國立成功大學
人工智慧科技碩士學位學程
碩士論文
(初稿)

以聯合語義語音詞嵌入強化中日文神經機器翻譯

Improving Chinese-Japanese Neural Machine Translation with Joint Semantic-Phonetic Word Embedding

研究生：王士杰　　Student : Shih-Chieh Wang

指導教授：賀保羅　　Advisor : Paul Horton

Master Degree Program on Artificial Intelligence,
National Cheng Kung University,
Tainan, Taiwan, R.O.C.
Thesis for Master of Science Degree
July, 2021

中華民國一百一十年七月

以聯合語義語音詞嵌入強化中日文神經機器翻譯

王士杰* 賀保羅†

國立成功大學人工智慧科技碩士學位學程

摘要

近年來，神經機器翻譯因引入編碼器-解碼器網路而逐漸完善，但是在中文及日文的翻譯任務中，由於訓練資料的缺乏，以及和西方語言的差異性，始終無法獲得像英文與其他語言之間的高翻譯品質。本篇論文嘗試使用聲音資訊作爲中日文的額外特徵，並將該特徵應用於中日文的神經機器翻譯系統當中，旨在透過以特徵工程的方式來加強翻譯品質。

本論文基於三種分詞方法，在不同的分詞下，提取出語料庫中的中文注音以及日文平假名做爲聲音資訊。接著，利用文字資訊以及聲音資訊，分別訓練出帶有語義以及帶有語音的詞嵌入。我們混合兩者，產生出同時帶有語義以及語音的「合併詞嵌入」，並將其投入至兩種主流的神經機器翻譯模型，進行訓練與進一步的特徵提取。

實驗採用雙語評估替補分數 (BLEU) 對翻譯結果進行評分，結果表明，使用合併詞嵌入進行訓練與微調的模型，皆獲得比單純使用語義或語音的詞嵌入還要更高的分數；我們亦對模型的翻譯結果進行案例分析，由合併詞嵌入產生出的翻譯能夠保留正確，甚至更爲精確的詞彙；也能夠保留片假名與英文的單詞，得到語義上的提升。實驗另外對合併詞嵌入和單純包含語義的詞嵌入進行四項分析，每項分析皆獲得正面回饋，顯示合併詞嵌入能夠保留，甚至超越語義詞嵌入所持有的向量涵義。

綜合翻譯分數以及詞嵌入分析，我們發現單純使用小型語料庫提取語音資訊作爲特徵，便能對中日文的詞向量帶來正面影響；此外，使用合併語義及語音的詞嵌入，能夠進一步有效提升中日文神經機器翻譯系統的效能。

關鍵字： 神經機器翻譯、詞嵌入、特徵工程、語音資訊

*學生
†指導教授

Improving Chinese-Japanese Neural Machine Translation with Joint Semantic-Phonetic Word Embedding

Shih-Chieh Wang* Paul Horton†

Master Degree Program on Artificial Intelligence
National Cheng Kung University

Abstract

Neural machine translation has been improved by the introduction of encoder-decoder networks in recent years. However, the translation between Chinese and Japanese has not yet achieved the same high quality as that between English and other languages due to the lack of parallel corpora and less similarity in languages. This paper attempted to use phonetic information as an additional feature in Chinese and Japanese. The aim is to enhance the translation quality by feature engineering. We first extracted Chinese Bopomofo and Japanese Hiragana as phonetic information from the corpus with three tokenization approaches. Second, word embeddings with semantics and word embeddings with phonetics are trained on the text and phonetic information, respectively. Third, we combined both embeddings to create a joint semantic-phonetic embedding and implemented it into two mainstream neural machine translation models for training and further extracting the features. The results show that the models trained and fine-tuned with joint embeddings yield higher BLEU scores than those using semantic or phonetic embeddings only. We also conducted case studies on the translation results. The translations generated with joint embeddings tend to select the correct and even more precise words and retain Japanese Katakana and English words, resulting in semantic improvements. Besides, four analyses conducted on joint embeddings and semantic embeddings all gave positive feedback, which showed that the joint embeddings could preserve and even outperform the vector meanings possessed by the semantic embeddings. The reveal of BLEU scores and embedding analysis demonstrates that simply using a small corpus to extract phonetic information as a feature can positively affect the Chinese and Japanese word vectors. In addition, the use of joint semantic-phonetic embeddings can effectively improve the performance of Chinese-Japanese neural machine translation systems.

Keywords: Neural Machine Translation, Word Embedding, Feature Engineering, Phoentic Information

*Student
†Advisor

Acknowledgements

During the months of writing the thesis, I have encountered many difficulties and obstacles, but now I feel a sense of accomplishment and warmth in my heart when I see the final draft.

First of all, I am deeply indebted to Dr. Paul Horton. Without his guidance, support and kindness, I would never have been able to pursue the Chinese-Japanese translation system as a thesis topic. He has given a lot of foresighted but insightful advice on topic content, essay writing, etc. Every time I communicate with him, I always gain a lot. 長い間、ご指導頂きまして、本當に感謝しております。

Secondly, I would like to thank Dr. Hung-Yu Kao and Dr. Wei-Ta Chu for agreeing to be my thesis committee despite their extremely busy schedule. I am grateful to them for taking the time to read my thesis and giving it all kinds of useful advice. They are all professors who are passionate about teaching and research, and I believe their academic achievements will continue to increase.

Thirdly, I would like to thank the owners of the information, literature, and ideas cited and referenced in this thesis. Without these constructive and prospective materials, I would not have been able to accomplish my thesis. Thank you to these researchers who are willing to share their knowledge with the world.

Lastly, I would like to thank my family and all my friends. Because of your encouragement, advice, and help, I was able to complete this thesis successfully.

Shih-Chieh Wang

CONTENTS

LIST OF TABLES

LIST OF FIGURES

Chapter 1

Introduction

The field of neural machine translation (NMT) between Chinese and Japanese is an area of active research with recent studies employing specific methods such as sub-character level features to improve the translation quality. Nevertheless, the quality of Chinese-Japanese machine translation is low compared to the high standard obtainable for language pairs such as English-French. The difficulties include a comparative lack of parallel corpora, and less overall similarity as languages. Another striking difference between translation between European languages versus Chinese-Japanese, is the complex orthography adopted by the latter two. Chinese and Japanese writing both employ thousands of characters, most having semantic significance, but some mostly or purely phonetic. This research explores phonetic information as an additional feature for improving the quality of Chinese-Japanese NMT systems.

1.1 Background

NMT, a popular area of natural language processing (NLP), has been proposed by using an end-to-end model which transforms a source sentence into a latent space and decodes it directly into a target sentence [Sutskever et al., 2014,Cho et al., 2014]. The model is called the encoder-decoder model or sequence-to-sequence model, and such models are widely used by large technology companies such as Google, Facebook, Microsoft, and DeepL.

1.1.1 Progress of Neural Machine Translation

The progress of NMT and NLP are inseparable. The development of models, tokenization methods, embeddings, and solutions to insufficient parallel corpora, all can help drive the progress of NMT.

Recurrent neural networks (RNNs), attention mechanisms, and transformers have been proposed sequentially throughout the progress of encoder-decoder models. An RNN that recursively passes states in its networks was first applied in the encoder-decoder model [Cho et al., 2014]. Attention-mechanism then addressed the problem of insufficient information in the latent space between encoder and decoder based on RNN [Bah-

danau et al., 2014]. Finally, transformers replaced the RNN structure with full attention-mechanism (i.e., self-attention) to achieve better results and used widely in NMT tasks [Vaswani et al., 2017].

Tokenization is one of the most important parts of any NLP task. It determines how a sentence is decomposed, and can affect the performance of downstream processing. Besides word-level and character-level tokenization, several subword-level tokenization algorithms have become mainstream. For example: Byte-Pair Encoding (BPE) [Sennrich et al., 2016b], Unigram Language Model [Kudo, 2018], WordPiece [Schuster and Nakajima, 2012], and SentencePiece [Kudo and Richardson, 2018]. This paper will utilize BPE, SentencePiece [Sennrich et al., 2016b,Kudo and Richardson, 2018] and the two word-level tokenizers (*Jieba*[1] and *Janome*[2]) as tokenization methods.

The concept of embeddings, also known as distributed representations, was first proposed by [Hinton et al., 1986,Bengio et al., 2003], but was difficult to implement due to hardware limitations. With the development of parallel computing and GPU, many embedding implementations have been proposed, such as Word2Vec [Mikolov et al., 2013a], GloVe [Pennington et al., 2014], and fastText [Bojanowski et al., 2017]. Contextualized word embedding is another concept that obtains context-dependent word embedding from the whole sentence, meaning that the same word with a different position can differentially be represented in the model. Well known word embedding tools include ELMo [Peters et al., 2018] and BERT [Devlin et al., 2019]. Here, we select Word2Vec [Mikolov et al., 2013a] as the tool for creating word embeddings because of its simplicity, rapidity, and convenience of analysis.

Several techniques have been developed to solve the problems like low-resources and noisy parallel data in NMT tasks. Back-translation [Sennrich et al., 2016a] is a data augmentation method that uses monolingual data of the target language to generate source data and offset the imbalance between encoder and decoder. Parallel corpus filtering was examined for a large number of NMT tasks [Koehn et al., 2018], using pre-filtering rules and scoring functions to retain good sentence pairs can effectively reduce the corpus size and obtained better translation results. This paper employs corpus filtering to retain quality training data and reduce corpus size to increase experimental efficiency.

[1] https://github.com/fxsjy/jieba
[2] https://mocobeta.github.io/janome

1.1.2 Chinese-Japanese Neural Machine Translation

NMT system has gained a lot of improvement in translating between English and other languages by utilizing the techniques described in section 1.1.1. However, the improvement in translating between Chinese and Japanese is limited. The main reasons are the inadequacy of the corpus and the differences in the writing systems of Chinese, Japanese, and Western languages.

Many studies have focused on improving the Chinese-Japanese (zh-ja) NMT system. In addition to using the methods [Imamura et al., 2018,Chu et al., 2017,Zhang et al., 2020] described in section 1.1.1, many feature engineering techniques have been proposed to utilize the features in Chinese Characters (known as *hanzi* in Mandarin and *kanji* in Japanese). For example, a character-level zh-ja NMT system has been improved by using radicals as character feature information [Zhang and Matsumoto, 2017]. Furthermore, the use of decomposed sub-character level information such as ideographs and strokes of Chinese characters, also improved the results [Zhang and Komachi, 2018].

1.1.3 Phonetic Information

Phonetic information is another feature that had been applied to NMT systems. [Khan and Xu, 2019] had suggested that a phonetic representation usually corresponds to semantically distinct characters or words. [Liu et al., 2019] had pointed out that phonetic information can effectively resist the homophone noises generated by typographical mistakes in Chinese sentences. Both papers had improved the performance of the NMT system between Chinese and other Western languages.

This paper attempts to use *Bopomofo* and *Hiragana* as Chinese and Japanese phonetic information to improve the performance of the zh-ja NMT system. Bopomofo also named *Zhuyin* (注音), is located in the Unicode block in the range U+3100–U+312F. It consists of 37 characters and 4 tone marks to transcribe all possible Chinese characters. Bopomofo is specifically designed to phonetically annotate Mandarin Chinese (in dictionaries and school reading primers etc.), but is not usually used when writing Chinese sentences. Thus phonetic information is not directly available to a machine reading normal Chinese sentences. Hiragana (平仮名, ひらがな) is a component of Japanese orthography, along with *katakana* and kanji. It consists of 46 base characters and is located in the Unicode block in the range U+3040–U+309F. Hiragana are phonetic characters, forming a complete phonetic syllabary of the Japanese language. For each hiragana character, there is a corresponding katakana character (analogous to lower and upper case Latin letters). Collectively, hiragana

and katakana are called *kana*. Kana plays two roles in Japanese: first, they both appear in normal Japanese text (so-called "仮名交じり文") where they complement kanji; second, kana can be used to phonetically annotate kanji. The second role is exactly analogous to the role of bopomofo in Mandarin, and like in Mandarin, a machine reading normal Japanese sentences does not have access to phonetic information regarding the kanji part of the text.

1.2 Objective

This paper aims to determine whether the use of phonetic information can help improve the performance of the zh-ja NMT system. We use embedding, which is commonly used to represent semantics, to represent the features of phonetic information. The *gensim* library [3] will be utilized to implement Word2Vec [Mikolov et al., 2013a] to extract both semantic and phonetic embedding. The embeddings will be trained on a small corpus (less than 1 million lines of sentences) to see if they are useful for the subsequent NMT task.

Combining the findings from other studies [Liu et al., 2019,Khan and Xu, 2019] described in section 1.1.3, we hypothesized that embeddings with the combination of semantics and phonetics (joint embedding) may improve the performance of the zh-ja NMT system more effectively than embeddings with only semantics or phonetics.

Here, we perform a series of experiments to test this hypothesis. First, we examine whether the joint embedding can improve the results of the zh-ja NMT system with different tokenization methods. Second, we conduct NMT tasks under four conditions: without any pre-trained embedding, with pre-trained semantic embedding, with pre-trained phonetic embedding, and with joint semantic-phonetic embedding. Third, we analyze the changes that occur when phonetic information is added to the general semantic embedding. The analyses include analogical reasoning, outlier detection, word similarity, and the influence on both homonyms and heteronyms.

1.3 Related Work

The core technique in this paper is to enhance the ability of word embeddings by utilizing feature engineering on phonetic information. Therefore, we review some studies that use additional features to improve word

[3] https://radimrehurek.com/gensim/index.html

embeddings. The review will be carried out from two perspectives, one is to improve embedding with the features of Chinese characters such as radicals and strokes, and the other is to improve embedding with the features of phonetics. The concept of Word2Vec [Mikolov et al., 2013a] has been reviewed elsewhere, but we will mention it in section 2.3.1 of Chapter 2.

1.3.1 Chinese Word Embedding

We review some studies which suggested that the rich, decomposed information of Chinese characters can be exploited to train the word embeddings with words or characters jointly. Some of the most popular studies in this field are: *CWE* [Chen et al., 2015], *MGE* [Yin et al., 2016], *JWE* [Yu et al., 2017], and *cw2vec* [Cao et al., 2018]. The following sections review the approaches from JWE and cw2vec because their concepts are relatively new and their performance is better.

Joint Learning Word Embedding Model (JWE)

The authors of JWE [Yu et al., 2017] proposed a model that combines word, character, and sub-character components to learn embeddings. They implemented the model (Figure 1.1) based on the Continuous Bag of Words (CBOW) proposed in Word2Vec [Mikolov et al., 2013a], and changed the default input words in CBOW by further adding the characters and sub-character components of input words.

