Contextual Sentence Filtering for Context-Aware Neural Machine Translation

修　　士　　論　　文

Contextual Sentence Filtering for Context-Aware Neural Machine Translation

(文脈を考慮するニューラル機械翻訳における文脈文抽出)

指導教員　　鶴岡 慶雅 教授

東京大学大学院情報理工学系研究科
電子情報学専攻

氏 名　　**48-186450 李　桐**

提 出 日　　**2020 年 1 月 30 日**

Abstract

Thanks to the involvement of neural network architecture and the increasing amount of available parallel corpora, it is considered that the quality of sentence-level neural machine translation has been pushed to an acceptable level in high resourced language pairs such as English-French. In context-aware machine translation, on the other hand, taking additional contextual information into account has been proved efficient to improve the performance. However, how to locate the contextual information preciously is still an open question.

Recently, transfer learning based on pre-trained language models is evolving rapidly and attracts more and more attention from the natural language processing research community. Deep models trained on large scale datasets such as BERT (Devlin et al., 2019) renewed the state of the art for various tasks, and methods based on such models have become the standard approach in a range of subfields of natural language processing.

Given such background, in this research, we first investigate the necessity of locating contextual sentences for context-aware neural machine translation with a newly created English-Japanese document-level aligned dataset. Our experiments reveal that most of the sentences can benefit from taking additional context into account and there is much room for improvement if we can select the proper contextual sentence for each source sentence in context-aware machine translation. Furthermore, we attempt to automate contextual sentence filtering with the help of pre-trained language models in a weakly supervised way.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Motivation

Language is often treated as the symbol of intelligence which seperates human being from other primates. Thus the objective of building computer systems that can understand human language is one of the steps towards artificial general intelligence. On the other hand, due to the rapid development of the Internet and social media, there is a humongous amount of text data being generated every single day. There is also a growing requirement of developing effective methods to analyze and mining knowledge from such a huge amount of data. Both of the objectives drive researchers in the area of natural language processing (NLP) and make progress rapidly.

Thanks to the highly growing computation power and an increasingly large amount of available data, various fields in the area of natural language processing (NLP) have archived significant performance gain with the help of neural networks (NNs). Among them, Machine Translation (MT), which aims at translating written text from one natural language into another automatically, is often used as the testbed for novel methodologies. Research in the field does bring various standard NN based tools into NLP (Sutskever et al., 2014; Bahdanau et al., 2015; Luong et al., 2015; Vaswani et al., 2017).

MT systems usually translate sentences in isolation, which is called sentence-level machine translation (Sent-MT). As more and more high-quality parallel corpora are available for high resourced language pairs, such as English-French, the quality of Sent-MT is considered to reach an acceptable level of quality. However, taking only a single sentence as input sometimes can be affected by the ambiguity existing in the source sentences, especially in some specific languages such as Japanese and Chinese. For example, Japanese speakers tend to omit some arguments of verbs when they are obvious from the context during a conversation, which makes it difficult to translate such sentences into other languages accurately. Therefore, context-aware machine translation or document-level ma-

chine translation (Doc-MT) is also actively researched (Tiedemann and Scherrer, 2017; Voita et al., 2018, 2019; Junczys-Dowmunt, 2019). Doc-MT is considered more difficult mainly due to, firstly, the amount of document-aligned parallel corpora is insufficient on size in most of the cases, and secondly, Doc-MT models which take longer sequences as input are more difficult to reach convergence during training.
The main approach for Doc-MT is taking more contextual information into account as input. As shown in Tiedemann and Scherrer (2017), extending the input by concatenating the previous sentence with the source sentence brought BLEU (Papineni et al., 2002) score gain in German-English translation. However, a higher BLEU score can nether tell whether all the input sentences benefited from the extended context, nor provide evidence about whether the previous one sentence used as the context is the correct one or not. It remains an open question to be answered by further experiments. Attempting to answer part of these questions is the main motivation of this research.

1.2 Research Objectives and Contributions

This thesis studies the problem in Doc-MT regarding the necessity and possibility of filtering contextual sentences automatically. Our main hypothesis of this thesis is:
The previous sentence is not always the contextual sentence for context-aware machine translation and locating the contextual sentences can improve the performance significantly.
First, we trained several baseline concatenation machine translation systems and compared their results. Our results show that at least in the domain of business conversation, the previous sentence is not always the proper contextual sentence. We also calculated oracle BLEU scores based on the baseline systems we trained to estimate the upper bound for a perfect context-aware MT system which takes one additional contextual sentence into account can reach. The result shows that there is much room for improvement if a context-aware MT system can select the correct contextual sentences.
Then, we proposed a multi-task fine-tuning framework based on BERT (Devlin et al., 2019) to filter contextual sentences inside a window size of 5. A comparison experiment based on the normal Next Sentence Prediction fine-tuning on BERT is also conducted. Our result shows that our proposed method works well in EN $\rightarrow$ JA translation but not as expected in JA $\rightarrow$ EN. We further analyzed the results and hypothesized possible causes of such results.

1.3 Thesis Outline

In Chapter 2, we provide an overview of background knowledge which is necessary to understand the contents of this thesis. We introduce the basic framework of machine translation from both of the traditional perspective of statistical machine translation and the state-of-the-art neural network-based approaches. We also introduce an automatic evaluation metric BLEU for machine translation systems. Furthermore, we conduct a literature view regarding transfer learning in the field of natural language processing, which is the main methodology used in our experiments.
In Chapter 3, we describe our investigation about the effectiveness of applying transfer learning on a character-level language model. We use a Japanese specific task, Japanese Word Segmentation, as our testbed and conduct experiments from both feature-based and fine-tuning based approaches.
Chapter 4 presents our work on contextual neural machine translation. First, we introduce a newly constructed clean English-Japanese document-level machine translation dataset. Then we introduce our experiments to investigate the necessity of contextual sentence selection in context-aware machine translation. Finally, we propose a method to automate the contextual sentence filtering procedure by fine-tuning a pre-trained language model with multiple tasks.
Chapter 5 finally summarizes our findings in the contextual sentence filtering experiments and provides an outlook into future directions.

Chapter 2

Background

2.1 Machine Translation

Machine translation (MT), especially Neural Machine Translation (NMT) is the main testbed where we conduct most of our experiments. In order to introduce basic ideas in MT field, this section provides a general introduction to several import topics, including evaluation (Subsec 2.1.1), traditional statistical machine translation (Subsec 2.1.2), neural network-based machine translation (Subsec 2.1.3).

2.1.1 Evaluation in Machine Translation

Machine Translation, which investigates about how to translate written text or speech from one natural language into another automatically, is one of the most challenging tasks in the field of NLP. One of the main obstacles in MT is the difficulty of evaluations on results because first, it is hard to define what a good translation is, and second, several alternative translations can be valid for a single source sentence while it is not possible to collect all of them.
The most reliable and ideal method is to evaluate the translation by human raters based on both fluency and accuracy. However, human evaluation, especially which is conducted by professional translators, is extremely expensive and time-consuming since the procedure has to be repeated every single time the MT system is modified.
To address this issue, several automatic evaluation metrics for text generation, the field which MT is part of, are proposed. Most of them try to determine the translation quality based on the distance between a translation result and one or more reference human translations. Among them, BLEU (Papineni et al., 2002) is the most popular one in the field of MT, and also the main metric we use to evaluate most of the experiments conducted in this research.

Candidate: the the the the the the the.
Reference 1: The cat is on the mat.
Reference 2: There is a cat on the mat.

Figure 2.1: A BLEU example (Papineni et al., 2002) with two references and one candidate

BLEU is calculated based on n-gram overlap; they use a so-called *modified n-gram precision* instead of the naive one which tends to assign a high score to sentences containing a lot of repetition of some words in the reference sentences. The modified n-gram precision is calculated as follows. First, the maximum number of times an n-gram occurs in any single reference translation is counted. Second, the total count of each candidate n-gram is clipped by its maximum reference count. Finally, all the clipped counts are summed up and divided by the total unclipped number of candidate n-gram. In the example shown in Figure. 2.1, given two references about a cat, a candidate with the only word "the" should get a modified unigram precision of $2/7$.

$$\text{MNP} = \frac{Count_{\text{clip}}}{Count_{\text{n-gram in candidate}}}$$

$$Count_{clip} = min(Count, \text{Max\_ref\_count})$$

Such modified precisions p_n are calculated using n-gram up to a length N, which is a hyper-parameter of BLEU and is usually set to 4, and then added up with positive weights w_n summing to 1.
Another important component in BLEU is the usage of an exponential brevity penalty factor. While evaluating the BLEU score on a given test corpus, the geometric mean of the corpus is calculated and then multiplied by a penalty factor BP which is formalized as follows, where c is the length of the candidate translation and r is the effective reference corpus length.

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

With both exponential brevity penalty factor and geometric mean of modified n-gram precisions, we can get the final formula of BLEU as follows.

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

Though these metrics can be done automatically, which is preferred during the implementation process of an MT system, they do have important disadvantages. For example,

since BLEU ignores not only the semantics of the sentence but also the global structure of the sentence, a MT system with a high score does not always produce correct translations in the sense of semantics. Therefore finding a better replacement metric for the evaluation is still an active research question in the field of MT.

2.1.2 Statistical Machine Translation

Statistical Machine Translation (SMT) used to dominate the research of MT for decades. Comparing to rule-based MT which requires handcrafted rules to translation from a source sentence S to a target sentence T, statistical MT relies on probabilistic models learned from parallel corpora. It has the advantage of robustness while it is also limited by the requirement of large quantities of training data.
In the most traditional SMT system, given a source sentence S, the most probable translation $\hat{T}$ is searched as follow:

$$\hat{T} = \arg\max_T P(T|S) = \arg\max_T \frac{P(S|T)P(T)}{P(S)} \tag{2.1}$$

$$\propto \arg\max_T P(S|T)P(T) \tag{2.2}$$

The final formula is composed of two components, a translation model $P(S|T)$ which encodes the faithfulness of T as a translation of S, as well as a language model $P(T)$ which encodes the fluency of T in the target language. A third component called decoder, or translation engine then searches the space of possible translation and find the best translation T in the target language for a source sentence S based on the translation model and the target language model.
Translation Model A normal translation model relies on translation tables which record translation probabilities for individual words. It is the initial approach to statistical machine translation and led to the development of the famous IBM Model (Brown et al., 1993; Och and Ney, 2000; Brown et al., 1990). Such a translation table, or called word alignments, can be learned from parallel corpora through *Expectation-Maximisation (EM)* algorithm. The basic idea works as:

- Assign uniform translation probabilities to all possible word pairs
- *Expectation Step*: Apply current probabilities to estimate possible alignments on the parallel sentences
- *Maximisation Step*: Revise the translation probabilities based on the estimated sentence alignments

- Iterate the two steps until a stable solution is archived

Language Model The fluency of a translated sentence T in the target language is estimated by a language model. In SMT, the statistical language model used to assign a probability distribution over sequences of words $w_1, w_2, \cdot, w_n$, is typically represented as n-grams as shown below. The formula stands on the chain rule, which means the probability of each word depends on the words occurring before it and is simplified by considering only the N previous words.

$$P(w_1^n) = \prod_{k=1}^{n} P(w_k|w_1^{k-1}) \approx \prod_{k=1}^{n} P(w_k|w_{k-1}^{k-N+1}) \tag{2.3}$$

Decoder Beyond the translation model and language model, a separate component, namely decoder, is also required to search for the best translation from the huge space of possible translations. An algorithm called *beam search* is deployed to perform an incremental decoding process by expanding translation hypotheses gradually. Beam search keeps track of only a limited number of "good' hypotheses, which are generated from the translation table, based on their cost and discards the rest of them. The cost of each hypothesis is evaluated by transitions between hypotheses and fluency estimated by the language model. Due to the greedy nature of the beam search, a global optimal result is not always guaranteed.
The basic mechanism seems straightforward, but there are several limitations to traditional SMT. For example, the translation models originally rely on translation probabilities for individual words, which can not account for idiosyncratic expressions well. Take English-Japanese translation as an example, "heavy" and "mental" are usually translated as "重い" and "金属" individually but "heavy metal" should be translated as "ヘヴィメタル" when it is used to refer to the genre of rock music. To address the issue, the method of using translation tables for entire phrases instead are proposed:

$$P(S|T) = \prod_{i=1}^{I} \phi(s_i|t_i) d(a_i - bi - 1) \tag{2.4}$$

$\phi(s_i|t_i)$ represents phrase probability given by the translation table and $d(a_i - bi - 1)$ represents distortion probability which shows relative distance between the phrase positions in the two languages.
However, these improvements don't change how a system is constructed in SMT. The nature of combining multiple components which have to be trained separately makes SMT systems hard to design, train, and maintain.

2.1.3 Neural Machine Translation

As various of tasks in the field of NLP have been boosted by the Neural Networks (NN), researchers also attempt to improve MT with the power of NN. For a long time, the improvement was done by integrating NN into SMT systems, e.g. replacing the standard n-gram LMs with Recurrent Neural Network (RNN) based ones (Cho et al., 2014b). Such attempts do bring improvement to SMT systems, but the difficulties of training and maintenance remain since all the changes are done under the traditional framework of SMT.
On the other hand, Neural Machine Translation, which aims at solving the MT task by training a single NN in an end-to-end behavior, reduces the effort required for training and maintaining a machine translation system significantly.

