit to the capacity of the dictionary, but there is no limit to the number of words. Words not included in the dictionary will become OOV (Out Of Vocabulary). In [18], OOV is considered as one of the six most important problems in machine translation. To solve the OOV problem, the easiest way is to replace all OOV words with UNK and use them as the input of the model, but this method loses a lot of information and result in poor performance in natural language processing tasks. In recent days, there are three main approaches to cope with OOV problem:

1. Expand the dictionary. For example, Jean et al. [15] proposed a method that does not increase the computational complexity while using a large dictionary. However, even more

words can be included, the training results of word vectors may still perform poorly due to their low frequency. Therefore, the method of expanding the dictionary cannot take into account the performance of word embedding while solving the OOV problem.

2. Either replace OOV words directly to the corresponding position, or use synonyms to replace OOV words. In [14, 23], the machine translation model will find the corresponding position of OOV words in the translated sentence and replace the word with OOV words in the corresponding position after translation. The information of OOV word cannot be learned by the model and has an impact on the overall result. Ngo et al. [29] replaces OOV words with the word that is closest to the word, possibly a synonym or a different form, with the help of an external dictionary. Even though OOV words is replaced with synonyms or root words, there are still differences in words that may lead to poor performance in natural language processing tasks.

3. Fine-grained level. This class of methods no longer uses words, but uses units smaller than words as embedding targets. At the fine-grained level, word will be broken down into parts such as prefixes, suffixes, roots, etc. Each part that is decomposed into a sub-word will get the corresponding embedding through language model training and form a dictionary. OOV words can also be decomposed into each sub-word when used, so as to find the corresponding embedding in the dictionary.

Byte-Pair Encoding Algorithm

As a typical fine-grained level method, Byte-Pair Encoding (BPE) [31] algorithm is widely applied in today's natural language processing tasks. This is based on the intuition that Words can be broken down into smaller units, and combinations of these units can form new words, such as word prefixes, suffixes, roots, and so on. Byte-Pair Encoding was firstly proposed by Gage [10], which was a data compression technique that iteratively selects the most frequent pairs and combine them into new bytes. On top of this idea, Sennrich et al. [31] modified the algorithm so that it can be applied to word segmentation to generate character sequences.

The algorithm firstly sets a symbol dictionary with character dictionary, by treating each word as a sequence of single character, and adds the special symbol '·' at the end, so that sub-

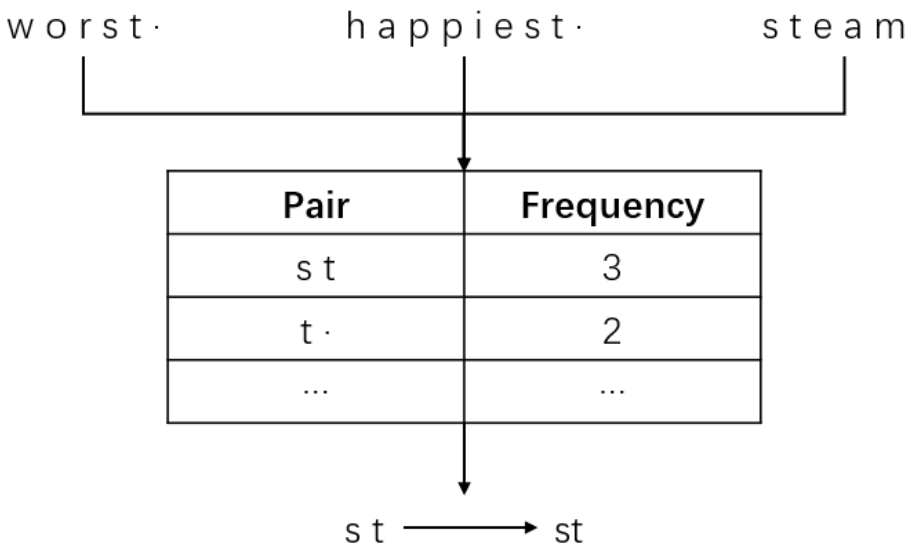

Figure 2.2: Example of BPE algorithm

words can be combined into words later. Then it will count all symbol pairs and replace the most frequent pair 'C', 'N' with a new symbol 'CN' in the dictionary. In this way, after each iteration, a new symbol can be obtained, and finally each word will become a collection of symbols, that is, a dictionary of sub-word. I show a simple English example in Figure 2.2 how a new symbol is generated in one iteration with three words.

Obviously, BPE algorithm relies on the co-occurrence frequency between symbols, which leads to the poor performance of the algorithm in some languages. Li et al. [20] show that character based embedding outperforms sub-word based and word based embedding due to the sparseness of Chinese word distribution in several natural language processing tasks. For this reason, many studies related to Chinese text processing use character embedding, or focus on smaller units such as character radicals and strokes [1, 24, 37, 17].

2.3 Glyph Feature Extraction

The characters of some languages are called hieroglyphs, and unlike other languages, these characters are developed from images and contain information related to meaning in the glyph. Chinese characters are a kind of hieroglyphs. When processing Chinese characters, if they are processed in the same way as English, the information contained in the glyph cannot be used. Therefore, some research has been conducted to extract the information in Chinese character glyph and apply it to the task of natural language processing [6]. Since Chinese characters can be

decomposed into smaller Chinese character structures, some studies segment Chinese characters into sub-character units, and embedding is composed of these units. Ke et al. [16] represented a word embedding using a sequence of radical level embedding instead of character embedding. Zhang and Komachi [39] used CJK character set to decompose Chinese characters into strokes of characters, and words are replaced by sequences of strokes. For example, word '物语' will be decomposed into '牛', '勿', '言', '五', '口'. Cao et al. [5] divided all strokes of Chinese characters into five categories. After decomposing the Chinese characters into these five categories of strokes, this sequence of strokes were input into the LSTM model according to the order to obtain the corresponding embedding. Methods such as these take advantage of the structural information of Chinese characters to process characters as text sequences. Although the information of glyph is incorporated into the embedding, the relative position information inside the structure is not taken into account, and it also depends on the constraints of some external knowledge.

Another class of methods takes Chinese characters as images and uses neural networks to extract the glyph features of Chinese characters from the images. In this case, deep learning models, which had previously performed well in the image domain, can be applied to this task.

2.3.1 CNN

CNN, which has performed well in many tasks of computer vision, is currently the most frequently used model in character glyph feature extraction. Liu et al. [22], Zhang and Lecunn [40] used CNN to extract glyph features and serve as character embedding. Meng et al. [25] designed a special CNN model for feature extraction and used image classification as an auxiliary training objective to solve the over-fitting problem. Sun et al. [33] used CNN to extract the features of images under three different fonts of the same character, and concatenated the three as the features of the glyph.

2.3.2 Vision Transformer (ViT)

ViT (vision transformer) [8] applies the transformer model [36] widely used in natural language processing tasks to the computer vision field [9, 41]. Since the transformer is suitable for sequence data, the image as input needs to be processed first. The image $x \in \mathbb{R}^{H\times W\times C}$ is reshaped into sequence of patches $x_{patch} \in \mathbb{R}^{N\times(P^2C)}$, where H, W represent the height and width of the image, C is the number of channels, P is the size of patch, and $N = (HW)/P^2$ is the length of the

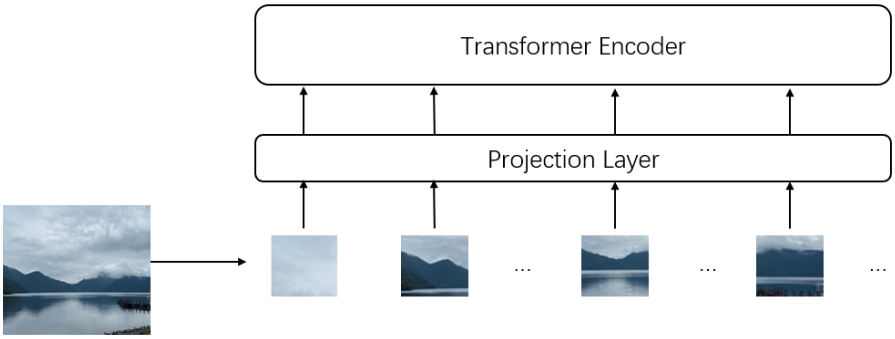

Figure 2.3: Structure of ViT

sequence consisting of patch. Then a linear projection layer is used to map the dimension of each patch to desire dimension. Finally the input sequence for the transformer encoder is obtrained. The structure of ViT is shown in Figure 2.3.

