Tokyo Metropolitan University

Doctoral Thesis

Neural Machine Translation Using Sub-Character Level Information

---

Author:

Longtu Zhang

Supervisor:

Associate Prof.

Mamoru Komachi

A thesis submitted in fulfillment of the requirements

for the degree of Doctor of Philosophy

in the

Department of Information and Communication Systems

Faculty of System Design

Graduate School of System Design (Doctoral Program)

February 11, 2021

Declaration of Authorship

I, Longtu Zhang, declare that this thesis titled, "Neural Machine Translation Using Sub-Character Level Information" and the work presented in it are my own. I confirm that:

- This work was done wholly or mainly while in candidature for a research degree at this university.
- Where any part of this thesis has previously been submitted for a degree or any other qualification at this university or any other institution, this has been clearly stated.
- Where I have consulted the published work of others, this is always clearly attributed.
- Where I have quoted from the work of others, the source is always given. With the exception of such quotations, this thesis is entirely my own work.
- I have acknowledged all main sources of help.
- Where the thesis is based on work done by myself jointly with others, I have made clear exactly what was done by others and what I have contributed myself.

Signed:

_____

Date:

_____

TOKYO METROPOLITAN UNIVERSITY

Abstract

Faculty of System Design

Department of Information and Communication Systems

Doctor of Philosophy

Neural Machine Translation Using Sub-Character Level Information

by Longtu ZHANG

Logographic and alphabetic languages (e.g., Chinese vs. English) have different writing systems linguistically. Languages of the same writing system share more similarities, which can facilitate natural language processing tasks such as neural machine translation (NMT). This paper takes advantage of logographic characters in Chinese and Japanese by decomposing them into smaller units, thus more optimally utilizing the shared information in the training of NMT systems in both encoding and decoding processes. Experiments show that the proposed method can improve the NMT performance of both "logographic" language pairs (JA–ZH) and "logographic–alphabetic" (JA–EN and ZH–EN) language pairs in both supervised and unsupervised NMT systems and finer decomposition granularities generally

leads to better performance. The findings suggest that finer granularity data with a higher portion of shared tokens (share token rate) and smaller vocabulary size can facilitate both encoding and decoding process in NMT training on the whole and facilitate each process individually. Meanwhile, longer sequence length can be a significant counter factor in training, and special multi-layer positional embedding (MPE) is introduced for Transformer NMT systems. We argue that a higher level of information sharing in the training data is the main reason for the improvements.

Acknowledgements

First of all, I would like to thank Professor Mamoru Komachi for his lead, guidance, and support for these four years. Without him, it is not possible for me to start my Ph.D. journey from a linguistic background and find my research interests that combined my past study experience and the new area of artificial intelligence and natural language processing. Especially during those days when I had to absorb so much new knowledge that I was not at all familiar with, his encouragements were powerful and had strengthened my confidence and resolution. Professor Komachi is one of the smartest people I have worked with, and his knowledge and attitude to research and life will always influence me in future study and career.

Moreover, I would like to thank my classmates: Tomoyuki Kajiwara, Yui Suzuki, Aizhan Imankulova, Masahiro Kaneko, Zhousi Chen, Yuting Zhao, Tosho Hirasawa, and many others I can not include here. All of them formed an excellent learning atmosphere in the lab, and I really appreciated it. It would not be possible for me to continue my learning and researching without the discussions and help from them.

Finally, I would like to thank my wife, my parents, and all my family members for their generous support during my Ph.D. years. I sacrificed so much for this, and I owe them a lot.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Language and Translation

Language is a *communication system* with *structures*. To use language to convey *meanings* is an activity that specific to humans. Languages can be used in forms of *spoken language*, *written langauge*, *sign language* and many other forms. Spoken language is the basic form of language, using structured sound to express meanings. Its flexibility allows speakers to express meanings, including emotions, exaggerations, sarcasm, etc., using different phonemes, syllables, tones, pitches, and intonations. On the other hand, because the sound is transmitted by air, it isn't easy to preserve before the invention of recorders . Written language is invented to represent the spoken language in written forms so that the meanings can be transmitted through long-distance and long-time. Sign language, on the other hand, uses gestures to represent meanings expressed by spoken language. There are also other reduced forms of languages such as *programming languages*, which are usually artificially designed for particular purposes. Nowadays, most information

stored in computers and communicated on the cloud are in written language.

There are thousands of human languages globally, in which 8 of them are spoken by more than 100 million population. [1] Assuming all the languages can express all possible meanings in the world, *translation* is to express the meaning in one language using another language. The translation is the bridge for communication between people speaking different languages.

Translators and interpreters are human experts for text translation and speech translation; moreover, simultaneous interpreters require more expertise to deliver instant speech translation. However, the training of translators and interpreters are expensive and time-consuming, and their workloads are usually very intense to produce high-quality translations. On the other hand, on many occasions, middle- or even low-quality translations can suffice the demands. Because of human translators and interpreters' efficiency and economic bottleneck, people start to look for novel methods to solve the problem.

1.2 History of Machine Translation

Thanks to the invention of modern computers, text data can be digitalized and processed more quickly than before. People started to think about how to use machines to do the translation. Machine translation task was formulated as building a model that can output a text in target language conditioned on the meaning of a source-language text.

[1] According to Wikipedia, these languages are Mandarin Chinese, Spanish, English, Hindi, Bengali, Portuguese, Russian, Japanese in descending order. https://en.wikipedia.org/wiki/List_of_languages_by_number_of_native_speakers

Along with the development of *Formalism Linguistics*, the rule-based machine translation system first emerged in 1970s (Hutchins, 1986). It utilized syntactic trees to map the *surface form* and dictionaries to map the *meanings* between the source and target languages. It can barely deal with the ambiguity in natural languages since there is no consideration of context. For example, sentence 1.1 can be interpreted in two meanings:

The dog saw a man in the park. (1.1)

1. The event of the dog's seeing a man happened in the park;
2. The man was in the park, and the dog saw him.

Without any context information, the rule-based machine translation system can not process such sentences. Moreover, it can only work on similar language pairs because there is no mechanism to guarantee the fluency of the translation.

In the 1990s, the IBM models were raised, representing the new era of statistical machine translation (Brown et al., 1993). The translation task from source language sentence f to target language sentence e is formulated as a noisy channel model. The translation probability can be written according to Bayes theorem in 1.2:

$$\Pr(e|f) = \frac{\Pr(e)\Pr(f|e)}{\Pr(f)} \tag{1.2}$$

To find the target translation $\hat{e}$ is to maximize the probability:

$$\hat{e} = \arg\max_{e} \Pr(e)\Pr(f|e) \tag{1.3}$$

where $\Pr(e)$ is the *language model* which can be learned from monolingual text and $\Pr(f|e)$ is the *translation model* which can be learned by parallel text. Statistical machine translation has been successful within the field for more than two decades. It became more complex as people try to introduce more sophisticated language and translation models. Also, another weakness is that it is hard to deal with long dependencies in the sentence.

Thanks to the revival of neural network techniques starting from 2010s, Sutskever et al. (2014) and Cho et al. (2014) demonstrated that using recurrent neural network (RNN) in encoder–decoder (or sequence-to-sequence) architecture is effective for machine translation task. Bahdanau et al. (2015) proposed the attention mechanism and significantly increased the translation performance of long sentences and sentences with long-dependencies. Luong et al. (2015) improved the attention-based NMT. Because the RNN-based NMT with attention significantly outperformed the traditional SMT and the implementation is straightforward in most occasions, it became the most popular model for machine translation tasks. Later in 2017, Vaswani et al. (2017) proposed the Transformer network that pushed NMT performance to a new level and greatly influenced other NLP tasks such as natural language understanding, classification, etc.

1.3 Current Research and Contributions

In this research, machine translation (NMT) is formulated as a machine learning task that predicts the target-language translation text based on the source-language text. In addition to NMT models based on long short-term memory (Hochreiter and

Schmidhuber, 1997) (LSTM) and Transformer network (Vaswani et al., 2017), we also investigate a special type of self-supervised NMT model called "unsupervised NMT" (Lample et al., 2018) (UNMT) models.

NMT training usually requires a tremendous amount of training text, which exhibits a long-tail distribution with a large vocabulary size (number of types), which poses a significant challenge for computation. In order to solve this problem, subword algorithms represented by BPE (byte-pair encoding) (Sennrich, Haddow, and Birch, 2016) are introduced, which breaks up the words into pieces and using subword vocabulary to train the model. BPE significantly alleviates the long-tail distribution and boosts more shared information.

However, most of these studies are done on alphabetic language pairs and overlooked the difference between logographic and alphabetic text. We introduce sub-character level data in NMT training that can further help alleviate the long-tail distribution in training data and boost the information sharing within the logographic text and across logographic language pairs. Our research hypothesis is that sub-character level data can increase NMT models' performance both in logographic language pairs and alphabetic-logographic language pairs, and the finer granularity data can lead to better performance.

Our results suggested that from almost all perspectives, using sub-character level information in (U)NMT tasks outperform the character level baselines:

1. Under same vocabulary size, models trained on sub-character level data outperform those on character level data;

2. Because sub-character level data can achieve smaller vocabulary size, the performance can further increase;

3. Sub-character level data will increase the performance under both logographic-logographic and alphabetic-logographic language pairs;

4. Although general patterns were found that finer granularity data outperform bigger granularity data (e.g., stroke level data outperform ideograph level data), the former will greatly increase the sequence length during training, which makes the training time longer and harder.

Our contributions are:

1. We propose a simple yet effective character decomposition method to transform character-level data to sub-character level data.

2. We use structural information in our character decomposition method, which has empirical gain over past sub-character methods in NLP tasks.

3. We demonstrate the generally positive relationship between finer granularity data and model performance.

4. We showed that our method could work on both NMT and UNMT scenarios.

5. We demonstrate that finer granularity data not only facilitates NMT model performance between logographic language pairs but also alphabetic–logographic language pairs. Thus, the proposed method may shed light on other NLP tasks, such as pre-trained models for natural language understanding (NLU), which only use the transformer network's encoder part.

1.4 Thesis Structure

The following thesis will be organized as follows:

- Chapter 1 provides a general introduction to the sub-character NMT problem. The organization of the thesis and contributions are also highlighted.
- Chapter 2 reviews the background of NMT and UNMT tasks in general, including the task formulation, the decomposability of logographic characters, and the machine translation models. It reveals that although most NMT models and popular methods are widely examined, they overlook the difference between alphabetic languages and logographic languages, which can further facilitate their performance if the logographic information can be used wisely. It also discusses the significance of the current study.
- Chapter 3 introduces the research method in detail. It first details the character decomposition method for logographic languages and then introduces a method to safely transform the decomposition sequence back to character sequences. Further, for Transformer models, it describes an extra positional encoding for the sub-character level sequences.
- Chapter 4 details the experimental settings. We tested our hypothesis on Chinese–Japanese, Chinese–English, and Japanese–English language pairs, representing NMT models between logographic languages and between alphabetic and logographic languages. We use a character decomposition dictionary (cjkvi-ids) to decompose logographic characters into three granularities: ideograph level data, finest ideograph level data, and stroke level data. We modified the dictionary with special markers to make sure the sub-character

sequence can be converted back safely. Lastly, we added an extra positional encoding layer to alleviate the problem of long sub-character level sequences.

- Chapter 5 shows the results. Our results suggested that almost from all dimensions, using sub-character level information in (U)NMT tasks outperformed the character level baselines:
  - Under the same vocabulary size, models trained on sub-character level data outperformed those on character level data.
  - Finer granularity data usually have better performance.
  - Sub-character level data was able to increase the performance under both logographic-logographic and alphabetic-logographic language pairs.
  - Similar tendencies were found in UNMT models trained on logographic language pairs.
- Chapter 6 discusses the results. The *smaller vocabulary size* and *more shared information* might be reasons for better performance. Extra control experiments have confirmed the benefits of both, which is only possible if sub-character level data is used. The finer the granularity of sub-character level data, the better performance the model tends to have. However, the performance can drop when the training sequence is too long in "stroke" level data. Additionally, we compared our results with many other baselines, which also tried to increase the shared information between source and target text to boost NMT performance, such as "character mapping" and "kana decomposition." The results suggested that our approach performs steadily better

than these control groups. We also discuss the potential reasons.

- Chapter 7 concludes the dissertation and gives out future research directions. This dissertation first notices the critical difference between alphabetic languages and logographic languages in (U)NMT training and uses a simple character decomposition method to transform character-level data to sub-character level data, and demonstrated significant and steady improvement in using sub-character level information in NMT tasks and shed light on other NLP tasks. In the future, we will try to apply more NMT techniques to sub-character level data, such as curriculum learning. Also, we will try to achieve better results in UNMT settings.

Chapter 2

Background

2.1 Overview of NMT Tasks

Machine Translation (MT) aims to automatically transform from source language text to target language text accurately and fluently. It is formulated as a machine learning task that makes sequential decisions on target tokens to form a target sentence conditioned on the source sentence.[1] *Statistical machine translation* (SMT) that combines phrase-table based translation models and n-gram language models has been a successful method (Koehn, 2010). With the development of deep learning techniques and increase of parallel data, modern *neural machine translation* (NMT) has achieved much better performance than SMT and became the most received MT method.