As shown in Figure 1.1 above, The JWE model is trained to predict the present word from the context words in the same way as CBOW. For example, the Chinese word 智能 (intelligence) has two characters 智 (wisdom) and 能 (able), decomposable into sub-character components: {知, 日} and {厶, 月, 匕x2}. The JWE model aims to maximize:

$$L(w_i) = \sum_{k=1}^{3} \log P(w_i \mid h_{ik})$$

the sum of 3 conditional probabilities from the context words (h_{i1}), characters (h_{i2}), and sub-characters (h_{i3}) respectively.

cw2vec Model

Inspired by fastText [Bojanowski et al., 2017], the authors of cw2vec [Cao et al., 2018] proposed an n-gram feature based on the strokes of Chinese characters. They first split and transform the text into stroke information, then map the strokes into 5 corresponding stroke ids, and finally merge the ids and generate n-gram

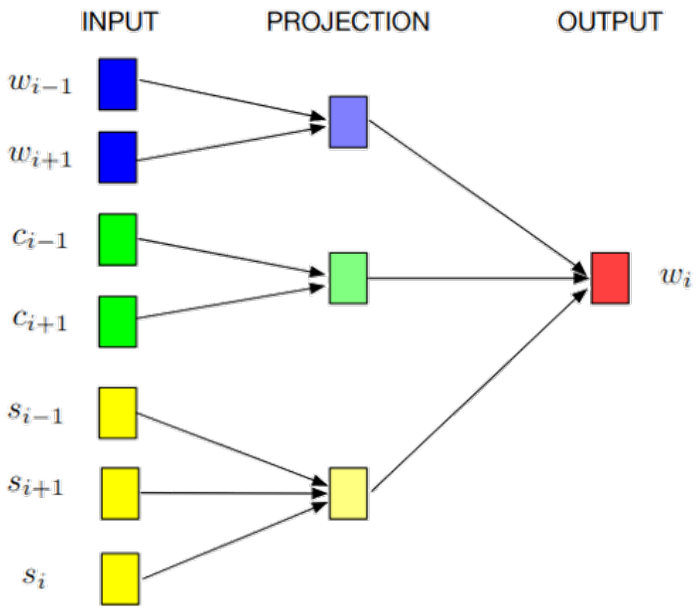


Figure 1.1: An illustration of the JWE model (reproduced from [Yu et al., 2017]). The symbol w represents words, w_i is the current word, w_{i-1} and w_{i+1} are the context words. The symbol c represents the characters of context words, c_{i-1} is the characters of w_{i-1}, and c_{i+1} is the characters of w_{i+1}. The symbol s represents the sub-characters of context words. s_i is the sub-character components of w_i, and s_{i-1}, s_{i+1} are the components of w_{i-1} and w_{i+1}.

features from these stroke ids. The diagram in the paper (Figure 1.2) shows the complete process.

The authors used Skip-Gram, a second method other than CBOW proposed by Word2Vec [Mikolov et al., 2013a], as the base model, and replaced word inputs with stroke n-grams (Figure 1.3). It is mentioned in the paper that the final embeddings come from the contextual word vectors.

These word embeddings designed for Chinese all utilized the radicals or strokes in Chinese characters. No research in Chinese word embeddings has so far used phonetic information extracted from Chinese or Japanese as features.

1.3.2 Phonetic Word Embedding

Next, we review studies that use phonetic information to construct word embeddings. These studies are closely related to our research, we can learn from their practical differences and use them as the cornerstone for our experiments.

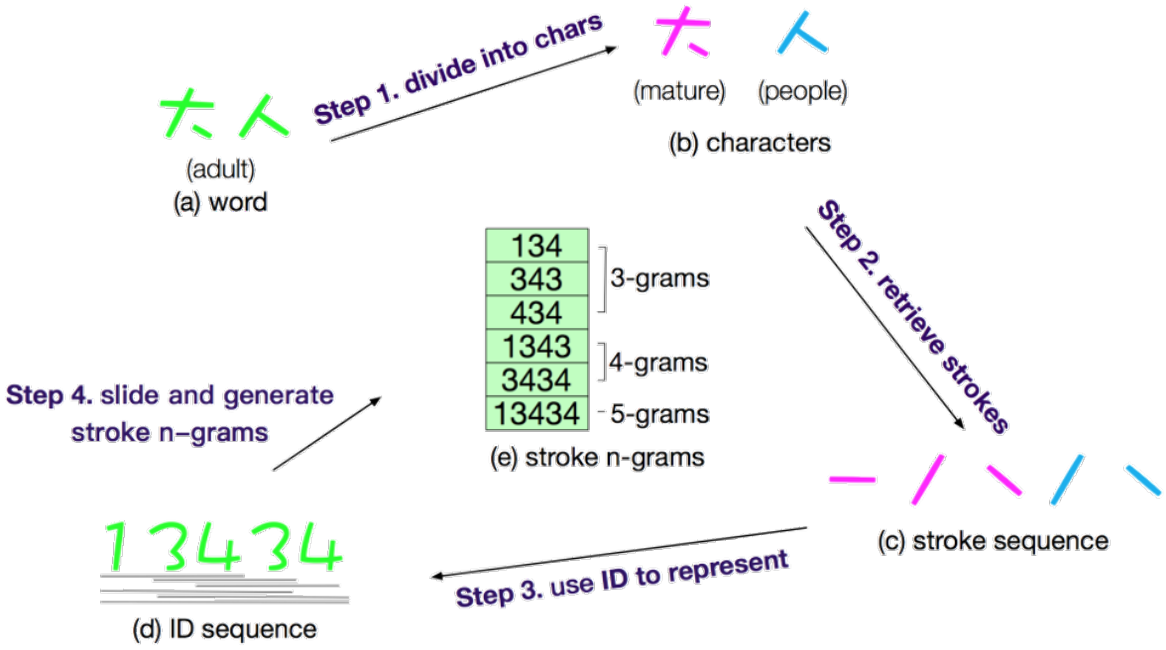


Figure 1.2: The Process of generating stroke n-grams in cw2vec (reproduced from [Cao et al., 2018])

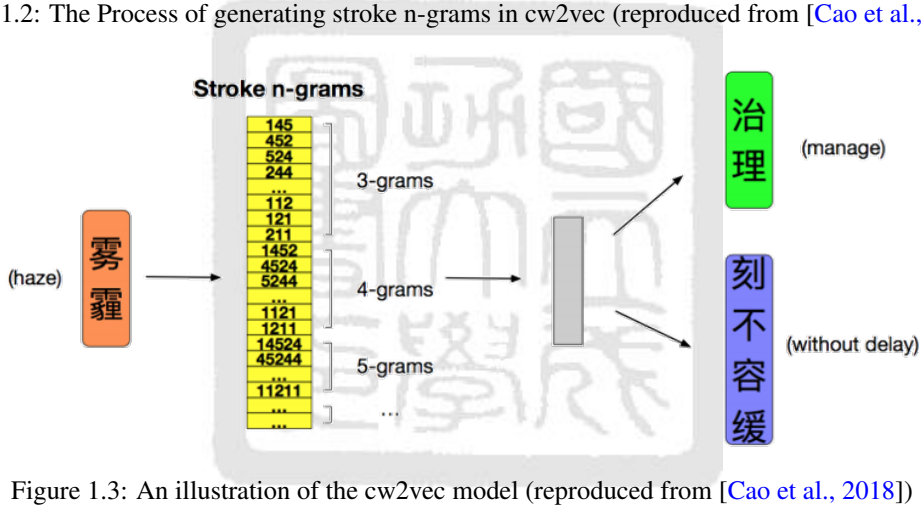


Figure 1.3: An illustration of the cw2vec model (reproduced from [Cao et al., 2018])

Phonetic Encoding

[Khan and Xu, 2019] extracted sentences from Chinese or Western languages into phonetic encodings [4], which used *Soundex*, *NYSIIS*, *Metaphone*, and *Hanyu Pinyin* as the extraction algorithm. After that, they applied BPE [Sennrich et al., 2016b] to tokenize the sentences and encodings, and trained separate embeddings from the sentences and each encoding (empty boxes in Figure 1.4). Lastly, they concatenated the embeddings and fed them into the NMT system.

Their paper did not explain the implementation of phonetic information in detail, but they had presented a hypothesis and verified it. That is, phonetics is a function that groups semantically distinct words. The words

[4]The logogram encoding is also used as a feature but is not explained here. Random clustering is used for comparison purposes.

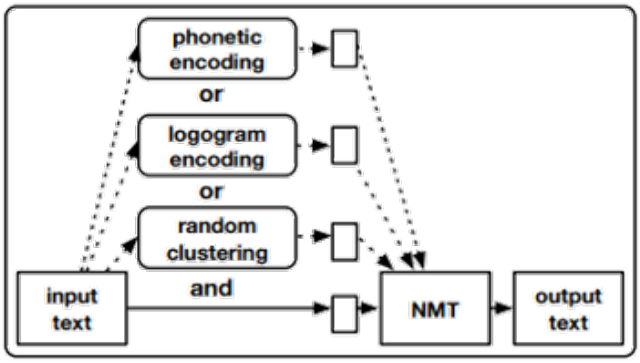


Figure 1.4: Implementation of phonetic encoding (reproduced from [Khan and Xu, 2019])

with the same pronunciation and spelling are usually distinguished by the different contexts.

Joint Textual and Phonetic Embedding

[Liu et al., 2019] focused on using phonetic embedding to address the homophone noise problem, which is a frequent problem in a parallel corpus. For instance, when a single word in the source sentence is replaced with a homophone (e.g., 有 (yǒu, to have) is wrongly replaced with 友 (yǒu, friend)), the model will read the wrong embedding and train in a wrong direction. The phonetic embedding can be seen as a feature to compensate for errors that occur in semantic embeddings with homophone noise.

[Liu et al., 2019] trained the text (denoted by a) as semantic embedding (denoted by $\pi(a)$) and phonetic embedding (denoted by $\psi(a)$), and combined two embeddings with a configurable parameter (denoted by β) as follows:

$$\pi([a, \psi(a)]) = (1 - \beta) \times \pi(a) + \beta \times \pi(\psi(a))$$

Their paper did not give the details of constructing embeddings, but they mentioned that the best result occurs when the β is 0.95. That is, when they used 5% semantic embedding and 95% phonetic embedding, they obtained the best performance in their NMT system.

To summarize the findings in these related works. The phonetic encodings can emphasize the difference between semantically diverse sentences. Joint semantic-phonetic embedding also shows the robustness to noise in parallel corpora. All of these features can help improve the performance of the NMT system. However, the methods of using phonetic information as embeddings had been explored mainly in Chinese and Western languages. They also used *Pinyin* instead of Bopomofo as the component to decompose Chinese characters. Ac-

cording to our study, no research has been proposed to apply and analyze phonetic information as an additional feature in Chinese and Japanese NMT systems.

Chapter 2

Method

This paper attempts to improve the zh-ja NMT system by utilizing the phonetic information hidden in the sentences. The complete structure of this paper is shown in Figure 2.1.

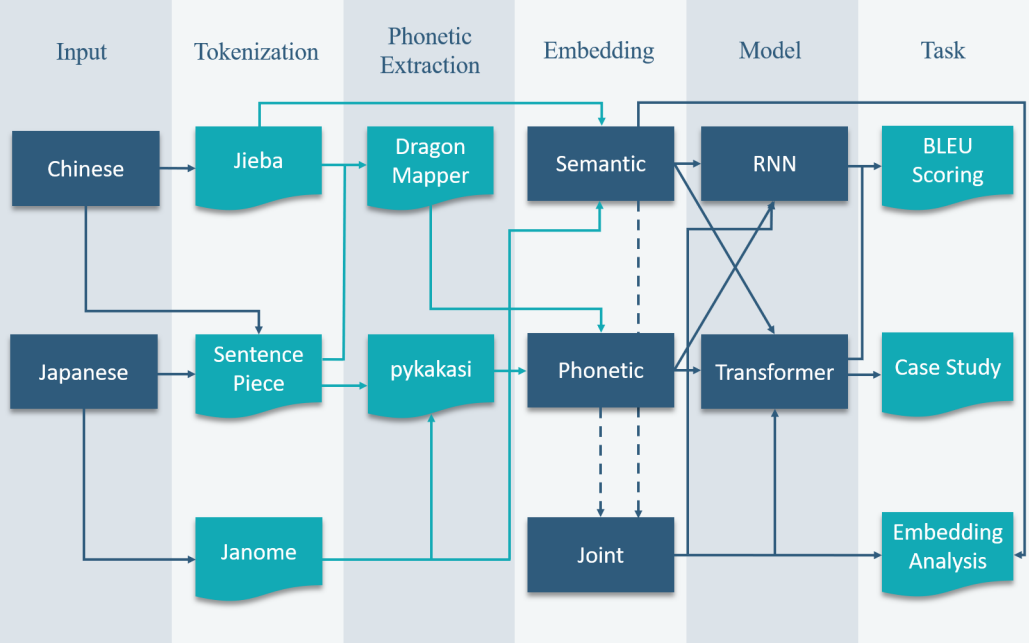


Figure 2.1: Structure of this paper.

Section 2.1 explains the tokenization methods we use to deconstruct sentences. Section 2.2 explains which phonetic extraction methods we use to transform plain sentences into phonetic encodings. Section 2.3 shows how we use Word2Vec as our algorithm to build and combine semantic and phonetic embeddings. Section 2.4 shows how we process data from the corpus to eliminate as much noise and reduce the size of the corpus as possible. Section 2.5 explains the details of two NMT models, which are the Attention-based Bi-GRU encoder-decoder Model and Transformer. Section 2.6 describes the methods we use to analyze the difference between semantic embeddings and joint semantic-phonetic embeddings.

2.1 Tokenization

This paper will examine the effectiveness of phonetic information under different tokenization methods. We use the `tokenizers` [1] library from the `HuggingFace` team as our main framework for tokenization. `SentencePiece` [2], `Jieba`, and `Janome` can be implemented as pre-tokenizers in the `HuggingFace Tokenizers`.

2.1.1 HuggingFace Tokenizers

The `HuggingFace Tokenizers` library has 5 components that allow users to customize their tokenization methods. Namely, normalizers, pre-tokenizers, models, post-processors, and decoders. Normalizers process an input string, performing tasks such as mapping to lower case and removal of spaces and symbols to normalize the text. Pre-tokenizers split an input string according to a set of rules, and this is the stage where we apply `SentencePiece`, `Jieba`, and `Janome`. Models are responsible for converting text into ids by using the rules learned in the corpus (e.g., WordPiece, BPE, Unigram). Post-processors help us to insert special tokens into the sentence, such as the start token *[BOS]* and the end token *[EOS]* in NMT tasks. Lastly, the job of Decoders is to reverse the ids to the original sentence.

2.1.2 Byte-Pair Encoding (BPE)

We use BPE as the model for merging Chinese and Japanese tokens. BPE builds a dictionary of all the words in the corpus and merges the most frequent words to generate new tokens until the maximum number of our dictionary is reached. We demonstrate the basic flow of BPE applied to Chinese in Tables 2.1 and 2.2.

Frequency	Vocabulary	Dictionary
5	区 __	__, 区, 地, 经, 济
7	地 区 __	
3	地 区 经 济 __	
6	经 济 __	

Table 2.1: A simple dataset for demonstrating BPE tokenization

Words are first tokenized by pre-tokenizers and loaded into the BPE vocabulary, with the underscore (__) representing the end of the words.