Transformer

After the patch passes through the linear layer, it will be input into the transformer model [36], and the transformer is core part of the vision transformer. It is a seq2seq (sequence to sequence) model consisting of a decoder and an encoder. The core components of the encoder and decoder are consisted of multi-layer multi-head attention models. In multi-head, each head projects the vectors into a unique vector space and concatenates them together. In each vector space, the attention mechanism is considered to learn the characteristics of different aspects of the data. Compared with RNN or LSTM, transformer is more efficient in parallel and avoids the problems caused by long text. As a result, in many tasks of natural language processing, transformers have outperformed other models. Shaheen et al. [32] compare differenet transformer-based model for Large Scale Legal Text Classification, and present new state-of-the-art results. Khandelwal et al. [38] use a single pre-trained transformer to achieve good result in abstract summarization task. The structure of transformer encoder in transformer model is the same as the transformer encoder in ViT for processing the output of projection layer. I show the structure of multi-head attention in Figure 2.4 and transformer in Figure 2.5.

Autoencoder

In order to use ViT to extract the information in the Chinese character image, and use the extracted information in combination with word embedding, autoencoder is used as an approach

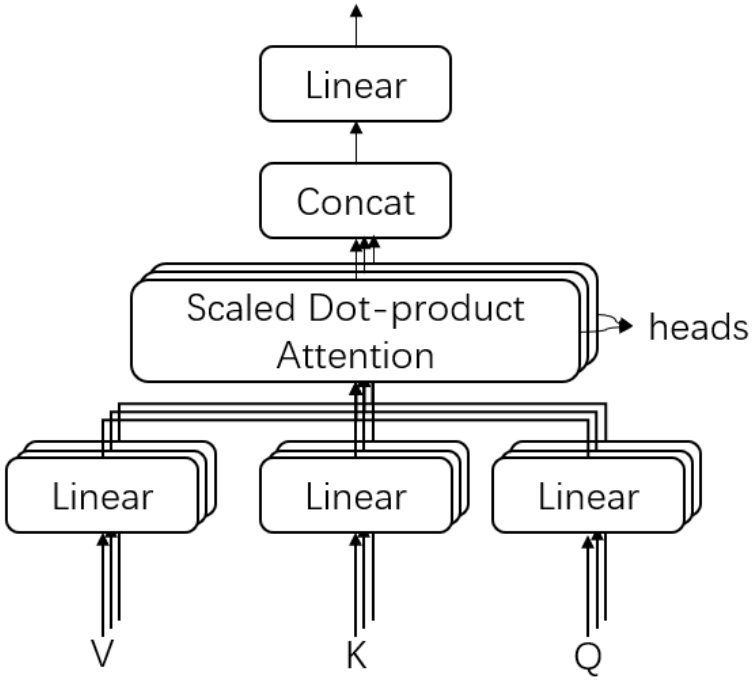

Figure 2.4: Structure of multi-head attention

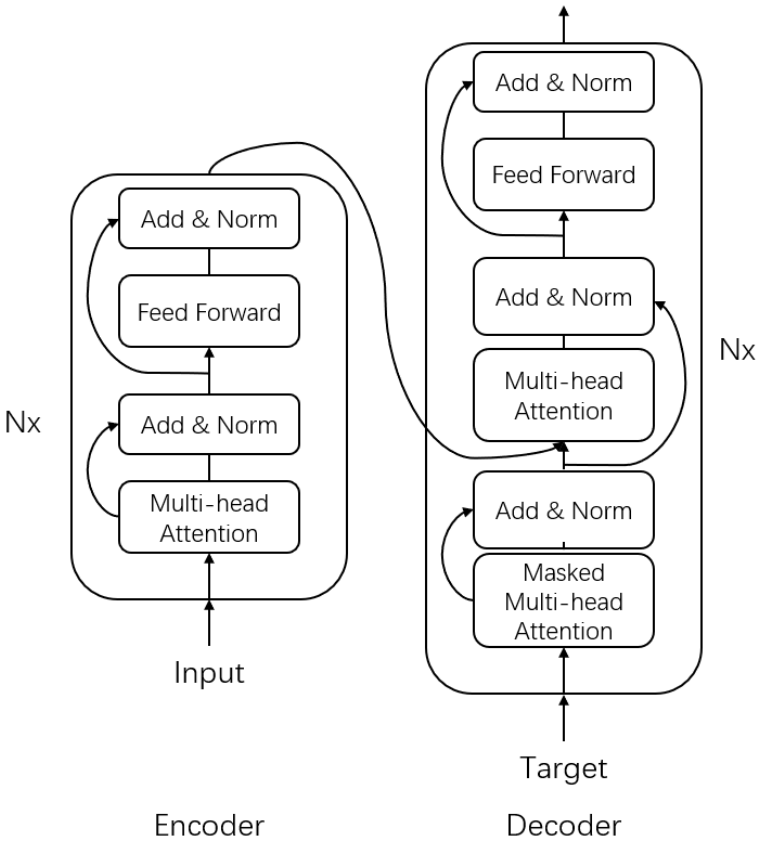

Figure 2.5: Structure of transformer

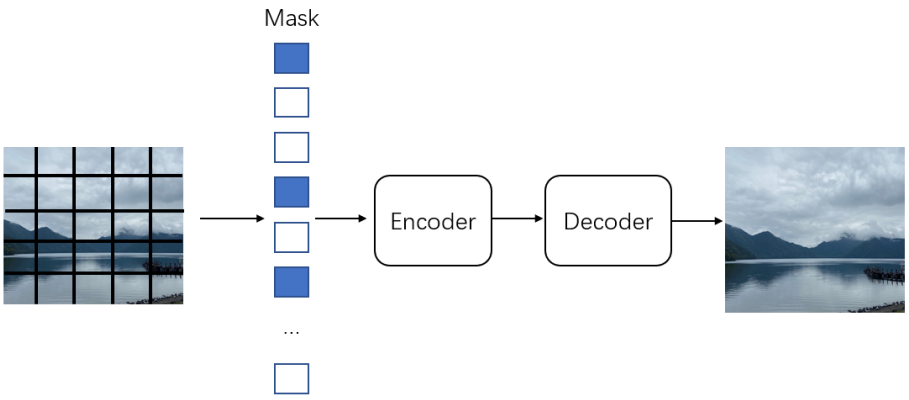

Figure 2.6: Structure of masked autoencoder

for pre-training and transfer learning. Autoencoder is a kind of structure for unsupervised representation learning. Its goal is to learn how to map high-level representations into low dimensional vector spaces, so that the model can reconstruct the original high dimensional representations approximately by low dimensional features. Typically, an autoencoder will contain an encoder and a decoder, which takes images as input into the encoder, and obtains as similar output as possible from the decoder, thereby learning the features of the input data.

Recently, He et al. [12] introduced the vision transformer into the autoencoder model as the encoder, and used a transformer as the decoder. They pre-trained autoencoder in an image reconstruction task by randomly masking partial patches in the encoder input.

In experiments, they fine-tuned the pre-trained autoencoder and apply it to the task of image classification. Compared to self-supervised pre-trained ViT [2], it has achieved better results. I show in Figure 2.6, the structure of masked autoencoder. This model takes an image as input, and randomly mask patches formed by segmenting the image. Then, through the encoder and decoder, a sequence of patches can be obtained to reconstruct the input image.