Most NMT models follow the encoder–decoder (or sequence-to-sequence) architecture (Cho et al., 2014; Sutskever, Vinyals, and Le, 2014) with attention mechanisms (Bahdanau, Cho, and Bengio, 2015; Luong, Pham, and Manning, 2015) using

[1] There are MT tasks in other modalities such as speech-to-text MT, image-to-text MT as well, however, they are beyond the range of the current dissertation.

recurrent neural networks (RNN) such as long short-term memory (LSTM) (Hochreiter and Schmidhuber, 1997) and gated recurrent unit (GRU) (Cho et al., 2014). The success of RNN-based NMT systems was marked by "Google's Neural Machine Translation" (GNMT) system (Wu et al., 2016). Recently, Transformer network (Vaswani et al., 2017) featured by multi-head attention, self-attention and positional embedding mechanisms abandoned recurrence and achieved better performance. Its powerful learning capability made it possible for building unsupervised NMT (UNMT) (Artetxe et al., 2018; Lample et al., 2018) systems using back-translation circuits. Moreover, its derivatives such as bidirectional encoder representations from Transformers (BERT) (Gehring et al., 2017), generative pre-trained Transformer (GPT) (Radford et al., 2019), etc., have pushed the natural language understanding (NLU) and natural language generation (NLG) task to a whole new level.

Besides the development of modeling techniques, the better use of training data is another *equally* important issue. Various curriculum learning (Bengio et al., 2009) algorithms help the training better and faster by changing the order of training data. Sub-word algorithms such as byte pair encoding (BPE) (Sennrich, Haddow, and Birch, 2016), sentence piece (Kudo, 2018), etc., have contributed to solving the old puzzling issue of the out-of-vocabulary (OOV) problem and further enhance the model performance by lowering the data granularity. Although the data sizes have become larger over time, which is essential to training, the better exploitation of the data themselves has never been less important.

The sub-word method is a text pre- and post-processing method that further breaks words into smaller units by discovering shared parts among themselves. The textual

characteristics of sub-word units can be better exploited during training.

2.2 Formulation of NMT Tasks

Given a *parallel* corpus $\mathcal{D}$ containing N sentence pairs of language 1 ($L1$) and langauge 2 ($L2$), such as:

$$\mathcal{D} = \{(X_1, Y_1), (X_2, Y_2), \cdots, (X_N, Y_N)\} \tag{2.1}$$

where the meaning of X_i is equivalent to the meaning of Y_i, when $i \in 1, \cdots, N$. Both X_i and Y_i are tokenized sentences (list of *tokens*) of length m and n respectively, such as:

$$\begin{aligned} X_i &= [\texttt{BOS}, x_1^{(i)}, x_2^{(i)}, \cdots, x_m^{(i)}, \texttt{EOS}] \\ Y_i &= [\texttt{BOS}, y_1^{(i)}, y_2^{(i)}, \cdots, y_n^{(i)}, \texttt{EOS}] \end{aligned} \tag{2.2}$$

for simplicity the superscript $^{(i)}$ will be omitted. The source and target vocabulary are unique *types* of source and target tokens:

$$\begin{aligned} V_{src} &= \{v_1^{(s)}, v_2^{(s)}, \cdots, v_{len(V_{src})}^{(s)}\} \\ V_{trg} &= \{v_1^{(t)}, v_2^{(t)}, \cdots, v_{len(V_{trg})}^{(t)}\} \end{aligned} \tag{2.3}$$

where the superscript $^{(s)}$ and $^{(t)}$ stand for source and target vocabulary and will be omitted for simplicity reasons.

A machine translation system will use a parallel corpus $\mathcal{D}_{train}$ to train and another parallel corpus of *same distribution* $\mathcal{D}_{test}$ to evaluate the performance. By the same

distribution, we mean they are sampled from the same corpus or the same domain.

Given one translation direction, such as from $L1$ to $L2$, the translation system f is asked to take source sentence X_i as input and output $\hat{Y}_i$ as candidate translation, a.k.a, *hypothesis* sentence:

$$\hat{Y}_i = f(X_i) \tag{2.4}$$

During training, the target sentence Y_i is taken as the *reference* sentence. The system learns to output $\hat{Y}_i$ that is as close to Y_i as possible. During testing, $\hat{Y}_i$ is predicted by $f(\cdot)$ incrementally. For example, in the first decoding step, only `BOS` is inputted to $f(\cdot)$ and get the first decoding output $\hat{Y}_i$. Then the $\hat{Y}_i$ will be concatenated with `BOS` again in the beginning to input to $f(\cdot)$ for decoding the next step. The decoding process will end until a pre-defined maximum sequence lengths is reached, or a `EOS` token is reached, and hence the full $\hat{Y}_i$ is generated. For evaluation, $\hat{Y}_i$ and Y_i are first de-tokenized to string form. After translating all the source sentences in $\mathcal{D}_{test}$, a corpus-level metric is computed on collections of string forms of $\mathcal{D}_{hyp} = \{(\hat{Y}_i, Y_i)\}$, i.e., the hypothesis corpus.

2.3 Leveraging Data in NMT Training

Because parallel training data for NMT tasks are usually very expensive and difficult to obtain, leveraging such data more efficiently becomes an important question. Generally speaking, there are two perspectives to better utilize the training data: through wise curriculum learning and tokenization of finer granularities.

The following sections will discuss these two topics.

2.3.1 Curriculum Learning

Curriculum Learning (Bengio et al., 2009) refers to the tendency that different learning order will result in different learning result. By carefully designing the learning order, i.e., the order that the training batches are feeding into the model, the model can learn the task faster and/or better.

The basic principle that Bengio et al. raised in their paper was that either filtering the noisy examples or feeding easy examples first will help the overall training.

For NMT tasks, there are two lines of research. One is on the *batch* level, that is how to re-order the batch to put the "easier" ones first to boost training. The other is on the *token* level, that is, when and how to use the sampled tokens in feeding to the next token in training.

Batching Curriculum

Almost directly following Bengio's idea, researchers looked for different selected features to decide which batch should go first during training. We call this batching curriculum.

Kocmi et al. (2017) sorted the batches according to target sentence length and syntactic complexity. They observed slight overall improvement in BLEU scores within one training epoch but a sharp drop of perplexity and BLEU scores between epochs. They believe the curriculum learning for NMT can get over-fitted on particular features quickly.

Zhang et al. (2017) tries to "boost" the training data in NMT using the perplexity scores during training. Training examples with high perplexity scores will be

up-sampled because they are considered "difficult" for training. Training examples with low perplexity scores will be down-sampled because they are considered "easier" examples. From the second epoch on, the training data is based on the previous training's training condition. This method is better at handling feature over-fitting but worse at stable training since it changes the training data distribution. In most cases, it has a faster convergence speed, but the overall increase in BLEU score is small. The author argues that this might be because the training data distribution will gradually converge to the original distribution.

Platanios et al. (2019) proposed a combinatory method of measuring word difficulty and learning competency during NMT training and selecting the next batch based on these measures. The difficulty measures include sentence length, word rareness, etc.; the competency measures were formulated as a function of learning loss. Thus the training can better capture the "easy-first" principle during training. Moreover, the over-fitting was alleviated. However, it could not completely overcome the feature over-fitting problem and added an extra feature engineering workload.

Sampling Curriculum

The sampling curriculum was proposed as a token level curriculum. The model will decide whether to use ground truth tokens from the target side to feed in the model for the next token prediction or use the model's previous prediction. This is sometimes referred to as "random teacher forcing" since when using the teacher forcing training method, the next token will always be the target side ground truth, regardless of what is the model prediction at this time. Generally speaking, this method is believed to increase the robustness of training.

Datasets	# Sents.	Vocab Size	
		L1	L2
wmt20.en–zh	39.40M	2.78M	3.23M
wmt20.en–de	45.81M	6.04M	11.17M
wmt19.en–de	38.77M	6.16M	10.02M
wmt19.en–ru	14.26M	3.10M	3.93M
wmt18.en–de	42.28M	6.34M	9.63M
wmt16.en–de	4.56M	1.03M	2.04M
wmt14.en–fr	40.84M	4.23M	4.00M
wmt14.en–de	4.47M	0.82M	1.68M

TABLE 2.1: WMT corpus sizes. The Chinese (`zh`) data is tokenized by `jieba` tokenizer with default dictionary; the rest of the data are tokenized by `SacreMoses` tokenizer. A trend of increasing vocabulary size can be observed since more data is added through the years, especially when datasets like `WikiTitles` is added which contains a lot of unique vocabularies.

Bengio et al. (2015) and Zhang et al. (2019) both used similar idea and achieved very promising results.

2.3.2 Sub-word Segmentation

Another method for efficiently exploiting NMT training data is sub-word algorithms. In the corpus of alphabetic languages, there is usually a large number of types in the vocabulary, which is difficult for computers to process.

Table 2.1 showed the vocabulary size information of major benchmark datasets. The word type frequencies in each dataset follow a long-tail distribution. Few types such as “the,” “a,” “is” in English have extremely high frequencies, and a more considerable amount of types only have very low frequencies. If the word type frequencies were plotted in decreasing order, most of the word types are in

the long-tail.

In order to solve the large vocabulary problem, a heuristic idea is to group the low-frequency types into one `<unk>` type so that the overall vocabulary size can be significantly reduced. Since most words (actual tokens) in the text are also high frequent ones, the BLEU (Papineni et al., 2002) score will not drop much even if we use `<unk>` in the testing hypothesis.

One shortage of this method is that the long-tail types are usually the most informative and important words in the sentence. A translating sentence without long-tail words is harder to understand, but a sentence without high-frequency words is relatively more comfortable to understand. Try to compare the following broken sentences:

1. A man looks through a `<unk>`. (The missing word is "telescope," which is a long-tail word.)

2. A man looks through `<unk>` telescope. (The missing word is "a," which is a high-frequency word.)

We will have almost no difficulty in the second sentence but have no idea about the first sentence. A good NMT model can not afford to lose too many long-tail words in order to keep its translation readable. Fortunately, this problem was partially solved by the application of *byte pair encoding* (BPE) algorithm in NMT (Sennrich, Haddow, and Birch, 2016).

BPE

Byte-pair-encoding was originally an information compression algorithm. When training a BPE model, it first decomposed all alphabetic words into letters and merged the most frequent letter sequences step by step until a target vocabulary size is satisfied. When applied to text, it breaks up the sequences of words into sequences of smaller pieces. Furthermore, after these sequences are processed by NMT models, they can be composed back to normal sequences of words. One analogy for BPE sub-word units is that they are like stems, suffixes, and prefixes in a word. To break up word-level sentences into sub-word level sentences can ensure every possible BPE unit get processed by the NMT model and no `<unk>` type is needed. The granularity of BPE data is significantly reduced comparing to word-level data.

To push this idea to the extreme, even character-level NMT is proposed for alphabetic language pairs (Cherry et al., 2018).

2.4 NMT Models in General

NMT models use neural networks to model the $f(\cdot)$ function in Equation 2.4. Assuming the source sentence X_i and the target sentence Y_i has already been transformed into an index based on the source vocabulary V_{src} and the target vocabulary V_{trg} respectively, usually, an NMT model contains the following parts, which we consider as the basic structure of sequence-to-sequence model:

1. Two embedders that transform the source index and target index into word embeddings E_{X_i} and E_{Y_i}.

2. A core neural network model learns to output a vector $E_{\hat{Y}_i}$ based on E_{X_i} and E_{Y_i}. It usually contains an encoder network and a decoder network.

3. Sometimes an attention function $a(\cdot)$ is applied to $E_{\hat{Y}_i}$ and output of the encoder to compare how close they are. This can serve as an analogy to word alignment scores in SMT. The $E_{\hat{Y}_i}$ will be updated based on this closeness information.

4. A projector network and a softmax layer that transform the $E_{\hat{Y}_i}$ into a distribution probability $P_{\hat{Y}_i}$ over target vocabulary size $len(V_{trg})$.

During training, loss is computed over $P_{\hat{Y}_i}$ and P_{Y_i}. The most popular loss for NMT training is cross-entropy (CE) loss. During testing, the beam search algorithm is done based on the $P_{\hat{Y}_i}$ scores. It is also possible to add length penalty and coverage penalty to the scoring function of the beam search algorithm (Wu et al., 2016).

2.4.1 LSTM-based Models

Recurrent neural networks takes the network's output as input, so that it can iteratively generating new output based on a initial state. However, naive recurrent neural networks suffered from gradient vanishing/explosion problem, making it difficult to model long sequences. The long short-term memory network (LSTM) was raised by Hochreiter et al. 1997, adding various gate functions and cell state to alleviate the problem. For quite a long period of time, LSTM is the most popular sequence modeling network due to its powerful learning ability and stable performance.

Let x_t be the input to LSTM network at time step t, Ws be the respective weights, bs be respective biases, the technical details of LSTM network are as follows:

$$\begin{aligned} i_t &= \sigma(W_{ii}x_t + b_{ii} + W_{hi}h_{t-1} + b_{hi}) \\ f_t &= \sigma(W_{if}x_t + b_{if} + W_{hf}h_{t-1} + b_{hf}) \\ g_t &= \tanh(W_{ig}x_t + b_{ig} + W_{hg}h_{t-1} + b_{hg}) \\ o_t &= \sigma(W_{io}x_t + b_{io} + W_{ho}h_{t-1} + b_{ho}) \\ c_t &= f_t \odot c_{t-1} + i_t \odot g_t \\ h_t &= o_t \odot \tanh(c_t) \end{aligned} \tag{2.5}$$

where c_t and h_t will be input to the same network for the next time step with new input x_{t+1}. In practice, we usually stack the LSTM cells for many layers for more powerful learning capability. For a given sequence, we also use LSTM to process it in the forward order and the backward order (called "bi-directional" LSTM) for better performance. Attention mechanism and residual connections were also commonly used in NMT models based on LSTM networks. LSTMs are good at modeling long sequences. However, due to gradient vanishing and explosion problems, it is hard to train. And because of this recurrence in c and h, it is difficult to update the weight in LSTM in parallel during training, which makes it harder for large scale training scenarios.