[1] https://github.com/huggingface/tokenizers
[2] https://github.com/google/sentencepiece

Total Frequency	Merge	New Dictionary
12	(区, __)	__, 区, 地, 经, 济, 区__
9	(济, __)	__, 区, 地, 经, 济, 区__, 济__
9	(经, 济__)	__, 区, 地, 经, 济, 区__, 济__, 经济__
7	(地, 区__)	__, 区, 地, 经, 济, 区__, 济__, 经济__, 地区__

Table 2.2: The process of merging tokens in BPE tokenization

The merge begins with 区 and __, which appear most frequently in the vocabulary. After merging, 区__will be added to the final dictionary and replaces (区 __) in the vocabulary. This process continues until the final dictionary reaches its maximum size.

2.1.3 SentencePiece

`SentencePiece` treats all text in the same Unicode format. It will escape the white space with a meta symbol '__' (U+2581). Therefore, the sentences in Chinese, Japanese, and English are considered to be in the same format, which achieving language independence.

`SentencePiece` is a purely data-driven method, which means it relies on the corpus to learn the tokenization. It is simple to implement `SentencePiece` in HuggingFace Tokenizers. First, Normalization Form Compatibility Composition (NFKC) normalizes the sentence, for example, by converting a symbol or text in the full-width form to a normalized form. Second, the Metaspace pre-tokenizer splits the sentence by white space and converts the white space into the '__' symbol. Last, BPE with dropout is applied to train with the corpus file. The dropout method will improve the robustness and accuracy.

```
from tokenizers.normalizers import NFKC
from tokenizers import Tokenizer, pre_tokenizers, decoders, trainers

tokenizer = Tokenizer(BPE(dropout=dropout, unk_token="[UNK]"))
tokenizer.normalizer = NFKC()
tokenizer.pre_tokenizer = pre_tokenizers.Metaspace(replacement="_",
    add_prefix_space=True)
tokenizer.decoder = decoders.Metaspace(replacement="_", add_prefix_space=
    True)

trainer = trainers.BpeTrainer(vocab_size=vocab_size)
tokenizer.train(corpus, trainer=trainer)
```

2.1.4 Jieba

`Jieba` is a famous Chinese tokenization Python library that has more than 26,000 stars on Github currently. `Jieba` uses a prefix dictionary to store the words and calculates the longest path from the Directed Acyclic Graph (DAG) created by the sentences and dictionary to return the most likely tokenized words. In addition, `Jieba` uses Hidden Markov Model (HMM) and Viterbi algorithm to tokenized the unknown words in the prefix dictionary. There are four states (B, M, E, S) in the HMM model, which represent the beginning, middle, end, and single (the character can represent a word) of a character. The Viterbi algorithm takes all the words as observation and outputs the states of each character from the input sentence.

A single line of code **`jieba.tokenize(sentence_str)`** can obtain the tokenized words from `Jieba`. We inserted it into Huggingface Tokenizers as a pre-tokenizer and trained the Chinese dictionary using BPE.

```
class JiebaPreTokenizer:
def jieba_split(self, i: int, normalized_string: NormalizedString) ->
    List[NormalizedString]:
    splits = []
    for _, start, stop in jieba.tokenize(str(normalized_string)):
        splits.append(normalized_string[start:stop])
    return splits

def pre_tokenize(self, pretok: PreTokenizedString):
    pretok.split(self.jieba_split)
```

2.1.5 Janome

Janome is a Japanese tokenization Python library that currently has 600 stars on Github. It applies the Japanese dictionary of another famous tokenization library, mecab [3]. For the methodology, Janome used the Minimal Acyclic Subsequential Transducer (MAST) as the internal dictionary data structure and the Viterbi algorithm to calculate the probability of tokenized words.

We inserted the Janome tokenizer as a pre-tokenizer to Huggingface Tokenizers and trained the Japanese dictionary using BPE.

```
ja_tokenizer = janome.tokenizer.Tokenizer()

class JanomePreTokenizer:
    def janome_split(self, i: int, normalized_string: NormalizedString)
        -> List[NormalizedString]:
        splits = []
        i = 0
        for token in ja_tokenizer.tokenize(str(normalized_string).strip()
            , wakati=True):
            splits.append(normalized_string[i: i+len(token)])
            i += len(token)
        return splits

    def pre_tokenize(self, pretok: PreTokenizedString):
        pretok.split(self.janome_split)
```

2.1.6 Comparison of Tokenizers

A sample of tokenized sentences using three tokenizers is listed in Table 2.3. Jieba and Janome can tokenize the words more precisely than SentencePiece. The better performance of Jieba and Janome over SentencePiece can be considered to be due to the small size and domain specificity of the corpus.

Input (Chinese)	平成１５年进行的研究内容如下。
SentencePiece	[__平成, __15, __年, 进行的研究, 内容, 如下, __。]
Jieba	[平成, 15, 年, 进行, 的, 研究, 内容, 如下, 。]
Input (Japanese)	平成１５年度に行なった研究内容は次の通りである。
SentencePiece	[__平成, __15, __年度, に行, なった, 研究, 内容, は次の通りである, __。]
Janome	[平成, 15, 年度, に, 行なっ, た, 研究, 内容, は, 次, の, 通り, で, ある, 。]

Table 2.3: Tokenized sentences using three tokenizers

[3] https://github.com/taku910/mecab

2.2 Phonetic Data Extraction

`Dragonmapper` [4] is used to extract information from Chinese sentences. `Pykakasi` [5] is used to extract information from Japanese sentences.

2.2.1 Dragonmapper

`Dragonmapper` is a Python library that can convert between Chinese characters, Bopomofo, Pinyin, and International Phonetic Alphabet (IPA). When we call **`dragonmapper.hanzi.to_pinyin(chinese_str)`**, it will convert the Chinese sentence to Bopomofo tokens based on *CC-CEDICT* [6] and *Unihan* [7] database. Table 2.4 shows the result of phonetic extraction in Chinese sentences. The extraction will be applied to the tokenized sentences to achieve better accuracy.

Input	数据详细计量对于节能推进的重要性
Tokenized	[数据, 详细, 计量, 对于, 节能, 推进, 的, 重要性]
Extracted	[ㄕㄨˋ ㄐㄩˋ, ㄒㄧㄤˊ ㄒㄧˋ, ㄐㄧˋ ㄌㄧㄤˋ, ㄉㄨㄟˋ ㄩˊ, ㄐㄧㄝˊ ㄋㄥˊ, ㄊㄨㄟ ㄐㄧㄣˋ, ㄓㄨㄥˋ ㄧㄠˋ ㄒㄧㄥˊ]
Input	分析项目是１９９７－２００１年是砷、镍、锰、铬、铍
Tokenized	[分析, 项目, 在, 1997, -, 2001, 年, 是, 砷, 、, 镍, 、, 锰, 、, 铬, 、, 铍]
Extracted	[ㄈㄣ ㄒㄧ, ㄒㄧㄤˋ ㄇㄨˋ, ㄕˋ ㄖˋ, 1997, -, 2001, ㄋㄧㄢˊ, ㄕˋ, ㄕˋ ㄣ, 、, ㄋㄧㄝˋ, 、, ㄇㄥˇ, 、, ㄍㄜˋ, 、, ㄆㄧ]
Heteronym Test	长大很快乐，音乐很长久
Tokenized	[长大, 很, 快乐, ,, 音乐, 很, 长久]
Extracted	['ㄓㄤˇ ㄉㄚˋ, ㄏㄣˇ, ㄎㄨㄞˋ ㄌㄜˋ, ,, ㄧㄣ ㄩㄝˋ, ㄏㄣˇ, ㄔㄤˊ ㄐㄧㄡˇ]

Table 2.4: Chinese Phonetic Extraction using Dragonmapper library

2.2.2 Pykakasi

`Pykakasi` is a Python library that implements algorithms in the `kakasi` [8] library. It is able to convert Kanji characters to Hiragana, Katakana, and Romaji through the `SKK` Japanese dictionary [9] and `UniDic` [10]. Table 2.5 shows the result of phonetic extraction in Japanese tokenized sentences.

[4] https://github.com/tsroten/dragonmapper
[5] https://github.com/miurahr/pykakasi
[6] https://cc-cedict.org/wiki/
[7] http://www.unicode.org/charts/unihan.html
[8] http://kakasi.namazu.org/index.html.en
[9] https://github.com/skk-dev/dict
[10] https://unidic.ninjal.ac.jp/

Input	技術以外にも，政策、法律、社会的側面の課題がある
Tokenized	[技術, 以外, に, も, ,, 政策, 、, , 法律, 、, , 社会, 的, 側面, の, 課題, が, ある]
Extracted	[ぎじゅつ, いがい, に, も, ,, せいさく, 、, , ほうりつ, 、, , しゃかい, てき, そくめん, の, かだい, が, ある]
Input	冠状動脈レントゲンでＴＩＭＩ血流３級が示された
Tokenized	[冠状, 動脈, レントゲン, で, TI, MI, 血流, 3, 級, が, 示さ, れ, た]
Extracted	[かんじょう, どうみゃく, れんとげん, で, TI, MI, けつりゅう, 3, きゅう, が, しめさ, れ, た]
Heteronym Test	一生，芝生で生ビールを飲む
Tokenized	[一生, ,, 芝生, で, 生, ビール, を, 飲む]
Extracted	[いっしょう, ,, しばふ, で, なま, びーる, を, のむ]

Table 2.5: Japanese Phonetic Extraction using Pykakasi library

2.3 Embedding

This paper uses Word2Vec as an embedding framework. Embedding is a distribution representation of words. Compared to using sparse vectors such as one-hot representations to represent words, embedding can represent words as high-dimensional vectors of real numbers. These vectors can be used to obtain similarity between words by computing distances such as cosine similarity.

First, We will explore the details of Word2Vec. Second, we explain how to train a general word embedding with semantics from a corpus. Third, we examine how to manipulate the phonetic information to build a phonetic embedding. Finally, we combine two embeddings to a joint embedding which can have both semantic and phonetic information.

2.3.1 Word2Vec

Word2Vec has two classic papers. The first one [Mikolov et al., 2013a] contributes two approaches to constructing an embedding, namely CBOW and Skip-gram. The second one [Mikolov et al., 2013b] contributes improved methods of embedding construction, such as Hierarchical Softmax and Negative Sampling. We will use Skip-gram and Negative Sampling to generate word embeddings in this paper.

Skip-gram model uses a neural network with hidden layers to predict the likelihood of context words $w_{t-2}, w_{t-1}, w_{t+1}, w_{t+2}$ of an input word w_t. The input word and context words will be paired, such as $(w_t, w_{t-2}), (w_t, w_{t-1})$ to perform supervised training. The model can be expressed as the following equation, where θ represents hidden states of the input word w_{center}, and $w_1, w_2, \cdots, w_C$ represents the context words of w_{center} within the window size C. The goal is to maximize the likelihood of the occurrence of context

words for each word w in the corpus by updating θ iteratively.

$$\underset{\theta}{\operatorname{argmax}}\, p\left(w_1, w_2, \ldots, w_C \mid w_{\text{center}}; \theta\right) \tag{2.1}$$

Skip-gram uses the softmax model to output a V-dimensional probability distribution for context word classification. We use the notation $p\left(w_{\text{context}} \mid w_{\text{center}}; \theta\right)$ in eq. 2.2 to denote the probability (a scalar value of range $[0, 1)$) of observing a context word given a center word. h is the hidden layer word vector for the input word w_{center} from the input embedding matrix, and W_{output} is a row vector for a context word from the output embedding matrix. $W_{\text{output}_{(c)}}$ corresponds to the row vector of the context word in (input, context) pair, while $W_{\text{output}_{(i)}}$ corresponds to the row vector of each word in the corpus of size V.

$$p\left(w_{\text{context}} \mid w_{\text{center}}; \theta\right) = \frac{\exp\left(W_{\text{output}_{(c)}} \cdot h\right)}{\sum_{i=1}^{V} \exp\left(W_{\text{output}_{(i)}} \cdot h\right)} \in \mathbb{R}^1 \tag{2.2}$$

The following equation shows that given a center word, the probability is computed V times to obtain its conditional probability distribution for each unique word in the corpus. W_{output} is the denominator in eq. 2.2.

$$\frac{\exp\left(W_{\text{output}} \cdot h\right)}{\sum_{i=1}^{V} \exp\left(W_{\text{output}_{(i)}} \cdot h\right)} \in \mathbb{R}^V = \begin{bmatrix} p\left(w_1 \mid w_{\text{center}}\right) \\ p\left(w_2 \mid w_{\text{center}}\right) \\ \vdots \\ p\left(w_V \mid w_{\text{center}}\right) \end{bmatrix} \tag{2.3}$$

In machine learning, it is a convention to minimize the loss function. The log is added to the equation to simplify and accelerate the calculation, and the formula is rewritten as minimizing a negative log-likelihood instead of maximizing a positive log-likelihood.

$$J\left(\theta; w^{(t)}\right) = -\log \prod_{c=1}^{C} \frac{\exp\left(W_{\text{output}\ (c)} \cdot h\right)}{\sum_{i=1}^{V} \exp\left(W_{\text{output}\ (i)} \cdot h\right)} \tag{2.4}$$

$$= -\sum_{c=1}^{C} \log \frac{\exp\left(W_{\text{output}\ (c)} \cdot h\right)}{\sum_{i=1}^{V} \exp\left(W_{\text{output}\ (i)} \cdot h\right)} \tag{2.5}$$

$$= -\sum_{c=1}^{C} \left(W_{\text{output}\ (c)} \cdot h\right) + C \cdot \log \sum_{i=1}^{V} \exp\left(W_{\text{output}\ (i)} \cdot h\right) \tag{2.6}$$

Figure 2.2 shows the Skip-gram model. we will learn word embedding vectors of size N given the vocabulary size V. The model uses one input word at a time for learning to predict one context word (output).

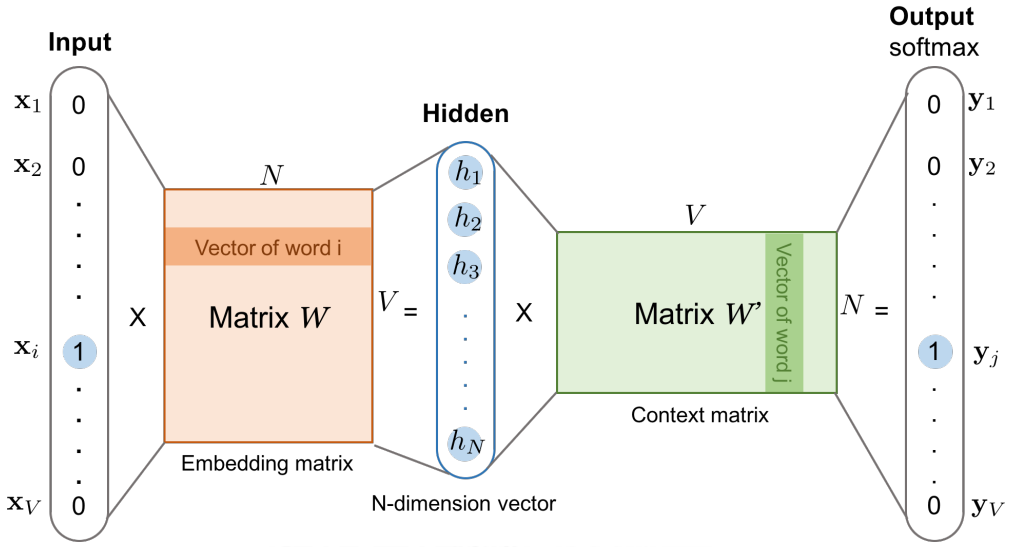


Figure 2.2: The Skip-gram Model[11]

Since the softmax model requires a lot of computational space and time, we also use negative sampling with the Skip-gram model. Negative sampling will convert the multi-classification problem that the model is designed to solve into a binary-classification problem. First, the method will randomly select k negative samples from the corpus that are irrelevant to the input word. k is a hyper-parameter, usually between 5 and 20. The model will update $(k+1) \times n$ parameters using the sigmoid function, where n is the dimension of the hidden layer h, and $+1$ accounts for the positive sample.