2.1.3.1 Sequence-to-Sequence

The recurrent continuous translation models proposed by (Kalchbrenner and Blunsom, 2013) was the first machine translation model in which the target sentence distribution is conditioned on a fixed-size representation of the source sentence. The idea gave rise to a new family of neural network architectures called encoder-decoder, or sequence to sequence (Seq2Seq) networks. It was also developed by successors (Sutskever et al., 2014) and become the de. facto tools in solving verities of NLP tasks. Machine translation systems using such type of neural networks are so-called neural machine translation systems. In NMT, given a source sentence $X = x_1^N$ and a target sentence $Y = y_1^M$, the probability distribution over the target sentences $P(Y|X)$ is factorized into conditionals:

$$P(Y|X) = \prod_{k=1}^{M} P(y_k|y_1^{k-1}, X)$$

The most traditional Seq2Seq network utilizes two recurrent neural networks (RNN), a variant of NN whose strength is modeling variable-length sequences of inputs such as natural language sentences, as the encoder and the decoder respectively. With that, the probability distribution can be further modeled as:

$$P(y_k|y_1^{k-1}, X) = f(y_k|h_k, y_{k-1}, c(X))$$

where h_k is the hidden state of the RNN decoder at time step k and $c(X)$ is the fixed-length vector compressed from X by the encoder. Furthermore, $f(\cdot)$ is a feedforward network with a softmax layer at the end which takes the decoder state h_k and an embedding of the previous target token y_{k-1} as input. Figure. 2.2 shows such a basic Seq2Seq network for MT.

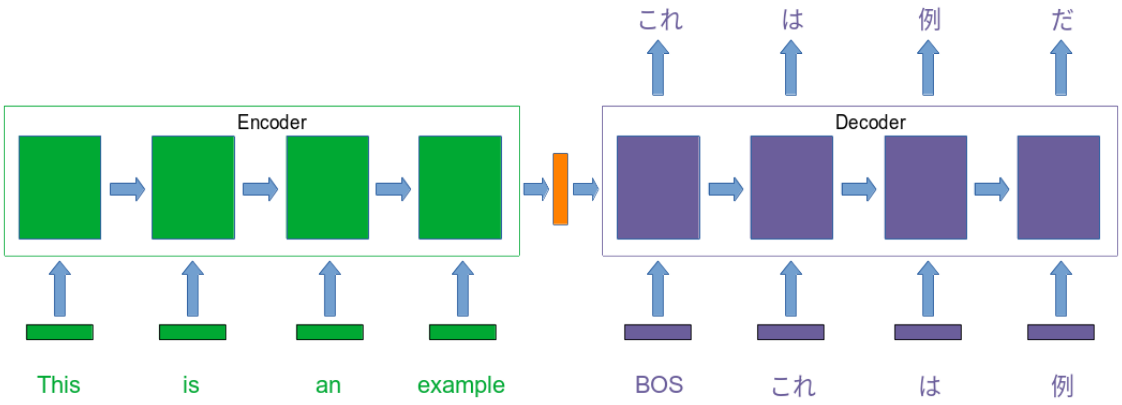


Figure 2.2: A simple Encoder-Decoder model in which the encoder compress the input sequence into a fixed-length vector and pass it to the decoder. Then decoder generate a sequence conditioned on the fixed vector.

2.1.3.2 Attention

One of the main drawbacks of early NMT models was poor performance while generating translations for long sequences. The fixed-length source sentence encoding was suggested by (Cho et al., 2014a) as the main cause. Further, they argued that though a fixed vector is enough for translating short sentences, it lacks the capability of encoding a long sequence with complicated structure and meaning. Then a concept called *attention* was introduced by (Bahdanau et al., 2015; Luong et al., 2015) to address this issue. Instead of the fixed-length source sentence encoding, their models can also place their attention on only parts of the source sentence which is crucial to producing the next token of the target sequence. Thus the context vector $c(X)$ is replaced by a series of vectors $c_k(X)$, which varies at different decoding steps.
Figure. 2.3 shows a demonstration about how encoder-decoder attention works in a RNN-based attentional Seq2Seq network. First, the encoder encodes a source sentence into a fixed-length vector and passes it to the decoder. What is different from the early Seq2Seq network is that the final hidden state of each encoding step s_1^N is also stored. Then at each decoding step t, a score for each encoder hidden state s_i is calculated based on a function $score(\cdot)$ which takes h_t and s_i as inputs. Then all the scores are normalized by a softmax

function

$$\sigma(\mathbf{z})_i = \frac{e^{z_i}}{\sum_{j=1}^{K} e^{z_j}} \text{ for } i = 1, \cdots, K \text{ and } \mathbf{z} = (z_1, \cdots, z_K)$$

to a weight w_t. Finally s_1^N are summed up by the w_t to form the final context vector $c_j(X)$.

Attention can also be abstracted as a mechanism to map m query vectors (hidden stats

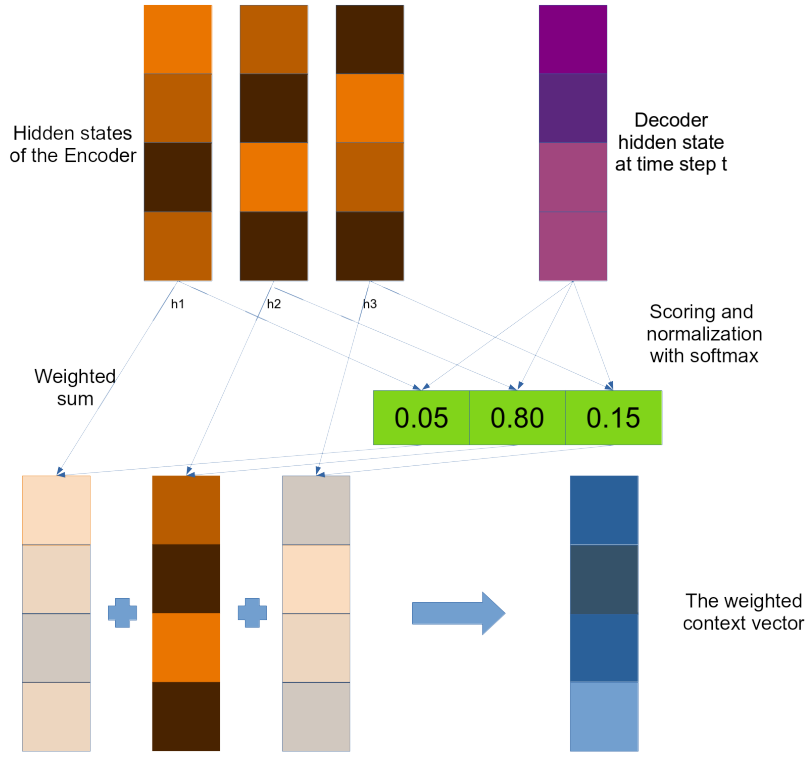


Figure 2.3: Demonstration about how attention works.

from decoder) to m output vectors (context vectors) through a mapping table of m key-value pairs. Following the terminology of (Vaswani et al., 2017), given three lists of vectors that can be stacked into three matrices $Q \in \mathbb{R}^{m \times d}$, $K \in \mathbb{R}^{n \times d}$, and $V \in \mathbb{R}^{n \times d}$, each output vector is computed as a weighted of the value vectors. Each weight is determined by a similarity score between the corresponding query and key vectors.

$$\text{Attention}(K, V, Q) = \text{Softmax}(score(Q, K))V$$

2.1.3.3 Transformer

(Vaswani et al., 2017) proposed an important generalization of attention namely *multi-head* attention. The core idea is performing attention operations for N, which is referred as the number of attention heads, times instead of a single one. The query, key, and value vectors for each attention head are linear transforms of Q, K, V. The outputs of each attention head are concatenated to form the final output of the multi-head attention. The mechanism can be described formally as follows:

$$\text{MultiHeadAttention}(K, V, Q) = \text{Concat}(head_1, \cdots, head_H)W^O$$

$$head_h = \text{Attention}(KW_h^K, VW_h^V, QW_h^Q)$$

(Vaswani et al., 2017) also proposed a so-called self-attention which can be applied in the Seq2Seq framework. Figure. 2.4 demonstrate how self-attention works with a toy example. Inside encoder and decoder, all three components of self-attention (queries, keys, and values) are derived from encoder or decoder state only respectively. For example, the decoder conditions on the previous output tokens y_1^{k-1} by attending to its own states from previous time steps. Most of the computation of self-attention can be parallelized and reduce the amount of sequential computation at the encoder side significantly.
Self-attention-based models are also challenged due to the nature of attention that it has no notion of orders. Positional embeddings are used by (Vaswani et al., 2017) to tackle the issue. They originally stacked sine and cosine functions of different frequencies to pass positional information to self-attention:

$$PE_{pos}(2i) = sin(\frac{pos}{10000^{\frac{2i}{d_{model}}}})$$

$$PE_{pos}(2i+1) = cos(\frac{pos}{10000^{\frac{2i}{d_{model}}}})$$

Alternatively, several works (Devlin et al., 2019) utilized a learned embedding matrix as the positional embedding alongside the normal token embedding layer. A $PE_{learned}(n)$ usually maps the absolute position n of the token in a sequence. However positional methods to use relative positional information were also proposed (Shaw et al., 2018).

2.2 Transfer Learning in NLP

2.2.1 Introduction

ImageNet (Deng et al., 2009), which was originally published in 2009, is a dataset for the task of image classification. It impacted the Computer Vision (CV) as well as the whole

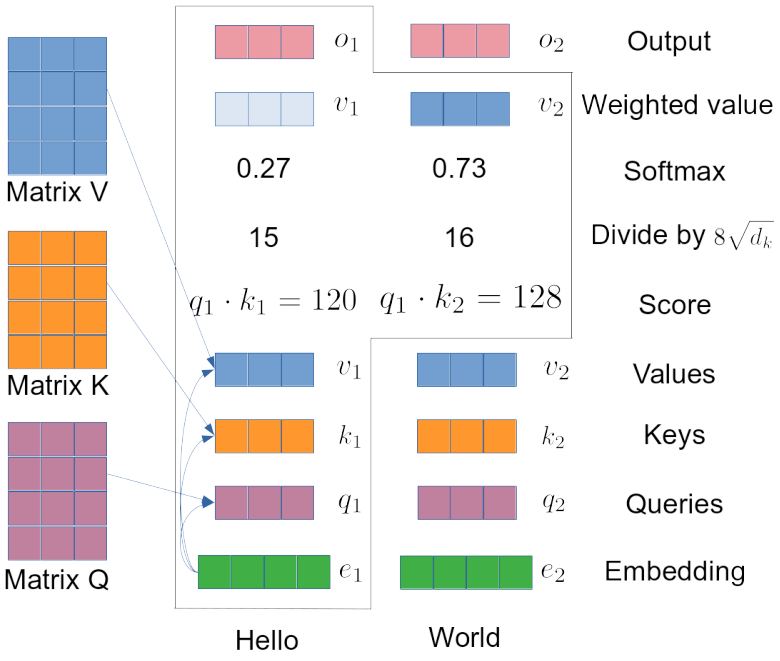


Figure 2.4: The self-attention mechanism introduced by (Vaswani et al., 2017)

course of machine learning greatly. Due to its humongous size (1.2 million images for training and 100 thousand for testing) and its variety (1,000 object classes), models pre-trained with ImageNet show impressive effectiveness in transfer learning. These models cannot only be deployed as feature extractors for other classification datasets, but also be fine-tuned to solve a variety of different tasks such as object detection, semantic segmentation, human pose estimation, and video recognition.
On the other hand, the application of pre-trained models in the field of Natural Language Processing (NLP) is mainly restricted to feature extracting, which is also called embedding, for a long time before 2018. Embeddings are used to convert string-based natural language tokens into distributional representations such as vectors and matrices. Methods like Word2vec (Mikolov et al., 2013) and GloVe (Pennington et al., 2014), which can capture context-independent features with shallow neural network models and large corpora become a key component in many NLP tasks. Start in 2018, models that pre-trained on tasks that are sensitive to contextual information like machine translation (CoVe) (McCann et al., 2017) and language model (ELMo) (Peters et al., 2018), can utilize context-

related features and improved state-of-the-art results in many NLP tasks with nearly no modification to the architecture of task models.
However, although contextualized embeddings show better performance than the classical context-independent ones, there remains a problem that these embeddings are used as fixed parameters and the main task models have to be trained from scratch. To address this issue, CV-like fine-tuning approaches (Alec et al., 2018; Howard and Ruder, 2018; Devlin et al., 2019) based on pre-trained models on Language Model (LM) are also proposed in 2018 and show great effectiveness in transfer learning for other NLP tasks.

2.2.2 Feature based approaches

The most widely used approach of applying pre-trained models into NLP tasks is through the feature-based methods, or saying embeddings. Unlike in the CV field where the object images can be represented as matrices and manipulated by machine learning (ML) models (e.g. neural networks) directly, inputs of most NLP tasks are string-based natural language tokens so that they have to be converted into vectors or matrices before got fed to ML models.
Traditional NLP approach to convert tokens into vectors before the deep learning booming was designing a rich set of features manually, but an approach was proposed in 2011, in which features are extracted with pre-trained models(Collobert et al., 2011). When applying pre-trained models through this approach, the models are first trained on a "fake" task. Then the parameters of the pre-trained model are fixed and used to extract features of words, sentences, or documents and input those features to task models, which are neural networks designed for solving individual NLP tasks.

2.2.2.1 Word2vec

The most classical methods can only capture context-independent information. The most widely known embedding method is called Word2vec (Mikolov et al., 2013) was one of the earliest attempts to capture word-level features with neural networks. It shows the great capability of neural networks in such a task when a large corpus is available. The features captured by Word2vec brought impressive improvement to a wide range of tasks. The theory where Word2vec is built on is the so-called **Distributional Hypothesis**. It hypotheses that words that occur in the same contexts tend to have similar meanings (Harris, 1954). To capture word meaning based on the theory, Word2vec algorithm deploys a shallow neural network with only one linear hidden layer and trains it on a task whose objective is finding representations that are useful for:

- Predicting nearby words when one word is given. More formally the objective is predicting w_t when $w_{t-c}, \ldots, w_{t-1}, w_{t+1}, \ldots, w_{t+c}$ are given.
- Or predicting a word when its nearby words are given. More formally the objective is predicting $w_{t-c}, \ldots, w_{t-1}, w_{t+1}, \ldots, w_{t+c}$ when w_t is given.