2.4 Combination of Word Embedding and Glyph Feature Vector

The characteristics of the glyph and the word vector obtained through text retain different information of Chinese respectively, and only the combination of the two can maximize the performance of the word embedding. Therefore, some researches focus on how to combine the extracted glyph features with word embedding. Sun et al. [33] obtain three glyph features from three different font images of same Chinese characters, combining the three features into one through a glyph layer, and then used a fusion layer to embed the pronunciation embedding, glyph

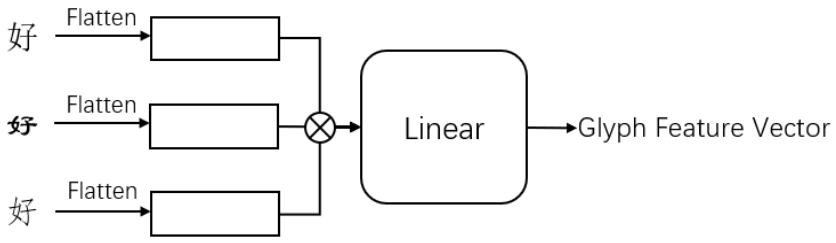

Figure 2.7: Structure of glyph layer

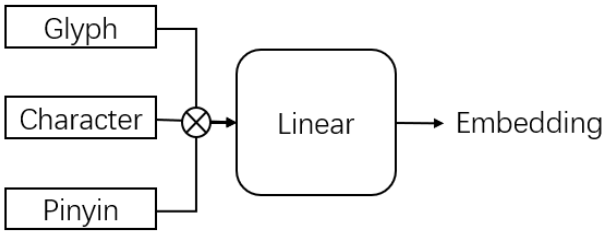

Figure 2.8: Structure of fusion layer

embedding, and character embedding into the final embedding. I show the structure of such a network in Figure 2.7 and Figure 2.8. Here, "Pinyin" denotes the pronunciation of character.

However in this method, Chinese needs to be processed at the character level, which is lack of word level information. Tao et al. [34] introduced glyph features at the level of word embedding. Each word will be split into Chinese characters. The features of Chinese characters are composed of the features of character images extracted by CNN and the features of strokes. The final word embedding is composed of the embedding of individual characters. However, the word embedding obtained in this way cannot learn context-related information in the text.

Chapter 3

Proposed Method

Here I will describe how to convert Chinese characters to radicals, how to pre-train the vision transformer, and how to combine the two to conduct evaluation.

3.1 Conversion of characters to radicals

Radical is an important part of a Chinese character, it may be the first stroke of a Chinese character, or a part of the structure of a Chinese character. Based on radicals, I can divide Chinese characters into different categories, and Chinese characters with the same radical often share some important information. I build a dictionary to find the corresponding radical of each Chinese character. Each radical in the dictionary corresponds to a list, and the list contains all the Chinese characters with the corresponding radical.

To build the dictionary, I first crawled all Chinese characters and their corresponding radicals from an online website Baidu Hanyu [1]. In this website, I can search for Kanji to obtain their corresponding radicals, strokes, and character images. However, due to the uncertainty of the website search results, Chinese characters may correspond to more than one radical. In this case, I take the first possible radical as the radical. On the other hand, when Chinese characters do not contain matching radicals in the website, or the query has no results, I use the first stroke of the Chinese characters as the radical. Finally, I obtain a dictionary with 289 different radicals corresponding to 2,606 commonly used Chinese characters in total. I show the structure of dictionary in Table 3.1.

After building the radical-character dictionary, I obtained all the texts of Chinese terms from

[1] https://hanyu.baidu.com/

Table 3.1: Structure of radical-character dictionary

Radical	Character
一	一 来 丁 云 不 与 甚 求 才 ...
亻	侯 他 传 何 但 们 候 似 作 ...
亠	亮 京 市 就 交 亦 享 六 主 ...
讠	误 调 计 识 议 该 记 请 让 ...
彳	得 往 行 德 彼 街 很 彻 御 ...
心	急 心 想 愿 总 忽 愈 憨 态 ...
足	跳 踪 跟 跑 路 踏 蹲 足 蹈 ...
王	班 珍 珠 现 玩 环 理 弄 球 ...
阝	限 险 那 院 陈 都 陷 阴 隐 ...
.....	

Wikipedia [2] including $1,410,142$ sentences, for generating sub-words, and training the embedding.

I first tokenize all the text to get a dictionary. Then, I calculate the occurrence frequency of all Chinese characters in the dictionary and arrange them in descending order. According to this order, I sequentially convert 100%, 60%, 30%, and 0% of the Chinese characters in the dictionary into the corresponding radicals. Throughout this paper, I define conversion ratio “0%" as converting all Chinese characters into its corresponding radicals, and “100%” as keeping each character unchanged. In the converted dictionary, I use the BPE algorithm to generate sub-words. In this way, I get a total of four different text dataset and sub-word dictionaries. Based on this, I use the Skip-gram model to train embedding belonging to individual dictionaries. When training the sub-word embedding, I set the output dimension of the Skip-gram model at 768, i.e., all trained

[2] htttps://www.wikipedia.org

sub-word embeddings have a dimension of 768. In Table 3.2 I show what a sentence looks like when its characters are converted according to different conversion ratios. In Table 3.3, I give the comparison of the sub-word in the dictionary and the original Chinese character according to different conversion ratios.

Generally, when encountering a new Chinese text, I firstly convert characters in text into radicals according to different conversion ratios, and then segment the whole text using constructed sub-word dictionary. For example, taking Chinese text "这个目标必须是最小的稳定性, 而不是胜利. 这似乎还是可以实现的." as input, and set conversion ratio as 100%, I can get "辶人目木丶彡日曰小白禾宀忄, 而一日月禾. 辶亻丿辶日口人宀王白." by converting characters into radicals. Finally, I segment it using sub-word dictionary, resulting in the sequence of sub-words: "'辶人', '目木丶', '彡日', '曰', '小白禾', '宀忄', ',', '而', '一日月禾', '.', '辶亻', '丿辶日', '口人', '宀王白', '.'"

3.2 Glyph Feature Extraction

There are two steps in my approach for extracting Chinese character glyph feature. In the first step, I pre-train a vision transformer as an encoder in autoencoder model through a Chinese character image reconstruction task. In the second step, I use the pre-trained transformer to extract the glyph features of Chinese characters.

3.2.1 Pre-training

I conduct pre-training in the task of Chinese character image reconstruction. The model used is autoencoder, where the encoder is composed of a vision transformer and the decoder is constructed by a transformer. During training, the character image will be firstly divided into patches, and be randomly masked at 70% according to [2] before input to the encoder. The objective of training is to make the error between each output of the decoder and the original patch as small as possible.

Table 3.2: Example of converted sentence

Conversion ratio	Converted sentence
0%	地球村不会缺少白痴。 那么德国会重归国家主义么? 在改革国有企业和自由化银行时，新领导人需要注意实体经济和扩张的金融部门之间的联系。
30%	土球村一人缺小白痴。 阝丿 德国人丿 彐国家亠丶丿？ 土改革国月企业禾自丨亻银彳日，斤领巳人需要氵音亠亻纟氵禾扌弓白金融阝门丶门白联系。
60%	土球木一人缶小白痴。 阝丿 德口人丿 彐口家亠丶丿？ 土改革口月人业禾自丨亻钅彳日，斤领巳人雨西氵音亠亻纟氵禾扌弓白金鬲阝门丶门白联糸。
100%	土王木一人缶小白疒。 阝丿 彳口人丿 彐口亠亠丶丿？ 土已革口月人业禾自丨亻钅彳日，斤页巳人雨西氵音亠亻纟氵禾扌弓白金鬲阝门丶门白耳糸。

Table 3.3: Example of sub-word

Conversion ratio	Original word	Sub-word
30%	有时候	月日亻
	教师	教师
	工会	工人
	灾难	宀隹
	他们自己	亻亻自己
60%	严重	一丿
	人类	人米
	的系	白系
	生产率	生产亠
	受到	又至
100%	以色	人色
	列人	刂人
	和	禾
	巴勒斯坦人	巳革斤土人
	百年来	白丿一