The probability that a word (c) appears within the context of the input word (w) can be defined as the following formula. The goal is to determine whether c is in the context window of w ($D = 1$) or not ($D = 0$) based on the input-context pairs (w, c).

$$p(D = 1 \mid w, c; \theta) = \frac{1}{1 + \exp\left(-\bar{c}_{\text{output}\,(j)} \cdot w\right)} \in \mathbb{R}^1 \tag{2.7}$$

2.3.2 Semantic Embedding

We use `Gensim` to implement Skip-gram with negative sampling from Word2Vec. We put the tokenized sentences generated by different tokenizers into `Gensim` to train the general word embedding with semantics. The following are the main settings of the parameters: *max_vocab_size* is 32000, *vector_size* is 300, *epoch_number*

[11] Source: https://lilianweng.github.io/lil-log/2017/10/15/learning-word-embedding.html

is 13, *window_size* is 5, the number of negative samples is 5, ignoring the words with total frequency lower than 5.

Some words may not be trained in the model, such as special tokens and low-frequency words. Therefore, the size of the trained embedding matrix and the tokenizer vocabulary is not identical, resulting in different coverage (Table 2.6). We calculate the mean and standard deviation of the whole embedding and randomly assign the empty word vectors using the normal distribution.

Language	SentencePiece	Jieba	Janome
Chinese	95% (30368/32000)	93% (29613/32000)	-
Japanese	97% (30940/32000)	-	93% (29801/32000)

Table 2.6: The coverage of semantic embedding in vocabulary

2.3.3 Phonetic Embedding

The training method for phonetic embedding is mostly the same as semantic embedding. First, we convert tokenized sentences into phonetic encodings using the phonetic extraction techniques described in Section 2.2. Second, we adopt the same training methods and parameters from semantic embedding. And last, we share the same vector of characters or words with the same pronunciation or spelling in the embedding matrix. Therefore, the coverage (Table 2.7) will be even higher than the semantic embedding. The empty vectors are also randomly assigned using the normal distribution with mean and standard deviation from the whole embedding.

Language	SentencePiece	Jieba	Janome
Chinese	99% (31753/32000)	97% (31068/32000)	-
Japanese	99% (31698/32000)	-	96% (30809/32000)

Table 2.7: The coverage of phonetic embedding in vocabulary

2.3.4 Joint Embedding

Several ways have been proposed to combine multiple embeddings, such as concatenation, training and blending separate embedding using the same model, and meta-embedding. Meta-embedding is a very vague term, but the core idea is to accept any kind of pre-trained embedding and fuse them into one (meta) embedding. Many studies on meta-embedding have been proposed [Kiela et al., 2018,Yin and Schütze, 2015,Muromägi et al.,

2017]. The methods of meta-embedding can range from very complicated to very straightforward. This paper uses a simple averaging method [Coates and Bollegala, 2018] to merge semantic and phonetic embedding.

2.4 Corpus Filtering

We define custom rules and apply alignment scores from `fast_align` [Dyer et al., 2013] to perform corpus filtering to reduce noise, dataset size, and resource burden during training in the NMT system. The original dataset had 672,315 sentence pairs, which were reduced to 557,685 using rules and 462,582 using alignment scores.

2.4.1 Pre-filtering rules

We apply the following rules to exclude the pairs of sentences:

1. Sentences that are too large in proportion to their length.
2. Sentences that are too short or too long.
3. Redundant (identical) sentences.
4. Sentences that cannot be correctly identified as Chinese or Japanese by the `fastText` identification model.
5. Sentences with more English and numeric characters than Chinese and Japanese.
6. Sentences that have more than one counterpart.

2.4.2 Scoring functions

`fast_align` is a tool based on *IBM alignment model 2* for training statistical machine translation and alignment model. The score is the probability of the source sentence given the target sentence and its alignment. The algorithm iteratively estimates the probability of being each other translation for source-target pairs and optimal alignment given the word-to-word translation probabilities.

In theory, the score should correlate with the degree of parallelism of the sentences. We remove the sentence pairs that are below a certain probability score threshold as shown in Table 2.8.

Alignment Score Range	Number of sentence pairs
$[-3, -81]$	173,100
$[-81, -160]$	287,882
$[-160, -238]$	93,134
$[-238, -317]$	3,540
$[-317, -395]$	29

Table 2.8: Distribution of Sentence Pair Alignment Scores

We sampled some sentence pairs corresponding to alignment scores in Table 2.9. We found low scores to be associated with longer sentences, and higher proportions of untranslated words and proper nouns.

Score	-52
Chinese	残留性化学物质的物质循环模型的构建和再利用·废弃物政策评价的应用
Japanese	残留性化学物質の物質循環モデルの構築とリサイクル・廃棄物政策評価への応用
Score	-151
Chinese	使用传统的轴组装工法的2层木制住宅,进行了室内甲醛浓度的测量,同时也研讨了上下楼层的甲醛浓度和换气量的关系。
Japanese	在来軸組工法木造2階建住宅を用いて,室内ホルムアルデヒド濃度の測定を行うとともに,上下階のホルムアルデヒド濃度と換気量の関係も検討した。
Score	-254
Chinese	当我尽快编写了PID控制程序并运行后发现,不仅温度不能很好地稳定,还出现了“不规则振荡”现象(重复出现大幅度偏离控制值的振动),控制彻底失败了。
Japanese	早速,PID制御のプログラムを書いて実行したところ,うまく温度が安定化しないどころかいわゆる“ハンチング”現象(制御値から大きく外れた振動を繰り返す)を起こしてしまい,見事,制御に失敗した
Score	-338
Chinese	(’关于葛根、黄芩、黄连汤之证,在『伤寒论』34条中叙述:“太阳病,桂枝证,医反下之,利遂不止,脉促者,表未解也,喘而汗出者,葛根黄芩黄连汤主之”。
Japanese	’葛根黄芩黄連湯証について,『傷寒論』34条に,「太陽病,桂枝の証,医反ってこれを下し,利遂に止まず,脈促のものは,表いまだ解せざるなり,ぜんして汗出づるものは,葛根黄芩黄連湯これを主る」という。

Table 2.9: Samples of Sentence Pairs with Alignment Scores

2.5 NMT Model

We will train and evaluate the NMT tasks in two classical models, the attention-based Bi-GRU model proposed by [Bahdanau et al., 2014], and the Transformer model proposed by [Vaswani et al., 2017]. We will explain the structure of models in this section. Furthermore, The use of parameters is presented in Section 3.3, the metric is explained in Section 3.4, the results are listed in Section 3.5, and the case study is conducted in Section 4.1.

The two models and the following illustrative diagrams are derived from *bentrevett/pytorch-seq2seq* [12].

[12]https://github.com/bentrevett/pytorch-seq2seq

We utilized their source code, but optimized some details, such as accelerating the decoding efficiency of the decoder.

2.5.1 Attention-based Bi-GRU encoder-decoder Model

This model has three main components, namely Encoder, Attention, and Decoder. We used the `nn.Module` provided by PyTorch to implement three components and PyTorchLightning to combine and train the entire model.

Encoder

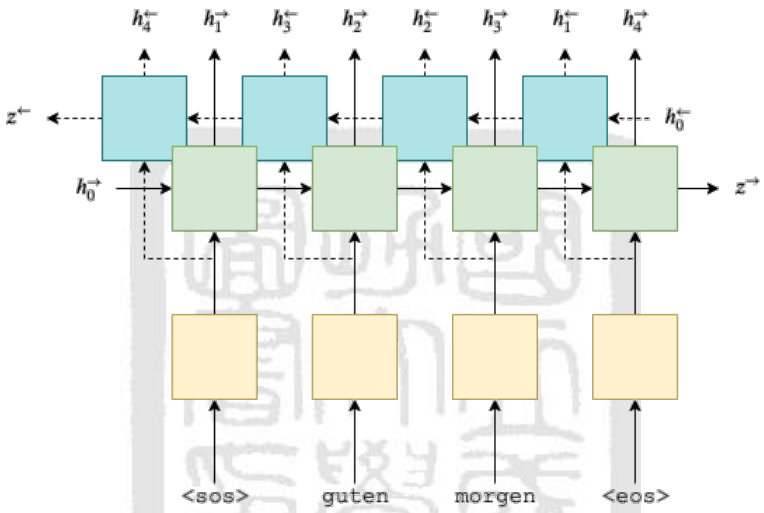


Figure 2.3: The Encoder of Attention-based Bi-GRU Model

The structure of the encoder (Figure 2.3) can be represented by the following equations. In eq. 2.8, we enter the embedded tokens into the bi-directional GRU model to calculate the forward and backward hidden states h.

$$h_{\overrightarrow{t}} = \overrightarrow{\text{EncoderGRU}}(\text{emb}(x_{\overrightarrow{t}}), h_{\overrightarrow{t-1}}), h_{\overleftarrow{t}} = \overleftarrow{\text{EncoderGRU}}(\text{emb}(x_{\overleftarrow{t}}), h_{\overleftarrow{t-1}}) \tag{2.8}$$

The output H (in eq. 2.9) represents the combination of hidden states in the last layer of the model and will be used to calculate the attention.

$$\text{output } H = \{h_1, h_2, \cdots, h_T\}, h_i = [h_{\overrightarrow{i}}; h_{\overleftarrow{i}}] \text{ for } i \in \{1, \cdots T\} \tag{2.9}$$

The obtained hidden states $\overrightarrow{z}$ and $\overleftarrow{z}$ (in eq. 2.10) will be used as the initial hidden states of the decoder.

$$\overrightarrow{z} = h_{\overrightarrow{T}}, \overleftarrow{z} = h_{\overleftarrow{T}} \tag{2.10}$$

Since the decoder is not bi-directional, we enter the hidden states $\overrightarrow{z}$ and $\overleftarrow{z}$ into a linear layer g and activation function $tanh$ (in eq. 2.11) to get the condensed context vector z, also called s_0.

$$z = \tanh(g(\text{cat}(\overrightarrow{z}, \overleftarrow{z}))) = s_0 \tag{2.11}$$

Attention

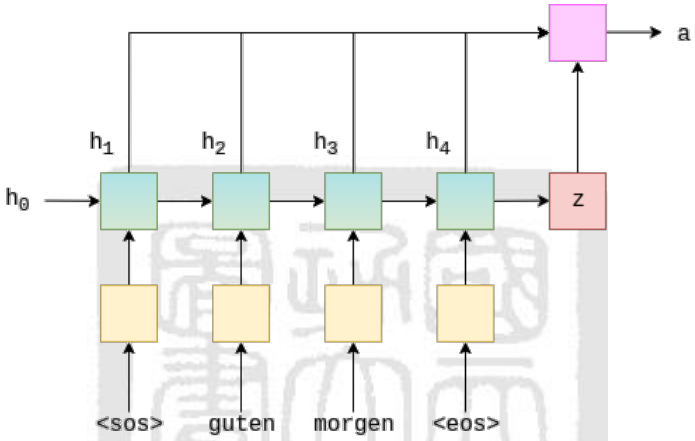


Figure 2.4: The Attention of Attention-based Bi-GRU Model

The attention layer (Figure 2.4) generates an array of the same length as the source sentence, representing how much attention was given to each token in the source sentence when predicting the next token $\hat{y}_{t+1}$. The attention layer is a linear layer, and each time it takes the previous hidden state s_{t-1} of the decoder (the first time is the z or s_0 from encoder), and the output H from encoder to compute an energy value E_t with $tanh$ activation function (in eq. 2.12).

$$E_t = \tanh(\text{attn}(s_{t-1}, H)) \tag{2.12}$$

Because the shape of the energy value E_t is (*src_len* [13], *hid_dim* [14]), it will enter into a linear layer v with shape (*hid_dim*, 1) (in eq. 2.13) to get the attention sequence $\hat{a}_t$ with shape (*src_len*). The parameters learned in the linear layer v can be imagined as the weight of the energy value E_t for each token in the source sentence.

[13] The length of source sentence
[14] The dimension of hidden layer

$$\hat{a}_t = v(E_t) \tag{2.13}$$

The attention sequence $\hat{a}_t$ will be passed to the softmax layer (in eq. 2.14) so that all the attention values add up to 1. The process combines with the mask to zero out the attention of the *[PAD]* tokens.

$$a_t = \text{softmax}(\hat{a}_t) \tag{2.14}$$

Decoder

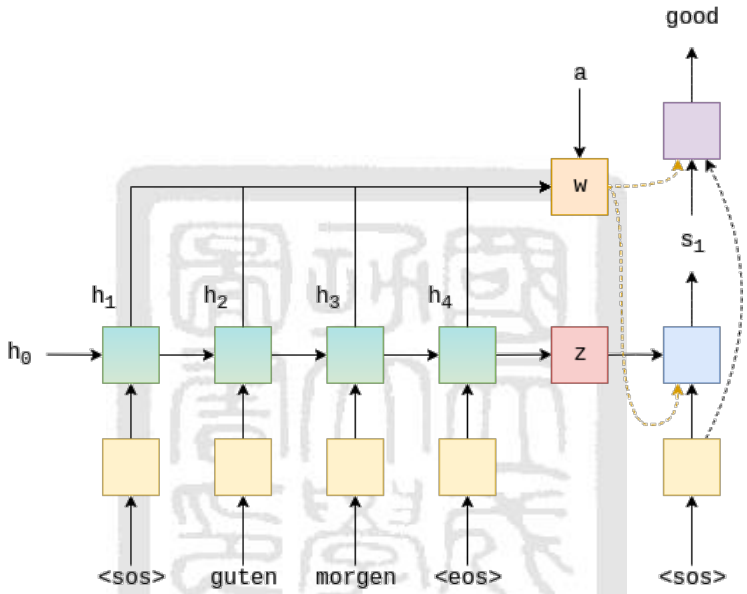


Figure 2.5: The Decoder of Attention-based Bi-GRU Model

The decoder (Figure 2.5) first uses the previous hidden state s_{t-1} and the encoder output H to compute the attention vector a_t for the current time-step t. The attention vector a_t will perform a matrix multiplication with H (in eq. 2.15) to produce the weighted sum w_t that represents the attention to each token in the source sentence.

$$w_t = a_t H \tag{2.15}$$

The decoder then takes the embedded target token $d(y_t)$, the weighted source vector w_t, and the previous hidden state s_{t-1} into the one-directional GRU model to compute the current hidden state s_t (in eq. 2.16).

$$s_t = \text{DecoderGRU}(d(y_t), w_t, s_{t-1}) \tag{2.16}$$

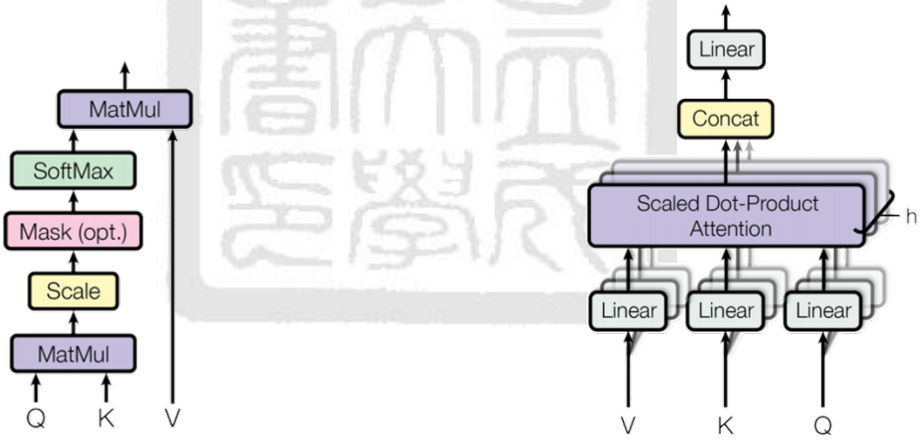
Finally, the decoder uses $d(y_t)$, w_t, s_t and a linear layer f to predict the next token $\hat{y}_{t+1}$ (in eq. 2.17). We will apply the teacher forcing technique, which has a 50% chance of using the predicted token and a 50% chance of using the ground truth token as the next target input token.

$$\hat{y}_{t+1} = f(d(y_t), w_t, s_t) \tag{2.17}$$

2.5.2 Transformer

The Transformer is an attention-driven model that uses the attention mechanism to process data in either encoder or decoder. We first describe the attention mechanism and then introduce it to the encoder and decoder.