Using LSTM networks, there are two majour NMT implementations.

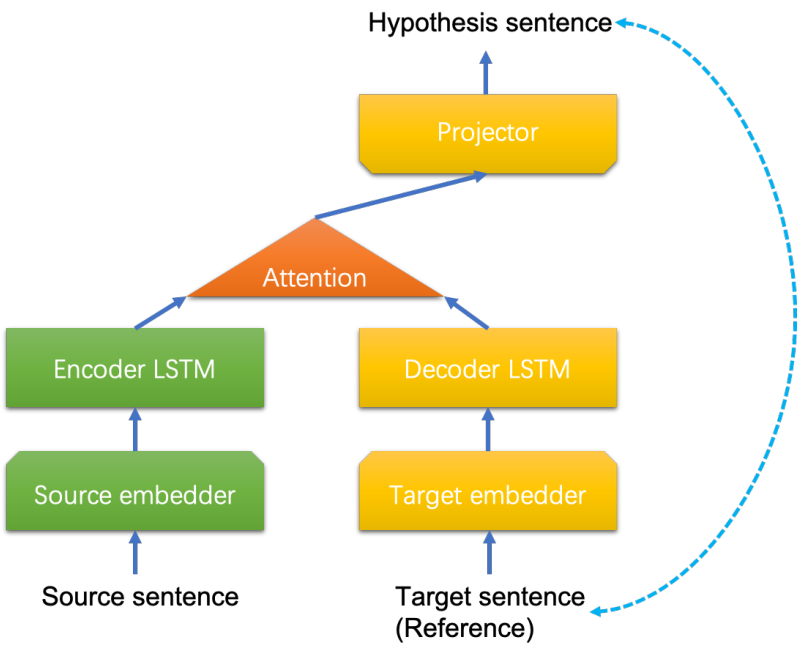


FIGURE 2.1: The structure of the RNNSearch models using LSTM network.

RNNSearch

The *RNNSearch* model is the most basic LSTM-based NMT model. It just replaces the core neural network model with two LSTM models. One is the encoder, and the other one is the decoder. An attention function is applied to the output of the decoder and the output of the encoder, in order to update the decoder output.

Figure 2.1 shows the structures of RNNSearch model. The solid blue lines indicate how the tensor flows in the model. The dotted blue lines indicate how the loss is computed. Luong et al. 2015 studied how different attention mechanism will affect NMT performance using the RNNSearch model and WMT14 dataset on English–German language pairs, which is regarded as a critical benchmark for the subsequent studies.

GNMT

The GNMT model is named after "Google's Neural Machine Translation" system (Wu et al., 2016). It utilized the following techniques that were regarded as best practices since its publication.

1. It is trained on sub-word sequence pairs (sentence piece model, which is very similar to BPE model).
2. It has one bi-directional encoder layer and six uni-directional encoder layers and eight uni-directional decoder layers.
3. It adds residual connections between LSTM layers.
4. It uses an additive attention network.
5. It uses length penalty and coverage penalty in beam search.
6. It uses learning rate scheduler during training.

Figure 2.2 showed the structure of GNMT models. The model achieved the state of the art performance at that time and was actually served at `Google Translate`. The BLEU scores reported on GNMT systems can approach human performance on several language pairs such as English–French and English–Spanish. Later, the model showed great power in multilingual NMT (Johnson et al., 2017) as well, which also known as zero-shot NMT.

2.4.2 Transformer Models

Transformer models (Vaswani et al., 2017) is another breakthrough for sequence modeling tasks.

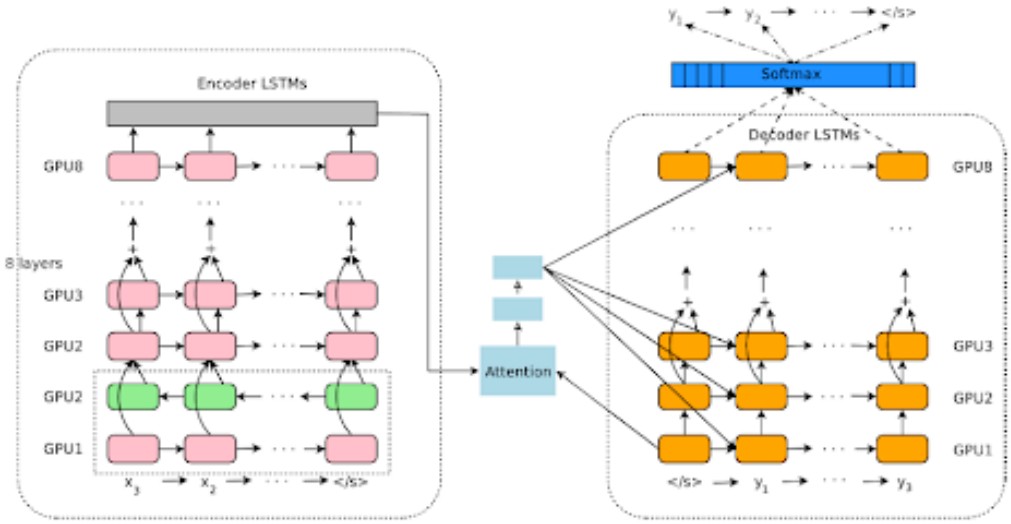


FIGURE 2.2: The structure of GNMT model. The first two layers in the encoder are forward and backward LSTM layers.

1. It abandoned the recurrent structure of RNN networks to enable full parallelism in training.

2. It intensively utilizes the attention mechanism between encoder and decoders and within encoder and decoder (called "self-attention").

3. It introduced novel positional encoding to help the model learn positional information.

4. It uses deep pointwise feed-forward networks at the end of each encoder and decoder layer.

5. It uses "layer normalization" and residual connections between each layer.

6. It uses learning rate schedules during training.

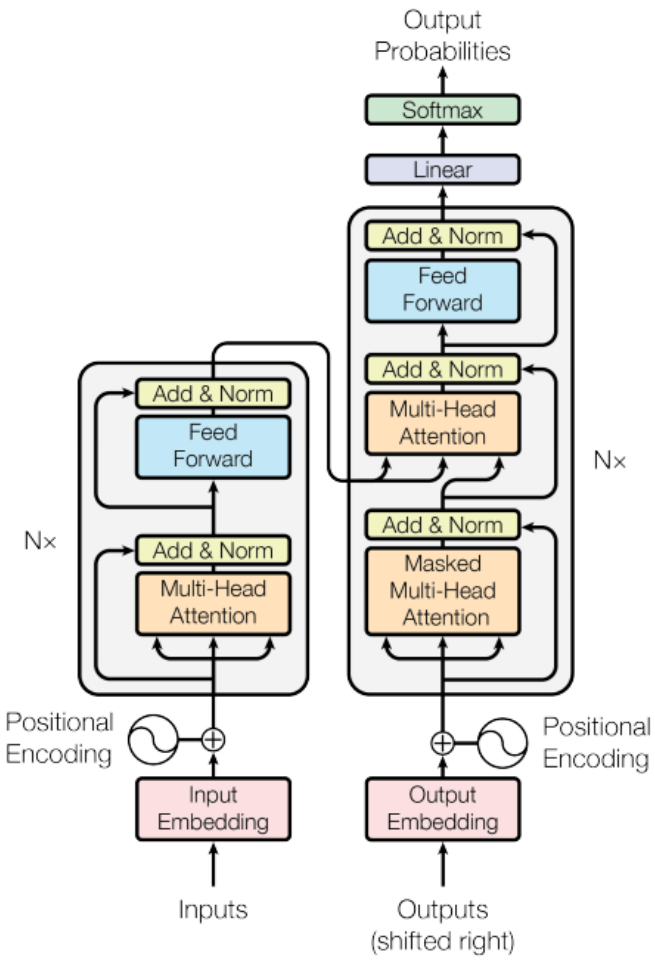


FIGURE 2.3: The structure of Transformer network.

The performance of the model in terms of BLEU scores surpass the GNMT baselines in many language pairs. What is more important, the influence of the Transformer model has gone beyond the NMT field. The famous pre-training models like "BERT" and "GPT" are all based on the Transformer network. Their performance on natural language understanding (NLU) tasks and natural language generation (NLG) tasks become the new state of the art system.

There are three novel designs in the Transformer network:

Positional embedding. The positional embedding matrix is computed by two trigonometric functions given the token position pos and the hidden index i, as shown in Equation 2.6, and then applied to normal pretrained embeddings by simple addition:

$$\begin{aligned} PE_{(pos,2i)} &= \sin(pos/10000^{2i/d_{model}}) \\ PE_{(pos,2i+1)} &= \cos(pos/10000^{2i/d_{model}}) \end{aligned} \tag{2.6}$$

where d_{model} is the model's dimensions (hidden size).

Multi-head attention. Functioning as an improved version of traditional attention mechanism (Equation 2.8), multi-head attention computes scaled attention scores on splitted *query*, *key* and *value* pairs according to Equation 2.7 (note QW_i^Q, KW_i^K and VW_i^V are Q_i, K_i and V_i projected by respective feed-forward networks) and then concatenates the results together.

$$\begin{aligned} MultiHead(Q,K,V) &= Concat(h_1,...,h_i)W^o \\ h_i &= Attention(QW_i^Q, KW_i^K, VW_i^V) \end{aligned} \tag{2.7}$$

Any single-head attention in the previous formulation uses the following formular to calculate its output:

$$Attention(Q, K, V) = softmax(QK^T/\sqrt{d_k})V \tag{2.8}$$

where d_K is the number of dimensions for a single head.

The multi-head attention (MA) that takes identical hidden states as Q, K and V is the so-called "self-attention"; and the MA that takes target states as Q, and source states as K and V are so-called "context attention," respectively.

Position-wise feed-forward network. The position-wise feed-forward network is combined by two feed-forward networks with a ReLU activation function in-between, as shown in Equation 2.9.

$$FFN(x) = max(0, xW_1 + b_1)W_2 + b_2 \tag{2.9}$$

In real implementation of transformer networks, each encoder layer contains one "self multi-head attention" and one position-wise feed-forward network; each decoder layer contains one "self multi-head attention," one "context multi-head attention" and one position-wise feed-forward network. Encoders will first embed the source sequence using source positional encoding and feed the output to a stack of encoder layers to get the encoder hidden state. The decoders will take the encoder hidden state as one input, and take the embedded target sequence as another input using target positional encoding, and then feed both of them to a stack of decoder layers to get the decoder state. Just like other NMT systems, a linear layer and a

softmax layer is used to project the decoder state to a probability distribution over target vocabulary size.

2.4.3 Other Non-auto-regressive Models

The models introduced in previous Sections are called "auto-regressive" models because they made sequential decisions on the next tokens conditioned on the previous input. However, there are some non-auto-regressive models for NMT focusing on parallelism and flexibility that are also worth noting.

Convolutional NMT Models

Figure 2.4 showed the structure of convolutional NMT models proposed by Facebook AI (Gehring et al., 2017). It is the first model to enable full parallelism on NMT training and get the stat of the art performance. Another advantage is that it can borrow ideas from computer vision research, which also extensively used CNN network in their model, such as how to deal with imbalanced training data, etc.

Levenshtein Transformer Models

Following the intuition that translation may not be generated in sequential order, there might be edits and changes before the translation is finalized, Levenshtein Transformer (Gu, Wang, and Zhao, 2019) modeled the "editing action," such as "adding," "deleting," just like the actions when calculating Levenshtein distance between two strings. Thus the NMT task can be formulated as generating a sequence of actions based on the source input.

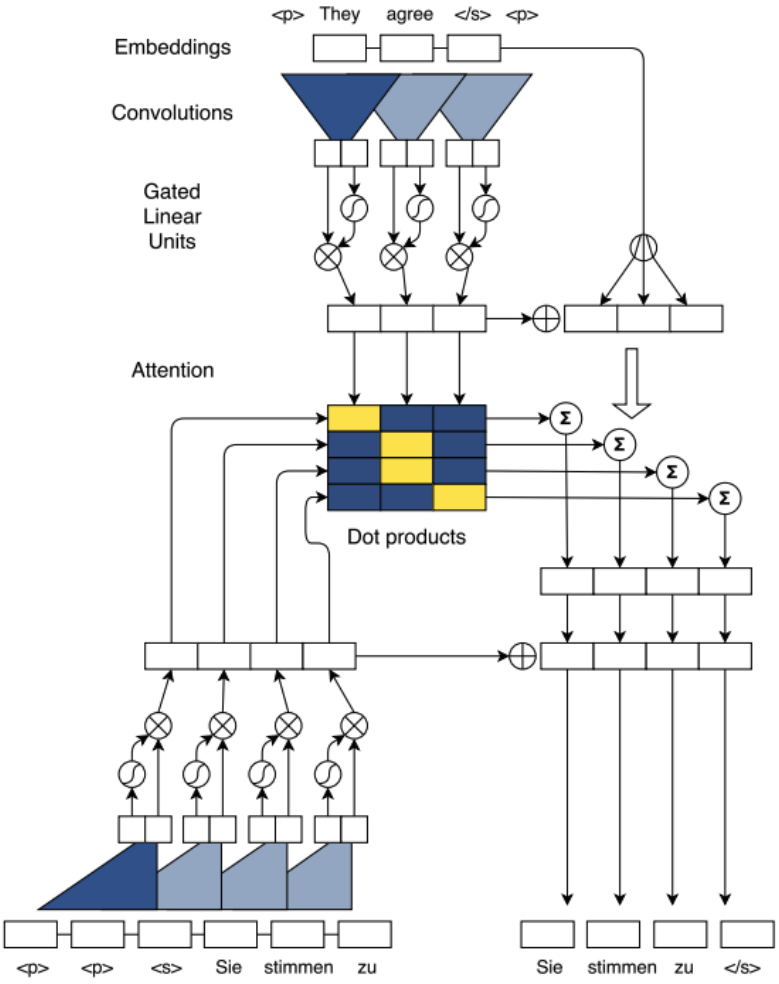


FIGURE 2.4: The structure of convolutional NMT network.