The c in these formulas is a windows size used to indicate the size of context during training time. The first approach is called *Continuous Bag-of-Words* and the second one is called *Continuous Skip-gram* and their architectures are shown in Figure. 2.5 and Figure. 2.6
In the skip-gram model, a word t is first converted into a so-called one-hot vector $w_t \in R^N$ (N is the size of the vocabulary), which consists of 0s in all cells except a single 1 in a cell used uniquely to identify the word. Then the one-hot vector is fed to the neural network and be projected as a vector $h_t \in R^d$ with a matrix $W_h \in R^{N \times d}$ where d is the number of features we want to use to represent a word. h_t is fed to a softmax layer to generate a distribution over the vocabulary. What we need to do during the training procedure is to maximize the probability of words that appear nearby t, which includes c words before t and c words after t, in the whole corpus.
Despite the simplicity of the objective for pre-training, Word2vec can capture many linguistic regularities and patterns. Many of these patterns can be represented as linear translations and simple addition can often produce meaningful results. For example, vec("Germany") + vec("capital") is close to vec("Berlin"), vec("man") - vec("woman") is close to vec("king") - vec("queen").

2.2.2.2 Recurrent Neural Language Model

Language Model is used to evaluate whether a sentence is "nature" or not by calculating the probability distribution of a given sequence of words or characters. Such a model is usually trained on a large scale monolingual corpus by optimizing the parameters of a model to reduce the negative log-likelihood of sequences from the corpus. Modeling the probability of a given sequence is usually done by the chain rule. In the deep learning era, the language modeling task is usually tackled by a recurrent neural network, especially one of its variants called long short-term memory (LSTM) (Hochreiter and Schmidhuber, 1997). A recurrent neural language model estimating the probability distribution of each token (word or character) with the information of all the tokens before is called a Forward Language Model, and the one utilizing the information afterward is called Backward Language Model. The formulation of these two kinds of LM is shown as follows. Given a sequence of N tokens $t_1, t_2, \ldots, t_{n-1}, t_n$:

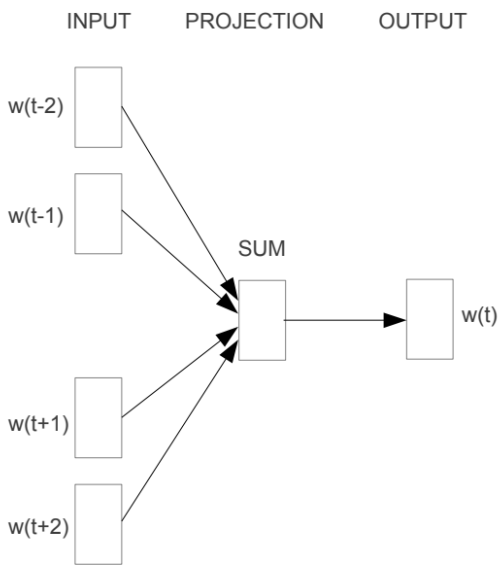


Figure 2.5: Features of contextual words are summed up to estimated the center word in continuous bag-of-words (Mikolov et al., 2013)

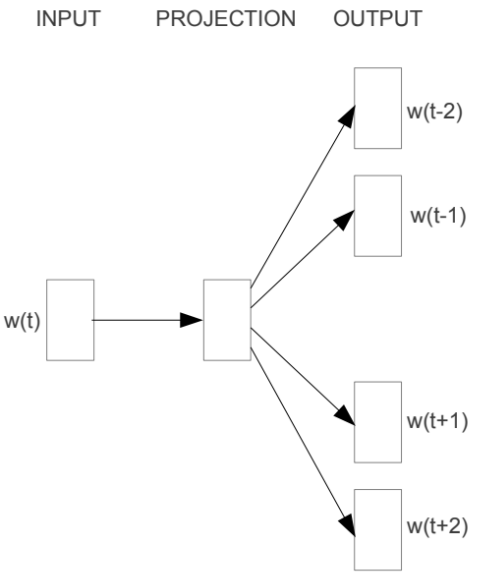


Figure 2.6: The feature of the center word is utilized to predict all the surrounding words inside a context window (Mikolov et al., 2013)

- A forward language model computes and maximize the probability of token t_k given history $(t_1, \ldots, t_{k-1})$ as follow. In the case that LSTM is used to solve the language model task, the probability can also be approximated by the below one in which θ^f is the parameters of the LSTM and h_t^f is the hidden state of the LSTM at time step t.

$$p(t_1, t_2, \ldots, t_N) = \prod_{k=1}^{N} p(t_k | t_1, t_2, \ldots, t_{k-1})$$

$$\approx \prod_{t=0}^{T} p(t_k | h_t^f; \theta^f)$$

- Likely, a backward language model computes and maximize the probability of token t_k given future $(t_{k+1}, \ldots, t_n)$. It can also be approximated with LSTM's pa-

rameters and the hidden state at time step t

$$p(t_1, t_2, \ldots, t_N) = \prod_{k=1}^{N} p(t_k | t_{k+1}, t_{k+2}, \ldots, t_N)$$

$$\approx \prod_{t=0}^{T} p(t_k | h_t^b; \theta^b)$$

2.2.2.3 Language Model-based Feature Extraction

Since Word2vec always converts the same word into a fixed vector without taking any contextual information into account, it can deal with phenomenons such as polysemous well. CoVe (McCann et al., 2017) is one of the several attempts to take contextual information into account during feature extraction with a pre-trained machine translation. It does bring performance improvement to several tasks, but it is also restricted by its requirement on parallel MT training data. To address this issue, Embeddings from Language Models (ELMo) (Peters et al., 2018) was proposed to utilize a model pre-trained with the language modeling task. Comparing to parallel corpora needed by the MT task, monolingual corpora used by language modeling tasks are much larger in size, which provides a great potential for language model to capture more contextual information than machine translation can do.
The base model of ELMo is trained as a bi-direction language model. Then ELMo embeddings are computed as a task-specific combination of the intermediate layer representations in the pre-trained biLM as follows. First, simply feed the sequence of tokens we are encoding into the LM and get intermediate representations $h_{k,j}^{LM}$ from all layers (k stands for the kth layer and j stands for the jth token) which is the concatenation of forwarding LM intermediate representation $\overrightarrow{h}_{k,j}^{LM}$ and backward LM intermediate representation $\overleftarrow{h}_{k,j}^{LM}$. Then a task-specific softmax-normalized weights s^{task} and a scalar parameter γ^{task} are learned tasks specifically to get the final specific context-aware embeddings. The formula is as follow and the whole procedure is demonstrated in Figure. 2.7.

$$\mathbf{ELMo}_k^{task} = \gamma^{task} \sum_{j=0}^{L} s_j^{task} h_{k,j}^{LM}$$

The state-of-the-art results for several tasks, including reading comprehension, textual entailment, semantic role labeling, co-reference resolution, named entity extraction, as well as sentiment analysis, were improved significantly by simply replacing normal Word2vec

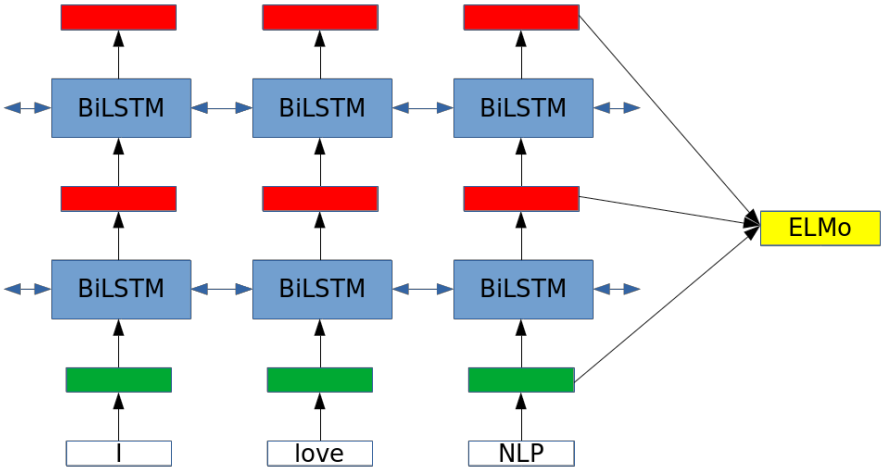


Figure 2.7: A 2-layer bi-directional LSTM language model is trained on a large scale monolingual dataset, then a task-specific weight is learned for each task to combine the three hidden states extracted by the language model.

embedding layer with ELMo in the task-specific models. Although most of the contextual feature extraction methods are based on word-level, a character-level language model based contextualized embedding method, called Contextual String Embeddings (Flair) (Akbik et al., 2018), was also proposed in 2018. Instead of converting words into vectors and feeding them in the biLMs, Flair encodes and uses characters as the input directly. And rather than concatenating intermediate representations corresponding to a word from each layer of the biLM like ELMo does, Flair extracts the output hidden state after the last character in the word from forwarding LM as well as the output hidden state before the first character in the word from backward LM and concatenates them into a single vector to represent the word. The procedure is demonstrated in Figure. 2.8. Flair outperformed than in tasks such part-of-speech tagging and named entity extraction for both English and German, which showed the great effectiveness of contextual information captured in language model at both word-level and character level.

2.2.3 Fine-tuning based approaches

Although pre-trained models for embedding, especially contextualized ones, perform extremely well in a range of NLP tasks, there remains an issue that only the first layer of task model benefits from pre-training and the other parts are still needed to be trained from scratch which requires a huge amount of data and time-consuming. To address this issue, several methods to directly fine-tune pre-trained language models on new tasks without

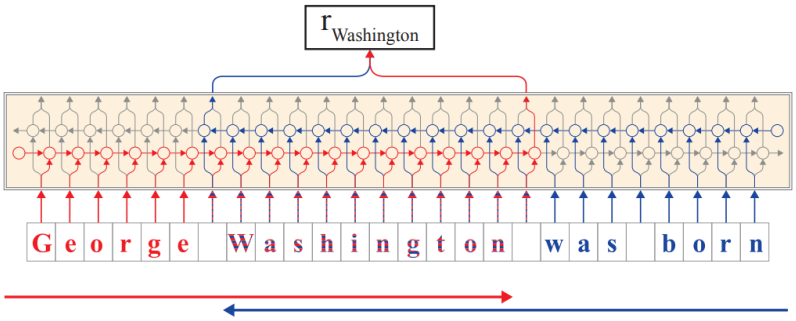


Figure 2.8: Flair embedding is constructed by concatenating the hidden state of the space before the first character in a word as well as the hidden state of the space after the last character in a word (Akbik et al., 2018).

task-specific models were proposed in 2018.

2.2.3.1 ULMFiT

Several attempts to perform transfer learning in NLP tasks via the fine-tuning based approach were conducted by researchers before. Unfortunately, the methods proposed by Dai and Le (Dai and Le, 2015) are limited in usefulness due to its requirement on millions of in-domain documents when trying to achieve good performance. However, (Howard and Ruder, 2018) argued that not the idea of fine-tuning pre-trained LM itself, but the lack of knowledge of how to perform the procedure has been hindering wider adoption. The fine-tuning methods, as well as novel training techniques proposed in ULMFiT, showed that fine-tuning over pre-trained LM can achieve CV-like transfer learning in a range of classification tasks.
Similar to ELMo and Flair, ULMFiT uses the same pre-training objective, which is language modeling, with a 3-layer bidirectional LSTM. After the pre-training, instead of computing embeddings based on hidden state outputs of the biLM by fixing all its parameters, ULMFiT first perform a procedure called target task LM fine-tuning. The motivation for adding such a procedure is the fact that no matter how diverse the data used for pre-training is, the data of the target will likely come from a different distribution. After the LM adapted to the idiosyncrasies of the target data, two additional blocks used for the target classification tasks are appended at the end of the pre-trained model and trained from scratch. These blocks take the pooled last hidden layer states as the input and produce a distribution over different labels used for classification. The whole procedure is shown in Figure. 2.9. Experiments are conducted on a variety of text classification tasks, including sentiment analysis, question classification, and topic classification, with

6 different datasets. The results showed that ULMFiT outperformed state-of-the-art on all datasets. The error rates were reduced by ranging from 8.3% to 22% relatively on smaller datasets and ranging from 2.0% to 23.7% on larger ones. It shows that not only the embedding layers can but also the other part can benefit from the knowledge.

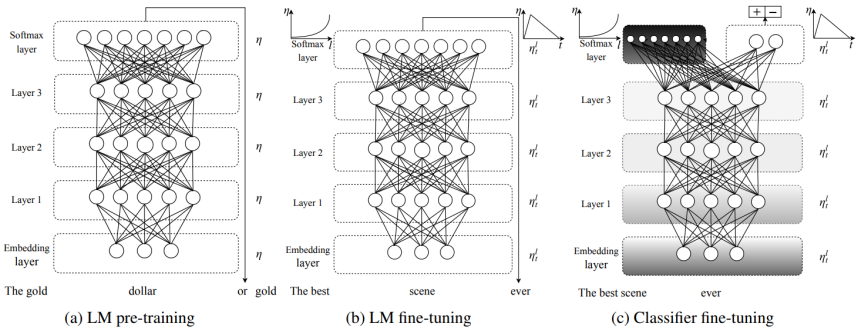


Figure 2.9: The fine-tuning procedure of ULMFiT for classification tasks, which includes a LM pre-training on large scale unlabeled data, LM fine-tuning on target dataset, and the fine step to adapte the whole model on the target task. (Howard and Ruder, 2018)

2.2.3.2 OpenAI GPT

Although ULMFiT shows impressive improvement over several classification tasks, the fine-tuning framework proposed by it is limited and difficult to be used on tasks other than classification. To provide a more general framework of LM-based fine-tuning, OpenAI GPT (Alec et al., 2018) was proposed then by replacing LSTM LM used in ULMFiT with a transformer-based (Vaswani et al., 2017) one and extending the framework to adapt a pre-trained LM to a target task.