Attention



(a) Scaled Dot-Product Attention

(b) Multi-Head Attention

Figure 2.6: The Multi-head Attention of Tranformer

In the Transformer, the attention layer requires three elements, namely query (Q), key (K), and value (V), as shown in Figure 2.6a. To begin with, the energy value is computed by taking the dot product between query Q and key K. It is then scaled by the square root of *head_dim* [15] ($\sqrt{d_k}$) to avoid gradient vanishing. After that, the energy value is fed into the softmax layer to get the attention sequence and finally multiplied by the value V to get the weighted sum (in eq. 2.18).

[15] The hidden state dimension of each head in Multi-Head Attention

$$\text{Attention}(Q, K, V) = \text{Softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V \tag{2.18}$$

The Transformer model splits query Q, key K, and value V into multiple heads with smaller dimensions and trains them separately (Figure 2.6b). They are divided into heads by the linear layers (W_i^Q, W_i^K, W_i^V). Each with a size equal to the dimension of the hidden state divided by the number of heads.

$$\text{head}_i = \text{Attention}(QW_i^Q, KW_i^K, VW_i^V) \tag{2.19}$$

After all the heads pass through the attention layer (in eq 2.19), they are re-concatenated and pass through the last linear layer W^O to get the final weighted sum (in eq 2.20).

$$\text{MultiHeadAttention}(Q, K, V) = \text{Concat}(\text{head}_1, \cdots, \text{head}_h)W^O \tag{2.20}$$

Encoder and Encoder Layers

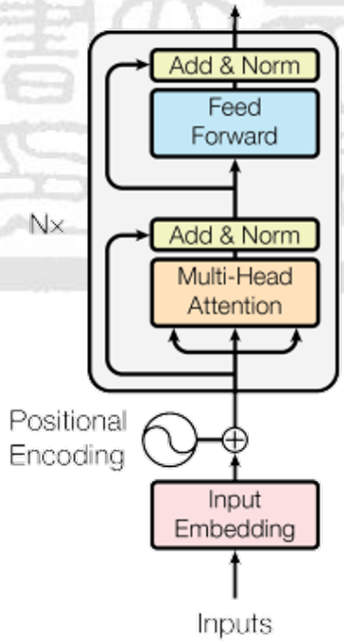


Figure 2.7: The Encoder of Transformer

Unlike the attention-based Bi-GRU model, the input tokens will generate the corresponding number of context vectors directly after passing through the encoder. For example, if the input has 5 tokens $X = (x_1, x_2, x_3, x_4, x_5)$, the encoder will generate 5 context vectors $Z = (z_1, z_2, z_3, z_4, z_5)$ for the decoder to

predict.

The tokens are passed to the word embedding and retrieved the position information by the positional embedding. The size of the positional embedding is equal to the length of the sentence; if the sentence length is 100, then the size of the positional embedding will be 100. The word embedding is added to the positional embedding to obtain the text and position information. However, the word embedding is multiplied by a scaling factor $\sqrt{d_{\text{model}}}$, which is used to reduce the variance of the embedding.

An encoder will have multiple encoder layers. Each of them contains an attention layer and a position-wise feedforward layer (which can be considered as a large linear layer). In addition to these two components, there are also residual connection and layer normalization, which aim to generalize all parameters and prevent gradients from disappearing, making it easier to train networks with a large number of parameters. Because the attention is calculated from the source sentence and itself, it is also called self-attention.

Decoder and Decoder Layers

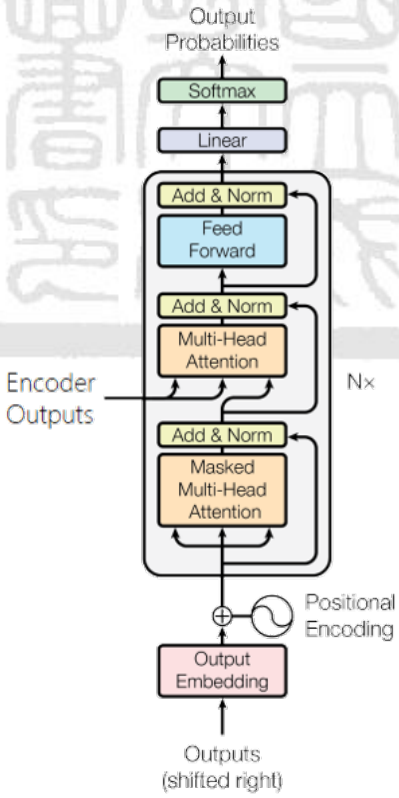


Figure 2.8: The Decoder of Transformer

The architecture of the decoder is almost identical to that of the encoder. The decoder feeds the embedded-target-tokens into several decoder layers, and predicts the next target token $\hat{y}$ with source context vector Z.

Following this, it computes the loss by the predicted token $\hat{y}$ and the answer token y, and updates all the weights of the network.

There are three main components in a decoder layer. First, the decoder has a masked multi-head attention layer, in which the target sentence will perform self-attention with itself and a special mask to prevent the decoder from seeing the future words in advance. Second, the multi-head attention layer is calculated with the context vector from the masked multi-head attention layer as the query Q and the encoder context vector as the key K and value V. Third, the last position-wise feedforward layer receives the results of the previous layer and outputs the final context vector to predict the next token $\hat{y}$.

2.6 Embedding Analysis

We will examine the effectiveness of joint semantic-phonetic embedding using four methods implemented in `Gensim`. We expect joint semantic-phonetic embedding to retain the word similarity, analogical reasoning, and outlier detection capabilities that semantic embedding possesses. We are also interested in whether the joint embedding can enhance the impact on homonyms and heteronyms. All examination findings will be discussed in Section 4.2.

2.6.1 Word Similarity

The purpose of word similarity is to evaluate the closeness and relatedness of any two words in the given embedding. This paper measures the word similarity by calculating the cosine distance, which is equal to 1 - cosine similarity; the higher the similarity, the shorter the distance. The cosine similarity estimates the relatedness between two vectors by measuring the cosine of the angle between them (in eq. 2.21). The cosine similarity of two vectors with the same direction is 1, the angle of 90° is 0, and the opposite direction is -1.

$$\text{similarity } = \cos(\theta) = \frac{A \cdot B}{\|A\|\|B\|} = \frac{\sum_{i=1}^{n} A_i \times B_i}{\sqrt{\sum_{i=1}^{n} (A_i)^2} \times \sqrt{\sum_{i=1}^{n} (B_i)^2}} \tag{2.21}$$

This paper uses similarity distance to compare the distance of synonyms under different embeddings to examine whether joint embedding can preserve semantic properties.

2.6.2 Analogical Reasoning

Analogical reasoning is to evaluate the ability of embeddings to apply arithmetic operations to reasoning. Given a set of three words, a, $a*$ and b, the task requires identifying the word $b*$, i.e., the relation b:$b*$ is the same as the relation a:$a*$. For example, when a = Tokyo, $a*$ = Japan, b = Taipei, the embedding need to be able to predict $b*$ = Taiwan. The analogical reasoning can be shown as the following eq. 2.22.

$$a - a* + b = b* \tag{2.22}$$

2.6.3 Outlier Detection

Outlier Detection tests the ability of embeddings to perform the task of word categorization. The embeddings are required to identify the semantically anomalous word from a defined word cluster. For example, when the word cluster contains "apple, bird, cat, dog", the embedding needs to predict the target word as "apple".

2.6.4 Homonym and Heteronym

This task is an extension of word similarity tasks. We expect the joint embedding with additional phonetic information reduces the impact of possible noise on the NMT system (i.e. the impact of homonyms, easily mistyped because they share the same pronunciation). The evaluation uses Pearson correlation to examine the relationship between homonyms. We expect the correlation coefficient will be higher in the joint embedding than in the semantic embedding. At the same time, we expect that phonetic information will help the joint embedding to increase the similarity distance between heteronyms. Heteronyms refer to words that have identical writing but different pronunciation, spelling, and meaning. Increasing the distance between heteronyms means strengthening the semantics between words.

Chapter 3

Experiment and Result

This chapter explains the details of the experiments. It includes the use of datasets (Section 3.1), the experimental environment (Section 3.2), the parameters of the NMT model (Section 3.3), and the metrics (BLEU score) of the NMT model (Section 3.4). Lastly, the BLEU scores of the translation system based on different tokenizations and word embeddings are shown in Section 3.5.

3.1 Dataset

Our datasets are acquired from the Asian Scientific Paper Excerpt Corpus (ASPEC) [Nakazawa et al., 2016] provided by the Japan Science and Technology Agency (JST) and the National Institute of Information and Communications Technology (NICT). ASPEC consists of a Japanese-English corpus (ASPEC-JE) with 3 million parallel sentences and a Japanese-Chinese corpus (ASPEC-JC) with 680,000 pairs.

We have selected ASPEC-JC as the dataset for our zh-ja NMT system, and the structure of the dataset is shown in Table 3.1. ASPEC-JC is constructed by manually translating the excerpts of Japanese scientific papers into Chinese. Papers used for translation are derived from the Japan Science and Technology Agency (JST) or the Japan Science and Technology Information Aggregator, Electronic (J-STAGE).

Data Type	File Name	Number of sentences
Train	train.txt	672,315
Validation	dev.txt	2,090
Validation-Test	devtest.txt	2,148
Test	test.txt	2,107

Table 3.1: ASPEC-JC dataset

3.2 Environment

We use `PyTorch` [1] and `PyTorch-Lightning` [2] [Falcon, 2019] as the main framework for constructing the entire NMT model (Section 3.2.1), as well as Weights & Biases (`wandb`) [3] [Biewald, 2020] for logging models, metrics, and dataflow (Section 3.2.2).

3.2.1 PyTorch and PyTorch Lightning

`PyTorch` has been widely used in academic research and production. We use `PyTorch` as a framework for processing data and designing the components (i.e., encoder, attention, decoder) of the attention-based Bi-GRU Model and Transformer.

`PyTorch-Lightning` offers many advantages as a wrapper for `PyTorch`: Dataflow can be integrated more efficiently in **`LightningDataModule`**; the main source code, including the components designed by `PyTorch`, can be integrated into **`LightningModule`**; using **`Trainer`** can directly utilize the **`LightningDataModule`** and **`LightningModule`**, and help us use the correct engineering code such as gradient update, data transfer between CPU and GPU. In addition, `PyTorch-Lightning` provides a lot of plugins to simplify research, such as multi-GPU or TPU training, checkpointing and logging of models, implementing 16-bit precision, early stopping, finding learning rate and batch size, etc.

3.2.2 Weights and Biases (wandb)

`Wandb`, similar to `Tensorboard` [4], is a powerful tool for logging model training and is used by OpenAI [5], the company designed GPT-3. `Wandb` can visualize all training metrics and system performance, record hyperparameters used in the experiments for easy replicating, integrate dataset in the cloud to facilitate the advantages of cross-platform. In the experiments, we use the **`WandbLogger`** provided by `PyTorch-Lightning` to integrate `wandb` and record the experimental results of all models, which will be presented in Section 3.5.

[1] https://pytorch.org/
[2] https://www.pytorchlightning.ai/
[3] https://wandb.ai/site
[4] https://www.tensorflow.org/tensorboard
[5] https://wandb.ai/site/articles/why-experiment-tracking-is-crucial-to-openai

3.3 Parameter

We implement eight (2×4) experiments for attention-based Bi-GRU Model and Transformer experiments. The experiments use two different methods for tokenization. one using SentencePiece for both Chinese and Japanese, and the other using Jieba and Janome for Chinese and Japanese respectively. Every experiment validates four scenarios, which are the model without any pre-trained embedding (i.e., the baseline), the model using only semantic embedding, the model using only phonetic embedding, and the model using joint semantic-phonetic embedding.

Some parameters are identical in all experiments. The dataset has 50,000 sentence pairs which filtered (described in Section 2.4) and sampled by Python `random` module. The dictionary size for tokenizer and embedding is fixed to 32,000 and the dimension of embedding is fixed to 300. The other hyperparameters are listed in Table 3.2.

Hyperparameters	Attention-based Bi-GRU	Transformer
dropout	0.1	0.1
hidden_dim (pf_dim)	512	512
learning rate	7e-4	5e-4
precision	16	32
attention heads	-	6
layers	1	3

Table 3.2: The hyperparameters

Catastrophic forgetting [Kirkpatrick et al., 2017] may occur during training with pre-trained embedding. In other words, the gradients in the first few batches will drastically change the embeddings and lose the initial meaning from vectors completely. We overcome the problem by freezing the weights of embeddings in the first n epochs, and fine-tuning the embeddings in the subsequent epochs. Therefore, n is also a hyperparameter of the attention-based Bi-GRU Model and Transformer. We apply $n = 3$ to RNN and $n = 1$ to Transformer to achieve the best results.

3.4 Metric

We evaluated our zh-ja NMT system using the BLEU (Bilingual Evaluation Understudy) score [Papineni et al., 2002]. The BLEU score is a number from 0 to 100 that measures the similarity between translated sentences from model and high-quality reference translations. A score of 0 means there is no overlap between machine

translation and reference translation, and a score of 100 means a complete overlap between the two. A common interpretation of the BLEU score can be referred to Table 3.3.