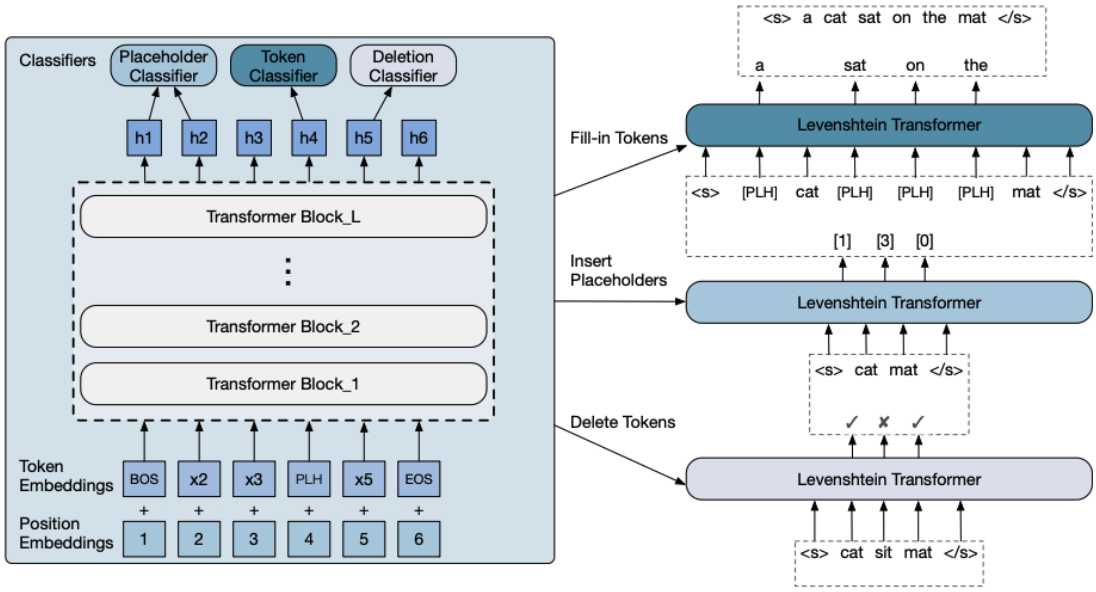


FIGURE 2.5: The structure of levenshtein Transformer network.

Levenshtein Transformer utilizes 3 actions: "deleting," "adding placeholder" and "filling placeholder". To apply these actions sequentially to a original sequence, the final hypothesized target sequence can be generated. Figure 2.5 showed the structure of levenshtein Transformer. It is especially useful if we would like to have more control of the target vocabulary.

2.5 Unsupervised NMT

One problem for NMT training is that it requires huge amount of parallel training data. Usually, this parallel alignment is annotated by human annotators, which is very expensive and time-consuming. Thus only data between major language pairs are available. It is impossible to directly train NMT models for minor language pairs simply due to lack of data.

Because NMT model training usually requires a large amount of training data, which is expensive and absent in a lot of minor language pairs, people start to think of a way to only rely on monolingual data to train an NMT model. As a special kind of self-supervised training method, the unsupervised NMT (UNMT) model pairs up two NMT models and use back-translation to generate pseudo-data for training (Lample et al., 2018; Artetxe et al., 2018).

It is formulated as follows that only takes monolingual data during training:

$$\begin{aligned} \mathcal{D}_{L1} &= \{X_1, X_2, \cdots, X_M\} \\ \mathcal{D}_{L2} &= \{Y_1, Y_2, \cdots, Y_N\} \end{aligned} \tag{2.10}$$

And during testing, when input sentence from one language, the model is asked to output a sentence from the other language with same meaning. BLEU score is also used here to evaluate the hypothesis translation with reference sentence as ground truth. Due to the missing of parallel sentences, the training of UNMT models usually resorts to self-supervised training, back-translation data augmentation, and distant supervised training techniques.

The key idea for UNMT's novel design is that during training, the model can take sequence from same lanugage as the ground truth of it's hypothesis in the same language, therefore, the losses can be calculated without parallel sentences.

Figure 2.6 showed the basics structures of UNMT models, where four training circuits can be formed. Note that the two ends of each circuits are all from one language, which we both take as input and ground truth.

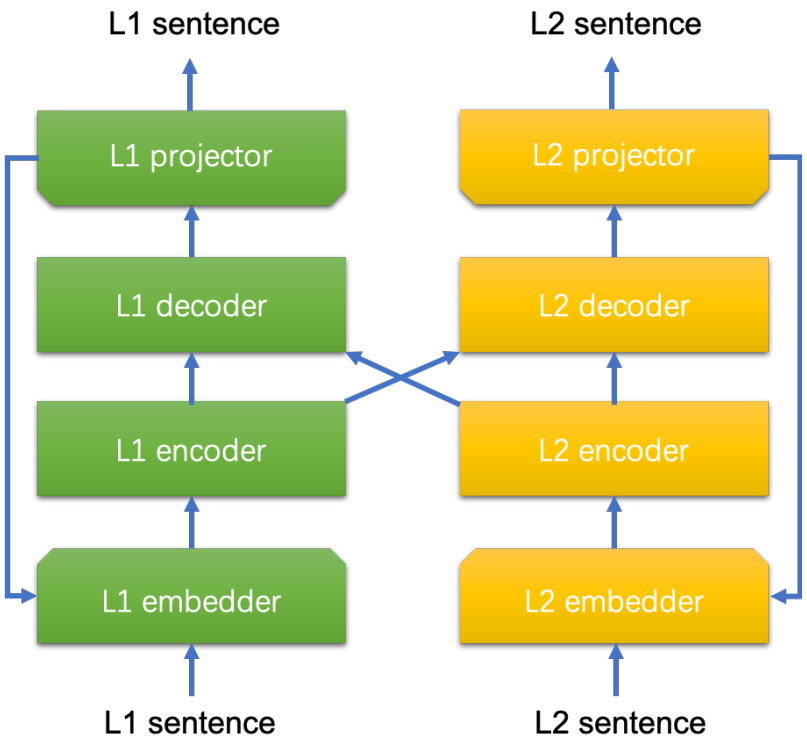


FIGURE 2.6: The structure of unsupervised neural machine translation models.

1. L1 encoder–decoder langauge model: L1 sentence → L1 encoder → L1 decoder → L1 projector → L1 sentence
2. L2 encoder–decoder langauge model: L2 sentence → L2 encoder → L2 decoder → L2 projector → L2 sentence
3. L1–L2 back translation model: L1 sentence → L1 encoder → L2 decoder → L2 projector → L2 sentence → L2 embedder → L2 encoder → L1 decoder → L1 projector → L1 sentence
4. L2–L1 back translation model: L2 sentence → L2 encoder → L1 decoder → L1 projector → L1 sentence → L1 embedder → L1 encoder → L2 decoder → L2 projector → L2 sentence

2.6 Evaluation Metrics

BLEU (Papineni et al., 2002; Post, 2018) is the most popular corpus-level evaluating metric in existing machine translation literatures. The BLEU metric considers the *n-gram* ($n = 1, 2, 3, 4$) representations of a sentence as *multiset*. For example, the sentence "`to be or not to be .`" can be represented as the following multiset when $n = 1, 2$:

$$\begin{aligned} n = 1 &: \{\texttt{to}, \texttt{be}, \texttt{or}, \texttt{not}, \texttt{to}, \texttt{be}, .\} \\ n = 2 &: \{(\texttt{to be}), (\texttt{be or}), (\texttt{or not}), (\texttt{not to}), (\texttt{to be}), (\texttt{be .})\} \end{aligned} \tag{2.11}$$

Let $n\text{-}gram(\mathcal{D})$ denote the n-gram tokens in the multiset of a given corpus $\mathcal{D}$, and $\{Y_i\}$ and $\{\hat{Y}_i\}$ denote all the tokenized sentences in reference corpus and testing

corpus, the n-gram precision of a hypothesis corpus is:

$$precision_n(\{Y_i\}, \{\hat{Y}_i\}) = \frac{|n\text{-}gram(\{Y_i\}) \cap n\text{-}gram(\{\hat{Y}_i\})|}{|n\text{-}gram(\{\hat{Y}_i\})|} \tag{2.12}$$

where $\{Y_i\}$ and $\{\hat{Y}_i\}$ are all the reference and hypothesis sentences in $\mathcal{D}$. When the length of $\{Y_i\}$ and $\{\hat{Y}_i\}$ are not equal, a length penalty LP is added:

$$LP = \min\left(\exp\left(1 - \frac{|n\text{-}gram(\{Y_i\})|}{|n\text{-}gram(\{\hat{Y}_i\})|}\right), 1\right) \tag{2.13}$$

The BLEU metric is simply the geometric mean of n-gram precision multiplies the LP:

$$BLEU(\{Y_i\}, \{\hat{Y}_i\}) = \mathbf{GM}_1^{n=4} precision_n(\{Y_i\}, \{\hat{Y}_i\}) \cdot LP(\{Y_i\}, \{\hat{Y}_i\}) \tag{2.14}$$

where **GM** refers to the geometric mean function.

Note that as we defined in Section 2.2, the $\mathcal{D}_{hyp}$ should be a corpus of strings, and $\{Y_i\}$ and $\{\hat{Y}_i\}$ should be tokenized sentences. The tokenization used by the BLEU metric is not necessarily the same with tokenization during training. Different tokenization might lead to different BLEU scores even if the actual translation string is the same. Therefore, Post et al. 2018 proposed a standard tokenization method for NMT benchmark comparisons.

Other corpus-level metrics also exist, such as

1. *GLEU*: (sentence level) the minimum of n-gram recall and precision.
2. *chrF*: (sentence level) Character n-gram F-score

3. *NIST*: (corpus level) arithmetic mean of n-gram precision with brevity penalty different from BLEU

4. *METEOR*: (corpus level) unigram F1 score based on exact matching, stem matching and WordNet matching.

2.7 Features of Logographic Languages

2.7.1 Two Writing Systems

There are two major writing systems, namely, the alphabetic writing system and the logographic writing system. The former uses letters as its basic unit. Sequential letters form words. Space-separated words form sentences. The representatives of alphabetic languages are most of the Western languages such as English, Spanish, French, etc. The latter uses strokes as its basic unit. Structured strokes form ideographs. Structured ideographs form characters. Sequential characters form words and sentences with no space inserted. The representatives of logographic languages are Chinese and Japanese.

For NMT tasks, most studies and benchmarks focus on alphabetic language pairs.

2.7.2 The Decomposibility of Logographic Characters

One crucial feature of characters in logographic languages is that they are decomposable to structured ideographs, and further to structured strokes. Specifically, strokes as the smallest units of logographic text can construct an ideograph; and

Unicode	Ideo./Char.	Composition	Meaning	Pronunciation
U+4EBB	亻	⿰丿丨	Side form of the person (人) radical.	rén
U+95E8	门	⿵丨丶冂	Door.	mén
U+4EEC	们	⿰亻门	Plural marker of person.	mén
U+7532	甲	⿱日丨	Armor.	jiǎ
U+95F8	闸	⿸门甲	Sluice.	zhá

TABLE 2.2: Composition of CJK Characters. Meanings and pronunciations are given based on simplified Chinese

ideographs can further construct a character. Ordered characters put together constitute words, and ordered words constitute sentences. Contrastively, alphabetic languages use letters as the smallest units in the text. Sequences of letters (only in left-right order) constitute words, and space-segmented words constitute sentences, which exhibit a natural boundary for words.

Ideographs usually carry meanings. They can be either semantic components to a character (or "semantic ideograph," similar to morphemes) or phonetic components that give hints on pronunciations (or "phonetic ideograph," similar to letters). Table 2.2 showed some examples of how semantic and phonetic ideograph work together. In Chinese, the ideograph "亻" and "门" combine together side-to-side to form "们," which takes the meaning of "亻" and pronunciation of "门," meaning "person (plural)". At the same time, when "门" is combined with "甲" and takes its sound, the resulting character is "闸," meaning "sluice."

Chapter 3

Method

3.1 Motivation

Data granularity has always been an important factor in the training of NMT models. Different granularities will result in different distributions in the training data of a (U)NMT task, which will significantly influence the performance. The primary motivation of this study is to create a safe alternation of the distribution of the training data in order to achieve better training performance.