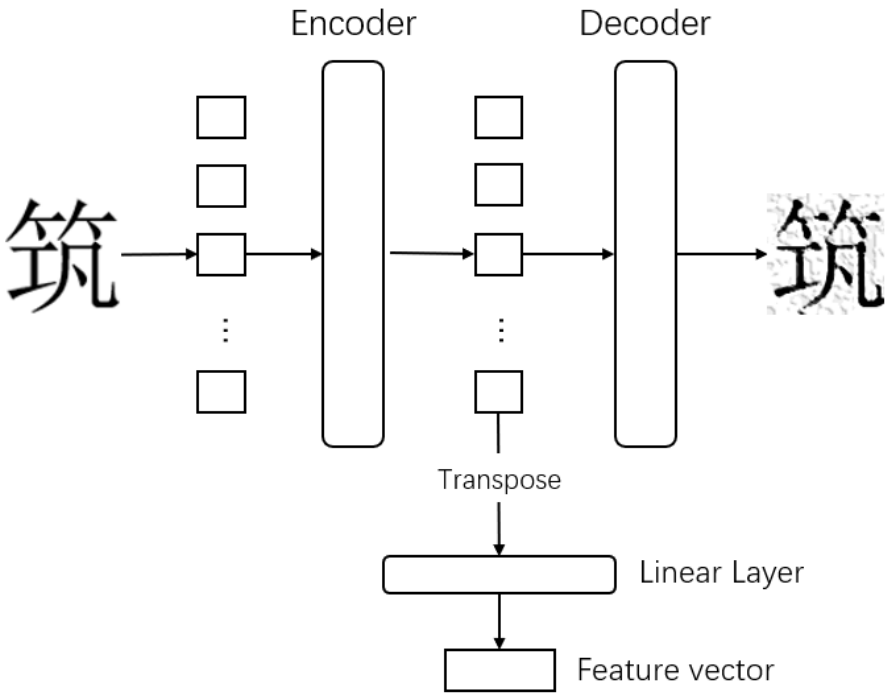

Figure 3.1: Structure of glyph feature extraction

3.2.2 Transfer Learning

After the pre-training is completed, the encoder in the autoencoder, i.e. the vision transformer, will be used to extract the features of Chinese character glyph. The output of the vision transformer is $x_{output} \in \mathbb{R}^{L \times D}$, where L is the number of patches and D is the dimension which is same as embedding dimension. In order to facilitate the combination with sub-word embedding later, I first transpose the output such that the dimension of x_{output} becomoes $D \times L$, and then map it to $D \times 1$ through a linear layer. After transposition, the size of the glyph feature vector extracted from each image becomes $x_{glyph} \in \mathbb{R}^{1 \times D}$. I show this process in Figure 3.1. The upper part of Figure 3.1 shows the process of pre-training ViT in the autoencoder model, where the Chinese character image is used as input, and after encoder and decoder, the reconstructed image is obtained.

3.3 Combination of Sub-word Embedding and Glyph Feature Vectors

Different from the research in [33], my embedding is not based on units of characters, but units of sub-words. For a radical based sub-word, I can get the sub-word constructed by original Chinese characters and extract each character's glyph feature in the sub-word. For example, sub-word "刂 人" is converted from "列人", and glyph features of characters "列" and "人" are extracted by vision transformer while the embedding of " 刂 人" is obtained from sub-word dictionary. Compared to original character, radical only contains part of character, i.e., part of character's information, while original character image used for glyph feature extraction keeps full character. Like this, a sub-word may include more than one character and correspond to plural glyph feature vectors, so directly using a linear layer for gathering glyph feature vectors is not effective, nor can it reflect the order of characters in the sub-word. To effectively handle this characteristic, I propose a new model for combining sub-word embedding and glyph feature vectors.

Proposed Model

After using the vision transformer to extract the glyph features for each character in the sub-word, I obtain a sequence of glyph feature vectors arranged in the order in the sub-word, where each vector's dimension is $x_{glyph} \in \mathbb{R}^{1\times D}$. Here, D is the dimension which is the same as the dimension of sub-word embedding. Then I concatenate these features in the first dimension, which generates $x_{glyphall}$ with the dimension of $L \times D$, here L is the length of sub-word. Then, I concatenate the sub-word embedding glyph feature vector sharing the same embedding dimension to form x_{all}, and the dimension of x_{all} is $(L+1) \times D$. This vector will be used as input into a RNN model, which can reflect the order of input, and the output of the RNN at last step. x_{output} with the dimension of $1 \times D$ will be used as the final embedding for sub-word.

I show the structure of proposed model in Figure 3.2. When encountering a sub- word, on the one hand, I obtain the embedding of sub-word from dictionary trained by Word2Vec model. On the other hand, glyph features of original characters corresponding to the radicals in the sub-word are extracted and concatenated together. For example, for a radical based sub-word "刂 人", its original characters are "列" and "人", and each character's glyph feature is extracted according to the order before concatenation. Finally, sub-word embedding and glyph feature vectors are

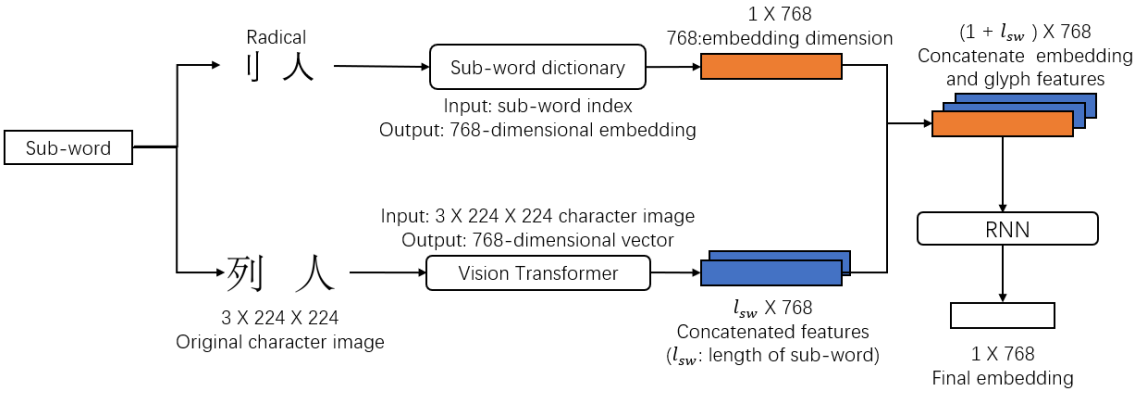

Figure 3.2: Structure of combination

concatenated together as input of RNN model. The output of RNN model is obtained as sub-word embedding.

Chapter 4

Experiments

The purpose of my experiment is to demonstrate whether my method is effective in handling OOV problem and improving performance of Chinese sub-word embedding Firstly, I describe the corpus and dataset used in my experiment. Then, I specifically introduced the model, various parameters and the metrics used in the evaluation. Finally, I present the results of the experiments, including size of corpus, the number OOV words and results of machine translation tasks.

4.1 Dataset

Here I describe the two dataset I used, including the details of the dataset and how they were divided into training, validation and test set.

4.1.1 Dataset for Glyph Feature Extraction

To conduct pre-training, I first construct a dataset consisting of Chinese character images. I use the Pygame library in Python to generate images of all Chinese characters according to the Unicode standard ranging from 4E00 to 9FA5. All Chinese character images are white background, black font color. When Chinese character images cannot be generated correctly, I use pure white image as an alternative. In addition, in order to improve the performance of the model and the ability to extract glyph features, I use two different Chinese fonts to generate images, namely "隶书" and "宋体". In the end, I obtained 41k Chinese character images, 90% of which will be used as training set, 5% as verification sets, and 5% as test set. I show an example of two

Figure 4.1: Example of font image

different fonts in Figure 4.1.

