Doctoral Dissertation

Interpretable Neural Machine Translation from Translation to Post-Editing

Hiroyuki Deguchi
Program of Information Science and Engineering
Graduate School of Science and Technology
Nara Institute of Science and Technology

Supervisor: Professor Taro Watanabe
Natural Language Processing Lab. (Division of Information Science)

Submitted on September 12, 2024

A Doctoral Dissertation
submitted to Graduate School of Science and Technology,
Nara Institute of Science and Technology
in partial fulfillment of the requirements for the degree of
Doctor of Engineering

Hiroyuki Deguchi

Thesis Committee:

Supervisor	Taro Watanabe
	(Professor, Division of Information Science)
Co-supervisor	Sakriani Sakti
	(Professor, Division of Information Science)
Co-supervisor	Hidetaka Kamigaito
	(Associate Professor, Division of Information Science)
Co-supervisor	Masao Utiyama
	(National Institute of Information and Communications Technology)
Co-supervisor	Satoshi Nakamura
	(Professor, The Chinese University of Hong Kong, Shenzhen)

Interpretable Neural Machine Translation from Translation to Post-Editing[1]

Hiroyuki Deguchi

Abstract

Neural machine translation (NMT) has achieved sufficient translation quality in the general domain, but not yet in the out-of-domain. Therefore, post-editing (PE), which manually corrects mistranslations, is still crucial, especially in fields where mistakes are not allowed, e.g., the medical domain. This dissertation tackles these problems from translation to post-editing using interpretable models. We firstly prevent the degradation of the translation quality in the out-of-domain. In previous work, kNN-MT adapted NMT models to various domains using the example-based approach; however, the example search is time-consuming and the decoding speed becomes two orders of magnitude slower than that of standard NMT. To improve the decoding speed of kNN-MT, we propose subset kNN-MT, which reduces the search space to the neighboring examples of the input sentence and employs an efficient computation method using the distance lookup table. Subset kNN-MT achieved a speed-up of up to 134.2 times and an improvement in BLEU score of up to 1.6 compared with kNN-MT in the De–En domain adaptation task. The other problem is to efficiently check and correct translation errors that still occur despite improvements in translation quality by subset kNN-MT. We then aim to improve the efficiency of human PE. Previous automatic

[1] Doctoral Dissertation, Graduate School of Science and Technology, Nara Institute of Science and Technology, September 12, 2024.

PE (APE) models attempt to correct the outputs of an MT model; however, many APE models are based on sequence generation, and thus their decisions are harder to interpret for human post-editors. We propose an edit-based PE model, which breaks the editing process into two steps, "error detection" and "error correction". The detector model tags each MT output token whether it should be corrected and/or reordered while the corrector model generates corrected words for the spans identified as errors. Experiments on the WMT'20 En–De and En–Zh APE tasks showed that our detector–corrector improved translation edit rate (TER) compared to not only an edit-based model but also a black-box sequence-to-sequence model by 0.7 points in En–De and En–Zh. Moreover, our model is more explainable than sequence-to-sequence models because it is based on edit operations.

Keywords:
machine translation, k-nearest-neighbor search, post-editing, explainability, natural language processing

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Background

Machine translation (MT) is one of the most important techniques studied since the dawn of computational linguistics, and is mainly used for two purposes: understanding information from texts written in a foreign language, called assimilation, and spreading information by converting texts written in one language into another language, called dissemination.

Until now, various types of machine translation have been studied. Early MT was rule-based MT (RBMT), which used manually defined the dictionary and grammar. Because RBMT requires checking that the dictionary and grammar are consistent, it is costly to add new translation rules or extend to other languages. Many modern MT systems employ corpus-based MT, which automatically acquires translation rules from parallel data. Example-based MT (EBMT), which refers to translation examples obtained from bilingual corpora at run time [84]; statistical machine translation (SMT), which uses statistical information learned from corpora [8]; and neural machine translation (NMT), which uses a neural network trained to generate the target sentence from the given source sentence [112, 3, 72, 121, 39, 113]. EBMT, SMT, and NMT are also called corpus-based MT, which acquires translation rules from bilingual corpora. Among them, NMT has achieved sufficient translation quality and is widely used. However, it sometimes makes errors [90] in the out-of-domain; therefore, post-editing (PE), which manually corrects mistranslations generated from MT systems, is still crucial in the real world, especially in fields where mistakes are not allowed, e.g.,

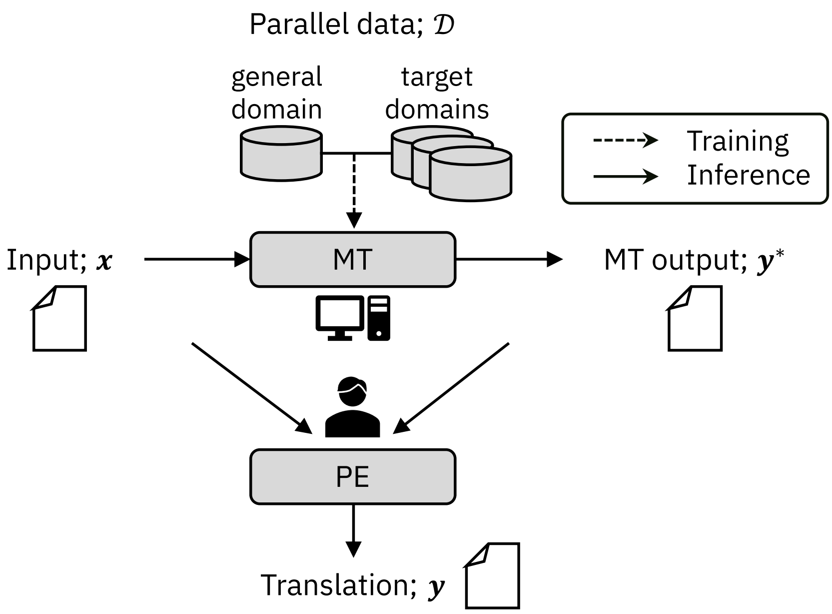


Figure 1.1: Overview of the translation process.

the medical domain. Figure 1.1 shows the translation process in the real world. Typically, MT is trained from the parallel data, including the general domain and various target domains, and then human translators check the output of the MT system and refine the translation. Note that the MT system needs to be additionally trained when new domain data is added.

1.2 Challenges in Neural Machine Translation

The challenges of translation performance of NMT include out-of-domain translation [17, 65], decoding methods to obtain the optimal translations [125, 32], context-aware and document-level translation [54, 77], low-resource languages [70, 77]. Here we focus on the in-domain and out-of-domain translation. In-domain translation has been improved by various methods: using syntactic information [33, 14, 23, 9], reranking the translation candidates to find the most promising candiate [67, 36], employing curriculum learning approaches [4, 82], etc. Even though translation performance is improved in the in-domain, that of out-of-domain is still low, and the improvement of translation quality for various domains is an open issue of NMT. To address the issue, example-based methods have been proposed, but k-nearest-neighbor machine translation, which achieved state-of-the-art translation performance in out-of-domain, is significantly slower than standard translation models, making it difficult to use in the real world.

In addition, industrial translation for information dissemination requires accurate translations in a variety of specialized domains. Currently, typical industrial translation uses MT systems to generate the translation draft and then do post-editing to refine the translation. While the translation quality of NMT has improved in recent years, the workload of post-editing is still heavy, and it is necessary to tackle the assistance of post-editing to reduce the human workload. To improve the productivity of post-editing, it is necessary to develop a model to assist in the editing process. For example, if it can help in the time-consuming task, e.g., finding mistranslations and omissions, it will reduce the human workload.

1.2.1 Domain Adaptation

Many NMT models are trained on large corpora of general domains such as web articles and news articles, and adapted to target domains such as medical and IT documents. Typically, NMT models are adapted by training on each target domain data, which requires additional training costs. The Workshop on Machine Translation (WMT), an international competition for machine translation, held the news translation task, but after 2022, it was replaced with the mixed-domain translation task [64]. Domain adaptation for various domains has attracted attention in the machine translation field.