BLEU Score	Interpretation
< 10	Almost useless
$10 - 19$	Hard to get the gist
$20 - 29$	The gist is clear, but has significant grammatical errors
$30 - 40$	Understandable to good translations
$40 - 50$	High quality translations
$50 - 60$	Very high quality, adequate, and fluent translations
> 60	Quality often better than human

Table 3.3: Interpretation of the BLEU score[6]

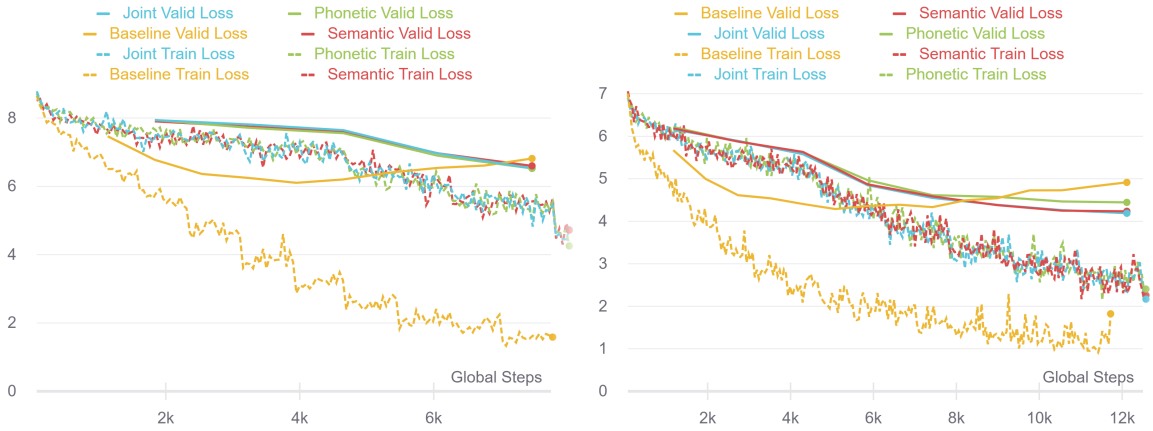
Mathematically, the BLEU score is defined as eq. 3.1 and eq. 3.2. The formula consists of two parts: the brevity penalty and the n-gram overlap. The brevity penalty penalizes the sentences that were translated too short, which were easier to score higher because of the lower denominator (in eq. 3.2) calculated by n-gram. The n-gram overlap counts the number of overlaps between the translated sentences and the corresponding unigrams, bigrams, trigrams, and four-grams in the reference translation ($\text{Count}_{\text{clip}}$). Unigrams account for the adequacy and longer n-grams account for the fluency of the translation.

$$\text{BLEU} = \underbrace{\min\left(1, \exp\left(1 - \frac{\text{reference-length}}{\text{output-length}}\right)\right)}_{\text{brevity penalty}} \underbrace{\left(\prod_{i=1}^{4} \text{precision}_i\right)^{1/4}}_{\text{n-gram overlap}} \tag{3.1}$$

$$\text{precision}_n = \frac{\sum_{\text{n-gram} \in C} \text{Count}_{\text{clip}}(\text{n-gram})}{\sum_{\text{n-gram}' \in C'} \text{Count}(\text{n-gram}')} \tag{3.2}$$

Here are several considerations for evaluating translation models using the BLEU score. First, BLEU is a corpus-based metric that requires evaluating multiple sentences (corpus) at once; the score for individual sentences will lose some meaning; in other words, statistics are accumulated across the sentences when computing the score. Second, BLEU cannot distinguish the errors between the function words and the content words; dropped function words such as “a”, “the”, and “of” carry the same penalties as important content words that are mistranslated or lost. Third, BLEU is not good at capturing the meaning and grammaticality of sentences; namely, the impact of single words such as “not” that can dramatically change the semantics, and long-range dependencies of distances greater than the length of the n-gram will be underestimated.

[6]Source: https://www.cs.cmu.edu/ alavie/Presentations/MT-Evaluation-MT-Summit-Tutorial-19Sep11.pdf



(a) Results using SentencePiece

(b) Results using Jieba & Janome

Figure 3.1: Train and Validation Loss of Attention-based Bi-GRU Model

3.5 Result

All NMT models are trained with a dictionary size of 32,000, 300-dimensional embedding, and sampled dataset that has 50,000 sentence pairs. We first examine the losses and BLEU scores produced by different tokenization methods and embeddings on the two models. Although there is no exact correspondence between the loss and the BLEU score, the loss is still a useful metric for evaluating the performance of models such as detecting overfitting. Finally, we look into the result using the best model and the complete, filtered dataset (462,582 sentence pairs) and compare it with other studies.

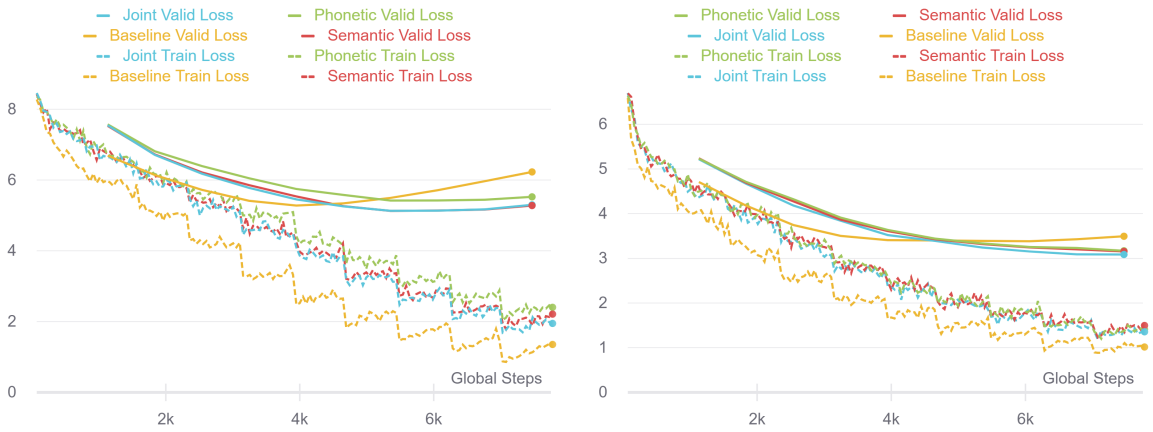
3.5.1 Attention-based Bi-GRU Model

Figure 3.1 shows the losses of using the two tokenization methods on the attention-based Bi-GRU model respectively. In both Figure 3.1a and 3.1b, it is observed that using any pre-trained embedding can effectively resist the overfitting that occurs in the baseline. Moreover, the loss of all three embeddings in Figure 3.1a is similar, but the loss of joint embedding in Figure 3.1b is the lowest.

Table 3.4 shows the BLEU scores for all combinations on the attention-based Bi-GRU model. Using `SentencePiece` as a tokenizer scores about 4 to 5 lower than the average score of `Jieba` + `Janome`. This is reasonable because SentencePiece is a data-driven tokenizer, and the lack of data in the small corpus has largely hampered its ability to train embedding and NMT systems. The use of joint embedding successfully outperforms baseline and the other two embeddings based on both tokenization methods. Furthermore, it improves performance over baseline by about 2 BLEU points when using the `Jieba` and `Janome` tokenizers

Tokenization	Baseline	Semantic	Phonetic	Joint
SentencePiece	21.63	21.66	21.32	**22.33**
Jieba + Janome	25.16	26.71	26.18	**27.05**

Table 3.4: BLEU Scores of Attention-based Bi-GRU Model



(a) The results using SentencePiece

(b) The results using Jieba & Janome

Figure 3.2: Train and Validation Loss of Transformer

3.5.2 Transformer

Figure 3.2 shows the losses of using the two tokenization methods on the Transformer. The addition of embeddings can resist overfitting in the Transformer as well as in the attention-based Bi-GRU Model. The semantic and joint embedding perform similarly when using `SentencePiece` as the tokenization method, but both are better than the embedding that only applies phonetic information (Figure 3.2a). On the other hand, the joint embedding is a bit better than semantic and phonetic embedding with `Jieba` and `Janome` (Figure 3.2b).

Table 3.5 shows the BLEU scores for all combinations on the Transformer. The joint embedding obtains the highest scores with either `SentencePiece` or `Jieba` + `Janome` as the tokenization method. In the case of using `Jieba` and `Janome`, the joint embedding improves performance by 1 and 3 BLEU points respectively over the semantic embedding and baseline system.

Tokenization	Baseline	Semantic	Phonetic	Joint
SentencePiece	24.32	25.72	23.48	**26.44**
Jieba + Janome	29.31	31.23	30.90	**32.48**

Table 3.5: BLEU Scores of Transformer

3.5.3 Best Model

We decided to use `Jieba` + `Janome` as the base tokenization method to train and re-evaluate the four scenarios using the complete and filtered dataset with 462,582 sentence pairs. The results are shown in Table 3.6. The use of joint embedding still achieves the best BLEU score, which is 0.35 higher than the baseline system.

Tokenization	Baseline	Semantic	Phonetic	Joint
Jieba + Janome	52.78	52.83	53.04	**53.13**

Table 3.6: BLEU Scores of Transformer with Complete, Filtered Dataset

We also used the baseline system of Workshop on Asian Translation 2020 (WAT) [Nakazawa et al., 2020] as a benchmark for evaluation, although we did not apply through its official submission system. For the ASPEC-JC task, WAT 2020 designed the baseline system using `OpenNMT` [7], BPE, and attention mechanisms. The BLEU score and three different tokenization methods are used for evaluating Chinese to Japanese translation. As shown in Table 3.7, our best model (53.13) outperforms the baseline system of WAT 2020 by about 6 BLEU points with the enhancement of tokenization, embedding, corpus filtering, etc.

	Juman 7.0	KyTea 0.4.6	Mecab 0.996
Baseline	46.87	47.30	47.00

Table 3.7: BLEU Scores for the WAT2020 Baseline System

[7] https://github.com/OpenNMT/OpenNMT

Chapter 4

Discussion

In Section 3.5, we have seen that the best performance of the zh-ja NMT system can be achieved by utilizing Jieba & Janome as the tokenization method and the joint embedding as an additional feature in the Transformer model. Based on the promising performance of the models, we will further examine the translation results produced by the best model and try to explain the reasons for its improved translation in Section 4.1. In addition, we will compare the joint embedding with the semantic embedding in Section 4.2 to further analyze the advantages that the joint embedding provides over traditional embedding.

4.1 Case Study

We compare the translation results using joint embedding with those using semantic embedding. The results show that the use of joint embedding is superior to that of semantic embedding in four aspects.

Source	使用有机溶剂的溶剂提取法作为物质的分离手段被广泛利用。
Target	有機溶媒を用いた溶媒抽出法は物質の分離手段として広く利用されている。
Semantic	有機溶媒を用いた溶媒抽出法は物質の分離手段として広く用いされている。
Joint	有機溶媒を用いた溶媒抽出法は物質の分離手段として広く利用されている。
Source	在本章中,将分布式组件间进行通信的中间件与Dragon进行比较。
Target	本章では,分散コンポーネント間通信を行うミドルウェアとDragonを比較する.
Semantic	本章では,分散コンポーネント間でを行うミドルウェアとDragonを比較する.
Joint	本章では,分散コンポーネント間通信を行うミドルウェアとDragonを比較する.
Source	LearnAR从背景知识B和观测结果O中获得行动规则的集合γ的集合H。
Target	LearnARは,背景知識Bと観測結果Oより行動規則の集合γの集合を獲得する.
Semantic	LearnARは背景背景知識Bと観測結果Oから行動ルールの集合γの集合H獲得する.
Joint	LearnARは,背景知識Bと観測結果Oから行動規則の集合γの集合H獲得する.

Table 4.1: Case Study - Retention of correct words

Table 4.1 compares the results of the two translations in terms of their ability to preserve correct words. In the first example, “利用され” (be used) is correctly translated by the joint embedding, while semantic embedding produces the ungrammatical string “用いされ”, which is perhaps an unfortunate combination of

“利用され” and an alternative translation “用いられ”. In the second example, “分散コンポーネント間通信” (communication between distributed components) is correctly translated by the joint embedding, and the semantic embedding omits “通信” (communication). The third example demonstrates that both the semantic and joint embedding can correctly retain the symbol H even though it is missing from the target sentence. However, the joint embedding translation is overall closer to the target. At least for these three examples, the joint embedding translation is consistently more fluent than those of the semantic embedding.

Source	构成这些乐曲时在可能范围里保证变化丰富。
Target	これらの楽曲は,可能な範囲でバリエーションが豊かになるように構成されている.
Semantic	これらの楽曲を,豊かな範囲で変化が豊かになる.にしされている.
Joint	これらの楽曲を,可能な範囲でバリエーションが豊かになるように構成されている.
Source	铝焊接烟气的特性
Target	アルミニウム溶接ヒュームの物性
Semantic	アルミニウム溶接煙の特性
Joint	アルミニウム溶接ヒュームの特性

Table 4.2: Case Study - Utilization of English loanwords

Table 4.2 demonstrates the ability of joint embedding to utilize English loanwords. Loanwords from English and occasionally German or other western languages play an important role in modern Japanese and occur with significant frequency in scientific-technical writing. Thus a natural translation of Chinese into Japanese is likely to include some of these loanwords, easily spotted due to the convention of using katakana to write them. Here we show a couple of examples in which the joint embedding uses such a loanword while the semantic embedding does not. In the first example, we focus on the translation of the “变化” in the Chinese “变化丰富”. Which is translated as “バリエション” by the target and joint embedding, but as “変化” by the semantic embedding. “变化” is a common word shared by Chinese and Japanese, which depending on context might be translated into English as: “change”, “alteration”, “inflection”, or “variation”. “variation” happens to be a word which has been borrowed into the Japanese language using the katakana spelling “バリエーション”. Although partially overlapping in meaning with “変化”, “バリエーション” is more specific and in particular tends to have a positive meaning in Japanese, which matches the positive connotation of “丰富” (abundant) in the source. These factors explain why “バリエーション” was selected in the target, instead of directly copying “変化” from the source. The second example involves the translation of “烟气” to either “ヒューム” or “煙”. “ヒューム” is the loanword “fume” borrowed from English, while “煙” is the general Japanese word for

"smoke". In an informal speech, a Japanese speaker might use "煙" to describe any smoke-like phenomenon (even steam), but the target sentence selects "ヒューム", as a more specific and technically correct word in the source sentence context of welding. The joint embedding also selects "ヒューム", but the semantic embedding selects "煙". Interestingly, in both of these examples the semantic translation more or less directly copies the Chinese word to produce a Japanese translation which is arguably not far off the mark semantically, but the word choice of the joint embedding is much preferred.

Source	RMCPRGIGCenter和RMCPRGIGTransceiver使用C语言安装,在二者间的通信中和RemoteGIG同样使用TCP/IP上的RMCP。
Target	RMCPRGIGCenterとRMCPRGIGTransceiverはC言語で実装し,両者間の通信にはRemoteGIGと同様にTCP/IP上のRMCPを用いた.
Semantic	RMCPRGIGCenterとRMCPRGIGTransceiはC言語を実装し,両者間通信通信でTCPTCPと同様にTCP/IPでRMCPを用いるて.
Joint	RMCPRGIGCenterとRMCPRGIGTransceiverをC言語を実装し,両者間の通信においてはRemoteGIGと同様にTCP/IP上でRMCPを用いた.
Source	另一方面,GUIServer和GUIClient是用Java语言进行安装。
Target	一方,GUIServerとGUIClientはJava言語で実装した.
Semantic	一方,GUISververやGUIIClientはJava言語で実装した.
Joint	一方,GUIServerとGUIClientはJava言語で実装した.