The transformer in the vision transformer consists of a total of 12 layers of multi-head attention model, each containing 12 heads. Transformer in decoder consists of 12 layers of multi-head attention model, each containing 8 heads. Before pre-training, The image will be divided into a sequence of patches, which will be used as the input of the vision transformer. The input of vision transformer is a Chinese character image of size 3 X 224 X 224, which is transformed into a sequence consisting of 768-dimensional vectors after segmentation and projection by linear layer. The output of vision transformer is a sequence of 768-dimensional vectors.

4.1.2 Dataset for Machine Translation

Chinese-English Dataset [35] [1] is ued in my machine translation task. There are 3.6 million sentence pairs in my dataset, 80% of which will be used as training set, 10% as verification sets, and 10% as test set.

4.2 Evaluation

I evaluate my methods on two aspects respectively. On the one hand, I apply the trained sub-word dictionary to tokenize new Chinese texts and obtain embedding of each sub-word. The Chinese characters in the text will be converted using the same conversion ratio as when generating the dictionary, that is, the Chinese characters that were converted to radicals during training embedding will also be converted to radicals at this time. Through this method, I compare the number of OOV words (words cannot be effectively decomposed into combinations of sub-words in the dictionary). After converting Chinese characters into radicals according to different conversion ratios. I report variations in the size of dictionary and the number of OOV words according to four conversion ratios.

On the other hand, I evaluate the performance of my embedding by a transformer model in

[1] https://opus.nlpl.eu/News-Commentary-v14.php

Chinese-English machine translation task. Including the comparison of embedding performance according to different conversion ratio. The embedding dimension is 768 in my experiment. In addition, I evaluate the performance of autoencoder in Chinese character image reconstruction task, and show the reconstructed image.

Internal evaluation tasks like analogical reasoning are not included in this paper, because of the lack of available datasets. Since words have specific and clear meanings and can be related with each other, these tasks are usually conducted at the word level. For example, Li et al. [19] proposed a dataset for internal evaluation tasks that Chinese words can be applied to. However, sub-words do not necessarily have specific meanings before forming words, so it is more difficult to construct effective dataset and conduct this kind of task.

To be able to evaluate my sub-word embedding on internal evaluation tasks, I need to use sub-word embeddings to form word-level embedding. There are many studies focusing on the method of using sub-word embeddings to form word embedding, such as using the average of the sum of sub-word embeddings. However, different composition methods will have a great impact on the final embedding performance, and no uniform and effective method has been proposed so far. Therefore, I did not use these techniques to conduct internal evaluation tasks at the word level, but will keep this topic as a future research task.

4.2.1 Characters Conversion

On the basis of Wikipedia dataset, I count the number of tokens in the corpus after characters being converted into Chinese radicals according to different conversion ratios. Afterwards, based on the corpus obtained by BPE algorithm, I segment the Chinese text of news commentary and count the number of OOV words. I show the result in Table 4.1.

4.2.2 Character Image Reconstruction

After pre-training the autoencoder, I apply the model to the test set of Chinese character images to evaluate the performance of the autoencoder in the Chinese character image reconstruction task and whether the vision-transformer can extract the features of the Chinese character image when it is used as an encoder. In order to better demonstrate the performance, I give a comparison between the image reconstructed by the model trained for 1,200 epochs and the original image. The result is shown in Figure 4.2.

Table 4.1: Corpus size for Wikipedia dataset and the number of OOV words in news commentary dataset.

Conversion ratio	Tokens	OOV
0%	341,779	24,260
30%	333,985	3,999
60%	333,792	3,578
100%	333,787	3,571

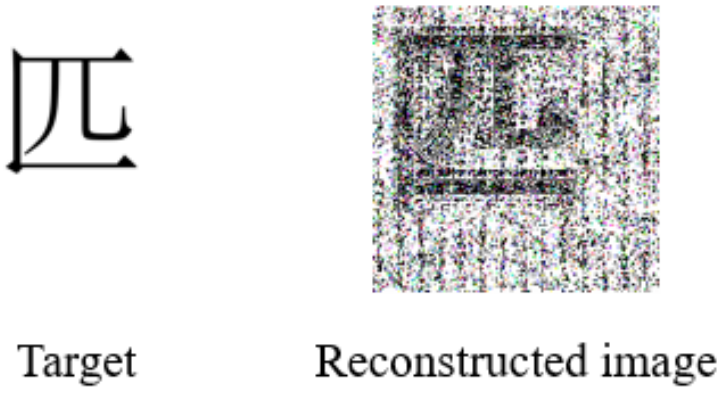

Figure 4.2: The right side is the image reconstructed from the target by the model after 1,200 epoch training.

Also in Figure 4.3, I show a failed example where the reconstructed image is not sharp enough at 800 epochs. Compared with 1,200 epochs, the result at 800 epochs is much more difficult to recognize. Although the boundary between the character and background can be recognized, the strokes in the character is not clear enough. Especially when the distribution of radicals is dense, as in the upper part of the Chinese character in Figure 4.3, the extraction of glyph features becomes more difficult and requires the model to spend more time on learning..

4.2.3 Machine Translation

The performance of the combined embedding will be evaluated in the Chinese-to-English machine translation task, using the transformer model. I take Chinese text as input by converting it into the sequence of sub-word embedding combined with glyph feature vector, and the output of my model is sequence of English sub-word embedding for building translated sentence. In the transformer model, the input and output dimensions are 768, both encoder and decoder contains

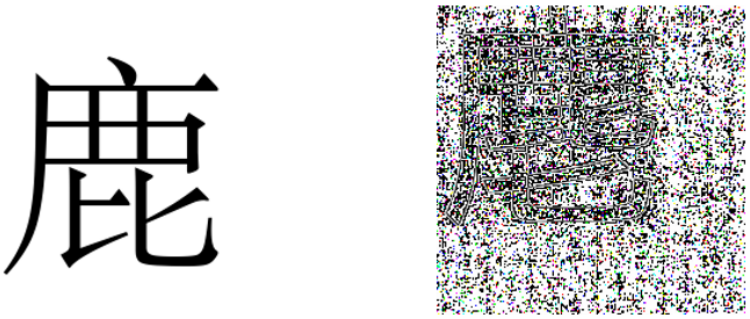

Figure 4.3: The right side is the image reconstructed from the left side image by the model after 800 epoch training.

6 layers of multi-head attention, each layer of multi-head contains 8 heads.

The Chinese text converted according to different ratio will be segmented based on the dictionary obtained by the BPE algorithm, and the result after word segmentation will be converted into an embedding obtained by training the Skip-gram model. Then corresponding Chinese characters extracted by the vision transformer features are combined as input.

For example, Chinese text “当然, 现在的情况和1989 年的情况明显不同了.” becomes sequence of sub-words “‘小灬’, ‘,’, ‘王土白亻’, ‘讠禾’, ‘1989’, ‘丿 ’, ‘白亻’, ‘讠日日’, ‘一冂了’, ‘.’” after conversion and segmentation at 100% conversion ratio. Each sub-word in the sequence is converted into corresponding embedding from the dictionary and combined with glyph feature vectors of characters in the sub-word to build final sub-word embedding. Finally, sequence of sub-word embeddings becomes the input of transformer model to conduct machine translation task, and the output of transformer is a sequence of English sub-word embeddings.

After getting the output of transformer model, i.e., a sequence of English sub-word embeddings, I first use Softmax algorithm to find the corresponding index of each sub-word embedding in an English sub-word dictionary constructed by a pre-trained tokenizer from HuggingFace website[2]. Then, each embedding is transformed to sub-word according to its index to form a sequence of English sub-words. Finally, sub-words are combined to form English text. In the example shown in the previous paragraph, the output of transformer model is a sequence of English sub-word embedding. After Softmax operation and converting index into sub-words, I get “‘Of’, ‘course’, ‘,’, ‘th’, ‘ere’, ‘are’, ‘ob’, ‘vious’, ‘diff’, ‘er’, ‘ences’, ‘be’, ‘tween’, ‘1989’, ‘and’, ‘now’, ‘.’” formed from English text “Of course, there are obvious differences between 1989 and now.”. In the training process, I take minimizing the cross entropy between the generated English

[2]https://huggingface.co/

index sequence and the real reference index sequence as the training objective.