The simplest way of domain adaptation is to prepare the domain data and fine-tune an MT model [71, 105, 18]. However, it requires additional training costs for each domain and the in-domain translation performance may be degraded by fine-tuning [44].

Some previous work tackled the problems of additional training costs and catastrophic forgetting in domain adaptation by using example-based approaches, which retrieve translation examples from parallel data or translation memory during decoding [130, 42, 61]. kNN-MT [61] can use any pre-trained NMT models without additional training and modification, and it has achieved state-of-the-art translation performance in domain adaptation. The reason why kNN-MT is so powerful is that it searches translation examples based on rich neural representations with context information, and it also allows flexible search by the token-level retrieval, whereas previous models [130, 42] retrieve similar sentences based on the edit distance. It not only improved the translation quality in out-of-domain but also improved interpretability through example-based generation. However, the translation speed was two orders of magnitude slower than the standard NMT, which is a major challenge.

1.2.2 Post-Editing

Post-editing (PE) is crucial in the real world, which corrects mistranslations, improves fluency, complements omitted translations, etc. In industrial translation, human translators creates the post-edited text by comparing the source text and the draft translation generated from MT systems. According to professional translators, despite recent advances in NMT that have greatly improved the translation quality, PE has saved only about 20% to 30% of the working time compared to translating from scratch. For example, Läubli et al. [66] investigated the productivity of post-editing, and they observed that post-editing only improves the speed by 9.26% compared with translating from scratch in German-to-Italian finance domain. This is because translators take time to read the source and MT sentences and look for mistranslations and omissions.

Automatic post-editing (APE) attempts to correct the MT model outputs (MT sentences) for the better translation quality. However, many APE models are based on sequence generation [58, 20, 106, 11, 12, 5], and their decision for correction is harder to interpret due to the black-box nature of the generation models.

In summary, if an APE model provides not only the correction but also editing processes, e.g., finding mistranslations, it would be helpful for human post-editors.

1.3 Research Objective

The objective of this dissertation is to improve the efficiency of the translation model in domain adaptation and the productivity of post-editing by providing the editing processes. In particular, we address the problem of the translation speed of kNN-MT, which is effective for domain adaptation, and the lack of interpretability of APE model due to black box predictions.

In the domain adaptation task, kNN-MT is focused in terms of the translation quality and interoperability, but its translation speed is more than two orders of magnitude slower than standard NMT models. We improve the translation speed of kNN-MT by narrowing down the search space to neighboring examples of the input sentence. In addition, we use the look-up table to calculate the distance between the query and keys efficiently when retrieving. Our subset kNN-MT achieved a speed-up of up to 134.2 times and an improvement in BLEU score of up to 1.6 compared with kNN-MT in the WMT'19 German-to-English general domain translation task, the domain adaptation tasks in German-to-English and English-to-Japanese with open-domain settings, and the Flores101 multilingual translation task.

Regarding post-editing, we improve the interpretability of APE models by using an edit-based model. Our "detector–corrector" breaks the editing process into two steps, "error detection" and "error correction". The detector model tags each MT output token whether it should be corrected and/or reordered while the corrector model generates corrected words for the spans identified as errors by the detector. Experiments on the WMT'20 English-to-German and English-to-Chinese APE tasks showed that our detector-corrector provides the editing process and outperforming black-box sequence-to-sequence APE model and previous edit-based model.

1.4 Structure of the Dissertation

The dissertation is organized as follows.

Chapter 2 shows the preliminary knowledge about machine translation and k-nearest-neighbor search.

Chapter 3 addresses the out-of-domain problem. We propose subset retrieval to speed up kNN-MT, which narrows down the search space of translation examples by retrieving neighboring sentences of the input sentence.

Chapter 4 aims to reduce the workload of human post-editing by using a novel explainable model that presents the editing process.

Chapter 5 summarizes the dissertation and discusses the future directions.

Chapter 2

Preliminaries

2.1 Machine Translation

The goal of machine translation is to convert the text written in the source language X to the text written in the target language Y. This section describes an overview of machine translation approaches.

2.1.1 Approaches

Example-based Machine Translation

Example-based machine translation [84] generates translations based on translation rules acquired from parallel data. Let $\boldsymbol{x} = (x_1, x_2, \ldots, x_{|\boldsymbol{x}|}) \in \mathcal{V}_X^*$ be the source sentence and $\boldsymbol{y} = (y_1, y_2, \ldots, y_{|\boldsymbol{y}|}) \in \mathcal{V}_Y^*$ be the target sentence where $|\cdot|$ is a length of a sentence, and $\mathcal{V}_X^*$ and $\mathcal{V}_Y^*$ are Kleene closures of vocabularies of the source language and target language, respectively. The most basic method, which acquires translation rules based on analogy, extracts the difference between two similar source sentences $\boldsymbol{x}$ and $\boldsymbol{x}'$, and their target sentences $\boldsymbol{y}$ and $\boldsymbol{y}'$ in the parallel data $\mathcal{D} = \{(\boldsymbol{x}^i, \boldsymbol{y}^i)\}_{i=1}^{|\mathcal{D}|}$. Table 2.1 shows an example of acquiring translation rules. When $\boldsymbol{x}$ is "私 は ぶどう が 好き 。", $\boldsymbol{x}'$ is "私 は テニス が 好き 。", $\boldsymbol{y}$ is "I like grapes .", and $\boldsymbol{y}'$ is "I like tennis ." (Table 2.1(a)), then, from the source side difference between $\boldsymbol{x}$ and $\boldsymbol{x}'$, and the target side difference between $\boldsymbol{y}$ and $\boldsymbol{y}'$, we get three translation rules: "私 は — が 好き 。" → "I like — .", "ぶどう" → "grapes", and "テニス" → "tennis" (Table 2.1(b)). During inference, the translation system refers to the acquired translation rules and generates the

Japanese	English
私 は ぶどう が 好き 。	I like grapes .
私 は テニス が 好き 。	I like tennis .

(a) Similar two translation examples in Japanese-to-English.

Japanese	English
私 は — が 好き 。	I like — .
ぶどう	grapes
テニス	tennis

(b) Translation rules acquired from the similar translation examples.

Table 2.1: An example of acquiring translation rules from Japanese-to-English parallel sentences.

target sentence.

Statistical Machine Translation

Statistical machine translation [8] learns statistical information from parallel data. The model generates $\boldsymbol{y}$ according to the conditional probability $P(\boldsymbol{y}|\boldsymbol{x})$, the source-to-target translation model, but since it is difficult to estimate the probability directly, instead the target-to-source translation model $P(\boldsymbol{x}|\boldsymbol{y})$ and the language model $P(\boldsymbol{y})$ of $P(\boldsymbol{y}|\boldsymbol{x}) \propto P(\boldsymbol{x}|\boldsymbol{y})P(\boldsymbol{y})$, decomposed by Bayes theorem, are used to compute the output probability, respectively:

$$\begin{aligned} \boldsymbol{y}^* &= \operatorname*{argmax}_{\boldsymbol{y}} P(\boldsymbol{y}|\boldsymbol{x}) \\ &= \operatorname*{argmax}_{\boldsymbol{y}} P(\boldsymbol{x}|\boldsymbol{y})P(\boldsymbol{y}), \end{aligned} \tag{2.1}$$

where $\boldsymbol{y}^*$ is the generated target sentence. The model parameters θ and ϕ are learned from parallel data:

$$\mathcal{L}(\theta) = \sum_{i=1}^{|\mathcal{D}|} \log p(\boldsymbol{x}^i|\boldsymbol{y}^i;\theta), \tag{2.2}$$

$$\theta^* = \operatorname*{argmax}_{\theta} \mathcal{L}(\theta), \tag{2.3}$$

$$\mathcal{L}(\phi) = \sum_{i=1}^{|\mathcal{D}|} \log p(\boldsymbol{y}^i;\phi), \tag{2.4}$$

$$\phi^* = \operatorname*{argmax}_{\phi} \mathcal{L}(\phi), \tag{2.5}$$

where θ^* and ϕ^* are the trained parameters learned from $\mathcal{D}$.