The pre-trained procedure is quite similar to the one used by ULMFiT. The only a difference worth noting is that unlike the bidirectional LSTM based LM used in ULMFiT, ELMo, and Flair, OpenAI choose to build a forward only LM with transformer, in which only left-to-right contextual information can be captured.

The biggest difference between OpenAI GPT and ULMFiT is the procedure to adapt the pre-trained LM for target tasks. Since OpenAI GPT was designed to solve tasks for both tasks in which the input is a single sentences as well as tasks that take sentences pairs as input. Solution for classification tasks is quite straightforward, which is feeding the last transformer block's hidden state to a softmax layer to generate a distribution over labels. For other tasks, a so-called traversal-style strategy is taken as:

- First, the several structured inputs are converted into a single order sequence which can be handled by the pre-trained language model. Randomly initialized special token [start] and [extractor] to mark the begin and end of the sequence are also included in the transformed sequence.

- **Textual entailment**: In this task, the model needs to read a pair of sentences and judge the relationship between them from one of entailment, contradiction or neutral. While fine-tuning for textual entailment, the two sentences are concatenated with a delimiter [delim] between. The remain parts work the same as classification tasks.

- **Sentences similarity**: Semantic similarity tasks involve predicting whether two sentences are semantically equivalent or not. Since there is no inherent ordering of the two sentences being compared, the input sequences are fed to the LM twice in different orders. Then before the two representations from the last transformer block are added element-wise before being fed into the softmax layer.

- **Question answering**: In this task, the model needs to choose the best answer while an English passage with associated one question and several answer candidates are given. To solve it with the LM, the passage and question are concatenated with each possible answer with a delimiter between the question and the answer in a way like $[p; q; \$; a_k]$ (p represents the passage, q represents the question, $\$$ represents the delimiter and a_k represents the kth answer candidate.). Each of the transformed sequences is processed independently and then normalized via a softmax layer to generate a distribution over possible answers.

Experiments showed that the performance of a variety of task, such as question answering, can be improved significantly through LM pre-training and fine-tuning. The performance improvement can be as large as 8.9%. The more general framework provided by OpenAI GPT, showed the great potential of CV-like fine-tuning methodology in the field of NLP.

2.2.3.3 BERT

OpenAI GPT shows the great effectiveness of the LM fine-tuning method in a range of tasks, however, the forward-only language model used in it restricted its ability to consider contextual information behind a word. Such restrictions are sub-optimal for sentence-level tasks and could be devastating when applying fine-tuning based approaches to token-level tasks such as SQuAD question answering dataset (Rajpurkar et al., 2016). The SQuAD contains more than 100 thousand crowdsourced question-answer pairs where the answer is a span in a given context. The objective of finding both start and end for an answer makes it crucial to incorporate context from both directions.

To complete the bidirectional language modeling lacked in OpenAI GPT, Bidirectional Encoder Representations from Transformers (BERT) (Devlin et al., 2019) is proposed to pre-train a transformer-based language model with two novel objectives. Unlike the bidirectional LM used in ULMFiT, in which hidden states from forwarding LM and backward LM are concatenated to produce a vector including both left-to-right and right-to-left contextual information, BERT uses a task so-called **masked LM**. Consider the case to train a language with the sentence "my dog is hairy". In normal bidirectional LSTM language models, saying we want to model the word "is", the forward LM will maximize the probability when "my" and " dog" is given, while the backward LM will maximize the probability when "hairy" is given. On the other hand, the masked LM will replace the "is" with a special toke [MASK] and let the model predict what it should be. This forces the model to consider contextual information from both sides simultaneously, which can produce a "deep" bidirectional LM.

The representation of the input for BERT is similar to OpenAI GPT, a special token [CLS] is used to mark the start of a sequence of tokens and another token [SEP] is used to separate two sequences when packing them together. [SEP] is also used to mark the end of a sequence. For a given token, its input representation is constructed by summing the corresponding token, segment and position embeddings as demonstrated in Figure. 2.10. Also, target tasks such as question answering and textual entailment need some kinds of

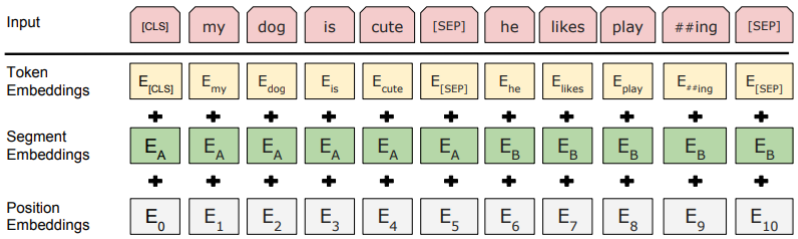
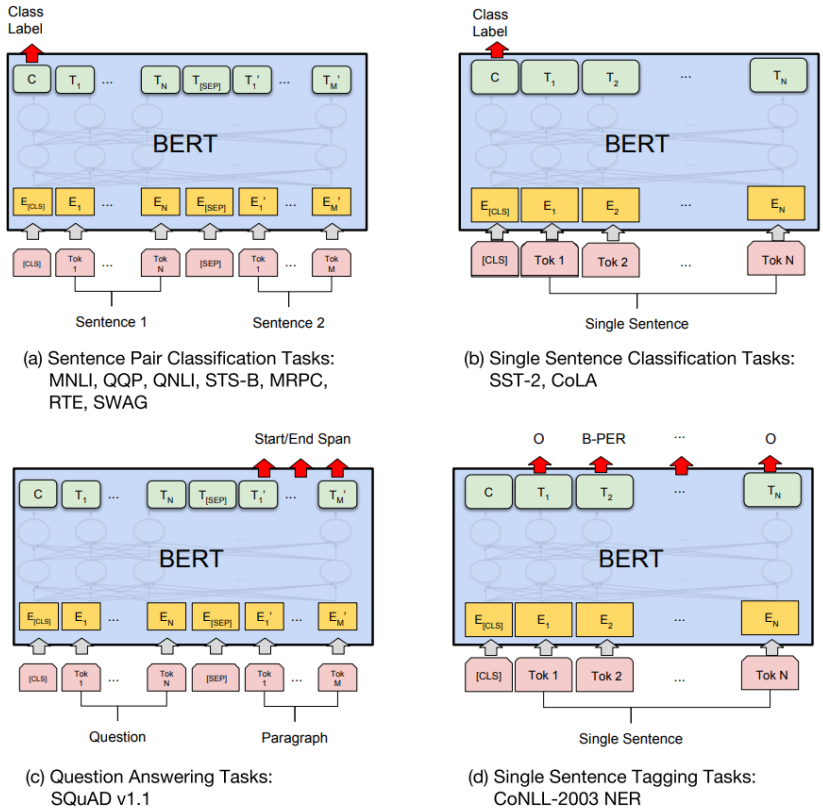


Figure 2.10: The embedding of BERT is consisted of three parts, 1) normal token embedding, 2) learned position embedding, 3) token type embedding to indicate which sequence the token is from. (Devlin et al., 2019)

understanding regarding the relationship between two text sentences. In order to train a model that can capture sentence relationships, a binary **next sentence prediction** task similar to the method proposed in (Kiros et al., 2015) that can be trivially generated from any monolingual corpus is also used to train the biLM. When choosing the sentences A and B for each pre-training example, 50% of the time B is the actual next sentence that follows A, and 50% of the time it is a random sentence from the corpus.

The fine-tuning procedure is shown in Figure. 2.11. For example, in classification tasks whose input is one sentences e.g. sentiment analysis or two sentences e.g. textual

entailment, the hidden state of [CLS] token from the last layer of BERT are fed to a softmax layer to generate a distribution over labels.



(a) Sentence Pair Classification Tasks: MNLI, QQP, QNLI, STS-B, MRPC, RTE, SWAG

(b) Single Sentence Classification Tasks: SST-2, CoLA

(c) Question Answering Tasks: SQuAD v1.1

(d) Single Sentence Tagging Tasks: CoNLL-2003 NER

Figure 2.11: Different strategy is deployed while fine-tuning BERT on different tasks (Devlin et al., 2019)

In the experiments conducted with fine-tuned BERT, the state-of-the-art of GLEU, a benchmark consisting of diverse natural language understanding tasks was improved by 4.7% absolute accuracy. BERT also outperforms the top leaderboard system of SQuAD by +1.5 F1 score which makes it +1.5 higher than human beings. The novel objective of masked LM and next sentence prediction shows a great improvement over ULMFiT and OpenAI GPT, as well as ELMo.

Chapter 3

Experiments on Japanese Word Segmentation

As we introduced in the last chapter, transfer learning based on language models, shows promising effectiveness at the word level. On the other hand, flair embedding (Akbik et al., 2018) also show that character-level language model can work even better on several tasks. However, most of the research is conducted on languages such as English and German. A question arises naturally is that how such language model-based methodologies fit language like Japanese and Chinese as well as their specific NLP tasks. In this chapter, we introduce trial and errors shown in our experiments which tackle the task of Japanese Word Segmentation (JWS) by language model-based transfer learning.

3.1 The Problem

Unlike languages such as English and German in which space exists between words, there is no explicit bound between words in the text of Japanese. While applying NLP to Japanese text, segment the whole input sentence into words is a necessary preprocessing before feeding them into any machine learning models. So that errors that happened at this phase may propagate and affect all the phases afterward. For example, the phrase ”この先生きのこるには”, where the correct word segmentation should be ”この先/生きのこる/には”, means ”How to survive from now on”. However, it can also be segmented as ”この先生/きのこ/る/には”, which is just a word list of ”this teacher, mushroom, to” and meaningless as a sentence. This kind of error will significantly change the results of tasks such as machine translation therefore high precision is required in the task of word segmentation.
The most widely used approach to solve JWS is modeling the task as a sequence labeling task which aims at assigning a proper label to each character in a sequence, then train-

ing machine learning models with supervised learning (Kitagawa and Komachi, 2017). However, since training data has to be collected and labeled manually by native speakers, most of the available JWS datasets are limited on size. It makes the trained model hard to handle unknown words which didn't show in the training data.
Considering the impressive performance of character-level LM shown in English and German (Akbik et al., 2018) and the character level labeling nature of JWS, we conduct experiments to investigate the possibility to solve the JWS task with a character-level trained on large unlabeled data from both feature-based and fine-tuning based approaches.

3.2 Japanese Word Segmentation

There are mainly two approaches to model the task of JWS, including

- **Lattice based**: In this approach, a lexicon that lists a pair of a word and its corresponding part-of-speech is assumed available. The lexicon is used to build a lattice, which represents all candidate paths. Every path consists of a candidate sequence of tokens where each token denotes a word as well as its part-of-speech. Then the objective is to find the best candidates path in the constructed lattice. Figure. 3.2 shows an example taken from (Kudo et al., 2004) which demonstrates how the lattice-based method works.

- **Character based** : Another straightforward method to solve JWS is modeling it as a simple sequence labeling task, in which every character is assigned a label indicating whether there is a word boundary or not. (Neubig et al., 2011) is one of the most widely used JWS tools which deploys such an algorithm.

Recently, due to the involvement of deep learning and the outstanding performance shown by neural networks, methods for integrating neural networks into JWS algorithms are also actively researched. (Morita et al., 2015) proposed a method that deploys a recurrent neural network language model to evaluate the likely-hood of a word sequence and then weighted sum the score to the normal lattice-based score to form a final score for each candidate path. Such an approach archive F-1 score gains range from 0.2 to 0.6 on several benchmark datasets. Furthurmore, (Kitagawa and Komachi, 2017) propose a method which utilizes an LSTM to tackle the JWS task in the sequence labeling approach without taking any external knowledge, e.g. manually collected lexicon used in (Kudo et al., 2004; Morita et al., 2015), into account as (Neubig et al., 2011) does and outperform it with an F1-score of 98.43.

3.3 Proposed Methods

While treating JWS as a sequence labeling task, there are mainly two types of strategies to assign labels to each character. The first one is called *Begin/Inside/End (BIS)* tagging in which each character is assigned a tag to indicate whether it is the beginning of a word, inside a word, or the end of a word. Another simpler setting is assigning a label from $0, 1$ to each character to indicate whether a word boundary exists or not. Figure. 3.2 shows an example to demonstrate how 0/1 labeling works with a real example. In this experiment, we follow the second strategy as (Neubig et al., 2011) does and propose a methodology to solve the task by transferring a pre-trained LSTM based bi-directional LM.

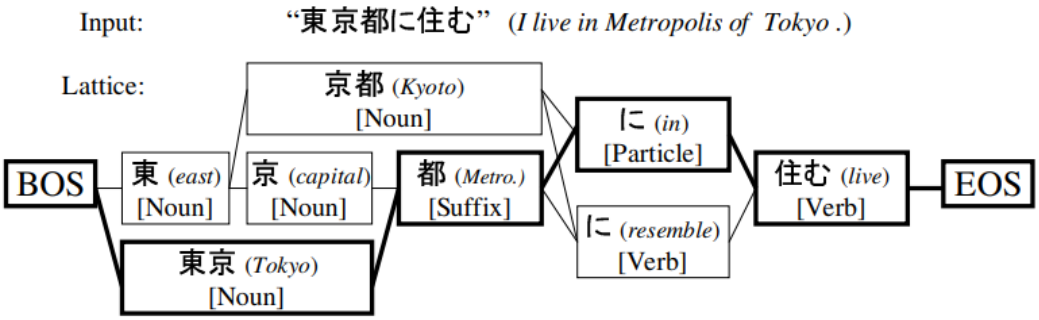


Figure 3.1: This figure shows how the lattice-based method works for JWS. A lattice is constructed with a lexicon (dictionary) and every path is scored. Then one single path with the highest score is selected from all the candidates (Kudo et al., 2004).