Any natural-language dataset will follow the "Zipf's Law" that when the overall word frequency is plotted in descending order, the curve is logarithmic where few words have the highest frequencies, and the majority of words are in the long tail. An ordinary English dataset typically has hundreds of thousands of words in its vocabulary, whereas the English Wikipedia contains millions. For NMT models, the larger vocabulary size usually means sparser data for the long-tail words and larger networks, which are difficult to train. Because of this, usually, the low-frequency words are grouped into one type called "unknown" words (represented

Language	Granularity	Vocab Size	AWL	ASL	MSL
JA	Word	188,253	6.10	78.86	402
	Sub-word	32,000	3.29	70.02	489
	Sub-word	4,000	1.33	85.11	597
	Character	3,083	1.00	99.41	664
EN	Word	361,590	8.86	152.97	684
	Sub-word	32,000	7.04	176.77	802
	Sub-word	4,000	5.32	187.32	850
	Character	216	1.00	299.09	1,346

TABLE 3.1: Different Granularities of Alphabetic and Logographic Text in ASPEC–JE Corpus. AWL = "average word length"; ASL = "average sentence length"; MSL = "max sentence length."

by `UNK`) and let the model focusing on the rest of the common words.

However, rare words in `UNK` often denote the most important meanings in the context which can not afford to lose too much. Byte pair encoding (BPE) (Sennrich, Haddow, and Birch, 2016) was introduced first to alphabetic text to break words into sub-word units as per how likely they are shared across different words. Using the sub-word vocabulary, no `UNK` is necessary anymore since all words can be decomposed to some sub-word sequences, and the NMT model can learn to model sub-words instead of words. Because sub-word vocabulary usually has a smaller pre-determined vocabulary size, and less sparse data with shorter tails, it greatly facilitates training. On the other hand, the sub-word sequence is usually longer than word sequences, which is inefficient for training. With careful handling of sub-word vocabulary size, which makes the sequence stay at a reasonable length, the overall effect can be positive. To push this idea to an extreme, even character-based (a.k.a., letter-based) NMT was proposed (Cherry et al., 2018).

Table 3.1 shows the trend that as granularity become smaller, the vocabulary size is smaller, the sentence length become longer[1] in ASPEC–JC corpus (Nakazawa et al., 2016). However, the phenomena are less significant in logographic languages than alphabetic languages because logographic words are generally shorter than alphabetic languages (Shen et al., 2016).

As discussed in Chapter 1, these shortcomings call for sub-character level NLP. From another perspective, this can be regarded as the trade-off of the gain brought by the new training distributions.

3.2 NMT Models in this Thesis

We experimented with 3 different models using sub-character level data based on the following reasons.

1. The LSTM-based GNMT (Wu et al., 2016) model lerned contextual information and positional information both by the recurrence of the RNN networks from character level data. We want to know can it also learn and predict based on sub-character level data? Also, more importantly, can the predicted sub-character sequences be safely transformed back to characters?

2. The Transformer (Vaswani et al., 2017) model lerned contextual information from multi-head attention mechanisms, and positional information from positional encoding networks. Intuitively, the sub-character level information can be better captured by the self multi-head attention mechanisms; is it

[1]The Japanese word level tokenizer is MeCab (https://taku910.github.io/mecab/) with "IPAdic."

Ture? Also, the positional information might be challenging for the nodel to learn especially when the sub-character sequence becomes longer; is it Ture? And can Transformer model outperform GNMT model like generally?

3. UNMT model is claimed to work better on similar language pairs. Since sub-character level data exhibits higher level of shared information, which can be regared as "more similar" to each other, can UNMT model achieve better performance using sub-character level data?

The following Sections will introduce our settings on different models one by one.

3.2.1 GNMT model

Standard GNMT settings in the orginal paper were used to form the NMT system. We use one bi-directional LSTM layer in the first place and stack it with six unidirectional LSTM layers in the encoder. In the decoder, we use eight uni-directional LSTM layers. We also apply an additive attention network over the output of the first decoder layer and concatenat the attention output to the input of each of the rest decoder layers. GNMT models based on LSTM units are known to model extremely long sequences ineffectively (Stosic et al., 2017; Cherry et al., 2018). We did not have any special treatment for this. Actually, our results showed that they are sufficient for the current experiment, and we will discuss this in the next Chapter.

3.2.2 Transformer model

Transformer model (Vaswani et al., 2017) with "positional encodings" is not able to process long sequences effectively (Kitaev, Kaiser, and Levskaya, 2020). This is potentially because for shorter sequences, the vectors at each position can be easily distinguished. However, at the longer positions, the neighboring positions tend to be very similar and become more difficulty for the model to learn.

The training sequence can be very long as the granularity becomes finer. In our pilot experiments, Transformer networks performed poorly on long sequences when the model needed high accuracy of the sub-character sequence orders to compose back to characters especially near the end of predictions.

To alleviate this problem, we designed a special treatement for sub-charcter level Transformer models. We use multi-layer positional embeddings (MPE) for the Transformer during training.

Since the original positional encoding is not sufficient, one special sub-character-level position features are added to introduce additional information into the source and target embeddings. During training, we have a special vocabulary for these features and add further dimensions to the embeddings for the vocabulary. The idea is similar to the OpenNMT implementation (Klein et al., 2018) for feature NMT.

Figure 3.1 showed how the extra positional embedding is added to form the MPE to alleviate this problem. Within each sub-character sequence decomposed from the same character, two special tokens "<b>" and "<e>" are added at the beginning and ending position. The other positions are labeled in a sequential order from "<1>,"

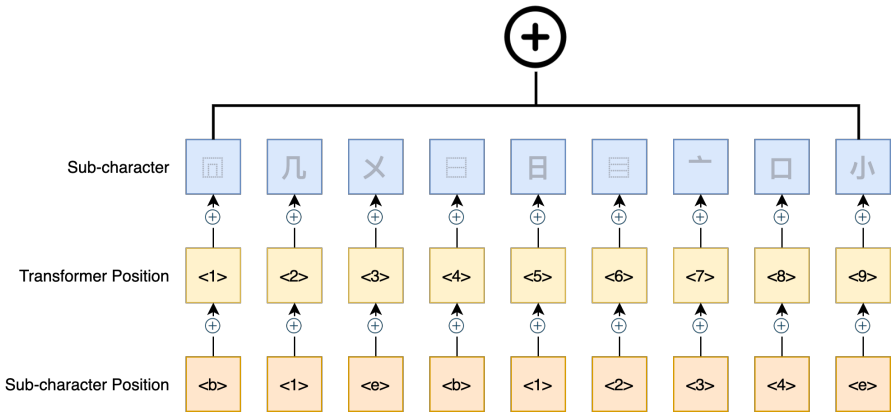


FIGURE 3.1: Extra positional embedding for sub-character sequence of "风景".

"<2>" to "<$(n-2)$>," where n is the length of the sub-character sequence. This embedding is added to the original Transformer positional embedding and finally added to the sub-character embedding. Therefore, both positions with regard to the whole sequence and internal positions of a sub-character sequence can be distinguished more clearly.

3.2.3 UNMT Model

Similar to NMT with Transformer units, we have the sub-character position features for each embedding whenever sub-character-level data are processed. In our implementation, we used a 4-layer unidirectional encoder and a 4-layer unidirectional decoder with an 8-head attention network for both self-attention and context attention. For simplicity, we do not use an adversarial discriminator.

3.3 Models and Training Hyperparameters

We trained the GNMT model, Transformer model and UNMT model following the settings in the original papers (Wu et al., 2016; Vaswani et al., 2017; Artetxe et al., 2018), except that we use greedy search for decoding in testing and add MPEs for Transformer. Some important hyperparameters are:

1. *GNMT model*. We used 8-layer encoder–decoder with both embedding dimension and hidden dimension are set to 512. The learning rate was set to 0.0002 with Adam optimizer.

2. *Transformer model*. We used 6 layer encoder–decoder with 8-head attentions. Both the embedding dimension and hidden dimension were set to 512. The MPE dimension was 10. "Noam" learning rate scheme was used with Adam optimizer. Xavior initialization (Glorot and Bengio, 2010) was applied to the weight matrix of the Transformer network before training.

3. *UNMT model*. We used 4-layer encoders and decoders and shared the first 3 layers of the encoders. The word embeddings were pre-trained by FastText.

All the trainings were done on a single GeForce GTX 1080 Ti GPU. The batch sizes and learning rates were adjusted according to the practice in Goyal et al. (2017). Character level BLEU scores for Chinese and Japanese, and word level BLEU scores for English were used as testing metrics, and "early stopping" were applied if validation BLEU score stops increasing for $500,000$ global steps.

Strokes	U+31C{0∼7}	㇀	㇁	㇂	㇃	㇄	㇅	㇆	㇇
	U+31C{8∼F}	㇈	㇉	㇊	㇋	㇌	㇍	㇎	㇏
	U+31D{0∼7}	㇐	㇑	㇒	㇓	㇔	㇕	㇖	㇗
	U+31D{8∼F}	㇘	㇙	㇚	㇛	㇜	㇝	㇞	㇟
	U+31E{0∼3}	㇠	㇡	㇢	㇣				
IDCs	U+2FF{0∼7}	⿰	⿱	⿲	⿳	⿴	⿵	⿶	⿷
	U+2FF{8∼B}	⿸	⿹	⿺	⿻				

TABLE 3.2: CJK Strokes and Ideographic Description Characters (IDC)

3.4 The Composition and Decomposition of Logographic Characters

Unicode 12.0 Standard has good representations of all these feathers for logographic characters, i.e., CJK (Chinese–Japanese–Korean) characters, together with stokes, ideographs, kana, and hangul (Korean letters). To describe the composition of characters, "Ideographic Description Character (IDC)" was introduced as symbolic representations of the structural relationship of the following two or three components (can be strokes, ideographs, or complex structures of ideographs). For example, "⿰" means two following components are placed side to side; "⿱" means three following components are placed in the up-middle-bottom structure; "⿴" refers to a surrounding structure and "⿻" refers to an intersect structure. All CJK strokes and IDCs are listed in Table 3.2.

In logographic language like Japanese, "kana" ("hiragana" and "katakana") are used as syllabaries together with characters to form the writing systems. They are syllabaries because each of them strictly corresponds to one sound in the Japanese language and can form syllables when there is no characters to use. Because the

kana system is a very accurate representation of spoken Japanese, it is often used to mark the readings of kanji or loan words as well.[2]

Cross-linguistically, similar characters and words often have similar meanings in different logographic languages (Chu, Nakazawa, and Kurohashi, 2012), but there are always many exceptions. For examples, "工程" means "engineering or project" in Chinese, but "work progress" in Japanese. "保险" in simplified Chinese and 保險" in traditional Chinese are variants of "保険" in Japanese, but in Chinese, it has several meanings, such as 1) to assure, to guarantee; 2) to be bound to; 3) safe, secure; 4) insurance; 5) certainly. And only the "insurance" meaning is used in Japanese. These mismatches add difficulty to the mutual intelligibility of different logographic languages and machine translation tasks.

In order to do sub-character level NMT/UNMT, logographic characters are first decomposed into different granularities of sub-characters and used during NMT and UNMT training. During testing, the sub-character sequences are composed back to characters.

CHISE[3] project offers a CJK character–ideograph mapping based on Unicode 12.0 Standard, which is more received and easy to reproduce[4]. The mapping contains 88,939 CJK characters. Each of them is decomposed into a sequence of

[2] https://en.wikipedia.org/wiki/Kana.

[3] http://www.chise.org, Character Information Service Environment.

[4] Comparing to previous methods such as "Wubi" (five-stroke) input method (https://en.wikipedia.org/wiki/Wubi_method), which is a Chinese input method using radicals as its input features; CNS11643 charset (http://www.cns11643.gov.tw/AIDB/welcome_en.do), which is published and maintained by the Taiwan government; and HTTPCN [4] (http://tool.httpcn.com/Zi/), etc.

Granularity	Chinese	Japanese
Word	风景	風景
Character	风 景	風 景
Ideo	⿰几乂 ⿰日京	⿰几⿱虫 ⿰日京
Ideo_finest	⿰几乂 ⿰日⿰亠口小	⿰几⿰日丿虫 ⿰日⿰亠口小
Stroke	⿰⿴丿乙⿻丿丶 ⿰⿰⿴丨冂⿰⿻一一⿰⿻丶一 ⿰⿴丨冂一⿻丨⿴丿丶	⿰⿴丿乙⿰日丿⿻⿰⿴⿴ 丨冂一⿴⿻日丨一丶⿰⿰⿴⿴ 丨冂⿰⿻一一⿰⿻日丶一⿰日⿴丨冂一 ⿻丨⿴丿丶

TABLE 3.3: Examples of sub-character decomposition in ASPEC–JC corpus

ideographs and IDCs. There are 9,986 ideographs in total, where 457 are standalone ideographs, which do not have further decompositions. To further extend our research to finer granularity, we manually decompose these ideographs into 36 Unicode CJK strokes and 12 IDCs. For Japanese Kanas, because linguistically, they are syllabaries, they should not be further decomposed theoretically. However, we decomposed it anyway and put it as one of the control groups in our experiments. Furthermore, the CHISE project uses circled numbers (① to ⑲) to represent the possible ideograph with this number of strokes when there is no Unicode glyphs available. There are 911 circled numbers in total, which are also decomposed.