Neural Machine Translation

Neural machine translation (NMT) directly estimates $P(\boldsymbol{y}|\boldsymbol{x})$ using a neural network. In NMT, encoder-decoder is the most common architecture and widely used [112, 3, 72, 121, 39, 113]. The encoder projects a source sentence $\boldsymbol{x}$ to the feature space, and the decoder generates target tokens $\boldsymbol{y}$ from the hidden vectors. The objective of NMT is to generate a target sentence based on the following probabilities:

$$\boldsymbol{y}^* = \underset{\boldsymbol{y}}{\operatorname{argmax}}\, P(\boldsymbol{y}|\boldsymbol{x}). \tag{2.6}$$

To calculate $P(\boldsymbol{y}|\boldsymbol{x})$, it is decomposed into the product of probabilities based on the chain rule. The t-th target token y_t is generated according to its output probability $P(y_t|\boldsymbol{x}, \boldsymbol{y}_{<t})$ over the target vocabulary, calculated from the source sentence $\boldsymbol{x}$ and generated target tokens $\boldsymbol{y}_{<t}$ as follows:

$$\boldsymbol{y}^* = \underset{y_1,\ldots,y_{|\boldsymbol{y}|}}{\operatorname{argmax}} \prod_{t=1}^{|\boldsymbol{y}|} P(y_t|\boldsymbol{x}, \boldsymbol{y}_{<t}), \tag{2.7}$$

where the output sequence $\boldsymbol{y}^*$ is approximated by search strategies like beam search.

The model parameters θ are trained to maximize the log-likelihood as follows:

$$\mathcal{L}(\theta) = \sum_{i=1}^{|\mathcal{D}|} \sum_{t=1}^{|\boldsymbol{y}^i|} \log p(y_t^i|\boldsymbol{x}^i, \boldsymbol{y}_{<t}^i; \theta), \tag{2.8}$$

$$\theta^* = \underset{\theta}{\operatorname{argmax}}\, \mathcal{L}(\theta), \tag{2.9}$$

where $\mathcal{D}$ is a parallel data and $(\boldsymbol{x}^i, \boldsymbol{y}^i) \in \mathcal{D}$ is an i-th translation pairs in the parallel data, and θ^* is the trained parameters. For neural networks of the encoder and decoder, the long short-term memory (LSTM) based models [112, 3, 72, 121], the convolution neural network (CNN) based model [39], and the Transformer model [113] are used.

2.1.2 Evaluation

When developing the MT model, it is expensive for a human translator to evaluate the quality of the MT-generated translations directly each time the model updates. For this purpose, automatic evaluation metrics are used to evaluate the translation quality of MT systems using reference translations.

Bilingual Evaluation Understudy (BLEU)

One of the most common evaluation metrics is bilingual evaluation understudy (BLEU) [92]. BLEU is computed by the modified n-gram precision and the brevity penalty using hypothesis translation and its reference translations. The modified n-gram precision first counts the maximum number of times a word occurs in any reference translation. Then, it clips the total count of each hypothesis word by its maximum reference count, adds these clipped counts up, and divides it by the total number of hypothesis words. Typically, the modified 1-gram to 4-gram precisions are used by calculating their geometric mean.

Translation Edit Rate (TER)

Translation edit rate (TER) [107] is a metric to evaluate the translation quality based on the edit distance between the translated text and the reference translation. In particular, TER is defined as the minimum number of edits from the translation hypothesis to the reference, as follows:

$$\frac{\text{Number of edits}}{\text{Number of reference words}}. \tag{2.10}$$

The edit operations of TER contain deletion, insertion, substitution, and shifts of word sequences, i.e., word reordering. TER iteratively reorders an input sequence to minimize the edit distance from the target sequence, called shift operation, then calculates the edit distance between the reordered input sequence and the target sequence.

Cross-lingual Optimized Metric for Evaluation of Translation (COMET)

BLEU and TER cannot evaluate the replacement of synonyms because they calculate scores using the surface of words. Rei et al. [98] presented cross-lingual optimized metric for evaluation of translation (COMET), which directly estimates the human judgments using a cross-lingual neural language model. Because COMET uses a neural network model, it is more computationally expensive than other metrics, but it can evaluate semantic similarity between a hypothesis translation and its reference translation and has a high correlation with human evaluation.

2.2 k-Nearest-Neighbor Search

2.2.1 Nearest Neighbor Search

k-nearest-neighbor (kNN) search retrieves top-k vectors from the vector set $\mathcal{K} \subseteq \mathbb{R}^D$ that are close to the given query vector $\boldsymbol{q} \in \mathbb{R}^D$. In this section, we assume $k = 1$ in squared Euclidean space. The most naive approach of kNN search is to compute the distance to all vectors $\boldsymbol{k} \in \mathcal{K}$ from the query $\boldsymbol{q}$.

$$\boldsymbol{k}^* = \underset{\boldsymbol{k}}{\operatorname{argmin}} \|\boldsymbol{q} - \boldsymbol{k}\|_2^2, \tag{2.11}$$

where $\boldsymbol{k}^*$ is the nearest neighbor vector. Note that the time and space complexity is $\mathcal{O}(ND)$ where $N = |\mathcal{K}|$.

2.2.2 Product Quantization

It is hard to load $\mathcal{K}$ into the main memory when N is large, e.g., one billion. For example, let N be one billion and D be 1024, the vector set $\mathcal{K}$ consumes 3.7 TiB. To reduce the space complexity, product quantization (PQ) [53], which approximates the vectors, is used.

PQ splits a D-dimensional vector into M sub-vectors and quantizes for each $\frac{D}{M}$-dimensional sub-vector. Codebooks are learned by k-means clustering of key vectors in each subspace. It is computed iteratively by: (1) assigning the code of a key to its nearest neighbor centroid (2) and updating the centroid of keys assigned to the code. The m-th sub-space's codebook $\mathcal{C}^m$ is formulated as follows:

$$\mathcal{C}^m = \{\boldsymbol{c}_1^m, \ldots, \boldsymbol{c}_L^m\},\ \boldsymbol{c}_l^m \in \mathbb{R}^{\frac{D}{M}}, \tag{2.12}$$

where L is the number of codes for each subspace. Typically, L is set to $2^8 = 256$, and quantized codes are represented as unsigned 8-bit integer (uint8). A vector $\boldsymbol{q} \in \mathbb{R}^D$ is quantized and its code vector $\bar{\boldsymbol{q}}$ is calculated as follows:

$$\bar{\boldsymbol{q}} = [\bar{q}^1, \ldots, \bar{q}^M]^\top \in \{1, \ldots, L\}^M, \tag{2.13}$$

$$\bar{q}^m = \underset{l}{\operatorname{argmin}} \|\boldsymbol{q}^m - \boldsymbol{c}_l^m\|_2^2,\ \boldsymbol{q}^m \in \mathbb{R}^{\frac{D}{M}}. \tag{2.14}$$

Chapter 3

Subset Retrieval Nearest Neighbor Machine Translation

3.1 Introduction

Neural machine translation (NMT) [112, 3, 72, 121, 113] has achieved state-of-the-art performance and become the focus of many studies. Recently, kNN-MT [61] has been proposed, which addresses the problem of degradation of translation quality in out-of-domain data by incorporating example-search into the decoding algorithm. kNN-MT stores translation examples as a set of key–value pairs called "datastore" and retrieves k-nearest-neighbor target tokens in decoding. The method improves the translation quality of NMT models without additional training. However, decoding is seriously time-consuming, i.e., roughly 100 to 1,000 times slower than standard NMT, because neighbor tokens are retrieved from all target tokens of parallel data in each timestep. In particular, in a realistic open-domain setting, kNN-MT may be significantly slower because it needs to retrieve neighbor tokens from a large datastore that covers various domains.