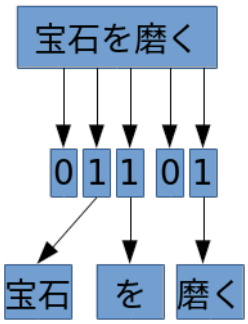


Figure 3.2: Segment a Japanese sentence by inferring whether a word boundary exists for each character.

We first train a bi-directional LSTM-based LM with unlabeled monolingual data from Wikipedia. From the feature-based approache, we extract hidden states responding to a character in a sequence from both of the forward and backward LM and concatenate them to form a feature embedding for that character. The method is shown in Figure. 3.3. The embedding is then fed to another LSTM-based sequence labeling model which is trained with a manually labeled word segmentation dataset. On the other hand, from the fine-tuning based approach, we simply replace the output layer of the LM with one for the 0/1 labeling task and train the LSTM network with JWS dataset.

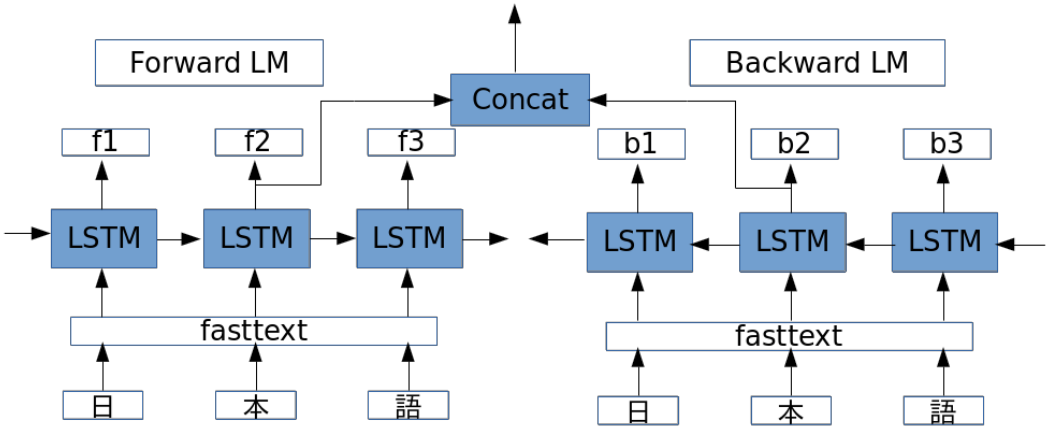


Figure 3.3: Feature extraction from a bi-directional character-level LM

3.4 Experiments

3.4.1 Datasets

For the experiment, we use the dump data of Japanese Wikipedia (wiki-dump) [1] Kyoto University Web Document Leads Corpus (KWDLC) [2] for LM pre-training and word segmentation model training. Sentence extraction from wiki-dump is done with wikiextractor [3], which yields a monolingual corpus with 9,112,239 Japanese sentences. No further preprocessing is applied. Sentences are segmented by character and fed to the LM. For KWDLC, we split the dataset into train, validation and test subsets randomly, which has 10,742, 2,000, 2,000 sentences respectively.

[1] https://dumps.wikimedia.org/jawiki/20181120/
[2] http://nlp.ist.i.kyoto-u.ac.jp/index.php?KWDLC
[3] https://github.com/attardi/wikiextractor

3.4.2 Model and Hyper-parameter

Experiment Setting	Embedding	LSTM
1-layer LSTM + fastText	100	256
1-layer LSTM + LM features	2048	256
1-layer LSTM + fastText + LM fatures	2148	256
2-layer LSTM + fastText	100	256
1-layer LSTM LM fine-tuning	1024	1024
2-layer LSTM LM fine-tuning	1024	1024

Table 3.1: Hyper-parameter setting up in the Japanese word segmentation experiments, include the embedding dimension and the hidden state dimention of LSTM

To compare with the proposed methods, we set up a baseline in which a fastText (Bojanowski et al., 2017) embedding is learned on the same Wikipedia data and used as a feature extractor to train a JWS model. In our preliminary experiments, we test different dimension choices (100, 200, 1024) for fastText embedding training but no significant performance difference is observed. Therefore we only report the experiment results with a 100-dimension fastText embedding. For the word segmentation model, we use a bi-directional LSTM as the LM. We also test the effection of the hidden state dimension choices (256, 512, 1024) on the performance. Similarly, there is no significant difference is observed so that all the experiments below are conducted with a fixed LSTM hidden state of 256 dimensions.
We trained a 1-layer as well as a 2-layer bi-directional LM for the pre-training. Due to the humongous size of the wikidump data, we split the whole training into 37 splits to make sure the data can be loaded into memory. We follow hyper-parameter settings of (Akbik et al., 2018), the optimization is done with Stochastic gradient descent (SGD) optimizer for 10 epochs over all the 37 splits. A learning rate scheduler with an initial learning rate of 20.0 is deployed. Performance on a validation set is monitored and the scheduler is responsible to reduce the learning rate by a factor of 0.25 while there is no improvement on the validation data for 25 training splits. During fine-tuning LM on the word segmentation dataset, we choose Adam (Kingma and Ba, 2014) as the optimizer. The fine-tuning strategy called *discriminative fine-tuning* (Howard and Ruder, 2018) is also deployed, therefore we set a learning rate of 0.005 for the LSTM part and one of 0.01 for the new appended 0/1 labeling projection layer.
The optimization of the LSTM based word segmentation model is proceeded with Adam optimizer and early stopping. Specifically, the initial learning rate is set to 0.01 and the learning rate is decayed with a factor of 0.5 while there is no improvement on the

validation set for 3 epochs. The training procedure will be stopped when the learning rate gets lower than 0.0001.
Detailed hyperparameters for each experiment settings are shown in Table. 3.1

3.4.3 Results and Analysis

In the word segmentation experiments, we report the F1-score on both validation and test subset, as well as the number of epochs it takes to reach the score for each setting in Table. 3.2. The first two lines in the table show the baselines where LSTM and fastText embedding trained on Wikipedia data are used to perform the word segmentation task. The 3rd and the 4th line show the results of using the pre-trained LM as the only feature extractor as well as combining it with a normal fastText embedding layer respectively. The last two lines show the results of fine-tuning character-level LMs directly to fit the word segmentation task.
As we can tell from the results, the transfer learning of LM through feature-based approaches does not yield any performance boosting over the baseline with a fastText embedding. Furthermore, lowe performance is observed while combining the features extracted from the LM and the fastText layer. The humongous size of input features may be considered as the cause. On the other hand, though performance increase is not observed during the 1-layer LSTM LM based fine-tuning experiment, the 2-layer LSTM LM based experiment shows the highest F1-score among all the experiment settings, including its baseline of 2-layer LSTM + fastText setting, with much less training epochs. It is considered that useful knowledge for word segmentation is learned during the pre-training on the LM objective.

Experiment Setting	F1-score (Valid)	F1-score (Test)	Epoch
1-layer LSTM + fastText	97.04	96.70	114
2-layer LSTM + fastText	97.06	97.10	114
1-layer LSTM + LM features	97.07	96.66	148
1-layer LSTM + fastText + LM features	96.86	96.49	139
1-layer LSTM LM fine-tuning	96.03	95.53	4
2-layer LSTM LM fine-tuning	**97.37**	**97.13**	7

Table 3.2: F1-score (on validation and test data) and the number of epochs for each experiment setting

3.5 Summary

In this research, we investigate the possibility to transfer a character-level LM trained on unlabeled Wikipedia data to tackle the task of Japanese Word Segmentation through both feature-based and fine-tuning based methods. The proposed method does outperform several baselines we set in our experiments, however, the performance gain does not meet our expectations. For the further work, there are mainly two ideas:

- Further error analysis to try to explain the reason for the performance decreasing happen while using the character-level in the feature-based approach.
- Due to the limited computation resource during the period of this research, we only attempt to train an LSTM-based bi-directional language model with Wikipedia dump data. Beyond that, we are also planning to train a Transformer based language model with a Masked Language Model task as BERT (Devlin et al., 2019) propose.

Chapter 4

Contextual Sentence Filtering for Context-Aware Machine Translation

4.1 Context-Aware Machine Translation

Recently, due to the success researchers archived at Sent-MT with the power of NNs, context-aware becomes one of the most actively researched subfields of MT. (Tiedemann and Scherrer, 2017) is one of the earliest attempts on context-aware machine translation by using extended context information under the normal attention-based NMT framework. They took two approaches to extend the context while training NMT models. The first one is the extended source, which means including previous sentences as context to improve the encoder part of the network only. The other approach is to extend translation units, in which both the input of the encoder and the decoder are increased to include more segments. When the size of the extended contextual unit is restricted to 1, the extended source and the extended translation units are also called Model 2+1 and Model 2+2, or 2-to-1 and 2-to-2 respectively. Their experiments on German-English translation with subtitle data shows that NMT framework can handle wider context and is also able to distinguish information coming from different segments and discourse history. The extended context can also help the model to archive higher scores on several metrics including BLEU, chrF3, precision, and recall.

Beyond such kind of context-aware MT systems that uses limited contextual units, recently another approach is getting noticed in which contextual units are fed to the NMT system as long as they fit into memory. (Junczys-Dowmunt, 2019) experimented with sequences of up 1000 subword segments with a deep Transformer based model. They used given document boundaries to concatenate parallel sentences into document sequences. The result document pairs are assured that there is the same number of sentences on both the source and the target sides. Specifically, they add special symbols for document start

($< BEG >$) and end ($< END >$), as well as for sentence separators ($< SEP >$). Another two special symbols ($< BRK >$) and ($< CNT >$) are used instead of ($< END >$) for the end of a sequence and ($< BEG >$) for the next sequence respectively, in cases where documents exceed their length limit of 1000 sub-word tokens.
The idea of including more contextual information does show performance, in the sense of BLEU score, in context-aware machine translation. However, there remain several open questions. In this research, we attempt to answer two of them with our experiments in English-Japanese translation, which are

- *whether concatenating previous sentences that are selected in a fixed way (e.g. always picking the one before) is a proper approach for*
- *whether all the sentences benefit from*

taking additional contextual information into account.

4.2 Dataset

4.2.1 Statistics

Although there are several large enough sentences aligned English-Japanese parallel corpora public available [4], there does not exist open document aligned corpus with sufficient size. In this research, we used a dataset that builds upon our previous work (Rikters et al., 2019), which introduced the initial smaller version of Bussiness Conversation Corpus (BSD). We extended it with the translated versions of the AMI Meeting Corpus (AMI) and the English part of OntoNotes 5.0 (ON) corpus. In the current version, we added more data to the BSD corpus, increasing its size by 1.5 (from 955 scenarios, 30,000 parallel sentences to 1462 scenarios, 44,800 parallel sentences), and combined all three parts into one conversation corpus of considerable size. Detailed statistics for the whole BSD is shown in Table 4.1, and that for the translated version AMI and OntoNotes are shown in Table 4.2.
The conversation corpus alone is not large enough to train real-world NMT sytems so that we supplemented it with a much larger in-house training dataset, which consists of document-aligned parallel news articles. The news corpus provides 298,597 sentence pairs and all the four corpora give us a total training data of 478,138 sentence pairs.
For validation and test data, we use the development and evaluation splits for BSD, which have 2051 and 2120 sentence pairs respectively, since it is the least noisy part of the conversation corpus. Table 4.3 shows an example of the corpus.

[4] http://www.phontron.com/japanese-translation-data.php

Scene	Scenarios	Sentences
JA → EN		
face-to-face	257	7,654
phone call	128	3,741
general chatting	149	4,806
meeting	146	4,943
training	26	989
presentation	9	267
sum	715	22,400
EN → JA		
face-to-face	237	7,161
phone call	154	4,525
general chatting	159	4,457
meeting	148	4,651
training	23	700
presentation	26	906
sum	747	22,400

Table 4.1: Statistics for the full version of BSD, where JA → EN represents scenarios which are written in Japanese then translated into English and EN → JA represents scenarios constructed in the reverse way.

4.2.2 Analysis

In order to identify the main issues of Doc-MT on conversation data, and considering that the full version of the corpus (all three parts) contains scenes other than business ones, we first extend the analysis conducted in Section 3 of (Rikters et al., 2019) to AMI and ON. However, in contrast to the original analysis, which mainly focuses on Japanese→English translation, we investigate the contextual information requirements for English→Japanese machine translation. To make the results comparable to the original work, we use Google Translate [5] to produce automatic translations for the analysis.
Since a single scenario in AMI and ON is much longer than the ones in BSD, instead of sampling based on scenarios, we randomly sample 200 and 100 sentence pairs from ON and AMI respectively. In the case of ON, half of the pairs are from BC and the other half are from Tele. Then we translate the English sentences into Japanese with Google Translate and check the translations for fatal translation errors, ignoring fluency or minor grammatical mistakes.
Unlike the results shown in Japanese→English translation, in which more than a half of

[5]https://translate.google.com/ (Nov 2019)

Dataset	Scene	Scenarios	Sentences
AMI	meeting	171	110,483
OntoNotes	BC	27	14,354
	Tele	46	14,075

Table 4.2: Statistics for translated version of AMI and ON corpora contained in this work. BC stands for broadcasting and Tele stands for telephone conversation.

Scene: general chatting about the welcome party for new colleagues

Japanese		English	
Speaker	Content	Speaker	Content
...	...	...	...
大東	となると、またいつもの居酒屋ですか？。	Oohigashi	Then should we go to the sama Izakaya as always?
高田	たまには別のところがいいな。	Takada	We could try somewhere else for a change.
大東	とういうジャンルにしますか？。	Oohigashi	What kind of food should we choose?.
市田	中華料理とかどう？。	Ichida	How about Chinese?
市田	円卓囲めば、より一層親睦を深められそうだ。。	Ichida	We could bond much easier if everybody sits facing one another at a round table.
大東	中華料理ですか、結構こってり系ですね。	Oohigashi	Chinese food, it's a little heavy.
市田	若い社員がメインだからね。。	Ichida	We have mostly young employees.
...	...	...	...