Table 4.3: Case Study - Preservation of English acronyms and technical jargon

Table 4.3 demonstrates the ability of joint embedding to cope with Chinese or Japanese text including English acronyms and technical jargon verbatim. When English words are present, semantic embedding often produces strange translations. It will lose English words, generate the wrong English words and grammar. Fortunately, joint embedding does not exhibit these problems.

Source	微小粒子测量装置的比较试验
Target	微小粒子測定装置の比較試験
Semantic	微小粒子計測装置の比較実験
Joint	微小粒子測定装置の比較試験
Source	胃运动机能和NERD的病状的关连性的相关讨论
Target	胃運動機能とNERDの病態との関連性に関する検討
Semantic	胃運動機能とNERDの病態のの関連性に関する議論
Joint	胃運動機能とNERDの病態との関連性に関する検討
Source	此外,用100对属性数进行了均等化。
Target	また,属性数を100で均一化した.
Semantic	また,属性数を100で均等化した.
Joint	また,属性数を100で均一化した.

Table 4.4: Case Study - More appropriate word selection

Table 4.4 shows that joint embedding will select more precise words for translation. In the first example, “测定装置” is commonly used to indicate the “measuring instrument” meaning of the source ”测量装置”, while “計測” is more often used independently to indicate “measurement”, that is, using instruments to measure. In addition, joint embedding also correctly maps the Chinese word “试验” (test) to its Japanese counterpart “試驗” (test) rather than “実験” (experiment). The second example involves whether to translate the Chinese word “讨论”, to the Japanese word “検討” or “議論”. Depending on context, “讨论” can mean “discussion” and in those case might be translated as “議論” which in Japanese means ‘’‘discussion” or “debate”. However, in the source sentence “讨论”, is used in the sense of “thoughtful consideration”, which is more appropriately translated as “検討”, found in the target and joint embedding translation, but not the semantic embedding. The third example involves whether to directly copy the Chinese word “均等化” as is in the Japanese translation, since “均等化” is a valid Japanese word with similar meaning, or to instead use an alternative Japanese word “均一化”. Both words are technical in nature, but joint embedding selection “均一化” concurs with the target sentence, so it may be considered preferable to simply copying “均等化” as done by the semantic embedding.

4.2 Embedding Analysis

We applied four analysis methods to compare joint embedding and semantic embedding. In all four analyses, joint embedding preserves the properties that semantic embedding should have and even outperforms semantic embedding.

4.2.1 Analogical Reasoning

The joint embedding can also produce correct answers in the analogical reasoning task. We found that the answers obtained from joint embedding received higher confidence values on average in both Chinese (Table 4.5) and Japanese (Table 4.6).

4.2.2 Outlier Detection

Joint embedding can answer outlier detection problems as well as semantic embedding, and sometimes gives better answers. For example, in the second case of Table 4.7, the joint embedding answers correctly, while

semantic embedding answers incorrectly.

Input			Output ($b*$)	
a	$a*$	b	Semantic	Joint
东京 (Tokyo)	日本 (Japan)	北京 (Beijing)	中国 (China) ($p = 0.492$)	中国 (China) ($\boldsymbol{p = 0.531}$)
长期 (long-term)	三年 (3 years)	短期 (short-term)	一年 (1 year) ($p = 0.379$)	两周 (2 weeks) ($\boldsymbol{p = 0.387}$)
进口 (import)	买入 (buy)	出口 (export)	卖出 (sell) ($p = 0.364$)	卖出 (sell) ($\boldsymbol{p = 0.442}$)

Table 4.5: Analogical Reasoning in Chinese

Input			Output ($b*$)	
a	$a*$	b	Semantic	Joint
男性(male)	女性 (female)	父親 (father)	母親 (mother) ($p = 0.487$)	母親 (mother) ($\boldsymbol{p = 0.508}$)
長期 (long-term)	年 (year)	短期 (short-term)	月 (month) ($p = 0.550$)	月 (month) ($\boldsymbol{p = 0.570}$)
左右(left-right)	前後 (front-rear)	水平 (horizontal)	垂直 (vertical) ($\boldsymbol{p = 0.432}$)	垂直 (vertical) ($p = 0.402$)

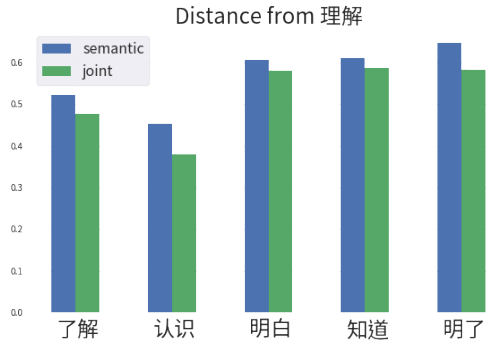
Table 4.6: Analogical Reasoning in Japanese

Word Cluster	Outlier (Output)	
	Semantic	Joint
鸟 (birds), 狗 (dogs), 猫 (cats), 花 (flowers)	花 (flowers)	花 (flowers)
可行(feasible), 不行 (not OK), 行 (OK), 可以 (can/OK)	可以 (can/OK)	不行 (not OK)
广岛 (Hiroshima), 名古屋 (Nagoya), 爱知(Aichi), 上海 (Shanghai, only this one is in China)	上海 (Shanghai)	上海 (Shanghai)

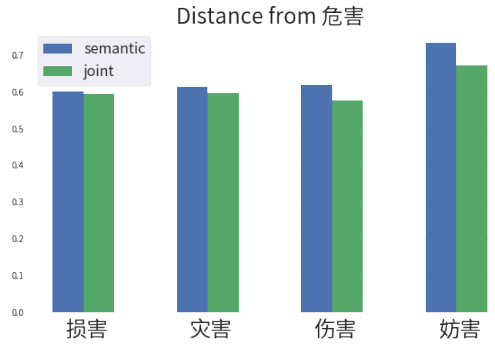
Table 4.7: Outlier Detection in Chinese

Word Cluster	Outlier (Output)	
	Semantic	Joint
生み (birth), 創造 (creation), 作る (making), 破壊 (destruction)	破壊 (destruction)	破壊 (destruction)
普通 (normal), 一般 (general), 通常 (normal), 異常 (abnormal)	異常 (abnormal)	異常 (abnormal)
平成 (Heisei era), 昭和 (Showa era), 大正 (Taisho era), 明治 (Meiji era), 京都 (Kyoto)	京都 (Kyoto)	京都 (Kyoto)

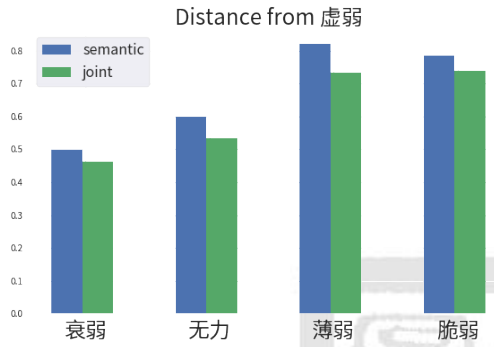
Table 4.8: Outlier Detection in Japanese



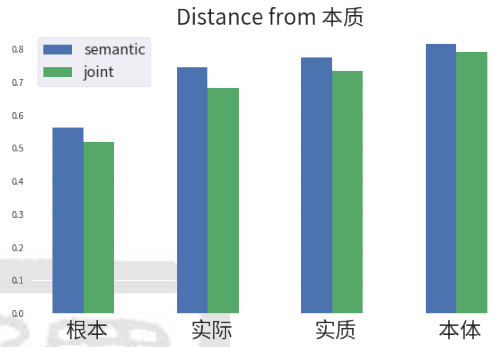
(a) Similarity distances between “理解” (comprehend) and its synonyms.



(b) Similarity distances between “危害” (jeopardize) and its synonyms.



(c) Similarity distances between “弱” (weak) and its synonyms.



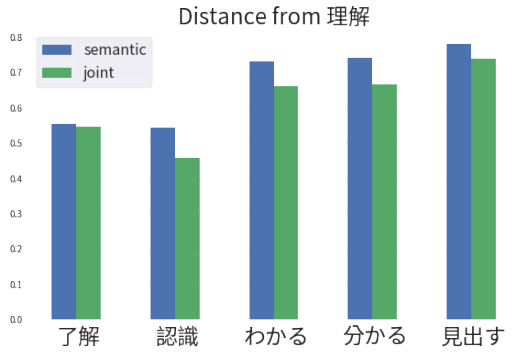
(d) Similarity distances between “本质” (essence) and its synonyms.

Figure 4.1: Similarity distances between synonyms in Chinese. In any cases joint embedding assigns shorter distances.

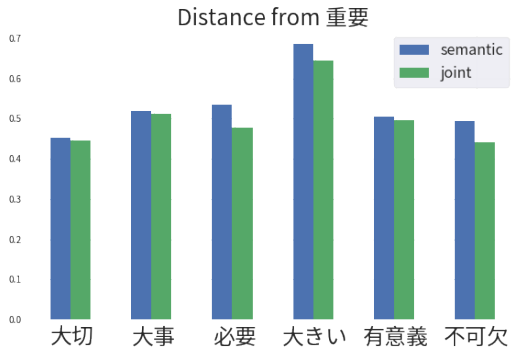
4.2.3 Word Similarity

The similarity distance between synonyms is not only preserved but also shortened. This finding indicates the joint embedding not only protects but also enhances the semantics by adding phonetic information. Figure 4.1 shows that the similarity distances of Chinese synonyms are shortened in the joint embedding. For example, Figure 4.1a shows that the distance between “了解 (understand), 认识 (recognize), 明白 (realize), 知道 (know), 明了 (clearly understand)” and “理解 (comprehend)” are reduced in the joint embedding.

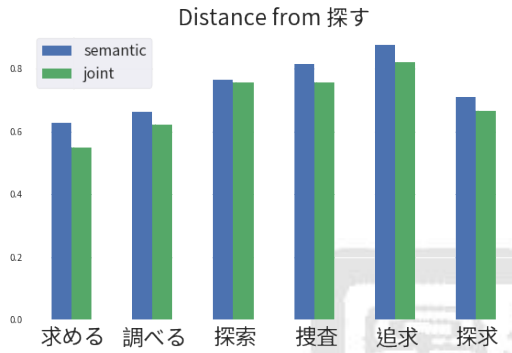
Similarly, the similarity distances of Japanese synonyms are also shortened with the use of joint embedding. Especially in Japanese, sometimes hiragana and kanji are used interchangeably. For example, in Figure 4.2d, “また” and “又 (また)” have the pronunciation and meaning (“again”). The joint embedding can find the relationship between them and reduce their distance significantly.



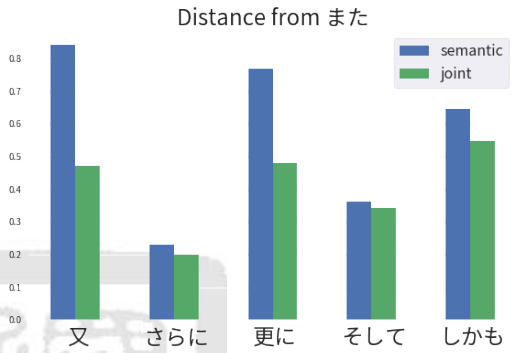
(a) Similarity distances between “理解” (comprehend) and its synonyms.



(b) Similarity distances between “重要”’ (important) and its synonyms.



(c) Similarity distances between “探す” (seek) and its synonyms.



(d) Similarity distances between “また” (again) and its synonyms.

Figure 4.2: Similarity distances between synonyms in Japanese. In any cases joint embedding assigns shorter distances.

4.2.4 Homonym and Heteronym

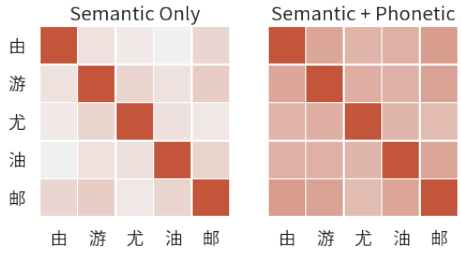
We also found encouraging results from tests on homonyms and heteronyms. The higher correlation between the homonym vectors indicates that the joint embedding can help the model be robust against noise (typographical errors). The increase in the similarity distance between heteronyms implies that the phonetic information effectively separates them, which is a sign of semantic enhancement.

Homonym

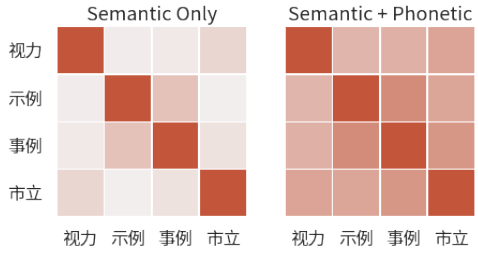
In Figure 4.3, we examine the correlations between Chinese homonyms in three scenarios: the correlation of homophonic characters (Figure 4.3a), the correlation of homophonic words (Figure 4.3b), and the correlation of words with the same pronunciation except for having different tones (Figure 4.3c). The results show that the correlation of these homonyms and near homonyms is greater under the joint embedding.

In Figure 4.4, we also examine the correlations between Japanese homonyms in three scenarios. Since there are no tones in Japanese, we tested the homonyms with single syllable (Figure 4.4a), homonyms with two

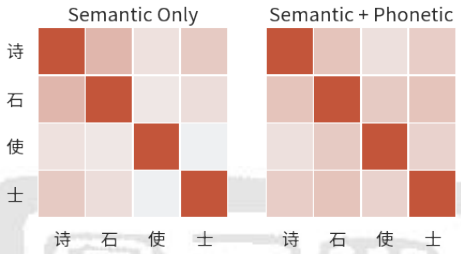
syllables (Figure 4.4b), and homophonic words (Figure 4.4c). All the results have shown that joint embedding can effectively improve the correlation between Japanese homonyms.



(a) Pearson correlation between characters pronounced ㄧㄡˊ (yóu).

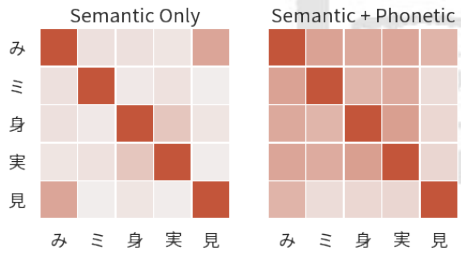


(b) Pearson correlation between words pronounced ㄕˋ ㄌㄧˋ (shì lì).

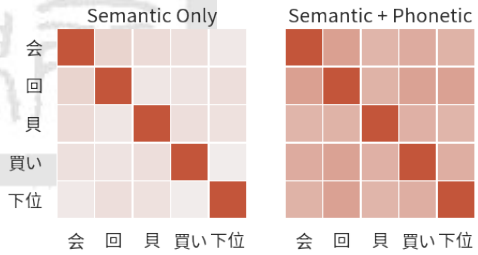


(c) Pearson correlation between characters pronounced ㄕ (shi).