We propose "Subset kNN-MT", which improves the decoding speed of kNN-MT by two methods: (1) retrieving neighbor target tokens from a subset that is the set of neighbor sentences of the input sentence, not from all sentences, and (2) efficient distance computation technique that is suitable for subset neighbor search using a look-up table. When retrieving neighbor sentences for a given input, we can employ arbitrary sentence representations, e.g., pre-trained neural encoders or TF-IDF vectors, to reduce the kNN search space. When retrieving target tokens in each decoding step, the search space in subset kNN-MT varies depending on the input sentence; therefore, the clustering-based search methods used in the original kNN-MT cannot be used. For this purpose, we use asymmetric distance computation (ADC) [53] in subset neighbor search.

Our subset kNN-MT achieved a speed-up of up to 134.2 times and an improvement in BLEU score of up to 1.6 compared with kNN-MT in the WMT'19 German-to-English general domain translation task, the domain adaptation tasks in German-to-English and English-to-Japanese with open-domain settings, and the Flores101 multilingual translation task. Our implementation, including both kNN-MT and subset kNN-MT, is available on GitHub[1].

3.2 kNN-MT

kNN-MT [61] retrieves the k-nearest-neighbor target tokens in each timestep, computes the kNN probability from the distances of retrieved tokens, and interpolates the probability with the model prediction probability. The method consists of two steps: (1) datastore creation, which creates key–value translation memory, and (2) generation, which calculates an output probability according to the nearest neighbors of the cached translation memory.

Datastore Construction kNN-MT stores pairs of D-dimensional vectors and tokens in a datastore, represented as key–value memory $\mathcal{M} \subseteq \mathbb{R}^D \times \mathcal{V}_Y$. The key ($\in \mathbb{R}^D$) is an intermediate representation of the final decoder layer obtained by teacher forcing a parallel sentence pair $(\boldsymbol{x}, \boldsymbol{y})$ to the NMT model, and the value is a ground-truth target token y_t. The datastore is formally defined as a set of tuples as follows:

$$\mathcal{M} = \{(f(\boldsymbol{x}, \boldsymbol{y}_{<t}), y_t) \mid (\boldsymbol{x}, \boldsymbol{y}) \in \mathcal{D}, 1 \leq t \leq |\boldsymbol{y}|\}, \tag{3.1}$$

where $\mathcal{D} = \{(\boldsymbol{x}^i, \boldsymbol{y}^i)\}_{i=1}^{|\mathcal{D}|}$ is parallel data and $f \colon \mathcal{V}_X^* \times \mathcal{V}_Y^* \to \mathbb{R}^D$ is a function that returns the D-dimensional intermediate representation of the final decoder layer from the source sentence and generated target tokens. In our model, as in

[1] https://github.com/naist-nlp/knn-seq

[61], the key is the intermediate representation before it is passed to the final feed-forward network.

Decoding During decoding, kNN-MT generates output probabilities by computing the linear interpolation between the kNN and MT probabilities, $p_{k\mathrm{NN}}$ and p_{MT}, as follows:

$$P(y_t|\boldsymbol{x}, \boldsymbol{y}_{<t}) = \lambda p_{k\mathrm{NN}}(y_t|\boldsymbol{x}, \boldsymbol{y}_{<t}) + (1-\lambda) p_{\mathrm{MT}}(y_t|\boldsymbol{x}, \boldsymbol{y}_{<t}), \tag{3.2}$$

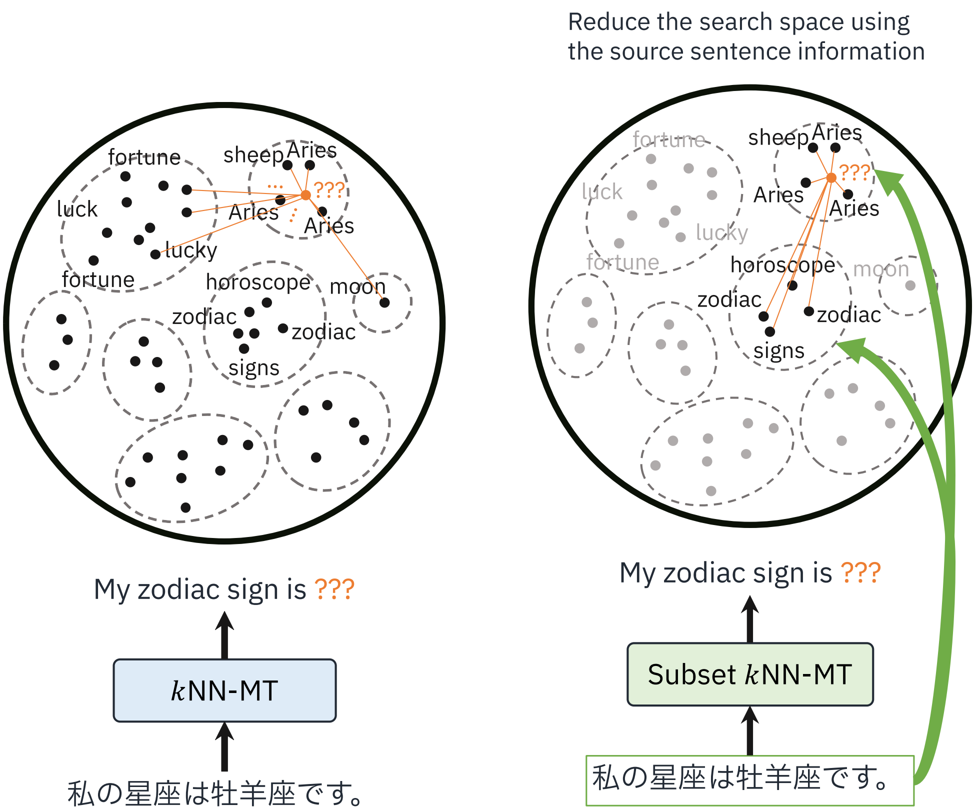
where λ is a hyperparameter for weighting the kNN probability. Let $f(\boldsymbol{x}, \boldsymbol{y}_{<t})$ be the query vector at timestep t. The k-nearest-neighbor tokens of the query are searched from the datastore and the top-k key–value pairs are obtained. The top i-th key and value in the k-nearest-neighbor are $\boldsymbol{k}_i \in \mathbb{R}^D$ and $v_i \in \mathcal{V}_Y$, respectively. Then $p_{k\mathrm{NN}}$ is defined as follows:

$$p_{k\mathrm{NN}}(y_t|\boldsymbol{x}, \boldsymbol{y}_{<t}) = \frac{1}{Z} \sum_{i=1}^{k} \mathbb{1}_{y_t = v_i} \exp\left(\frac{-\|\boldsymbol{k}_i - f(\boldsymbol{x}, \boldsymbol{y}_{<t})\|_2^2}{\tau}\right), \tag{3.3}$$

$$Z = \sum_{i=1}^{k} \exp\left(\frac{-\|\boldsymbol{k}_i - f(\boldsymbol{x}, \boldsymbol{y}_{<t})\|_2^2}{\tau}\right), \tag{3.4}$$

where τ is the temperature for $p_{k\mathrm{NN}}$, and we set $\tau = 100$.

Note that this kNN search is seriously time-consuming[2] [61]. This is because the kNN tokens are searched for each timestep in generating a target sentence. For example, let $|\mathcal{M}|$ be one billion and $|\boldsymbol{y}|$ be 30. If we naively search the kNN tokens, we need to calculate the distance between the query and key $|\mathcal{M}| \times |\boldsymbol{y}| = 30$ billion times, i.e., the time complexity is $\mathcal{O}(|\mathcal{M}||\boldsymbol{y}|)$. In other words, the speed problem of the kNN-MT is due to the large search space $|\mathcal{M}|$ [3].