Table 4.3: An example of the Japanese-English business conversation parallel corpus.

the errors are due to Zero Anaphora[6], there are mainly two types of causes for errors we detected in this analysis, namely phrase ambiguity (PA) and absence of world knowledge (AWK). Detailed results are shown in Table 4.4.
As we can tell from the results, most of the errors (15 out of 18) are caused by phrase ambiguity, for which taking a context sentence(s) into account can be considered as a possible solution. On the other hand, the scenarios in ON-BC contain a variety of named entities (an example from our analysis is Shia, one of the two main branches of Islam) and abbreviations (such as CPC, which stands for Communist Party of China in one example we meet). To solve such an issue, a more broad variety of training data would be required or other additional mechanisms that take world knowledge into account.

[6]A grammatical phenomenon happened in Japanese, in which some arguments of verbs are often omitted from the phrases when they are obvious from the context.

Dataset	Scene	PA	AWK
AMI	meeting	6	0
OntoNotes	BC	5	3
	Tele	4	0

Table 4.4: Two types of causes for fatal errors detected in the English→Japanese machine translation error analysis.

4.3 The One Sentence Before is not Always the Contextual One

To answer the first question, we start by identifying whether one sentence before is the best choice for extending the context. In our setting, given a document pair $D_s = X_1^M$ and $D_t = Y_1^M$ where X_i and Y_i are aligned parallel sentences. With this definition, a normal 1to1 system takes X_i and Y_i as source and target respectively. Then a simple 2-to-1 system used by (Tiedemann and Scherrer, 2017) can be defined as a system takes X_{i-1} and X_i for source and the correspond target Y_i of X_i for target.

To compare with the normal 2-to-1 system, we furthermore trained 5 additional systems which take $X_{i-2}, X_{i-3}, X_{i-4}, X_{i-5}$ as the extended context input respectively for both EN $\rightarrow$ JAand JA $\rightarrow$ EN directions. We concatenate the context sentence and the source sentence with a special symbol "@@CONCAT@@" and feed them into a single encoder. An example should look like "I am a master student . @@CONCAT@@ And I am working on NLP .". For the first k-th sentences where the bias of the context sentence is set to k, we concatenate them with the special symbol only which yields an example looks like "@@CONCAT This is an example.". We then uses Data$_{X_{i-k}}$ to denote the data where X_{i-k} and X_i $(k = 1, \cdots, 5)$ are concatenated as the source sequence. Model that is trained with Data$_{X_{i-k}}$ can therefore be denoted as Model$_{X_{i-k}}$.

We used Transformer architecture based encoder-deocder model. What is worth to note here is that, in the original paper of (Vaswani et al., 2017), after each operation such as multi-head attention and feed-forward projecting, three types of regularization methods are applied in the order of Dropout (Srivastava et al., 2014), residual connection (He et al., 2016), and layer normalization (Lei Ba et al., 2016). However, in the reference implementation code released as tensor2tensor [7], they suggested that moving the layer normalization before each operation as a preprocessing and postprocessing the output with dropout and residual connection only is more robust during learning. In our experiments, we followed the approach taken in the released code.

We used the fairseq toolkit (Ott et al., 2019) to conduct all our MT experiments. The

[7] https://github.com/tensorflow/tensor2tensor

data was tokenized by a single Sentencepiece (Kudo and Richardson, 2018) that is jointly learned on the training data in both English and Japanese. The shared vocabulary size was set to 32,000. For hyperparameter, we followed (Rikters et al., 2019). Both the encoder and the decoder consist of 6 multi-head self-attention layers, each layer has 8 attention heads. We set the dimension of word embeddings and hidden layers to 512 and apply a dropout of 0.2. The maximum sentence length is restricted to 128 symbols for both 1-to-1 and 2-to-1 systems. Thanks to the half-precision training supported by fairseq, we were able to use a larger batch size of 2048 words and a checkpoint frequency of 4000 updates. To train every model until they reach convergence, we used a scheduler to decay the learning rate by a factor of 0.7 while there is no improvement on the validation split for 10 times. The initial learning was set to 0.0001.

During the evaluation, we use a beam search size of 6 to generate output from the source sentences. The output is retokenized with Sentencepiece and the Japanese sentences are further tokenized into words with Mecab (Kudo et al., 2004). The BLEU scores for each system is computed with sacreBLEU [8] on both validation and test splits. In Table 4.5, we report the BLEU when evaluating Model$_{X_{i-k}}$ with Data$_{X_{i-k}}$ with the best models bolded. First of all, we can tell most of the 2-to-1 systems outperform the 1-to-1 baseline which re-confirm the fact that taking additional information into account does improve the performance in most cases. Additionally, we can tell from the results that the most widely used setting of Model$_{X_{i-k}}$ did not give out the best score on both EN $\rightarrow$ JA and JA $\rightarrow$ EN translation directions. It turns out that Model$_{X_{i-5}}$ and Model$_{X_{i-4}}$ archive the best scores in EN $\rightarrow$ JA and JA $\rightarrow$ EN respectively. We can observe that the best-performed models take sentences which locate rather far from the source sentence. Such a result is surprising since it does not match the intuition that sentences closed to each other should be more relevant to each other. It shows that at least in the domain of business conversation, context-aware machine translation needed to consider contextual information more than the one-sentence before.

We furthermore conducted a so-called cross-evaluation to confirm our observation. Other than test Data$_{X_{i-a}}$ which in the same format as the training data that Model$_{X_{i-a}}$ is trained with, we feed all variant of test data Data$_{X_{i-k}}$ where $k = 1, \cdots, 5$ to Model$_{X_{i-a}}$ and evaluate BLEU scores on the generated translations. This can be considered as a robustness test that can be used to identify the best one among the five 2-to-1 systems for each direction. The results for EN $\rightarrow$ JA and JA $\rightarrow$ EN are reported in Table 4.6 and Table 4.7 in which each column represents different models and each row represent the variants of data format. We also report the mean of scores that a single model archives on different variants of data.

From the cross-evaluation results, we can tell that Model$_{X_{i-5}}$ and Model$_{X_{i-4}}$ archives most highest BLEU scores and the highest average score on the five variants of data in

[8] https://github.com/mjpost/sacreBLEU

Models	$\text{Model}_{\text{1-to-1}}$		$\text{Model}_{X_{i-1}}$		$\text{Model}_{X_{i-3}}$	
Data Subset	valid	test	valid	test	valid	test
EN → JA	15.11	14.65	14.82	15.33	14.94	15.20
JA → EN	16.90	17.10	17.13	17.21	17.31	**17.46**

$\text{Model}_{X_{i-3}}$		$\text{Model}_{X_{i-4}}$		$\text{Model}_{X_{i-5}}$	
valid	test	valid	test	valid	test
15.01	14.98	14.69	15.38	**15.46**	**15.43**
17.14	16.89	**17.51**	**17.46**	16.73	17.15

Table 4.5: BLEU scores of baseline models on validation and test splits of BSD

EN → JA	$\text{Model}_{X_{i-1}}$	$\text{Model}_{X_{i-2}}$	$\text{Model}_{X_{i-3}}$	$\text{Model}_{X_{i-4}}$	$\text{Model}_{X_{i-5}}$
$\text{Data}_{X_{i-1}}$	15.33	15.41	15.31	15.31	**15.46**
$\text{Data}_{X_{i-2}}$	15.08	15.20	15.40	**15.53**	15.44
$\text{Data}_{X_{i-3}}$	15.00	**15.45**	14.98	15.37	15.43
$\text{Data}_{X_{i-4}}$	15.18	15.41	15.22	15.38	**15.50**
$\text{Data}_{X_{i-5}}$	14.92	15.01	14.94	15.24	**15.43**
Mean	15.10	15.30	15.17	15.37	**15.45**

Table 4.6: BLEU scores of 2-to-1 cross evaluation for EN → JA translation

EN → JA and JA → EN respectively. Such a result confirms the observation acquired from the normal evaluation. Moreover, it reveals the fact that the model achieves the best result also owns the robustness to handle data in other formats, sometimes even outperform the model that is trained on that data format. The fact imposes that selecting proper contextual sentences during training is crucial not only for performance improvement but also for the robustness of the resulted model.

4.4 Contextual Sentence Filtering by Fine-tuning BERT

Considering the facts observed in the experiments as described above, the question about how to identify the proper context sentences arises naturally. We then investigate the necessary by applying transfer learning on pre-trained language model BERT.
At first, we calculate the oracle BLEU scores based on our 2-to-1 NMT models to show how far a perfect contextual sentence filtering model can archive if it can locate the correct context for each sentence. For each example in the validation and test set, we compute sentence-level BLEU for the output of all the give 2-to-1 models corresponding to the

JA → EN	Model$_{X_{i-1}}$	Model$_{X_{i-2}}$	Model$_{X_{i-3}}$	Model$_{X_{i-4}}$	Model$_{X_{i-5}}$
Data$_{X_{i-1}}$	17.21	17.21	16.99	17.30	**17.31**
Data$_{X_{i-2}}$	17.00	17.46	16.98	**17.57**	17.18
Data$_{X_{i-3}}$	17.07	**17.31**	16.89	17.13	17.24
Data$_{X_{i-4}}$	17.11	17.24	16.86	**17.46**	17.04
Data$_{X_{i-4}}$	16.92	**17.47**	16.98	17.41	17.15
Mean	17.06	17.34	16.88	**17.37**	17.18

Table 4.7: BLEU scores of 2-to-1 cross evaluation for JA → EN translation

example and select the one which achieves the highest score as the oracle example. Then we get a list of oracle translation which is generated by the best one of our models for each sentence and recalculate the corpus BLEU of this oracle output.
The additional contextual sentence used by a 2-to-1 model also brings randomness which a 1-to-1 does not have. To omit such randomness, we trained a noised 2-to-1 model in which a random sentence sampled from another corpus (JESC [9] is used in our experiments) is taken as the contextual sentence. Such a 2-to-1 model should treat the additional sentence as noise and omit it while generating the target. During oracle score calculation, we first feed Data$_{X_{i-k}}$ to the noised 2-to-1 model to get a base score, namely $Score_{\text{base\_}X_{i-k}}$, in which X_{i-k} should be treated as noise only. Then Data$_{X_{i-k}}$ is fed to Model$_{X_{i-k}}$ to acquire a normal score $Score_{X_{i-k}}$. When selecting oracle sentences that get the highest score among all the models, we use a so-called $Score_{\text{real}}$, which keeps the same as the normal score for Model$_{\text{1-to-1}}$ and is calculated as follows for all the other models:

$$Score_{\text{real}} = Score_{X_{i-k}} - (Score_{\text{base\_}X_{i-k}} - Score_{\text{1-to-1}})$$

We show the final results in Table 4.8. The numbers reveal the upper bound performance of such a context sentence filtering based approach.
We also illustrate the distribution of the "correct" contextual sentences' location in Figure 4.1 and Figure 4.2. From the graph, we can tell several facts about the context information requirement in context-aware machine translation:

- Different language has different needs for contextual information. As our data shows, EN → JA tends to refer to sentences that have a distance as far as 5 sentences before (on test split). On the other hand, JA → EN tends to refer to the sentences that short distance such as 2 or 3.

- The requirements on the contextual information even vary for the same language on a different subset of data. For example, in JA → EN translation on BSD data,

[9] https://nlp.stanford.edu/projects/jesc/index_ja.html

	Valid	Test
EN → JA	21.13	20.84
JA → EN	23.26	23.44

Table 4.8: Oracle scores on validation and test data if there exists a perfect model that can always select the correct contextual sentence.

sentences in the validation split tend to benefit from the information for the sentence 2 sentences before while the number is 3 for the test split.

The results showed that the contextual information requirement can be so complicated that a rule-based context selector is not enough in most cases. Such a fact imposes that an ML model may be needed to address the issue of contextual sentence filtering for MT systems.

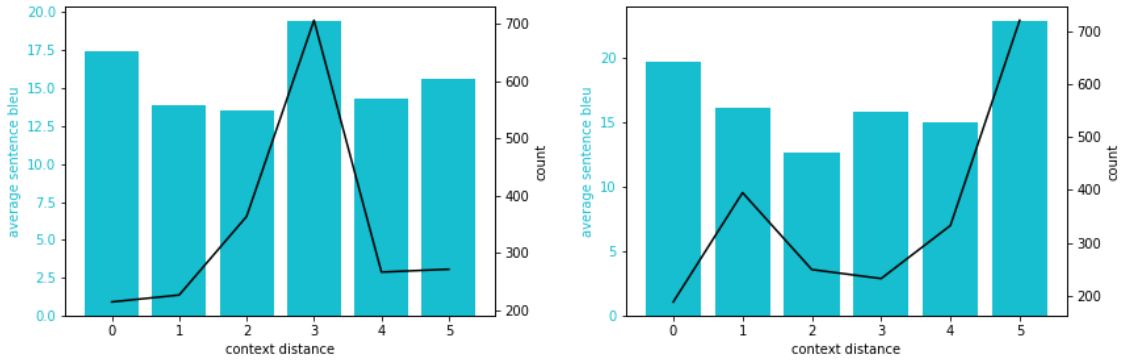


Figure 4.1: The distribution of contextual sentences' location in the oracle results in EN → JA . Figure on the left is for the validation split and the one on the right is for the test split. Label 1 to 5 means that distance between the source sentence and the contextual sentece which yield the oracle score. Label 0 stands for sentences archive best BLEU without taking additional information into account. For each graph, the left y-axis represents average sentence BLEU scores and the right y-axis represents count for each contextual sentence distance.