I trained 20 epochs for each case and compared them by the following two metrics.

Metrics

Unigram F1 and BLEU [30] are applied to compare my machine translated results.

1. Unigram F1: F1 is the harmonic mean of Precision and recall. It represents whether each word is translated correctly in the translation result. I show the computation of unigram precision and recall in Equations 4.1, 4.2.

$$Precision = \frac{w_{match}}{w_{translation}} \tag{4.1}$$

$$Recall = \frac{w_{match}}{w_{reference}} \tag{4.2}$$

Here w_{match} is the number of words in the translation that match the reference, $w_{translation}$ and $w_{reference}$ are the number of words in the translation and words in the reference.

2. BLEU: BLEU is widely used in machine translation tasks. Compared with unigram F1, BLEU evaluates the translation accuracy of various n-grams. I show the computation of BLEU in Equations 4.3, 4.4, 4.5.

$$P_n = \frac{\sum_i^E \sum_k^K min(h_k(c_i), min_{j \in M} h_k(S_{i,j}))}{\sum_i^E \sum_k^K min(h_k(c_i))} \tag{4.3}$$

$$BP = \begin{cases} 1 & l_c > l_s \\ e^{1-\frac{l_s}{l_c}} & l_c <= l_s \end{cases} \tag{4.4}$$

$$BLEU = BP \times exp(\sum_i^n W_n log(P_n))$$

(4.5)

P_n is used to calculate the n-gram score, where S_j represents the reference translation, and j belongs to M, indicating that there are M reference in total. C_i represents the translation result, and i belongs to E, indicating that there are E translation results in total. k represents the k-th phrase of the sentence in n-gram, and there are K phrases in total. $h_k(c_i)$ represents the number of times the k-th phrase appears in c_i. $h_k(S_{i,j})$ represents the number of times the k-th phrase appears in the standard answer $S_{i,j}$. BP is used to avoid the case where the sentence is too short and the result is high. l_c indicates the length of the translation result. In the final calculation, BLEU adopts uniform weighting, W_n is $1/N$, corresponding to n-gram, in actual application, the upper limit of N is 4.

Here, I show the results of the model on the test set for four different methods in Table 4.2.

Table 4.2: Chinese-English translation performance (BLEU, Unigram F1)

Conversion ratio	BLEU	Unigram F1
0%	29.79	66.45
30%	**30.14**	**66.92**
60%	29.23	66.37
100%	29.19	66.22

Chapter 5

Discussion

In this chapter, I will first analyze the results of the above experiments, including converting Chinese characters into radicals to handle the OOV problem. Then I analyze the performance of vision transformer as a model in extracting Chinese character glyph features, and performance of sub-word embedding in machine translation task. In the following ablation study, I compared the results of using the vision transformer to extract the glyph feature with not using the vision transformer model. Finally, I will explain the shortcomings that still exist in this research.

5.1 Analysis

5.1.1 Variations in Corpus Size and Number of OOV Words

I converted the Chinese characters in the dataset into radicals according to different conversion ratios, and finally generate a corresponding fixed-size dictionary through the BPE algorithm. Due to the different conversion ratios, the size of the final generated dictionaries are also different. In addition, in practice, when I generate sub-words, I ignore some sub-word units with low frequency (≤ 5), because there may be some special characters in the character set, which form noise.

In Table 4.1, I can clearly find that when the ratio of Chinese characters converted into radicals is continuously increasing, the size of the dictionary is continuously decreasing. After all the characters are converted into radicals, the size of the dictionary reaches the minimum value. Therefore, it can be considered that after the same number of merge operations, the higher the ratio of Chinese characters converted into radicals, the more frequent the found combinations appear in the dictionary, that is, more words in the dictionary can be decomposed into different

Table 5.1: Number of characters according to occurrence frequency under different conversion ratio.

Occurrence Frequency \ Conversion ratio	0%	30%	60%	100%
>20000	3194	1985	1560	1360
10000 - 20000	631	622	515	439
5000 - 10000	686	674	673	547
0 - 5000	18318	14283	13976	13375

sub-word combinations. Moreover, between 30% and 0%, the size of the dictionary changes most drastically. It can be observed that only a small part of Chinese characters needs to be converted to have a great impact on the performance of BPE. In addition I compare the number of OOV words encountered when processing the Chinese data of news commentary dataset according to different conversion ratios. OOV words here include only words that cannot be concatenated from sub-words in the dictionary. When the proportion of Chinese characters converted into radicals gradually increases, the number of OOV words gradually decreases. Among them, the change between 30% and 0% is the most obvious. When 30% of Chinese characters are converted into radicals, OOV problem can be greatly alleviated.

In Table 5.1, I compare the occurrence frequency of characters in the Wikipedia dataset after conversion according to different ratios. I divide the characters into four different regions according to the occurrence frequency, calculate the total number of corresponding characters and compare them.

In the Table 5.1, the characters with high and low occurrence frequency occupy most part of all characters. However, compared with the result without converting to radicals, after conversion, this problem is alleviated, and the distribution of characters becomes more even. Especially after all the characters are converted into radicals, the occurrence frequency of characters is closer to the middle area, which means that the number of low-frequency characters is reduced. In addition, it can be observed that after 30% are converted into radicals, the distribution of characters has greatly changed compared to 0%.

Figure 5.1: Original image and reconstructed image at different steps

5.1.2 Image Reconstruction

When evaluating the task of image reconstruction, I mainly rely on directly comparing the reconstructed image with the original image. In Figure 4.2 and Figure 4.3, I show a pair of pictures, which are the original picture and the reconstructed picture. It can be observed that after 1,200 epoch training, the autoencoder has been able to clearly construct the image of Chinese characters, and show its Structure. From this, it can be considered that the vision transformer has successfully learned the glyph feature of Chinese characters as an encoder.

In order to more clearly demonstrate the model's learning of Chinese glyph feature during the entire training process, I show different images reconstructed from the same image as the target under different epochs in Figure 5.1.

In this series of reconstructed images, I can see that the character image is constantly becoming clearer. At the beginning, the model focuses on learning the same background for all character images, and then focuses on learning the position of the character in the image, as well as the glyph feature of the character. After 1,200 epochs training, the model has learned the glyph features of the characters. The vision transformer as the encoder can successfully extract the feature of the Chinese character image, and as a result can extract the glyph feature of the Chinese character contained in the image.

5.1.3 Machine Translation

Unigram F1 is usually used to describe the accuracy of the translation of each word in the translation results, which to some extent reflects the performance of the word embedding. Especially in the case of OOV, when OOV word is composed of different sub-word units, unigram F1 will reflect this effect. Table 4.2 shows the performance of embedding combined with glyph feature vectors in Chinese-English translation according to different conversion ratio. Among them, 4 different ratios have achieved good results on unigram F1. It can be considered that after converting into radicals, combining embedding with glyph feature vectors can effectively express the

characteristics of sub-words. This result also shows the effectiveness of the method proposed in this study to combine embedding and glyph feature vectors in practice. In addition, in the case of retaining 70% of Chinese characters, the score of unigram F1 reached the highest 66.92, and there is a gap of about 0.6 with other results. At the same time, the conversion of Chinese characters into radicals to a greater extent affects the performance of embedding in machine translation to a certain extent. Compared with the result of keeping all Chinese characters, the BLEU score of converting all Chinese characters into radicals is reduced by 0.6. Although the OOV problem is better handled, the information loss in the characters is not compensated.

Compared with unigram F1, BLEU evaluates the overall results of translation more comprehensively. It can be observed from the results that according to the four different ratios, the difference between BLEU is not huge, and the difference between the highest value and the lowest value is only 0.95. This shows that in translation, all four embedding achieve acceptable levels. Among them, 30% got the highest BLEU score with 30.14, same as unigram F1. It can be known that 30% found the best balance between the loss of character information and the alleviation of OOV problem. Compared with 30%, the result of keeping all Chinese characters got the closest result, while the result of converting all Chinese characters into radicals had a gap less than 1 against the best result.