(a) Overview of kNN-MT. (b) Overview of subset kNN-MT.

Figure 3.1: Overview of kNN-MT (left) and our subset kNN-MT (right). Subset kNN-MT finds the k-nearest-neighbor tokens from the reduced search space related to the input sentence.

3.3 Proposed Model: Subset kNN-MT

Our Subset kNN-MT drastically accelerates vanilla kNN-MT by reducing the kNN search space by using sentence information as shown in Figure 3.1. In particular, subset retrieval (Section 3.3.1) retrieves the neighboring sentences of the given input sentence and reduces the search space from all sentences to only the

[2]In our experiments on the WMT'19 German-to-English, the datastore has 862M tokens, the vocabulary size is 42k, and the batch size was set to 12,000 tokens. While a normal Transformer translates 2,000 sentences in 7.5 seconds, kNN-MT takes 2446.0 seconds.

[3]The original kNN-MT actually uses an approximate nearest neighbor algorithm, but it is still time-consuming.

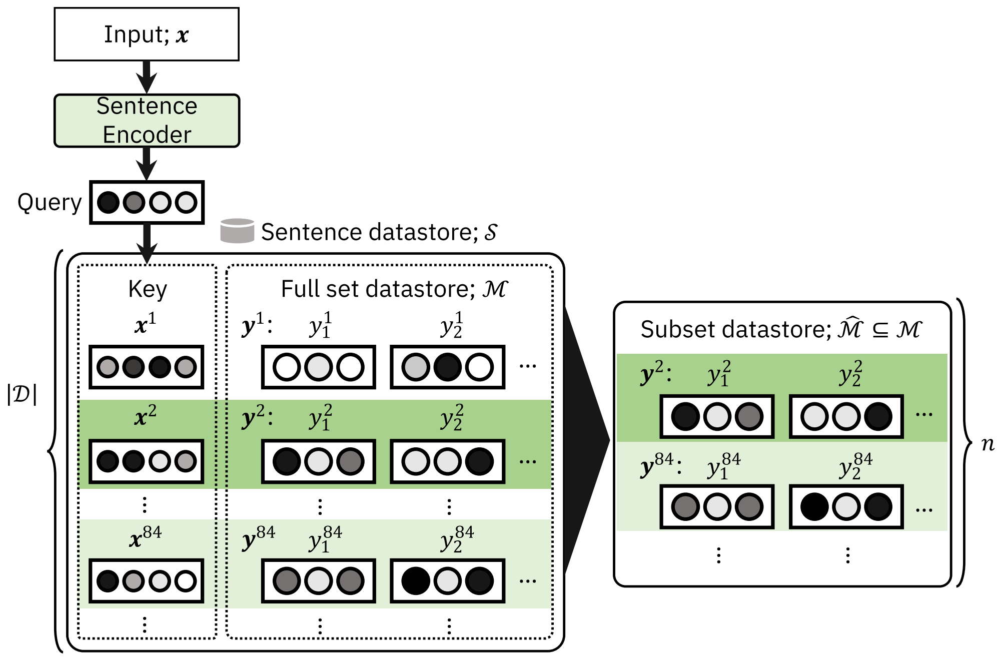


Figure 3.2: Susbet kNN-MT reduces the search space of kNN-MT by retrieving the neighboring sentences of the input sentence. The green boxes in the sentence datastore mean the neighboring examples of the input sentence. The search space is reduced from $|\mathcal{D}|$ sentences to the $n(\ll |\mathcal{D}|)$ neighboring sentences.

neighboring sentences, shown in Figure 3.2. Additionally, we employ the efficient method to compute the distance between a query and keys using a look-up table when retrieving the kNN tokens from the reduced datastore (Section 3.3.2). While the original kNN-MT employs a data structure and an algorithm optimized for the fixed search space, i.e., the full set of the datastore, our subset kNN-MT employs an algorithm that efficiently searches the subset datastore that varies dynamically depending on the input sentence.

3.3.1 Subset Retrieval

Sentence Datastore Construction We construct a sentence datastore that stores pairs comprising a vector representation of a source sentence and key–value pairs of target tokens. Concretely, a sentence datastore $\mathcal{S}$ is defined as follows:

$$\mathcal{S} = \{(h(\boldsymbol{x}^s), \mathcal{M}^{(s)}) \mid (\boldsymbol{x}^s, \boldsymbol{y}^s) \in \mathcal{D}\}, \tag{3.5}$$

$$\mathcal{M}^{(s)} = \{(f(\boldsymbol{x}^s, \boldsymbol{y}^s_{<t}), y^s_t) \mid 1 \leq t \leq |\boldsymbol{y}^s|\} \tag{3.6}$$

where $h\colon \mathcal{V}_X^* \rightarrow \mathbb{R}^{D'}$ represents a sentence encoder, which is a function that returns a D'-dimensional vector representation of a source sentence, and $s \in \{1, \ldots, |\mathcal{D}|\}$ denotes the identifier of s-th sentence pairs in the parallel data. Note that $\mathcal{M}^{(s)} \subset \mathcal{M}$ is the datastore which is created from only the s-th sentence pairs.

Decoding At the beginning of decoding, the model retrieves the n-nearest-neighbor sentences of the given input sentence by calculating the distances between the input sentence vector and the source sentence vectors of the sentence datastore $\mathcal{S}$. Let $\mathcal{N} \subset \{1, \ldots, |\mathcal{D}|\}$ be the set of sentence numbers of the n-nearest-neighbor sentences. The nearest neighbor search space for target tokens in kNN-MT is then drastically reduced by obtaining the datastore as follows:

$$\hat{\mathcal{M}} = \bigcup_{s \in \mathcal{N}} \mathcal{M}^{(s)} = \{(f(\boldsymbol{x}^s, \boldsymbol{y}^s_{<t}), y^s_t) \mid s \in \mathcal{N}, 1 \leq t \leq |\boldsymbol{y}^s|\}, \tag{3.7}$$

where $\hat{\mathcal{M}} \subset \mathcal{M}$ is the reduced datastore for the translation examples coming from the n-nearest-neighbor sentences. During decoding, the model uses the same algorithm as kNN-MT except that $\hat{\mathcal{M}}$ is used as the datastore instead of $\mathcal{M}$. The proposed method reduces the size of the nearest neighbor search space for the target tokens from $|\mathcal{D}|$ to n ($\ll |\mathcal{D}|$) sentences.

Note that our method needs to store sentence representations in addition to the original datastore that stores the token representations. However, the number of sentences is usually one order of magnitude smaller than the number of tokens, i.e., $|\mathcal{D}| \ll |\mathcal{M}|$; thus, the memory and storage usages will not be increased significantly. Additionally, subset kNN-MT requires the neighboring sentence search, but it is not time-consuming because it only searches once before the decoding iterations and the search space of the neighboring sentence search is smaller than that of all target tokens, as mentioned above.

3.3.2 Efficient Distance Computation Using Look-up Table

Subset kNN-MT retrieves the k-nearest-neighbor target tokens by an efficient distance computation method that uses a look-up table. In the original kNN-MT, inverted file index (IVF) is used for retrieving kNN tokens. IVF divides the search space into N_{list} clusters and retrieves tokens from the neighbor clusters. In contrast, in subset kNN-MT, the search space varies dynamically depending on the input sentence. Therefore, clustering-based search methods cannot be used; instead, it is necessary to calculate the distance for each key in the subset. For this purpose, we use asymmetric distance computation (ADC) [53] instead of the usual distance computation between floating-point vectors. In ADC, the number of table look-up is linearly proportional to the number of keys N in the subset. Therefore, it is not suitable for searching in large datastore $\mathcal{M}$, but in a small subset $\hat{\mathcal{M}}$, the search is faster than the direct calculation of the L2 distance.