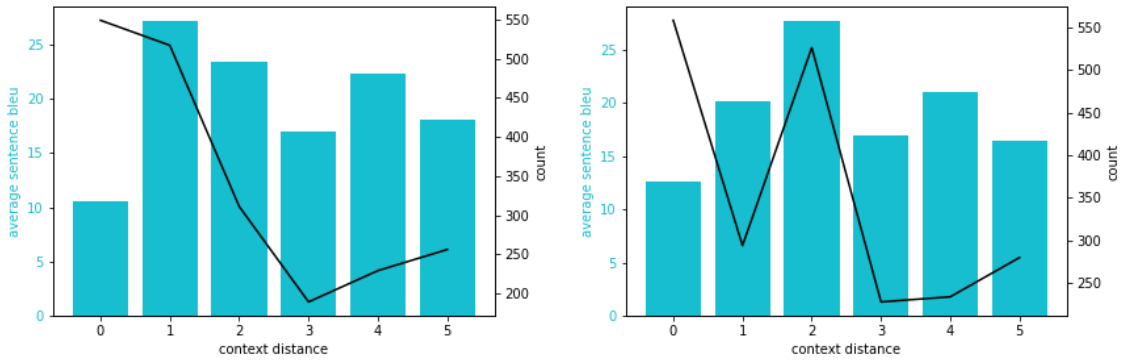


Figure 4.2: As same as Figure 4.1 but for JA $\rightarrow$ EN

4.4.1 Next Sentence Prediction Fine-tuning

The problem of contextual sentence filtering can be treated as a sequence pair classification task. For each source sentence X_i, we fed it with one sentence before it X_{i-k} within a context window w and output a label from $\{0, 1\}$ to indicate whether X_{i-k} is the contextual sentence for X_i. Under such a setting, BERT (Devlin et al., 2019) which utilized the task called Next Sentence Prediction (NSP) during pre-training was chosen as our start point.

We first fine-tune BERT with NSP objective with the same training data as that used in MT experiments for 2 epochs. During fine-tuning, the pair of X_{i-1} and X_i are treated as a positive example and the pair consists of X_ia sampled sentence among all the sentences before it are treated as a negative exmaple. Then we take an approach similar to (Sun et al., 2019), feed every X_{i-k} along with X_i and score it with the output of BERT. The difference between (Sun et al., 2019) and our method is that we used the hidden state corresponding to label "1" before the softmax function instead of the probability corresponding to "1" after the softmax. After the scoring procedure, we select the X_{i-k} with the highest score as the contextual sentence for X_i. The scoring and selection are performed on the whole training, validation, and test to form a novel data format which can be denoted as Data$_{filtered}$. A 2-to-1 NMT system is then trained and validated with the filtered data.

We used the uncased BERT-Base for EN $\rightarrow$ JA. For JA $\rightarrow$ EN, instead of using the multi-lingual BERT released by Google, we use a Japanese-only BERT model released by Tohoku University [10]. During fine-tuning, we use AdamW optimizer (Loshchilov and Hutter, 2017) with a learning rate of 0.00005. We set the window size w to 5 for the contextual sentence filtering procedure. For 2-to-1 NMT model training, we used the same

[10]https://github.com/cl-tohoku/bert-japanese

	BERT FT Filtered		Best from the baselines	
Data subset	Valid	Test	Valid	Test
EN → JA	14.70	15.28	**15.46**	**15.43**
JA → EN	17.40	**17.97**	**17.51**	17.46

Table 4.9: BLEU score from normal evaluation for the BERT Fine-tuning (FT) based contextual sentences filtering.

hyperparameter as all the baselines and report the BLEU score the final NMT model in Table 4.9.
As we can tell from the results, 2-to-1 systems trained with the BERT filtered data outperformed the best of the baselines in both JA → EN by a margin as large as 0.51 BLEU score. On the other hand, the same result did not hold on the validation split. In the case of EN → JA, the decrease of BLEU score reached 0.73 and 0.48 on validation and test splits resepctively. Such a result reveals that the NSP fine-tuning does provide some knowledge for contextual sentence filtering for JA → EN but it does not generalize well on different languages and different subsets of data.

4.4.2 Multi-Tasked Fine-tuning

We blame the poor generalization of NSP fine-tuning to the absence of knowledge related to MT during the procedure of fine-tuning. Therefore BERT only learned about how to distinguish sentence pairs with are adjacent to each other from those are not. We then hypothesized that to make sure the fine-tuned BERT be able to identify contextual sentences that can provide information crucial to translation, MT related knowledge should also be learned during fine-tuning of BERT. Therefore we extend the standard fine-tuning framework by comprehending it with a single layer LSTM based decoder. As illustrated in Figure 4.3, comparing to the normal NSP fine-tuning method in which the hidden state of [CLS] is only fed to the classifier, we also use that vector as the initial hidden state of the LSTM decoder and feed the target sentence to the decoder. Additionally, we also concatenate the hidden state of [CLS] to the token embedding at the target side before feeding them into the decoder as (Artetxe and Schwenk, 2019) suggested.
We use the same hyperparameter as the NSP fine-tuning experiments except that we apply a larger learning rate of 0.001 for the decoder. The window size setting at inference time keeps the same as 5. The BLEU scores on the 2-to-1 NMT models trained with the data filtered with our proposed method are reported in Table 4.10.
By adding the decoder into the fine-tuning framework, significant improvements of 0.21 and 0.57 BLEU is observed on the validation and test splits respectively are observed in

	Multi-task FT Filtered		Best from the baselines	
Data subset	Valid	Test	Valid	Test
EN → JA	14.94	**15.52**	**15.46**	15.43
JA → EN	17.10	17.33	**17.51**	**17.46**

Table 4.10: BLEU score from normal evaluation for the proposed filtering method.

EN → JA translation. However, the performance on the validation split is still lower than the strongest baseline model. Furthermore, in JA → EN translation, the introduction of decoder brought a drop of 0.3 and 0.64 on the two subsets. Such a result reveals that our multi-task modification to the fine-tuning framework still can not solve the poor generalization issue. There drives us to the next reform.

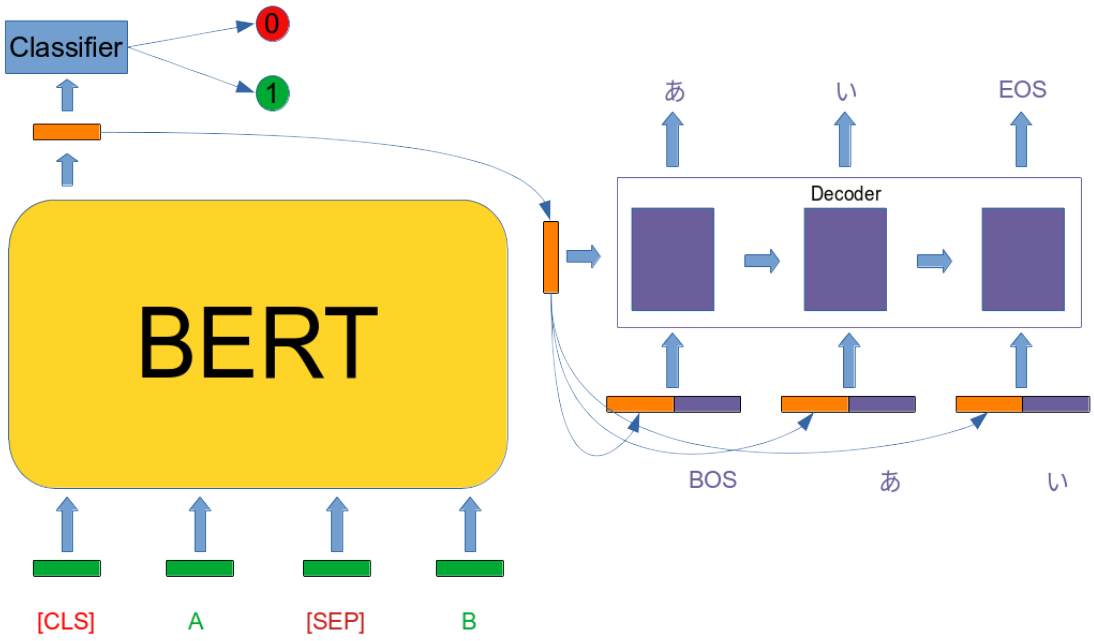


Figure 4.3: Illustrations about the proposed multi-task fine-tuning on BERT.

4.4.3 Better Negative Examples

In the basic multi-task fine-tuning framework we proposed, the same target sentence is fed to the decoder for both the positive and the negative example for a source sentence. Such

	Multi-task FT + Translated Source		Best from the baselines	
Data subset	Valid	Test	Valid	Test
EN → JA	**15.47**	**15.58**	15.46	15.43
JA → EN	**17.68**	**17.72**	17.51	17.46

Table 4.11: BLEU score from normal evaluation for the proposed filtering method with translated source as target for negative examples.

a setting did not improve the generalization capability of NSP fine-tuing. Therefore we hypothesized that the sentence on the target side can not provide enough teacher signal to the source side BERT so that the crucial knowledge to identify contextual sentences can not be acquired during the fine-tuning. Some types of penalties should be introduced to help BERT distinguish between good contextual sentences and bad ones. Our idea here is instead of feeding the same target which is the correct one to decoder for both positive and negative examples, we feed a pseudo target sentence which is translated from the source sentence by a sentence-level NMT model. We hypothesized such an approach can provide more teacher signals to BERT and therefore help it acquire knowledge that is crucial for contextual sentence filtering.

For the pseudo target sentence generation, we used the base model that is released by NTT which is trained with JParaCrawl [11]. All the other hyperparameter settings are kept the same as the basic multi-task fine-tuning experiment. The BLEU scores of the resulted 2-to-1 NMT models are reported in Table 4.11.

As we can tell from the table, in both EN → JA and JA → EN , the performance of the sentence filtering based NMT model is boosted by the multi-tasked fine-tuning framework and the refined negative examples. The BLEU scores exceed all the baselines models on both validation and test splits. However, the score on the test split for JA → EN did not exceed the one yielded by the NSP fine-tuning based filtering model.

The results reveal that the full version of our proposed method, including the multi-task fine-tuning framework and the idea of using a translated source as the target sentence for negative examples, did show significant improvement in both EN → JA and JA → EN , as well as solved the generalization issue among different subsets of data. But it did not beat the NSP fine-tuning baseline in all experiment settings.

[11] http://www.kecl.ntt.co.jp/icl/lirg/jparacrawl/

4.4.4 Analysis

The results of our experiments reveal that the NSP fine-tuning and our proposed multi-task fine-tuning have different behavior and they have strength at different translation directions. A natural question arises here is what is the differences between the contextual sentences selected by the two kinds of models. We collect the filtered results on the test split for our best performed multi-task fine-tuning based model and normal NSP fine-tuning based one. The graph on the left of Figure 4.4 illustrates the distribution of the selected contextual sentences' relative positions selected by both of the models for EN $\rightarrow$ JA . And the green histograms in the graph on the right side also shows the number of cases in which our proposed method picks the different contextual sentences from the NSP fine-tuning based one. The observation can be mainly summarized as:

- Even being comprehended by a decoder, the fine-tuning procedure in our proposed method is still highly relevant to the normal NSP. As we can tell from the graph, both the proposed method and the NSP only method produce a context filter that similarly picks contextual sentences. Especially both of them are most likely to pick the sentence before. The proposed method even picked one sentence before more often.

- On the other hand, as we can tell the graph on the right side, these two methods are picking different contextual sentences for the same sentence in many cases even they share a similar overall distribution. This fact reveals that even they are highly related, they are still filtering contextual sentences in significant different behaviors.

Figure 4.5 shows the same graphs for JA $\rightarrow$ EN. What is confusing here is that the NSP fine-tuning based filter model decide that the source sentence does not require a contextual sentence mostly, regardless the fact that it is trained with a objective to select the one sentence before. We do not have a proper hypothesis to explain this for now. Further exploration is still necessary.
With the positions of the oracle sentences, which are identified as described at the start of this section, as ground truth of the position of the contextual sentences, we also exam how many does each of the two methods select correctly for the test split. We report the results in 4.12. From this point of view, the high performance of JA $\rightarrow$ EN test split can be caused by this high accuracy. On the other hand, the accuracy does not vary significantly in EN $\rightarrow$ JA for both our proposed method and the NSP fine-tuning based approach.
It is deserving to note that the NSP fine-tuning based filter does produce a distribution of the contextual sentences' positions which is similar to the oracle one in JA $\rightarrow$ EN. Such similarity could also be the reason for the high score of the NSP fine-tuning approach on the test split for JA $\rightarrow$ EN. However, such effectiveness does not seem robust since it could not generalize well even the distribution of the valid split in JA $\rightarrow$ EN is more similar to

	Multi-task FT + Translated Source	BERT NSP FT
EN → JA	241 / 2120	246 / 2120
JA → EN	274 / 2120	339 / 2120

Table 4.12: Precision of two types of filters we experimented on the test split, with the oracle examples as the ground truth.

that of filtered contextual sentences' positions. Further exploration is required to confirm whether the cause for the high performance on JA → EN test split is randomness or not.

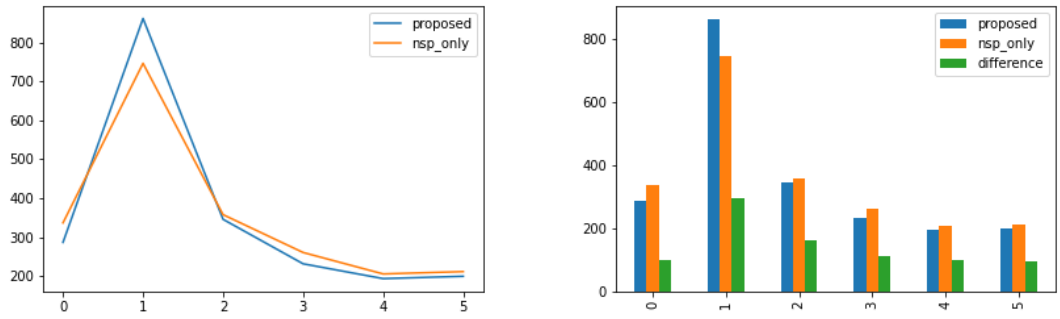


Figure 4.4: The left shows the distribution of contextual sentences filtered by our proposed method and the normal NSP fine-tuning for EN → JA on the test split. The figure on the right also shows the number of cases in which the two methods picked different contextual sentence for the same sentence.