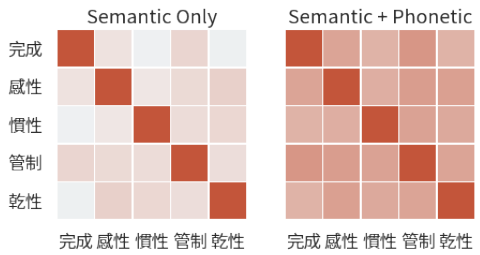
Figure 4.3: Pearson correlation between homonyms in Chinese. In all three cases shown, the correlation between homonyms or near homonyms is greater in the joint embedding than the semantic embedding.



(a) Pearson correlation between characters pronounced み (mi).



(b) Pearson correlation between characters and words pronounced かい (kai).



(c) Pearson correlation between words pronounced かんせい (kansei).

Figure 4.4: Pearson correlation between homonyms in Japanese. In any cases the correlation is higher under joint embedding.

Heteronym

In Table 4.9, we compare the similarity distance between Chinese heteronyms. The first example is "長" meaning "length" when pronounced one way (ㄔㄤˊ) and growth when pronounced another way (ㄓㄤˇ). The second example is "樂", meaning joyful (ㄌㄜˋ) or musical sound (ㄩㄝˋ). The third example "中" stands for middle (ㄓㄨㄥ) or hitting a target (ㄓㄨㄥˋ) respectively. In all three examples, the similarity distances are greater in the joint embedding, which implies that the phonetic information can help distinguish between heteronyms.

A	B	Similarity Distance	
		Semantic	Joint
長 (ㄔㄤˊ) 度	長 (ㄓㄤˇ) 大	0.826	**0.853**
樂 (ㄌㄜˋ) 趣	音樂 (ㄩㄝˋ)	0.636	**0.682**
中 (ㄓㄨㄥ) 午	中 (ㄓㄨㄥˋ) 毒	0.842	**0.866**

Table 4.9: Similarity distance between heteronyms in Chinese

In Table 4.10, the same advantage also appears in Japanese. We have compared the different pronunciations and meanings of "生", a character notorious for having so many distinct readings. We compared the readings (なま) "raw" , (しょう) "life" , (う) "birth" , and (き) "unprocessed". All the results show that joint embedding can increase the similarity distance between them.

A	B	Similarity Distance	
		Semantic	Joint
生 (なま)	一生 (しょう)	0.879	**0.889**
生 (なま)	生 (う) む	0.830	**0.839**
生 (なま)	生 (き) 地	0.769	**0.867**

Table 4.10: Similarity distance between heteronyms in Japanese

There is a special feature in Japanese where many kanji are assigned a pair of readings; one originally derived from Chinese called on-yomi (音読み) and another of more purely Japanese origin (i.e. not from Chinese) called kun-yomi (訓読み). In general, the on-yomi and kun-yomi pronunciations of characters are entirely unrelated, but the meaning is usually very close. In many cases, the kun-yomi pronunciation is used when the word is used along, while the on-yomi is used as an element to form compound words. A situation is roughly analogous to what English might look like if we used a special water symbol to write both "water" and "hydro".

We examined the (on-yomi, kun-yomi) pairs in Table 4.11. For example, when “土” is pronounced as “つ ち” or “と”, it means soil; when “海” is pronounced as “う み” or “か い”, it means sea; when “時” is pronounced as “と き” or “じ”, it means time. Despite having completely different pronunciations, these semantically close (on-yomi,kun-yomi) pairs are closer to each other under the joint embedding.

A	B	Similarity Distance	
		Semantic	Joint
土 (つち)	土 (と) 地	0.719	**0.672**
海 (うみ)	深海 (かい)	0.707	**0.637**
時 (とき)	時 (じ) 間	0.578	**0.498**

Table 4.11: Similarity distance between kun-yomi and on-yomi in Japanese

Chapter 5

Conclusion and Future Work

This chapter will summarize the findings and contributions and further discuss some potential future improvements. We will summarize the results and contributions in Section 5.1. After that, we propose a series of methods that can be further studied or improved as our future work in Section 5.2.

5.1 Conclusion

Our work aimed to improve the neural machine translation between Chinese and Japanese by applying feature engineering to texts, constructing a word embedding with phonetic information, and combining the phonetic embedding with general word embedding that represents semantics. The resulting joint semantic-phonetic embedding obtained positive results in the neural machine translation model and the embedding analysis.

We compared translation performance using different models and tokenization methods. The use of joint embedding achieved better BLEU scores than the use of semantic or phonetic embedding. Moreover, compared to semantic embedding, the translations generated by the best model with joint embedding tended to be more natural and grammatical, and showed the ability to utilize Western loanwords (written in katakana) in Japanese translations while faithfully preserving English acronyms and jargon.

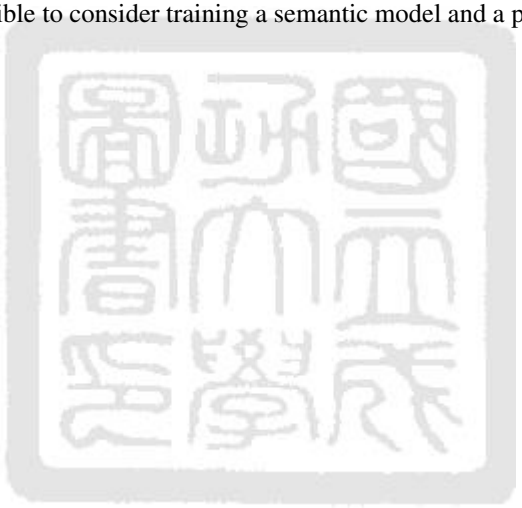
Embedding analysis further identified the effectiveness of joint embedding. The tests from analogical reasoning and outlier detection demonstrated that the joint embedding not only had retained the capabilities but also enhanced the performance of the semantic embedding. Furthermore, the similarity distance and Pearson correlation of synonyms, homonyms, and heteronyms were observed to be better in joint embedding, indicating improvement in semantics and noise resistance.

Together, our results and analysis reveal demonstrate that joint semantic-phonetic embedding is a highly promising technique to improve Chinese-Japanese neural machine translation systems.

5.2 Future Work

A variety of research can be further investigated based on the method of this paper. Here we list a few techniques and approaches that may further improve performance.

- Using any basic attempts to refine the research. For instance, using other tokenization methods, phonetic extraction methods, NMT models, and embedding training methods.
- Referring to the model for training Chinese word embedding as described in Section 1.3.1. Training an embedding by combining phonetic information with Chinese characters and various sub-character features such as strokes and radicals.
- Using ELMo or BERT as the base embedding framework. Since they are both embeddings that require a binding model, it is possible to consider training a semantic model and a phonetic model and combining the two.



REFERENCES

[Bahdanau et al., 2014] Bahdanau, D., Cho, K., and Bengio, Y. (2014). Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*.

[Bengio et al., 2003] Bengio, Y., Ducharme, R., Vincent, P., and Janvin, C. (2003). A neural probabilistic language model. *The journal of machine learning research*, 3:1137–1155.

[Biewald, 2020] Biewald, L. (2020). Experiment tracking with weights and biases. Software available from wandb.com.

[Bojanowski et al., 2017] Bojanowski, P., Grave, E., Joulin, A., and Mikolov, T. (2017). Enriching word vectors with subword information. *Transactions of the Association for Computational Linguistics*, 5:135–146.

[Cao et al., 2018] Cao, S., Lu, W., Zhou, J., and Li, X. (2018). cw2vec: Learning chinese word embeddings with stroke n-gram information. In *Proceedings of the AAAI Conference on Artificial Intelligence*.

[Chen et al., 2015] Chen, X., Xu, L., Liu, Z., Sun, M., and Luan, H. (2015). Joint learning of character and word embeddings. In *Twenty-Fourth International Joint Conference on Artificial Intelligence*.

[Cho et al., 2014] Cho, K., van Merrienboer, B., Gulcehre, C., Bougares, F., Schwenk, H., and Bengio, Y. (2014). Learning phrase representations using rnn encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078*.

[Chu et al., 2017] Chu, C., Dabre, R., and Kurohashi, S. (2017). An empirical comparison of domain adaptation methods for neural machine translation. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 385–391.

[Coates and Bollegala, 2018] Coates, J. and Bollegala, D. (2018). Frustratingly easy meta-embedding – computing meta-embeddings by averaging source word embeddings. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*, pages 194–198, New Orleans, Louisiana. Association for Computational Linguistics.

[Devlin et al., 2019] Devlin, J., Chang, M.-W., Lee, K., and Toutanova, K. (2019). 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, Minneapolis, Minnesota. Association for Computational Linguistics.

[Dyer et al., 2013] Dyer, C., Chahuneau, V., and Smith, N. A. (2013). A simple, fast, and effective reparameterization of ibm model 2. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 644–648.

[Falcon, 2019] Falcon, WA, e. a. (2019). Pytorch lightning. *GitHub. Note: https://github.com/PyTorchLightning/pytorch-lightning*, 3.

[Hinton et al., 1986] Hinton, G. E. et al. (1986). Learning distributed representations of concepts. In *Proceedings of the eighth annual conference of the cognitive science society*, volume 1, page 12. Amherst, MA.

[Imamura et al., 2018] Imamura, K., Fujita, A., and Sumita, E. (2018). Enhancement of encoder and attention using target monolingual corpora in neural machine translation. In *Proceedings of the 2nd Workshop on Neural Machine Translation and Generation*, pages 55–63.

[Khan and Xu, 2019] Khan, A. R. and Xu, J. (2019). Diversity by phonetics and its application in neural machine translation. *arXiv preprint arXiv:1911.04292*.

[Kiela et al., 2018] Kiela, D., Wang, C., and Cho, K. (2018). Dynamic meta-embeddings for improved sentence representations. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 1466–1477, Brussels, Belgium. Association for Computational Linguistics.

[Kirkpatrick et al., 2017] Kirkpatrick, J., Pascanu, R., Rabinowitz, N., Veness, J., Desjardins, G., Rusu, A. A., Milan, K., Quan, J., Ramalho, T., Grabska-Barwinska, A., et al. (2017). Overcoming catastrophic forgetting in neural networks. *Proceedings of the national academy of sciences*, 114(13):3521–3526.

[Koehn et al., 2018] Koehn, P., Khayrallah, H., Heafield, K., and Forcada, M. L. (2018). Findings of the wmt 2018 shared task on parallel corpus filtering. In *Proceedings of the Third Conference on Machine Translation: Shared Task Papers*, pages 726–739.

[Kudo, 2018] Kudo, T. (2018). Subword regularization: Improving neural network translation models with multiple subword candidates. In *Proceedings of the 56th Annual Meeting of the Association for Computa-*

tional Linguistics (Volume 1: Long Papers), pages 66–75, Melbourne, Australia. Association for Computational Linguistics.

[Kudo and Richardson, 2018] Kudo, T. and Richardson, J. (2018). SentencePiece: A simple and language independent subword tokenizer and detokenizer for neural text processing. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 66–71, Brussels, Belgium. Association for Computational Linguistics.

[Liu et al., 2019] Liu, H., Ma, M., Huang, L., Xiong, H., and He, Z. (2019). Robust neural machine translation with joint textual and phonetic embedding. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 3044–3049, Florence, Italy. Association for Computational Linguistics.

[Mikolov et al., 2013a] Mikolov, T., Chen, K., Corrado, G., and Dean, J. (2013a). Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*.

[Mikolov et al., 2013b] Mikolov, T., Sutskever, I., Chen, K., Corrado, G., and Dean, J. (2013b). Distributed representations of words and phrases and their compositionality. *arXiv preprint arXiv:1310.4546*.

[Muromägi et al., 2017] Muromägi, A., Sirts, K., and Laur, S. (2017). Linear ensembles of word embedding models. *arXiv preprint arXiv:1704.01419*.

[Nakazawa et al., 2020] Nakazawa, T., Nakayama, H., Ding, C., Dabre, R., Higashiyama, S., Mino, H., Goto, I., Pa, W. P., Kunchukuttan, A., Parida, S., et al. (2020). Overview of the 7th workshop on asian translation. In *Proceedings of the 7th Workshop on Asian Translation*, pages 1–44.

[Nakazawa et al., 2016] Nakazawa, T., Yaguchi, M., Uchimoto, K., Utiyama, M., Sumita, E., Kurohashi, S., and Isahara, H. (2016). ASPEC: Asian scientific paper excerpt corpus. In *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC'16)*, pages 2204–2208, Portorož, Slovenia. European Language Resources Association (ELRA).

[Papineni et al., 2002] Papineni, K., Roukos, S., Ward, T., and Zhu, W.-J. (2002). 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.

[Pennington et al., 2014] Pennington, J., Socher, R., and Manning, C. D. (2014). Glove: Global vectors for word representation. In *Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543.

[Peters et al., 2018] Peters, M. E., Neumann, M., Iyyer, M., Gardner, M., Clark, C., Lee, K., and Zettlemoyer, L. (2018). Deep contextualized word representations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 2227–2237, New Orleans, Louisiana. Association for Computational Linguistics.

[Schuster and Nakajima, 2012] Schuster, M. and Nakajima, K. (2012). Japanese and korean voice search. In *2012 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 5149–5152.

[Sennrich et al., 2016a] Sennrich, R., Haddow, B., and Birch, A. (2016a). Improving neural machine translation models with monolingual data. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 86–96, Berlin, Germany. Association for Computational Linguistics.

[Sennrich et al., 2016b] Sennrich, R., Haddow, B., and Birch, A. (2016b). Neural Machine Translation of Rare Words with Subword Units. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1715–1725, Berlin, Germany. Association for Computational Linguistics.

[Sutskever et al., 2014] Sutskever, I., Vinyals, O., and Le, Q. V. (2014). Sequence to sequence learning with neural networks. In *Advances in Neural Information Processing Systems*, pages 3104–3112.

[Vaswani et al., 2017] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L. u., and Polosukhin, I. (2017). Attention is all you need. In Guyon, I., Luxburg, U. V., Bengio, S., Wallach, H., Fergus, R., Vishwanathan, S., and Garnett, R., editors, *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc.

[Yin et al., 2016] Yin, R., Wang, Q., Li, P., Li, R., and Wang, B. (2016). Multi-granularity chinese word embedding. In *Proceedings of the 2016 conference on empirical methods in natural language processing*, pages 981–986.

[Yin and Schütze, 2015] Yin, W. and Schütze, H. (2015). Learning meta-embeddings by using ensembles of embedding sets. *arXiv preprint arXiv:1508.04257*.

[Yu et al., 2017] Yu, J., Jian, X., Xin, H., and Song, Y. (2017). Joint embeddings of chinese words, characters, and fine-grained subcharacter components. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 286–291.

[Zhang et al., 2020] Zhang, B., Nagesh, A., and Knight, K. (2020). Parallel corpus filtering via pre-trained language models. *arXiv preprint arXiv:2005.06166*.

[Zhang and Matsumoto, 2017] Zhang, J. and Matsumoto, T. (2017). Improving character-level japanese-chinese neural machine translation with radicals as an additional input feature. In *2017 International Conference on Asian Language Processing (IALP)*, pages 172–175.

[Zhang and Komachi, 2018] Zhang, L. and Komachi, M. (2018). Neural machine translation of logographic language using sub-character level information. In *Proceedings of the Third Conference on Machine Translation: Research Papers*, pages 17–25, Brussels, Belgium. Association for Computational Linguistics.