Product Quantization (PQ) The kNN-MT datastore $\mathcal{M}$ may become too large because it stores high-dimensional intermediate representations of all target tokens of parallel data. For instance, the WMT'19 German-to-English parallel data, which is used in our experiments, contains 862M tokens on the target side. Therefore, if vectors were stored directly, the datastore would occupy 3.2 TiB when a 1024-dimensional vector as a key[4], and this would be hard to load into RAM. To solve this memory problem, product quantization (PQ) [53] is used in both kNN-MT and our subset kNN-MT, which includes both source sentence and target token search.

PQ splits a D-dimensional vector into M sub-vectors and quantizes for each $\frac{D}{M}$-dimensional sub-vector. Codebooks are learned by k-means clustering of key vectors in each subspace. It is computed iteratively by: (1) assigning the code of a key to its nearest neighbor centroid (2) and updating the centroid of keys assigned to the code. The m-th sub-space's codebook $\mathcal{C}^m$ is formulated as follows:

$$\mathcal{C}^m = \{\boldsymbol{c}_1^m, \dots, \boldsymbol{c}_L^m\},\ \boldsymbol{c}_l^m \in \mathbb{R}^{\frac{D}{M}}. \tag{3.8}$$

[4] 3.2 TiB $\simeq$ 862.6M tokens $\times$ 1024 dimension $\times$ 32 bits (float size)/8 bits (byte size)/1024^4.

In this work, each codebook size is set to $L = 256$[5]. A vector $\boldsymbol{q} \in \mathbb{R}^D$ is quantized and its code vector $\bar{\boldsymbol{q}}$ is calculated as follows:

$$\boldsymbol{q} = \left[\boldsymbol{q}^{1\top}, \ldots, \boldsymbol{q}^{M\top}\right]^\top, \ \boldsymbol{q}^m \in \mathbb{R}^{\frac{D}{M}}, \tag{3.9}$$

$$\bar{q}^m = \underset{l}{\operatorname{argmin}} \|\boldsymbol{q}^m - \boldsymbol{c}_l^m\|_2^2, \tag{3.10}$$

$$\bar{\boldsymbol{q}} = [\bar{q}^1, \ldots, \bar{q}^M]^\top \in \{1, \ldots, L\}^M. \tag{3.11}$$

Note that naive PQ may result in poor approximation accuracy because it ignores dimension correlations. To address this problem, vector transformation methods such as optimized PQ (OPQ) [38] and principal component analysis (PCA) are used. Details of the settings we employed are listed in Table 3.1 in the Experiments section.

Asymmetric Distance Computation (ADC) Our method efficiently computes the squared Euclidean distance between a query vector and quantized key vectors using ADC [53] (Figure 3.3 and Algorithm 1). ADC computes the squared Euclidean distance between a query vector $\boldsymbol{q} \in \mathbb{R}^D$ and N key codes $\bar{\mathcal{K}} = \{\bar{\boldsymbol{k}}_i\}_{i=1}^N \subseteq \{1, \ldots, L\}^M$. First, the distance look-up table $\boldsymbol{A}^m \in \mathbb{R}^L$ is computed by calculating the distance between a query $\boldsymbol{q}^m$ and the codes $\boldsymbol{c}_l^m \in \mathcal{C}^m$ in each sub-space m ("distance table" in Figure 3.3 and line 4 in Algorithm 1), as follows:

$$A_l^m = \|\boldsymbol{q}^m - \boldsymbol{c}_l^m\|_2^2. \tag{3.12}$$

Second, the distance between a query and each key $d(\boldsymbol{q}, \bar{\boldsymbol{k}}_i)$ is obtained by looking up the distance table ("looked up disntances" in Figure 3.3 and line 8 in Algorithm 1), as follows:

$$d(\boldsymbol{q}, \bar{\boldsymbol{k}}_i) = \sum_{m=1}^{M} d_m(\boldsymbol{q}^m, \bar{k}_i^m) = \sum_{m=1}^{M} A_{\bar{k}_i^m}^m. \tag{3.13}$$

[5]Codes are represented as unsigned 8bit integers, i.e., an array of `uint8`. We chose the $L = 256$ and $M = 64$ (described in Table 3.1 in the Experiments section) according to the prior work [53]. They reported that $M = 8$ is a reasonable choice when $D = 128$; therefore, the codebook represents $\frac{D}{M} = 16$ dimensional subspace, which is the same as our settings: $M = 64$ and $D = 1024$.

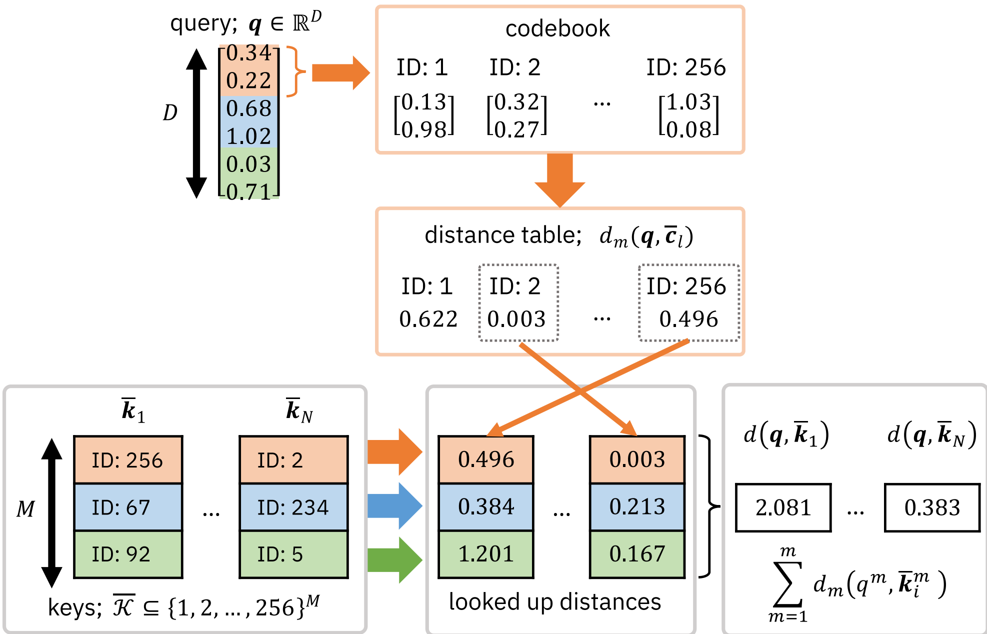


Figure 3.3: Distance computation using asymmetric distance computation (ADC).

A look-up table in each subspace, $\boldsymbol{A}^m \in \mathbb{R}^L$, consists of the distance between a query and codes. The number of codes in each subspace is L and a distance is a scalar; therefore, $\boldsymbol{A}^m$ has L distances. And the table look-up key is the code of a key itself, i.e., if the m-th subspace's code of a key is 5, ADC looks-up A_5^m. By using ADC, the distance is computed only once[6] (Equation 3.12) and does not decode PQ codes into D-dimensional key vectors; therefore, it can compute the distance while keeping the key in the quantization code, and the k-nearest-neighbor tokens are efficiently retrieved from $\hat{\mathcal{M}}$.

[6]The direct distance computation requires N times calculations according to $\|\boldsymbol{q}-\boldsymbol{k}\|^2$, i.e., the time complexity is $\mathcal{O}(ND)$. ADC computes the distance only $L \ll N$ times, i.e., the time complexity for creating the distance table is $\mathcal{O}(L \times \frac{D}{M} \times M) = \mathcal{O}(LD)$, and just looks-up the table MN times in the constant time $\mathcal{O}(1)$. Therefore, the complexity is reduced from $\mathcal{O}(ND)$ to $\mathcal{O}(LD)$.