There are other options for us to use in the character decomposition, such as CNS11643 charset we used in (Zhang and Komachi, 2018). The shortcomings comparing to CHISE+UNICODE is that they lack structural information in decompositions. In our empirical results, the structural information is important to distinguish similar characters and reduce the noise when composing the sub-characters back to characters.

After the decomposition process, three sub-character level granularities can be achieved:

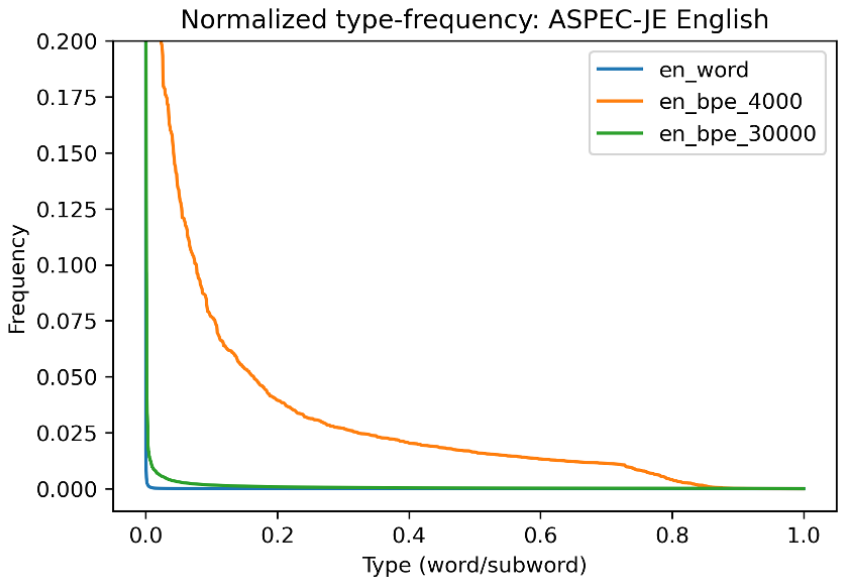
1. **Ideo**. The original ideograph decomposition data in CHISE.
2. **Ideo_finest**. After further decomposing the data in "ideo" recursively until all the ideographs in the ideograph data are stand-along ideographs.
3. **Stroke**. After further decomposing ideographs in "ideo_finest" data to strokes level.

Table 3.3 showed the decomposition examples. When granularity becomes smaller, more similar components are shared between Chinese and Japanese language. At the word level, the Japanese token "风景" and Chinese token "風景" are completely different tokens. At the character level, they share the character "景" which means 50% of the information are shared. At the sub-character level, more common parts appeared in the two sequences.

BPE is applied to all granularities of sub-character level data to get proper vocabulary before training, which is a good tool to control data granularity by setting different vocabulary sizes. The largest granularity is at the word level data, which has the largest vocabulary size. The finest granularity is at the "character" level data, which has the smallest vocabulary size[5]. Every BPE vocabulary size set in between will have intermediate granularities. If we plot the vocabulary–frequency curve, the longer tails the data exhibits, the curve will move to the left–bottom corner, and vice-versa. Figure 3.4 shows the vocabulary-frequency curves that

[5]This literarily splits every character apart. For alphabetic data, it splits based on letters; for logographic data, it split based on characters. For sub-character level logographic data, it splits based on ideographs or strokes.

English



Japanese

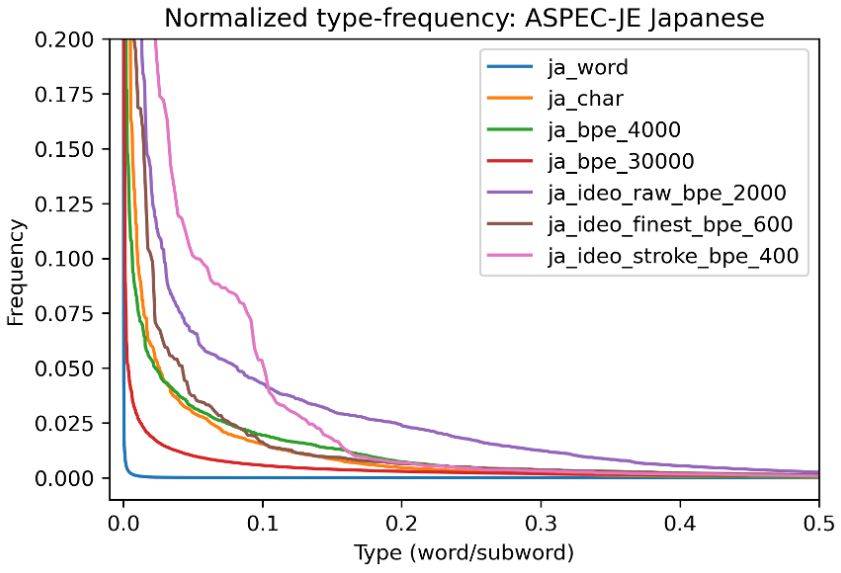


TABLE 3.4: Vocabulary–frequency graph of English, Japanese data of different granularities in ASPEC–JE corpus.

data of different granularities might have. The word frequencies and vocabulary sizes are normalized to the $(0, 1)$ range. For clearer comparison, we zoom in to $(0, 0.5)$ at Y-axis for Japanese data. The data set exhibits more long-tail tendency if the vocabulary-frequency curve is more towards the left-bottom corner, and vice-versa. For English data, when BPE is applied, the curve is moving right-upper. With smaller BPE vocabulary size, the curve is even moving further. For Japanese character level data, the tendency towards right-up corner is not so obvious in character-level data. However, when apply BPE to sub-character level data, the curve significantly moves to the right-up corner. We can see that finer granularity data tend to have less severe long-tail tendencies, and for logographic languages, sub-character level data further exhibits less severe long-tail tendency.

During testing time, sub-character sequences are composed back to characters during testing using the reverse CHISE mappings with the help of IDCs and special suffixes. IDCs helped to distinguish mappings like "叻 → ⿰口力" and "另 → ⿱口力". In addition for some minor duplicated cases, we add different number suffixes to the decomposed sequences to distinguish them, e.g., "土 → ⿱十一**_0**" and "土 → ⿱十一**_1**". Therefore, theoretically, all the decomposed sequences can be safely composed back. In practice, for those generated sub-character sequences that can not be found in the mappings, we replace them with `UNK`.

Chapter 4

Experimental Settings

4.1 NMT between Logographic Languages

We first refine our language pairs between logographic languages, such as Chinese and Japanese.

Asian Scientific Paper Excerpt (ASPEC) (Nakazawa et al., 2016) corpus contains 0.67 million Chinese–Japanese parallel sentences ("ASPEC–JC") and 3 million English–Japanese parallel sentences ("ASPEC–JE"). All the ASPEC–JC data were used to train Chinese–Japanese and Japanese–Chinese NMT models, together with 2,090 validation and 2,107 testing sentences. Chinese and Japanese text in these corpora were pre-processed into character and sub-character levels using the methods introduced in Section 3.4. Data of all granularities were tokenized by the BPE algorithm. The vocabulary size varies according to the granularities. In addition, character level data are tokenized by word tokenizers (the vocabulary is set to the most frequent 32,000 words). [1] The maximum training sequence length is set to

[1] `MeCab` for Japanese; `Jieba` for Chinese and `NLTK word tokenizer` for English.

"$mean(Lengths) + std(Lengths)$" heuristically to increase training efficiency.

Table 4.1 shows the vocabulary sizes and lengths information of all the training data, including the data used by control groups discussed in Section 4.5.

Table 4.2 showed the details of vocabulary sizes and length information of all training data in different granularity levels in ASPEC–JE dataset.

United Nations (UN) corpus contains 15 million Zh–En parallel sentences, which were used to train Zh–En and En–Zh NMT models. The size of the validation and testing set are both $4,000$.

Table 4.3 showed the details of ASPEC–JE datasets.

For Japanese–English data, we also try to apply BPE tokenization of different vocabulary size as well when necessary. But we have not implemented any data transformation method as we described in Section 4.5. It will make no sense to use character mapping between CJK characters when it pairs up with English. Also the kana decomposition method is not tested for similar reasons.

4.2 UNMT between Logographic Languages

Similar to NMT settings, we use ASPEC–JC dataset to do our experiment on UNMT settings. Although we are using the same ASPEC–JC parallel dataset comparing to previous NMT tasks, we disregard all the parallel information when we use it. We do this because we want to keep a relative fair comparison between our supervised baselines and try not to attribute our results to the noises in our

Data	Granularity	JA		ZH	
		Vocab Size	Max. Length	Vocab Size	Max. Length
Char	Word	32,000	57	32,000	42
	BPE	32,000	37	32,000	36
	Character	4,173	97	5,977	73
CharMap	Word	32,000	60	32,000	48
	BPE	32,000	37	32,000	36
	Character	4,016	97	5,539	73
Ideo	BPE	32,000	96	32,000	73
	BPE	40,00	117	4,000	75
	Character	1,811	268	2,029	268
Ideo_finest	BPE	32,000	96	32000	73
	BPE	4,000	119	4000	74
	Character	609	404	656	421
Stroke	BPE	32,000	96	32,000	73
	BPE	4,000	126	4,000	75
	Character	398	739	425	884
Stroke_decomp	BPE	32,000	96	*N/A*	*N/A*
	BPE	4,000	129	*N/A*	*N/A*
	Character	398	807	*N/A*	*N/A*

TABLE 4.1: Statistics of ASPEC–JC corpus.

Data	Granularity	JA Vocab Size	JA Max. Length	EN Vocab Size	EN Max. Length
Char	Word	188253	145	361590	113
	BPE	32000	166	32000	180
	Character	3000	220	214	594
Ideo	BPE	32000	321	32000	180
	BPE	3000	397		
Ideo_finest	BPE	32000	321		
	BPE	3000	398		
Stroke	BPE	32000	321		
	BPE	3000	402		

Table 4.2: The details of Japanese–English dataset (ASPEC–JE).

Data	Granularity	JA Vocab Size	JA Max. Length	EN Vocab Size	EN Max. Length
Char	Word	188253	502	361590	174
	BPE	32000	512	32000	202
	Character	3080	634	214	489
Ideo	BPE	32000	503	32000	180
	BPE	3000	995		
Ideo_finest	BPE	32000	503		
	BPE	3000	986		
Stroke	BPE	32000	503		
	BPE	3000	996		

Table 4.3: The details of Chinese–English dataset (UN corpus).

Algorithm 1: Sharing Rate Sampling

Data: source/target sentences
Input: r, k, N
Output: source/target sentences with r sharing rate ($sample$)
Init: $current\_r, vocab, shared\_vocab, sample$;
while $len(sample) < N$ **do**
    $current\_sample \sim$ randomly sample $8 \times k$ sentences;
    calculate sentence level sharing rate s_r based on $shared\_vocab$;
    sort $sample$ in descending order of s_r;
    if $current\_r < r$ **then**
        select top k sentences;
    else
        select bottom k sentences;
    end
    add selected sentences to $sample$;
    update $current\_r, vocab, shared\_vocab$;
    remove $current\_sample$ from datasets;
end

training data. One shortcomings for such usage is that the number of training data is fewer than that is described in the original paper in Lample et al. 2018.

4.3 Controlling Shared Tokens

Lample et al. (Lample et al., 2018) have successfully made 95% of the BPE tokens in English–German language pair shared across the training set, indicating that the more proportion of token sharing, the better the UNMT systems will perform. This study sampled from same dataset with controlled token sharing rate to get a better understanding of this notion. Algorithm 1 takes controlled token sharing rate r, top-k value k and sample size N as parameters.

4.4 Experiments and Expected Results

Based on the previous settings, the following experiments will be done:

4.4.1 NMT

GNMT and Transformer models with extra positional codings for sub-characters will be trained on Chinese–Japanese, Chinese–English, and Japanese–English data, respectively. These experiments will mainly investigate the effectiveness of sub-characters in NMT tasks between logographic language pairs and alphabetic-logographic language pairs.

Moreover, within each division of writing systems, different granularities of the data were tested. We tested from larger granularities, such as word and character-level data, to smaller granularities, such as ideographic and stroke level data. Under each granularity, we try first to have a large vocabulary size generated from the BPE algorithm that ties every granularity, and also a smaller vocabulary size that approaches to the size of characters in that granularity level. Thus we can try our best to distinguish the effect brought by sub-character level data: Whether it is because of the smaller vocabulary size or because of the more shared information?

The expected result would be the finer granularity the data is, the better the performance would be. And both smaller vocabulary size and higher shared information facilitate the training.

4.4.2 UNMT

Sub-character level UNMT The baseline is a UNMT system trained on Chinese–Japanese monolingual data, which are first pre-tokenized into words, and then BPE'ed using fastBPE. [2] We call it character level baseline because no sub-character level units are involved. The experiments are to compare it against UNMT systems trained on sub-character level data, which are directly decomposed from character-level data and then BPE'ed using fastBPE. In sub-character level data, the presence of structural information was also controlled by adding or removing IDCs.

UNMT with different token sharing We sampled data ($N = 300,000$) from same monolingual corpus using Algorithm 1 with controlled token sharing rate r of 0.5, 0.7 and 0.9, respectively. This is because UNMT systems trained on stroke level data with IDCs achieved the best performance in preliminary experiments.