Combining the results of these two different metrics, it can be found that when the conversion ratio is 30%, the combination of embedding and glyph feature vectors achieves the best results. Compared with 30%, the performance of other ratios have decreased to a certain extent. When a larger ratio of Chinese characters is converted into radicals, more character information will be lost. Obviously, the glyph feature cannot fully complement this part of the information. For example, compared to original sub-word “百年来”, its corresponding radical based sub-word “白丿一” only contains a part of characters’ information while vision transformer capturing glyph feature from full original character image. When more Chinese characters are reserved, due to the sparseness of Chinese characters, the performance on handling OOV is even worse. 70% characters and 30% radicals will be the best balance, while alleviating the OOV problem.

5.2 Ablation Study

Here I will analyze the performance of the glyph feature extracted by the vision transformer on the final embedding. I will compare the performance of embedding without glyph feature and

Table 5.2: Comparison of translation performance (BLEU, Unigram F1).

Conversion ratio	ViT	BLEU	Unigram F1
0%	With	29.79	66.13
	W/O	28.82	65.43
30%	With	**30.14**	**66.92**
	W/O	24.43	63.75
60%	With	29.23	66.37
	W/O	23.96	63.13
100%	With	29.19	66.22
	W/O	23.89	63.17

embedding with glyph feature on machine translation tasks.

In order to reflect the impact of whether to use glyph feature on the performance of embedding, I use exactly the same settings as in previous experiments in machine translation, including the parameters of the transformer used for training and the hyper-parameters during training. In addition, the training set and test set are also consistent.

I show the results of embedding with 4 different ratios in Table 5.2 and compare them with the results of embedding combined with glyph feature vectors. In the Table, **With** represents the case that ViT is used to extract glyph feature, and **W/O** indicates that ViT is not applied.

Based on these results, I can evaluate the effectiveness of vision transformer for complementing the missing information when converting characters to radicals. Among results without using vision transformer, the score of BLEU and F1 are continuously decreasing while the conversion ratio keeps increasing. Converting more Chinese characters leads to more missing information. For example, after converting all characters to radicals, “这个目标必须是最小的稳定性, 而不是胜利. 这似乎还是可以实现的.” (Meaning: The goal must be minimal stability, not victory. This still seems achievable) becomes “辶人目木丶彡日曰小白禾宀忄, 而一日月禾. 辶亻丿

辶日口人宀王白.". It is difficult to recognize the original Chinese characters and understand the meaning of the sentence, which leads to a decrease in the embedding effect. However, after combining embedding with glyph feature vectors, the score of BLEU and F1 is significantly improved, and it can be seen that the missing information can be complemented by glyph features of Chinese characters. For example, I can obtain the left half information of the Chinese character "列" by using glyph feature, which is lost during converting from the original Chinese character "列" to "刂". Except for the case of retaining all Chinese characters, there is a big difference between using glyph feature and not using glyph feature. The biggest difference occurs at 100%, BLEU increases from 23.89 to 29.19, and the score of F1 increases from 63.17 to 66.22.

When all Chinese characters are retained, the use of glyph features can also effectively improve the translation performance. Although the improvement is not so obvious compared with the performance of converting Chinese characters into radicals, it can still be seen that the glyph feature retains another part of the information in the Chinese word. The glyph feature extracted by the vision transformer model not only complement the information loss when converting to radicals, but also extract the information of words in the character sense.

5.3 Shortcomings

Here I will discuss some shortcomings in the study, including deficiencies in embedding training, deficiencies in evaluation, and possible solutions to these problems.

5.3.1 Sub-word Embedding

In practice, I only use the Skip-gram model to train Chinese text to obtain word embedding. When using the Skip-gram model, sub-word with a frequency below 5 will be ignored, and this setting should be related to the actual situation of the text. Inappropriate settings may have a negative impact on the trained sub-word embedding.

BERT, as the most widely used pre-training language model at present, has not been selected for training sub-word embedding in this study due to its high requirements for text and hardware. In order to further improve the performance of embedding in tasks such as machine translation, I recommend using BERT and other pre-training language model to train the sub-word embedding.

5.3.2 Evaluation Tasks

In the evaluation, only the embedding according to four conversion ratios are used for comparison. Due to the long evaluation period, I have not been able to compare the performance of embedding combined with glyph feature vectors in more cases in more detail, and the results obtained only represent the optimal solutions in these four cases. I believe that by dividing by 10% and experimenting, more comprehensive results can be achieved.

In addition to machine translation, other NLP tasks can be used to evaluate my embedding, such as text generation, text classification, sentiment analysis, etc. Particularly, more diverse evaluation metrics should be applied to the evaluation tasks for a more comprehensive evaluation of my proposed method.

Chapter 6

Conclusion

6.1 Summary

In this study, I propose a new method for processing Chinese text, and a method of using vision transformer to extract Chinese character glyph feature through transfer learning. Glyph feature vectors are combined with embedding to improve the performance in downstream task. Using glyph features of Chinese characters, I transform Chinese characters into corresponding radicals, alleviating the sparsity problem of Chinese and the OOV problem when processing. Moreover, after I converted Chinese characters into radicals according to four different conversion ratios, I compared the embedding performance and the number of OOV words with each other. My experiments show that converting Chinese characters into radicals can indeed reduce the number of OOV words when processing Chinese text, and using the vision transformer to extract glyph features and combine embedding with it can improve the sub-word embedding performance.

The main contributions of this paper are as follows:

1. I propose a new approach to Chinese processing, which converts Chinese characters in text into their corresponding radicals. Based on the converted text, I use BPE algorithm to generate sub-word. Compared with Chinese characters, there are fewer kinds of radicals, so as to alleviate the sparsity problem of Chinese characters and reduce the number of OOV words that may be encountered when processing.

2. A method of applying the vision transformer as an autoencoder to the glyph feature extraction of Chinese characters is proposed. The vision transformer pre-trained in the Chinese

character image reconstruction task is introduced into the extraction of glyph features using transfer learning.

3. I propose a novel approach to combine character sub-word embedding with glyph feature vectors. Compared with the previous ones, the combined embedding are more effective in machine translation tasks.

4. I have shown that converting Chinese characters into radicals according to a low ratio can greatly reduce the number of OOV words while improving the performance of sub-word embedding.

6.2 Future Work

In the experiment, I only used the Word2Vec model to train the sub-word embedding, while pre-training models such as BERT are currently more general methods. Therefore, I will add the BERT model for sub-word embedding in future experiments. At the same time, in this study, the embedding performance of four different conversion ratio were used for comparison, but there are still more cases that have not been included. I will continue to experiment with other conversion ratios and compare the results. Finally, in order to obtain more comprehensive results, in addition to machine translation, other tasks will also be added to the experiment to compare the performance of different embedding.

Acknowledgements

Affected by the corona virus, the two years of studying for a master's degree at the University of Tsukuba were full of various uncertainties. This kind of uncertainty is not only reflected in scientific research, but also in life, which has caused me various difficulties. And it was my supervisor and tutors in the who helped me overcome these difficulties.

First of all I would like to thank my supervisor Associate Professor Yohei Seki. With his help, I realized my research plan step by step and achieved the desired goal. From him, I learned the rigor and earnestness of a researcher, and was infected by his enthusiasm for scientific research. At the same time, I would also like to thank Associate Professor Kei Wakabayashi for his valuable advice on my research and for his support and help in my research.

Finally, I would like to thank my tutor Tetsuya Ishida, who gave me a lot of help as a foreigner who has never lived in Japan. He not only solved the difficulties I encountered in my research, but also gave me advice on various problems in my life. At the same time, as a friend, he also played an active role in relieving me from the loneliness and research pressure.

There are many other people who gave me all kinds of help and care, including classmates, teachers, friends and relatives. Without them, I would not have the various achievements I have made in the past two years. I hope they can stay healthy during the epidemic.