Algorithm 1 ADC look-up

Require:

query; $\boldsymbol{q} \in \mathbb{R}^D$

quantized keys; $\bar{\mathcal{K}} = \{\bar{\boldsymbol{k}}_i\}_{i=1}^N \subseteq \{1, \ldots, L\}^M$

codebook; $\mathcal{C} = \{\mathcal{C}^1, \ldots, \mathcal{C}^M\}$, where $\mathcal{C}^m = \{\boldsymbol{c}_l^m\}_{l=1}^L \subseteq \mathbb{R}^{\frac{D}{M}}$

Ensure:

distances; $\boldsymbol{d} \in \mathbb{R}^N$

function COMPUTE_DISTANCES($\boldsymbol{q}$, $\bar{\mathcal{K}}$, $\mathcal{C}$)
  for $m = 1, \ldots, M$ **do**
    for $l = 1, \ldots, L$ **do**
      $A_l^m \leftarrow \|\boldsymbol{q}^m - \boldsymbol{c}_l^m\|_2^2$
    end for
  end for
  for $i = 1, \ldots, N$ **do**
    $d_i \leftarrow \sum_{m=1}^M A_{\bar{k}_i^m}^m$
  end for
  return $\boldsymbol{d}$
end function

3.3.3 Sentence Encoder

In our subset kNN-MT, a variety of sentence encoder models can be employed. The more similar sentences extracted from $\mathcal{M}$, the more likely the subset $\hat{\mathcal{M}}$ comprises the target tokens that are useful for translation. Hence, we need sentence encoders that compute vector representations whose distances are close for similar sentences.

In this work, we employ two types of representations: *neural* and *non-neural*. We can employ pre-trained neural sentence encoders. While they require to support the source language, we expect that the retrieved sentences are more similar than other encoders because we can use models that have been trained to minimize the vector distance between similar sentences [101]. An NMT encoder can also be used as a sentence encoder by applying average pooling to its intermediate representations. This does not require any external resources, but it is not trained from the supervision of sentence representations. Alternatively, we can also use non-neural models like TF-IDF. However, it is not clear whether TF-IDF based similarity is suitable for our method. This is because even if sentences with close surface expressions are retrieved, they do not necessarily have similar meanings and may not yield the candidate tokens needed for translation.

3.4 Implementation Details

We use FAIRSEQ [91] to implement kNN-MT and subset kNN-MT model, and FAISS [55] to retrieve the kNN tokens in kNN-MT and for neighbor sentence search in subset kNN-MT. Existing kNN libraries including FAISS and algorithms like IVF that are used in the original kNN-MT are designed for full search but not for subset search [78]; therefore, we implement the subset search and ADC look-up by using PyTorch.

Subset Caching Quantized key codes and value tokens of the subset are read at the beginning of decoding and cached during decoding. Therefore, a billion-scale large array is accessed only once during decoding. Note that the subset depends only on the input sentence, and the cache size does not change with beam sizes.

Distance Look-up in Beam Search Decoding During decoding by beam search, the queries in the beams have different representations because a query vector is computed depending on the generated target tokens. Let B be a beam size and $\mathbf{Q} \in \mathbb{R}^{B \times D}$ be the queries. Note that we regard $\mathbf{Q}_i$ as the column vector by transposing the i-th row of the matrix $\mathbf{Q}$, i.e., $\mathbf{Q}_i \in \mathbb{R}^D$, and $\mathbf{Q}_i^m$ as the m-th subspace of M subspaces in $\mathbf{Q}_i$, i.e., $\mathbf{Q}_i^m \in \mathbb{R}^{\frac{D}{M}}$. The distance table of a subspace in PQ is computed from a query ($\in \mathbb{R}^{\frac{D}{M}}$) and codebook $\mathcal{C}^m$; thus, the table $\mathbf{A}^m \in \mathbb{R}^{L \times B}$ is computed for each beam.

$$\mathbf{A}_l^m = \left[\|\mathbf{Q}_1^m - \boldsymbol{c}_l^m\|_2^2, \|\mathbf{Q}_2^m - \boldsymbol{c}_l^m\|_2^2, \dots, \|\mathbf{Q}_B^m - \boldsymbol{c}_l^m\|_2^2\right]^\top . \tag{3.14}$$

In contrast, the keys in the subset $\{\bar{\boldsymbol{k}}_1, \dots \bar{\boldsymbol{k}}_N\} \subseteq \{1, \dots, L\}^M$ are the same across beams because they are not changed by generated target tokens. Then, the distances between a key and the queries for each beam $\mathbf{d}(\mathbf{Q}, \bar{\boldsymbol{k}}) \in \mathbb{R}^B$ are obtained as follows:

$$\mathbf{d}(\mathbf{Q}, \bar{\boldsymbol{k}}) = \left[d(\mathbf{Q}_1, \bar{\boldsymbol{k}}), \ldots d(\mathbf{Q}_B, \bar{\boldsymbol{k}})\right]^\top, \tag{3.15}$$

$$d(\mathbf{Q}_i, \bar{\boldsymbol{k}}) = \sum_{m=1}^{M} d_m(\mathbf{Q}_i^m, \bar{k}^m) = \sum_{m=1}^{M} \mathbf{A}^m_{\bar{k}^m}. \tag{3.16}$$

From Equation 3.15 and 3.16, the distance table is looked up $M \times B$ times at each timestep during decoding. We parallelize this look-up using `torch.gather()` in PyTorch. However, to perform parallel look-up, the keys must be replicated to each beam leading to multiple copies proportional to the beam size. To avoid increasing the memory usage of key vectors, we designed not to allocate new memory by copying multiple instances for each beam, but only create a new view of the tensor by using `torch.expand()`. The number of keys takes the number of target tokens in the neighboring sentences, e.g. 10,000. Therefore, this technique is helpful in that it saves memory usage even if the beam size is increased.

3.5 Experiments

3.5.1 Setup

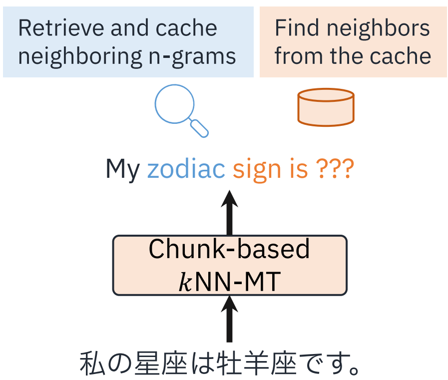
We compared the translation quality and speed of our subset kNN-MT with those of the conventional kNN-MT in open-domain settings that assume a domain of an input sentence is unknown. The translation quality was measured by sacreBLEU[7] [94] and COMET [98]. The decoding speed was evaluated by the number of tokens generated per second (tok/s) on a single NVIDIA V100 GPU. The time measurement includes all processes since the source tokens are given until the output sequence is obtained by beam search; that is, in kNN-MT, it includes the time to search the k-nearest-neighbor tokens for each timestep in addition to the forward computation of the NMT model, and in subset kNN-MT, it includes the time to compute sentence vectors, search the neighboring sentences, look-up the distance table, etc. The speed, tok/s, is calculated by dividing the number of all generated tokens by the time it took to translate the entire test set. We varied the batch size settings: either 12,000 tokens (B_∞), to simulate the document