For pre-tokenization of the data: Jieba [3] was applied to Chinese using the default dictionary; MeCab[4] was applied to Japanese using the IPA dictionary. For BPE training, the vocabulary size was set to 30,000. We use 4-layer standard Transformer (Vaswani et al., 2017) units as our two encoders and decoders. The embedding size was 512; the hidden size of the fully connected network was 2048; the weights of the last 3 layers of the encoders are shared; the number of multi-attention head was 8. During training, the dropout rate was set to 0.1, and both vocabularies and embeddings were shared. 10% of input and output sentences were

[2] https://github.com/glample/fastBPE
[3] https://github.com/fxsjy/jieba
[4] http://taku910.github.io/mecab/

randomly blanked out to add noise to language model training. We used Adam optimizer with a learning rate of 0.0001.

4.5 Control Groups

To further validate our result, there are three control groups from different perspectives:

1. *Logographic Language Pairs vs. Alphabetic–Logographic Language Pairs.* We would like to see whether the sub-character information only facilitates logographic language pairs or even only with one side of the data has logographic languages; the model can also get some benefit from it. The expected result would be in both cases, sub-character level data can facilitate the training.

2. *Chinese–Japanese Character Mapping* ("Charmap"). As we discussed before, logographic languages often share a large proportion of characters, but sometime exhibit confusing meanings. We would like to test if we map all Chinese characters into Japanese counterpart as in Chu et al.(2012), can we get better performance or not. The expected result would be no better performance can be observed from similar experiment settings.

3. *Kana Decomposition Data* ("kana"). Although kanas are syllabaries in Japanese, we still decompose them to stroke level for comparison. We would like to test if the model can learn further longer sequences or not.

Chapter 5

Results

5.1 NMT between Logographic Language Pairs

Table 5.1 shows the results of the GNMT model and Transformer model on ASPEC–JC corpus. Almost all scores from Transformer models outperform that of GNMT models. Models trained on sub-character level data consistently outperform character level baselines, whenever they have equal or smaller vocabulary sizes. Within sub-character level data, generally, the finer the granularity, the better the performance. Exceptions are mainly found on data with long sequence lengths. Models trained on character-mapping data do not have a significant performance difference comparing to character data. Models trained on kana decomposition data do not have a significant performance difference comparing to stroke data. [1]

[1]Since kana decomposition is only applied to Japanese text, we pair it with Chinese stroke level data and train our models. Therefore the "N/A" is in the Chinese vocabulary size and maximum lengths cells.

Data	Granularity	JA–ZH BLEU		ZH–JA BLEU	
		GNMT	Transformer	GNMT	Transformer
Char	Word	38.11	38.96	45.22	47.66
	BPE	42.02	44.69	47.01	49.15
	Character	45.84	47.02	50.77	52.09
CharMap	Word	34.13	34.79	41.13	42.86
	BPE	41.88	43.37	47.15	48.81
	Character	45.80	47.73	49.20	51.83
Ideo	BPE	46.30	47.76	50.40	52.98
	BPE	46.90	48.57	51.99	**53.79**
	Character	44.93	46.91	51.09	52.75
Ideo_Finest	BPE	46.41	47.61	50.69	52.04
	BPE	46.75	**49.02**	50.80	52.99
	Character	44.39	46.72	49.95	51.67
Stroke	BPE	46.57	48.11	**50.83**	51.16
	BPE	**46.81**	48.45	49.05	53.37
	Character	43.26	45.72	48.51	49.70
Kana	BPE	46.43	48.67	50.53	51.25
	BPE	45.21	48.19	47.34	48.42
	Character	38.64	40.19	42.98	43.90

TABLE 5.1: BLEU scores of NMT models using ASPEC–JC data.

Data	JA–EN		EN–JA	
	GNMT	Transformer	GNMT	Transformer
Char	24.47	25.62	37.90	39.18
Ideo	26.22	27.49	38.15	39.17
Ideo_finest	**26.45**	**27.72**	**38.92**	**39.98**
Stroke	26.22	27.43	38.37	39.94
Data	ZH–EN		EN–ZH	
	GNMT	Transformer	GNMT	Transformer
Char	28.28	29.55	44.12	45.58
Ideo	30.53	32.09	46.33	47.21
Ideo_finest	30.59	32.17	46.45	47.52
Stroke	**31.19**	**31.87**	**46.75**	**47.70**

TABLE 5.2: BLEU scores of NMT models of JA–EN and ZH–EN data. All data are tokenized by BPE model with vocabulary size of 32,000.

5.2 NMT between Logographic and Alphabetic Language Pairs

Table 5.2 shows the results of the GNMT model and Transformer model on ASPEC–JE and UN ZH–EN corpus. Models trained on sub-character level data usually outperform models trained on character level data. Within the sub-character groups, the finer the granularity, the better the performance.

Data	Granularity	JA–ZH	ZH–JA
Char	BPE.32000	32.38	41.02
Ideo_raw	BPE.32000	33.22	41.91
	BPE.2300	35.60	43.88
Ideo_finest	BPE.32000	33.91	42.20
	BPE.800	37.54	44.39
Stroke	BPE.600	35.82	42.35

TABLE 5.3: The BLEU scores of UNMT model under different granularities.

r	JA–ZH	ZH–JA
0.5	19.72	25.23
0.7	23.60	28.32
0.9	23.04	28.84

TABLE 5.4: The BLEU scores of different token sharing rate on test set.

5.3 UNMT between Logographic Language Pairs

5.3.1 Sub-character Level UNMT

Table 5.3 showed the UNMT model's performance of different granularities. Generally speaking, the finer granularity the data, the better the performance. Best performance was observed at "Ideo_finest" level data with BPE vocabulary size of 800.

5.3.2 UNMT with Different Share Token Rate

Table 5.4 shows the results for UNMT systems using data of different share token rate. When $r = 0.5$, the system recorded the lowest performance; however, when

r increased to 0.7 and 0.9, the performance differences become minor. In contrast with Lample et al. (2018), in our previous sub-character experiments, only 66% to 68% of the tokens are shared and can get a relatively good BLEU score.

Chapter 6

Discussions

6.1 Model Trained on Finer, Sub-character Level Data Performed Better

We have experimented on both character and sub-character level data in logographic–logographic and logographic–alphabetic language pairs using GNMT, Transformer, and UNMT models. In almost all of these scenarios, we found that finer granularity data can facilitate training and got higher BLEU scores (despite that the UNMT model does not converge for logographic–alphabetic language pairs). On one hand, the results confirmed that on character level, data tokenized by BPE models and by character outperform the word tokenizers; on the other hand, it showed that sub-character level data could further improve the performance.

Are the improvements coming from the finer sub-character level granularity or simply from the reduction of vocabulary size? We believe in both. When controlling

the vocabulary size to $32,000$ (BPE), sub-character models outperform the character models. What's more, because sub-character data can naturally achieve smaller vocabulary size, when balancing it with sequence length, the models usually get even better performance. We also believe that the worse performance of some finest level data, such as "stroke" data tokenized by character, might be because the sequence is too long for the model to learn, and the learning took much longer time. From Table 5.1, roughly when the maximum length of the training sequence is longer than 130, the performance would be harmed. This is also reported as "gradient vanishing" problem for LSTM neural networks (Koehn and Knowles, 2017) and smaller distinctive features in latter positions in Transformer positional embedding (Kitaev, Kaiser, and Levskaya, 2020).

There are different preferences over what level of sub-character level data to generate the best performance over different models. The best scores are observed for "ideo_finest" level data when the target language is Chinese or Japanese, and "stroke" level when it is English. Considering that the Transformer models always outperform LSTM models on all the tasks, the improvements we achieved are highly promising. Further, Table 5.2 showed that when only one side of the training data is at sub-character level, similar patterns of improvements can be achieved. This suggests that our method can also be used in the BERT (Gehring et al., 2017) or GPT (Radford et al., 2019) models, where only encoder and decoder networks are used. Finally, despite good performance on logographic language pairs, our UNMT models on Japanese–English and Chinese–English did not converge. This confirms previous observation (Artetxe et al., 2018) that current UNMT models cannot handle remote languages well because they rely heavily on the shared information

between two vocabularies.

However, the difference between the "stroke" model and the "ideograph" level model is that the lengthy sequences of data with finer granularity can complicate Transformer decoding. Taking Transformer NMT models as examples, when English is on the target side, the performance of all models increased as the granularity became finer and the "stroke" level data performed the best. Under this condition, the decoding sequence is always English sub-word sequences of which the lengths are not substantial. This enables the model to exploit the finer granularity of the data fully, resulting in improved learning. On the other hand, when Chinese or Japanese were on the target side, the model was ineffective on "stroke" level data. The best performance was obtained with "ideo_finest" level data. This might be because Transformer units were unable to process long sequences satisfactorily (Dai et al., 2019); i.e., although "stroke" level data were easier to learn, they were more difficult to decode.

The improvements could be attributed to the increase in the share token rate and smaller vocabulary size. As mentioned before, data with finer granularity naturally exhibit a higher share token rate. Our results appear to suggest a strong relationship between the share token rate and the performance of the model when the character level and sub-character level data had the same vocabulary size. When sub-character level data have smaller vocabulary sizes, they often resulted in better performance after striking a balance with the deficits of longer sequence lengths.

Data	JA–ZH Sharing	Sharing within JA	Sharing within ZH
Char	60.28%	52.70%	30.79%
Ideo	68.76%	48.53%	42.73%
Ideo_finest	81.36%	62.26%	63.82%
Stroke	87.74%	71.14%	75.98%

TABLE 6.1: Information sharing of different granularity data of ASPEC–JC corpus.

6.2 Higher Level of Information Sharing in Finer Granularity Data

The token sharing can happen across different languages, but can also happen within one language. When we calculate cross-language share token rate, we calculate how many tokens both appearing in both text. When we calculate the share token rate within one language, we usually focus on specific sentences and look for the duplicated parts in that sentence.

Finer granularity data exhibit a higher level of information sharing between and within each training sequence, which might be the reason for better NMT performance. Table 6.1 calculates the percentage of information sharing by counting how many tokens in the sequence are duplicated. For example, in the sequence "`[a, b, c, d, a]`," "`a`" appeared twice, so that the percentage of information sharing is $2/5 = 40\%$. As the granularity became finer, both information shared across the source and target text as well as within source or target text increased. This seems to suggest a high correlation with the increasing performance of NMT models. Intuitively, a higher proportion of shared information can facilitate the model to better learn the mappings between input and output, which may collectively

Granularity	JA–ZH	+ MPE (difference)	ZH–JA	+ MPE (difference)
Character	48.65	48.45 (−0.20)	55.49	55.32 (−0.17)
Ideo	47.21	50.07 (+2.86)	53.91	56.77 (+2.68)
Ideo_finest	34.56	51.11 (+16.55)	41.88	57.28 (+15.40)
Stroke	23.19	49.06 (+25.87)	29.23	52.49 (+23.26)

TABLE 6.2: Performance of Transformer models with/without MPE. BLEU score reported on ASPEC–JC corpus.

contribute to the higher BLEU scores.

6.3 Orders within Sub-character Sequence can be Mostly Learned

How can we guarantee that all the outputs can be transform back to characters safely, especially in the sub-character situation? Theoretically, it is possible that the sub-character sequence produced by the model can not be composed to the original character for BLEU computing, because just one wrong prediction in the sub-character sequence can lead to mistakes. However, in real practice we found that the absolute majority of characters can be recovered, which means the model learns the sub-character order very well. For example, in the Chinese–Japanese ideograph level GNMT model using 32000 vocabulary size, 99.57% of the characters in the hypothesis can be safely converted back to character level data. We think the BLEU scores itself could be a good indicator of the successful transforming. But it is still possible that in finer granularities, the successful transforming rate becomes a little bit lower.

The extra multi-layer positional embedding (MPE) we proposed in Section 3.2.2

was very important as Table 6.2 showed how the BLEU scores dropped dramatically without MPE.

6.4 Character-mapping and Kana Decomposition

In statistical machine translation (SMT), character-mapping between Japanese kanjis and Chinese characters plus a carefully designed dictionary can have better performance (Chu, Nakazawa, and Kurohashi, 2012). We use the same character-mapping to pre-process our character level data as one control group in experiments. However, according to Table 5.1, its performance is not significantly different to the models trained on the original character-level data. We believe this is because character-mapping might increase the similarity between the source and target text, it also introduced uncontrollable noise because the same characters and words might have different meanings in different languages. Interestingly, the reduction of character/kanji types increased the vocabulary size of the word-level data. In SMT systems, a larger phrase table is an advantage, which might be the reason for its gain in BLEU score but not in NMT. Similarly, kana decomposition does not help Japanese stroke level NMT much since the training data became much longer and noisier.

6.5 IDCs

IDCs are also one factor that we can successfully train our models on sub-character level information. Zhang et al. (Zhang and Komachi, 2018) has pointed out that

Granularity		JA–ZH	ZH–JA
Character		24.18 *(29.60)*	29.79 *(40.00)*
Ideograph	w/ IDCs	25.76*	32.61*
	w/o IDCs	25.14* *(32.00)*	32.17* *(42.60)*
Stroke	w/ IDCs	**26.39***	**32.99***
	w/o IDCs	24.75* *(32.10)*	30.59* *(42.20)*

TABLE 6.3: BLEU scores ($*$ for statistically significant score against baseline at $p < 0.0001$) of UNMT (larger fonts) and supervised NMT systems (Zhang and Komachi, 2018) (smaller fonts in brackets) on test sets.

without IDCs in character decomposition, there will be many more possible duplications in sub-character sequence. In our experiments, this point is confirmed and if we trained on data without IDCs, usually poor performance will be got.