References

[1] Gustavo Aguilar, Bryan McCann, Tong Niu, Nazneen Rajani, Nitish Keskar, and Thamar Solorio. Char2subword: Extending the subword embedding space using robust character compositionality. *arXiv preprint arXiv:2010.12730*, 2020.

[2] Sara Atito, Muhammad Awais, and Josef Kittler. Sit: Self-supervised vision transformer. *arXiv preprint arXiv:2104.03602*, 2021.

[3] Yoshua Bengio, Réjean Ducharme, and Pascal Vincent. A neural probabilistic language model. *Advances in Neural Information Processing Systems*, 13:1137–1155, 2000.

[4] Xueting Bi and Xiaojun Liu. Chinese character captcha sequential selection system based on convolutional neural network. In *2020 International Conference on Computer Vision, Image and Deep Learning (CVIDL)*, pages 554–559. IEEE, 2020.

[5] Shaosheng Cao, Wei Lu, Jun Zhou, and Xiaolong Li. cw2vec: Learning chinese word embeddings with stroke n-gram information. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 32, pages 5053–5061, 2018.

[6] Falcon Dai and Zheng Cai. Glyph-aware embedding of chinese characters. In *Proceedings of the First Workshop on Subword and Character Level Models in NLP*, pages 64–69, 2017.

[7] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, 2019.

[8] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly,

Jakob Uszkoreit, and Neil Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*, 2020.

[9] Jun Fu, Jing Liu, Haijie Tian, Yong Li, Yongjun Bao, Zhiwei Fang, and Hanqing Lu. Dual attention network for scene segmentation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 3146–3154, 2019.

[10] Philip Gage. A new algorithm for data compression. *C/C++ Users Journal*, 12(2):23–38, 1994.

[11] Yoav Goldberg and Omer Levy. word2vec explained: deriving mikolov et al.'s negative-sampling word-embedding method. *arXiv preprint arXiv:1402.3722*, 2014.

[12] Kaiming He, Xinlei Chen, Saining Xie, Yanghao Li, Piotr Dollár, and Ross Girshick. Masked autoencoders are scalable vision learners. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 16000–16009, 2022.

[13] Geoffrey E Hinton. Learning distributed representations of concepts. In *Proceedings of the 8th Annual Conference of the Cognitive Science Society*, volume 1, pages 1–12. Amherst, MA, 1986.

[14] Saki Ibe, Yoshitatsu Matsuda, and Kazunori Yamaguchi. Replacement of unknown words using an attention model in japanese to english neural machine translation. *Journal of Natural Laguage Processing*, 25(5):511–525, 2018.

[15] Sébastien Jean, Kyunghyun Cho, Roland Memisevic, and Yoshua Bengio. On using very large target vocabulary for neural machine translation. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 1–10, 2015.

[16] Yuanzhi Ke and Masafumi Hagiwara. Radical-level ideograph encoder for rnn-based sentiment analysis of chinese and japanese. In *Asian Conference on Machine Learning*, pages 561–573. PMLR, 2017.

[17] Yoon Kim, Yacine Jernite, David Sontag, and Alexander M Rush. Character-aware neural language models. In *Thirtieth AAAI conference on artificial intelligence*, pages 2741–2749.

[18] Philipp Koehn and Rebecca Knowles. Six challenges for neural machine translation. In *Proceedings of the First Workshop on Neural Machine Translation*, pages 28–39, 2017.

[19] Shen Li, Zhe Zhao, Renfen Hu, Wensi Li, Tao Liu, and Xiaoyong Du. Analogical reasoning on Chinese morphological and semantic relations. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 138–143, Melbourne, Australia, July 2018.

[20] Xiaoya Li, Yuxian Meng, Xiaofei Sun, Qinghong Han, Arianna Yuan, and Jiwei Li. Is word segmentation necessary for deep learning of chinese representations? In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 3242–3252, 2019.

[21] Yang Li and Tao Yang. Word embedding for understanding natural language: a survey. In *Guide to Big Data Applications*, pages 83–104. Springer, 2018.

[22] Ying-Tian Liu, Yuan-Chen Guo, Yi-Xiao Li, Chen Wang, and Song-Hai Zhang. Learning implicit glyph shape representation. *IEEE Transactions on Visualization and Computer Graphics*, pages 1–12, 2022.

[23] Minh-Thang Luong, Ilya Sutskever, Quoc Le, Oriol Vinyals, and Wojciech Zaremba. Addressing the rare word problem in neural machine translation. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 11–19, 2015.

[24] Bing Ma, Qi Qi, Jianxin Liao, Haifeng Sun, and Jingyu Wang. Learning chinese word embeddings from character structural information. *Computer Speech & Language*, 60:101031, 2020.

[25] Yuxian Meng, Wei Wu, Fei Wang, Xiaoya Li, Ping Nie, Fan Yin, Muyu Li, Qinghong Han, Xiaofei Sun, and Jiwei Li. Glyce: Glyph-vectors for chinese character representations. *Advances in Neural Information Processing Systems*, 32:2746–2757, 2019.

[26] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*, 2013.

[27] Tomas Mikolov, Martin Karafiát, Lukas Burget, Jan Cernockỳ, and Sanjeev Khudanpur. Recurrent neural network based language model. In *Interspeech*, volume 2, pages 1045–1048, 2010.

[28] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. Distributed representations of words and phrases and their compositionality. *Advances in Neural Information Processing Systems*, 26:3111–3119, 2013.

[29] Thi-Vinh Ngo, Thanh-Le Ha, Phuong-Thai Nguyen, and Minh Le Nguyen. Overcoming the rare word problem for low-resource language pairs in neural machine translation. In *Proceedings of the 6th Workshop on Asian Translation*, pages 207–214, 2019.

[30] Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, pages 311–318, 2002.

[31] Rico Sennrich, Barry Haddow, and Alexandra Birch. 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, 2016.

[32] Zein Shaheen, Gerhard Wohlgenannt, and Erwin Filtz. Large scale legal text classification using transformer models. *arXiv preprint arXiv:2010.12871*, 2020.

[33] Zijun Sun, Xiaoya Li, Xiaofei Sun, Yuxian Meng, Xiang Ao, Qing He, Fei Wu, and Jiwei Li. Chinesebert: Chinese pretraining enhanced by glyph and pinyin information. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 2065–2075, 2021.

[34] Hanqing Tao, Shiwei Tong, Tong Xu, Qi Liu, and Enhong Chen. Chinese embedding via stroke and glyph information: A dual-channel view. *arXiv preprint arXiv:1906.04287*, 2019.

[35] Jörg Tiedemann. Parallel data, tools and interfaces in opus. In *The International Conference on Language Resources and Evaluation*, volume 2012, pages 2214–2218, 2012.

[36] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in Neural Information Processing Systems*, 30:6000–6010, 2017.

[37] Han Xu, Wang Hongsu, Zhang Sanqian, Fu Qunchao, and Liu Jun. Sentence segmentation for classical chinese based on lstm with radical embedding. *The Journal of China Universities of Posts and Telecommunications*, 26(2):1–8, 2019.

[38] Shusheng Xu, Xingxing Zhang, Yi Wu, and Furu Wei. Sequence level contrastive learning for text summarization. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 36, pages 11556–11565, 2022.

[39] Longtu Zhang and Mamoru Komachi. Using sub-character level information for neural machine translation of logographic languages. *ACM Transactions on Asian and Low-Resource Language Information Processing*, 20(2):1–15, 2021.

[40] Xiang Zhang, Junbo Zhao, and Yann LeCun. Character-level convolutional networks for text classification. *Advances in Neural Information Processing Systems*, 28:649–657, 2015.

[41] Hengshuang Zhao, Yi Zhang, Shu Liu, Jianping Shi, Chen Change Loy, Dahua Lin, and Jiaya Jia. Psanet: Point-wise spatial attention network for scene parsing. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 267–283, 2018.

[42] 周梅森. 人民的名义. 北京出版集团, 2017.