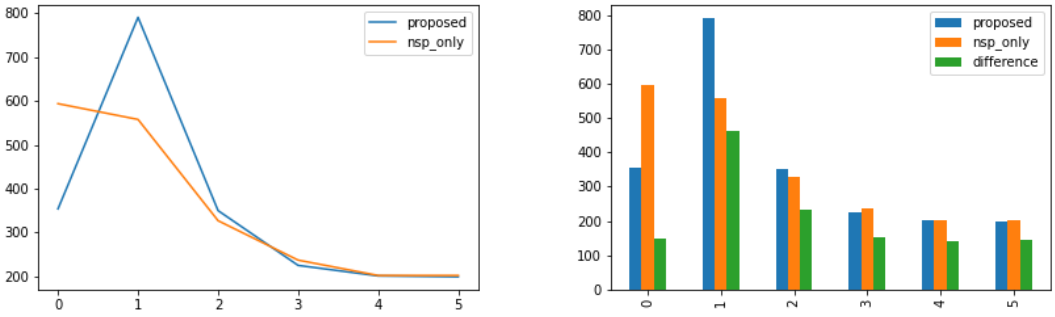


Figure 4.5: As same as Figure 4.4 but for JA $\rightarrow$ EN

Chapter 5

Conclusion

5.1 Summary of Findings

This research investigates the effectiveness of applying transfer learning based on language models on two tasks. Over the course of this thesis, we present both our preliminary example about applying transfer learning on character-level language model to solve the Japanese Word Segmentation task, as well as our main experiments to investigate the necessity and possibility of contextual sentence filtering for context-aware machine translation. Our main findings can be mainly summarized as:

- Additional information is usually required to acquire accurate translations in context-aware machine translation. As we can tell from the analysis of our 12 baseline concentration machine translation systems, most of the sentences can acquire higher BLEU scores by taking previous sentences into account. Another fact revealed by our experiment that selecting proper contextual sentences is not only crucial for performance but also the robustness of the final model.
- Different language has different requirements for contextual information. It is shown by the oracle sentence distributions on our test set for both EN $\rightarrow$ JA and JA $\rightarrow$ EN translation. For oracle results we computed based on our baseline models, context-aware translation for EN $\rightarrow$ JA benefits most from taking the first previous sentence or the sentence before that one. On the other hand, JA $\rightarrow$ EN translation sometimes requires further context sentences as far as a previous fifth one (the window size we set in our experiments) at least in business scene conversations.
- The proposed multi-task based fine-tuning on BERT shows effectiveness and robustness in both EN $\rightarrow$ JA and JA $\rightarrow$ EN by beating all the baselines with fixed-distance contextual sentences. Translated source sentences on the target side provide more teacher signals and become the key element for the performance-boosting.

However, it does not outperform the NSP fine-tuning based filtering model on test split in JA $\rightarrow$ EN , which requires more exploration to identify whether the reason is randomness or not.

5.2 Future Work

Although the proposed multi-task fine-tuning method shows effectiveness in contextual sentence filtering in EN $\rightarrow$ JA translation, there remains a variety of questions require to be answered by further investigation.

First, for the reason why the same methodology does not work in JA $\rightarrow$ EN direction, one of the possible explanations in our mind is that Japanese requires sentences further than the one before as contextual information. The property or features of such contextual sentences is not well learned since we use the previous sentence as a positive example during the fine-tuning. However, more analysis is required to verify the real cause of the difference.

Second, as mentioned above, we use the previous one sentence as a positive example during fine-tuning which is oversimplified and can be considered problematic. Better methodology to label the training data for our contextual sentence filtering model is required to improve the performance further. One possible approach in our mind is fully utilizing the 12 baseline models to generate positive labels based on metrics such as BLEU scores and the negative log-likelihood loss.

Third, in this thesis, we only experiment with simple concatenation context-aware machine translation systems. Beyond that, we are planning to train document-level systems with a limited size of crucial contextual information filtered by our proposed method and compare it with systems utilizing the whole document.

Forth, all the evaluation in our MT experiments is conducted with BLEU, which sometimes can be irrelevant to the semantics of a sentence. As the next step, we want to perform human evaluation to identify whether NMT systems trained with the proposed methods can produce better translation than baselines in the sense of semantics.

Last but not least, we only test our proposed method on limit sized English-Japanese data in news and conversation domains. Further experiments to investigate the effectiveness of the proposed method on other language pairs, as well as on larger datasets and datasets from other domains.

Bibliography

Alan Akbik, Duncan Blythe, and Roland Vollgraf. 2018. Contextual string embeddings for sequence labeling. In *COLING 2018*.

Radford Alec, Narasimhan Karthik, Salimans Tim, and Sutskever Ilya. 2018. Improving language understanding by generative pre-training.

Mikel Artetxe and Holger Schwenk. 2019. Massively multilingual sentence embeddings for zero-shot cross-lingual transfer and beyond. *TACL*, 7:597–610.

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

Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. 2017. Enriching word vectors with subword information. *TACL*.

Peter F. Brown, John Cocke, Stephen A. Della-Pietra, Vincent J. Della-Pietra, Frederick Jelinek, John D. Lafferty, Robert L. Mercer, and Paul Rossin. 1990. A statistical approach to machine translation. *Computational Linguistics*, 16(2):76–85.

Peter F. Brown, Stephen A. Della-Pietra, Vincent J. Della-Pietra, and Robert L. Mercer. 1993. The mathematics of statistical machine translation. *Computational Linguistics*, 19(2):263–313.

Kyunghyun Cho, Bart van Merriënboer, Dzmitry Bahdanau, and Yoshua Bengio. 2014a. On the properties of neural machine translation: Encoder–decoder approaches. In *Proceedings of SSST-8, Eighth Workshop on Syntax, Semantics and Structure in Statistical Translation*, pages 103–111, Doha, Qatar. Association for Computational Linguistics.

Kyunghyun Cho, Bart van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014b. Learning phrase representations using RNN encoder–decoder for statistical machine translation. In *Proceed-*

ings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP), pages 1724–1734, Doha, Qatar. Association for Computational Linguistics.

Ronan Collobert, Jason Weston, Léon Bottou, Michael Karlen, Koray Kavukcuoglu, and Pavel Kuksa. 2011. Natural language processing (almost) from scratch. *J. Mach. Learn. Res.*, 12:2493–2537.

Andrew M Dai and Quoc V Le. 2015. Semi-supervised sequence learning. In C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, and R. Garnett, editors, *NIPS 2015*, pages 3079–3087.

J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. 2009. ImageNet: A Large-Scale Hierarchical Image Database. In *CVPR 2009*.

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

Zellig S. Harris. 1954. Distributional structure. *WORD*, 10(2-3):146–162.

K. He, X. Zhang, S. Ren, and J. Sun. 2016. Deep residual learning for image recognition. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 770–778.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation*, 9(8):1735–1780.

Jeremy Howard and Sebastian Ruder. 2018. Universal language model fine-tuning for text classification. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 328–339, Melbourne, Australia. Association for Computational Linguistics.

Marcin Junczys-Dowmunt. 2019. Microsoft translator at WMT 2019: Towards large-scale document-level neural machine translation. In *Proceedings of the Fourth Conference on Machine Translation (Volume 2: Shared Task Papers, Day 1)*, pages 225–233, Florence, Italy. Association for Computational Linguistics.

Nal Kalchbrenner and Phil Blunsom. 2013. Recurrent continuous translation models. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1700–1709, Seattle, Washington, USA. Association for Computational Linguistics.

Diederik Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *International Conference on Learning Representations*.

Ryan Kiros, Yukun Zhu, Ruslan R Salakhutdinov, Richard Zemel, Raquel Urtasun, Antonio Torralba, and Sanja Fidler. 2015. Skip-thought vectors. In *NIPS 2015*.

Yoshiaki Kitagawa and Mamoru Komachi. 2017. Long short-term memory for japanese word segmentation. *arXiv:1709.08011*.

Taku Kudo and John Richardson. 2018. Sentencepiece: A simple and language independent subword tokenizer and detokenizer for neural text processing. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 66–71.

Taku Kudo, Kaoru Yamamoto, and Yuji Matsumoto. 2004. Applying conditional random fields to japanese morphological analysis. In *EMNLP 2004*.

Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E. Hinton. 2016. Layer Normalization. *arXiv e-prints*, page arXiv:1607.06450.

Ilya Loshchilov and Frank Hutter. 2017. Decoupled weight decay regularization. In *ICLR*.

Minh-Thang Luong, Hieu Pham, and Christopher D. Manning. 2015. Effective Approaches to Attention-based Neural Machine Translation.

Bryan McCann, James Bradbury, Caiming Xiong, and Richard Socher. 2017. Learned in translation: Contextualized word vectors. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *NIPS 2017*, pages 6294–6305.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *NIPS 2013*.

Hajime Morita, Daisuke Kawahara, and Sadao Kurohashi. 2015. Morphological analysis for unsegmented languages using recurrent neural network language model. In *EMNLP 2015*.

Graham Neubig, Yosuke Nakata, and Shinsuke Mori. 2011. Pointwise prediction for robust, adaptable Japanese morphological analysis. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 529–533, Portland, Oregon, USA. Association for Computational Linguistics.

Franz Josef Och and Hermann Ney. 2000. Improved statistical alignment models. In *Proceedings of the 38th Annual Meeting of the Association of Computational Linguistics (ACL)*.

Myle Ott, Sergey Edunov, Alexei Baevski, Angela Fan, Sam Gross, Nathan Ng, David Grangier, and Michael Auli. 2019. fairseq: A fast, extensible toolkit for sequence modeling. In *Proceedings of NAACL-HLT 2019: Demonstrations*.

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

Jeffrey Pennington, Richard Socher, and Christopher D Manning. 2014. Glove: Global vectors for word representation. In *EMNLP 2014*.

Matthew E. Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. In *NAACL 2018*.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. Squad: 100,000+ questions for machine comprehension of text. In *EMNLP 2016*.

Matīss Rikters, Ryokan Ri, Tong Li, and Toshiaki Nakazawa. 2019. Designing the business conversation corpus. In *Proceedings of the 6th Workshop on Asian Translation*, pages 54–61, Hong Kong, China. Association for Computational Linguistics.

Peter Shaw, Jakob Uszkoreit, and Ashish Vaswani. 2018. Self-attention with relative position representations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*, pages 464–468, New Orleans, Louisiana. Association for Computational Linguistics.

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15:1929–1958.

Chi Sun, Luyao Huang, and Xipeng Qiu. 2019. Utilizing BERT for aspect-based sentiment analysis via constructing auxiliary sentence. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 380–385, Minneapolis, Minnesota. Association for Computational Linguistics.

Ilya Sutskever, Oriol Vinyals, and Quoc V Le. 2014. Sequence to sequence learning with neural networks. In Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 27*, pages 3104–3112. Curran Associates, Inc.

Jörg Tiedemann and Yves Scherrer. 2017. Neural machine translation with extended context. In *Proceedings of the Third Workshop on Discourse in Machine Translation*, pages 82–92, Copenhagen, Denmark. Association for Computational Linguistics.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *NIPS 2017*, pages 5998–6008.

Elena Voita, Rico Sennrich, and Ivan Titov. 2019. When a good translation is wrong in context: Context-aware machine translation improves on deixis, ellipsis, and lexical cohesion. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 1198–1212, Florence, Italy. Association for Computational Linguistics.

Elena Voita, Pavel Serdyukov, Rico Sennrich, and Ivan Titov. 2018. Context-aware neural machine translation learns anaphora resolution. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1264–1274, Melbourne, Australia. Association for Computational Linguistics.

Publication List

- Matīss Rikters, Ryokan Ri, Tong Li and Toshiaki Nakazawa. Designing the Business Conversation Corpus. In *Proceddings of the 6th Workshop on Asian Translation*, pages 54-61, Hongkong, China, November 2019. Association of Computational Linguistics.
- 李桐, 鶴岡慶雅. 文字レベル言語モデルの転移学習による日本語単語分割. 言語処理学会第 25 回年次大会, 2019.

Acknowledgements

The two years of the master's program are too short for starting to work on research projects in an unfamiliar area. I am thankful for all the people I met and accompanied me during the short period. First, I would like to thank my supervisor Prof. Tsuruoka and the leader of our context-aware machine translation research project Prof. Nakazawa. Without the freedom you gave me during research topic selection, I could not enjoy these two years by working on projects driven by my interests. Your sensible and wisdom advice guided me to adjust the direction of my research when it was not going well.
I want to thank my family to support me to proceed to graduate school in a foreign country. Live and study abroad for the first time is difficult for both me and my family. I can not make the way without their support at home in China.
I am fortunate to be accompanied by all the members of my lab, especially the senior students Kawamu-san and Mizutani-san, as well as my colleagues Haoyu Zhang, Kano-san, Laige Peng, Ryokan-san, Yasui-san. and they not only provided advice to my research and study but also helped me to get used to life in Japan in a short time. I am also thankful for the advice provided by all the post-doctor researchers, including Eriguchi-san, Hashimoto-san, and Matīss Rikters. Matīss, thank you for your information and experiment setting which helped me solve a lot of issues at the early stage of this thesis.
I also want to thank the Rotary Yoneyama Memorial Foundation, who not only generally provided scholarship to support my life over the two years, but also provided a platform for me to communicate with students working in different areas who comes from other countries. With their support, I lived for two years without financial stress and met lots of foreign friends.
Last but not least, I would like to express my gratitude to the project, by whom this research is supported, "Research and Development of Deep Learning Technology for Advanced Multilingual Speech Translation" of the Commissioned Research of National Institute of Information and Communications Technology (NICT), JAPAN.