Table 6.3 showed the performance with/without IDCs in Chinese–Japanese Language pairs.

6.6 Shared Information and Proportion of Shared Tokens in UNMT

Zhang et al. (2018) showed that shared information brought by sub-character level information could help supervised NMT systems; this study found a similar phenomenon, although with different granularity preference.

Table 6.4, Table 6.5 and Table 6.6 showed the examples from all the models that we tested. From each tables we listed the ground truth parallel sentences, the models

Type	Lang.	Sentence
Ground Truth	JA	図3に「会」が固有表現であるか否かを判定する2つの例文を示した.
	ZH	图3所示的是2个关于判断“会”是否是固有表达的例句。
Ideo GNMT	JA	図3に,「会」が固有表現であるかどうか判断する例文を2つ示す.
	ZH	图3表示的是判定“会”是固有表现的2个例句。
Stroke GNMT	JA	図3に,「会」が固有表現の有無を判定する例文を2つ示す.
	ZH	图3显示了判断“会”是否是固有表达的2个例句。
English		Figure 3 shows two example sentences that were used to judge whether “会” is an inherent expression.

Table 6.4: Translation examples from GNMT models.

translations from both directions and the English translation. For brevity, we only listed the ideo level models and stroke level models in each experiment settings.

Despite that most of translations are fluent and accurate, we can still see some differences from the translations. For example, in Table 6.6, the stroke model was even slightly better than the ideograph model because it translated Japanese “表現” (“expression”) into Chinese “表达” (“expression”), which was more precise than ideograph model’s “名词” (“noun”). Also, in Table 6.5, the stroke model was even correctly translated Japanese “判定する” (“judge”) into Chinese “判断” (“judge”), ideograph model’s translation “判定” (“determine”).

However, current unsupervised models still perform poorly on distant language pairs. If the shared information between distant languages can be improved, UNMT

Type	Lang.	Sentence
Ground Truth	JA	図3に「会」が固有表現であるか否かを判定する2つの例文を示した.
	ZH	图3所示的是2个关于判断“会”是否是固有表达的例句。
Ideo Transformer	JA	図3に,「会」が固有表現であるかどうか否かを判断する例文を2つ示す.
	ZH	图3表示的是2个判定“会”是固有表现的例句。
Stroke Transformer	JA	図3に,「会」が固有表現であるかどうか否かを判定する例文を2つ示す.
	ZH	图3显示了2个判断“会”是否是固有表达的例句。
English		Figure 3 shows two example sentences that were used to judge whether “会” is an inherent expression.

TABLE 6.5: Translation examples from 3 unsupervised NMT models in 6 translation directions.

may work for a more general purpose. Additionally, the low proportion of shared tokens can harm the performance, but the high proportion does not linearly improve the performance either.

6.7 Translation Quality in UNMT

In both translation directions, there are a lot of synonymous expressions produced which lowered the BLEU score. However, according to native speakers' judgement, they tend to be good translations in terms of grammaticality, fluency, and naturalness. For example in Table 6.6, the character-level system's Chinese translation “中 显示” (“in which shows”) was very close to the reference “所示” (“as shown in”) semantically, and was consistent in ideograph- and stroke-level models. A

Type	Lang.	Sentence
Ground Truth	JA	図3に「会」が固有表現であるか否かを判定する2つの例文を示した.
	ZH	图3所示的是2个关于判断“会”是否是固有表达的例句。
Ideo UNMT	JA	図3に示すように2つの判断「会」が固有表現であるかどうかについての例文を示す.
	ZH	图3中显示了判定“会”是否是固有名词的2个例句。
Stroke UNMT	JA	図3に示すのは,2つの判断について「会」が固有表現の例文であるかどうかである
	ZH	图3中显示了判定“会”是否是固有表达的2个例句。
English		Figure 3 shows two example sentences that were used to judge whether “会” is an inherent expression.

Table 6.6: Translation examples from 3 unsupervised NMT models in 6 translation directions.

similar example would be “判断” (“judge”) in reference and “判定” (“determine”) in the hypothesis. This might be due to the encoder-decoder language models, which successfully grasp the language features and express them in the translation. Consequently, if semantic metrics could be introduced, the performance of unsupervised NMT might be better reflected.

Chapter 7

Conclusion

7.1 Concluding Remarks

This study showed that sub-character level information can be successfully applied to Transformer NMT models and UNMT models, with sub-word position features applied to the model. This study also tested the difference between levels of granularity in the data and found that when the training sequence is not too long, the models can learn better from finer granularity data. All these improvements might be attributed to the naturally higher shared information between and within the source and target text.

The topic of shared information has not yet been studied in detail. A method capable of increasing the this information should not only be applicable to data preparation but also to training techniques and model design. Additional future work in this direction is expected to be promising.

7.2 Future Works

In the future, we would like to discover more about the share token rate and based on that find better ways of utilizing the NMT training data. For example, can we better use share token rate information in designing better learning curriculum? Also, we would like to find more on how sub-character data can help with the down stream tasks, such as BERT, in NLU and NLG tasks.

Appendix A

List of Publications

- Longtu Zhang and Mamoru Komachi (2020), Using Sub-Character Level Information for Neural Machine Translation of Logographic Languages, in ACM Transactions on Asian and Low-Resource Language Information Processing (to appear)
- Longtu Zhang and Mamoru Komachi (2019). Chinese-Japanese Unsupervised Neural Machine Translation Using Sub-character Level Information. In Proceedings of the 33rd Pacific Asia Conference on Language, Information and Computation, Sep, 2019
- Yuting Zhao, Longtu Zhang and Mamoru Komachi (2019). Application of Unsupervised NMT Technique to Japanese-Chinese Machine Translation. In Proceedings of the 33rd Annual Conference of the Japanese Society for Artificial Intelligence, June 6, 2019
- Longtu Zhang and Yuting Zhao and Mamoru Komachi (2018). TMU Japanese-Chinese Unsupervised NMT System for WAT 2018 Translation Task. In

Proceedings of the 5th Workshop on Asian Translation, HongKong, China. December 3, 2018

- Longtu Zhang and Mamoru Komachi (2018). Neural Machine Translation of Logographic Language Using Sub-character Level Information. In Proceedings of the Third Conference on Machine Translation, Brussels, Belgium. October 31 – November 1, 2018 (Acceptance rate 28.7%)

- Longtu Zhang (2016). Short-long/Long-short Preferences in English/Japanese Processing Revisited. In Young Researchers Symposium on Natural Language Processing (YRSNLP) 2016, Osaka, Japan. December 10, 2016

Bibliography

Artetxe, Mikel et al. (2018). "Unsupervised Neural Machine Translation". In: *Proceedings of the 6th International Conference on Learning Representations*.

Bahdanau, Dzmitry, Kyunghyun Cho, and Yoshua Bengio (2015). "Neural Machine Translation by Jointly Learning to Align and Translate". In: *Proceedings of the third International Conference on Learning Representations*.

Bengio, Samy et al. (2015). "Scheduled Sampling for Sequence Prediction with Recurrent Neural Networks". In: *Proceedings of the 28th International Conference on Neural Information Processing Systems*, pp. 1171–1179.

Bengio, Yoshua et al. (2009). "Curriculum learning". In: *Proceedings of International Conference on Machine Learning*. Vol. 382. ACM International Conference Proceeding Series, pp. 41–48.

Brown, Peter F. et al. (1993). "The Mathematics of Statistical Machine Translation: Parameter Estimation". In: *Computational Linguistics* 19.2, pp. 263–311.

Cherry, Colin et al. (2018). "Revisiting Character-Based Neural Machine Translation with Capacity and Compression". In: *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pp. 4295–4305.

Cho, Kyunghyun et al. (2014). "Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation". In: *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing*, pp. 1724–1734.

Chu, Chenhui, Toshiaki Nakazawa, and Sadao Kurohashi (2012). "Chinese Characters Mapping Table of Japanese, Traditional Chinese and Simplified Chinese". In: *Proceedings of the Eighth International Conference on Language Resources and Evaluation*, pp. 2149–2152.

Dai, Zihang et al. (2019). "Transformer-XL: Attentive Language Models beyond a Fixed-Length Context". In: *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pp. 2978–2988.

Gehring, Jonas et al. (2017). "Convolutional Sequence to Sequence Learning". In: *Proceedings of International Conference on Machine Learning*. Vol. 70, pp. 1243–1252.

Glorot, Xavier and Yoshua Bengio (2010). "Understanding the difficulty of training deep feedforward neural networks". In: *Proceedings of the thirteenth international conference on artificial intelligence and statistics*. Vol. 9, pp. 249–256.

Goyal, Priya et al. (2017). "Accurate, Large Minibatch SGD: Training ImageNet in 1 Hour". In: *CoRR* abs/1706.02677.

Gu, Jiatao, Changhan Wang, and Junbo Zhao (2019). "Levenshtein Transformer". In: *Proceedings of Advances in Neural Information Processing Systems*. Vol. 32, pp. 11181–11191.

Hochreiter, Sepp and Jürgen Schmidhuber (1997). "Long Short-Term Memory". In: *Neural Computation* 9.8, pp. 1735–1780.

Hutchins, W. J. (1986). *Machine Translation: Past, Present, Future*. John Wiley and Sons, Inc. ISBN: 0-470-20313-7.

Johnson, Melvin et al. (2017). "Google's Multilingual Neural Machine Translation System: Enabling Zero-Shot Translation". In: *Transactions of the Association for Computational Linguistics* 5, pp. 339–351.

Kitaev, Nikita, Lukasz Kaiser, and Anselm Levskaya (2020). "Reformer: The Efficient Transformer". In: *Proceedings of International Conference on Learning Representations*.

Klein, Guillaume et al. (2018). "OpenNMT: Neural Machine Translation Toolkit". In: *Proceedings of the 13th Conference of the Association for Machine Translation in the Americas*, pp. 177–184.

Kocmi, Tom and Ondřej Bojar (2017). "Curriculum Learning and Minibatch Bucketing in Neural Machine Translation". In: pp. 379–386.

Koehn, Philipp (2010). *Statistical Machine Translation*. Cambridge University Press. ISBN: 978-0-521-87415-1. URL: http://www.statmt.org/book/.

Koehn, Philipp and Rebecca Knowles (2017). "Six Challenges for Neural Machine Translation". In: *Proceedings of the First Workshop on Neural Machine Translation*, pp. 28–39.

Kudo, Taku (2018). "Subword Regularization: Improving Neural Network Translation Models with Multiple Subword Candidates". In: *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics*, pp. 66–75.

Lample, Guillaume et al. (2018). "Phrase-Based & Neural Unsupervised Machine Translation". In: *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pp. 5039–5049.

Luong, Thang, Hieu Pham, and Christopher D. Manning (2015). "Effective Approaches to Attention-based Neural Machine Translation". In: *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pp. 1412–1421.

Nakazawa, Toshiaki et al. (2016). "ASPEC: Asian Scientific Paper Excerpt Corpus". In: *Proceedings of the Tenth International Conference on Language Resources and Evaluation*.

Papineni, Kishore et al. (2002). "BLEU: a Method for Automatic Evaluation of Machine Translation". In: *Proceedings of the 40th annual meeting of the Association for Computational Linguistics*, pp. 311–318.

Platanios, Emmanouil Antonios et al. (2019). "Competence-based Curriculum Learning for Neural Machine Translation". In: *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pp. 1162–1172.

Post, Matt (2018). "A Call for Clarity in Reporting BLEU Scores". In: *Proceedings of the Third Conference on Machine Translation: Research Papers*, pp. 186–191.

Radford, Alec et al. (2019). "Language Models are Unsupervised Multitask Learners". In:

Sennrich, Rico, Barry Haddow, and Alexandra Birch (2016). "Neural Machine Translation of Rare Words with Subword Units". In: *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics*.

Shen, Mo et al. (2016). "Consistent Word Segmentation, Part-of-Speech Tagging and Dependency Labelling Annotation for Chinese Language". In: *Proceedings*

of the 26th International Conference on Computational Linguistics: Technical Papers, pp. 298–308.

Stosic, Dusan et al. (2017). "QRNN: q -Generalized Random Neural Network". In: *IEEE transactions on neural networks and learning systems* 28.2, pp. 383–390.

Sutskever, Ilya, Oriol Vinyals, and Quoc V. Le (2014). "Sequence to Sequence Learning with Neural Networks". In: *Proceedings of Advances in Neural Information Processing Systems 27*, pp. 3104–3112.

Vaswani, Ashish et al. (2017). "Attention is All you Need". In: *Proceedings of Advances in Neural Information Processing Systems 30*, pp. 6000–6010.

Wu, Yonghui et al. (2016). "Google's Neural Machine Translation System: Bridging the Gap between Human and Machine Translation". In: *CoRR* abs/1609.08144.

Zhang, Dakun et al. (2017). "Boosting Neural Machine Translation". In: *Proceedings of the Eighth International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, pp. 271–276.

Zhang, Longtu and Mamoru Komachi (2018). "Neural Machine Translation of Logographic Language Using Sub-character Level Information". In: *Proceedings of the Third Conference on Machine Translation: Research Papers*, pp. 17–25.

Zhang, Wen et al. (2019). "Bridging the Gap between Training and Inference for Neural Machine Translation". In: *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pp. 4334–4343.