[7]Signature: `|nrefs:1|case:mixed|eff:no|tok:13a|smooth:exp|version:2.3.1`

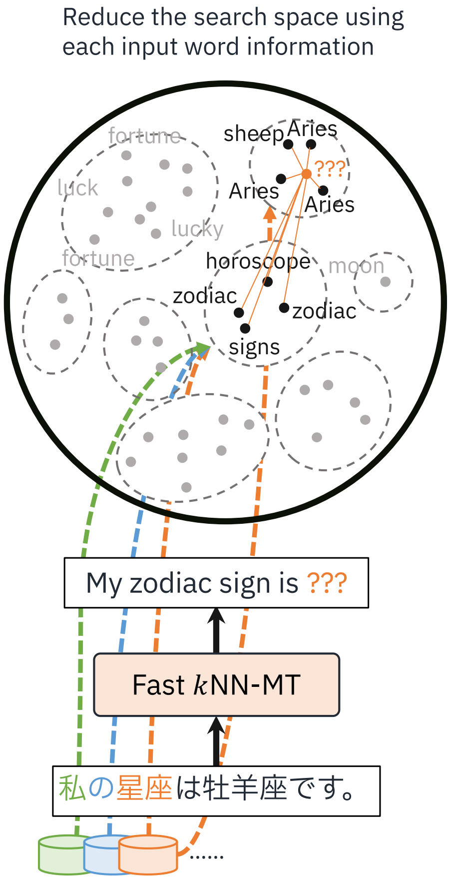
translation scenario, or a single sentence (B_1), to simulate the online translation scenario. The beam size was set to 5, and the length penalty was set to 1.0. In the result tables, the best score is emphasized with bold font, and the second best score is underlined.

k-Nearest-Neighbor Search In kNN-MT, we set the number of nearest neighbor tokens to $k = 16$. We use approximate distance computed from quantized keys instead of full-precision keys in Equation 3.3 and 3.4 following the original kNN-MT [61] implementation. The kNN-MT datastore and our sentence datastore used IVF and optimized PQ (OPQ) [38]. OPQ rotates vectors to minimize the quantization error of PQ. The subset kNN-MT datastore is not applied clustering since we need to extract subset tokens. In this datastore, the 1024-dimensional vector representation, i.e., $D = 1024$, was reduced in dimensionality to 256-dimensions by principal component analysis (PCA), and these vectors were then quantized by PQ. At search time, a query vector is pre-transformed to 256-dimensions by multiplying the PCA matrix, and then the kNN target tokens are searched by ADC. The subset of a datastore can be loaded into GPU memory since it is significantly smaller than the original kNN-MT datastore, so we retrieved k-nearest-neighbor tokens from a subset on a GPU.

Sentence Encoder We compared 4 different sentence encoders: LaBSE, AvgEnc, TF-IDF, and BM25. LaBSE [35] is a pre-trained sentence encoder, fine-tuned from multilingual BERT. AvgEnc is an average pooled encoder hidden vector of the Transformer NMT model, which is also used for translation. TF-IDF [56] and BM25 [57] compute vectors weighted the important words in a sentence. We used the raw count of tokens as the term frequency and applied add-one smoothing to calculate the inverse document frequency, where a sentence was regarded as a document. We counted the number of words segmented by the `scikit-learn` tokenizer [93]. We set $k_1 = 2.0, b = 0.75$ in BM25 [57]. Both TF-IDF and BM25 vectors were normalized by their L2-norm and their dimensionality was reduced to 256-dimensions by singular value decomposition (SVD). In particular, we used truncated SVD also known as latent semantic analysis for the dimension reduction.



(a) Overview of chunk-based kNN-MT [76].



(b) Overview of fast kNN-MT [81].

Figure 3.4: Overview of chunk-based kNN-MT and fast kNN-MT.

3.5.2 In-Domain Translation

We evaluated the translation quality and speed of subset kNN-MT in the WMT'19 De–En translation task (newstest2019; 2,000 sentences) and compared them with the original kNN-MT [61] and other prior work [81, 76]. Chunk-based kNN-MT [76] (Figure 3.4(a)) reduces the number of retrieval times by caching the n-grams of neighboring tokens. Fast kNN-MT [81] (Figure 3.4(b)) retrieves the source-side neighbor tokens by querying each input token and reduces the search space by using the retrieved source-side tokens and their source-to-target word alignment. We used a trained Transformer big implemented in FAIRSEQ [91] as

	kNN-MT	Subset kNN-MT	
	DS; $\mathcal{M}$	Sentence DS; $\mathcal{S}$	DS; $\hat{\mathcal{M}}$
Search method	IVF	IVF	ADC
Vector transform	OPQ [38]	OPQ [38]	PCA: $1024 \rightarrow 256$ dim
# of PQ sub-vectors; M	64	64	64
# of centroids; N_{list}	131,072	32,768	—
# of probed clusters by IVF	64 clusters	64 clusters	—
Size of search target	$\sum_{\boldsymbol{y} \in \mathcal{D}} \lvert\boldsymbol{y}\rvert$	$\lvert\mathcal{D}\rvert$	$\sum_{s \in \mathcal{N}} \lvert\boldsymbol{y}^s\rvert$

Table 3.1: Details of kNN indexes. "DS" indicates "Datastore".

the base MT model. We constructed the datastore from the parallel data of the WMT'19 De–En news translation task. We removed all empty lines and sentences of the parallel data longer than 250 tokens. We also removed all sentences in which the sentence length in one language was more than 1.5 times longer than that in the other language, i.e., the ratio of tokens between the source and target was > 1.5. The datastore contained 862.6M target tokens obtained from 29.5M sentence pairs. The subset size was set to $n = 512$. For fast kNN-MT, we constructed additional source side datastores for each source token, and used `fast_align` [31] to obtain the source-to-target word alignment, following Meng et al. [81]. Then, we retrieved the 512 nearest neighbor source tokens from the source side datastores for each input token in the decoding time of fast kNN-MT. Note that the total size of the source side datastores is close to the kNN-MT datastore; thus, it consumes twice as much storage and memory compared to the original kNN-MT. In chunk-based kNN-MT, the chunk size was set to 16, the hyperparameters that determine the interval of retrieval i_{max} and i_{min} were set to 2 and 16, respectively, following Martins et al. [76]. The details of the kNN indexes are shown in Table 3.1.

Table 3.2 shows our experimental results. The table shows that, although kNN-MT improves 0.9 BLEU point from the base MT without additional training, the decoding speed is 326.1 times and 51.7 times slower with the B_∞ and B_1 settings, respectively. In contrast, our subset kNN-MT (h: LaBSE) is 111.8 times (with

			↑tok/s	
Model	↑BLEU	↑COMET	B_∞	B_1
Base MT	39.2	84.56	6375.2	129.14
kNN-MT	<u>40.1</u>	**84.73**	19.6	2.5
Chunk-based kNN-MT	39.5	84.33	74.6	22.3
Fast kNN-MT	**40.3**	<u>84.70</u>	286.9	27.1
Ours: Subset kNN-MT				
h: LaBSE	<u>40.1</u>	84.66	2191.4	118.4
h: AvgEnc	39.9	84.68	1816.8	97.3
h: TF-IDF	40.0	84.63	2199.1	113.0
h: BM25	40.0	84.60	1903.9	108.4

Table 3.2: Results of translation quality and decoding speed in the WMT'19 De–En translation task. "h:" shows the type of sentence encoder used. The best score is emphasized with bold font, and the second best score is underlined.

B_∞) and 47.4 times (with B_1) faster than kNN-MT with no degradation in the BLEU score. Subset kNN-MT (h: AvgEnc) achieved speed-ups of 92.7 times (with B_∞) and 38.9 times (with B_1) with a slight quality degradation (-0.2 BLEU and -0.05 COMET), despite using no external models. We also evaluated our subset kNN-MT when using non-neural sentence encoders (h: TF-IDF, BM25). The results show that both TF-IDF and BM25 can generate translations with almost the same BLEU score and speed as neural sentence encoders.

In the table, neural encoders, i.e., LaBSE and AvgEnc, and non-neural encoders, i.e., TF-IDF and BM25, have similar calculations, respectively, but their speeds are different. One of the reasons is a difference in the number of retrieved tokens. In total, LaBSE and AvgEnc retrieved 27,910,815 and 34,234,900 tokens, respectively; thus, the ratio of the number of tokens is 1.227 and is close to the speed ratio, $2191.4/1816.8 = 1.206$. Similarly, the number of retrieved tokens in TF-IDF and BM25 is 20,423,819 and 22,576,161, respectively, and its ratio 1.105 is close to $2199.1/1903.9 = 1.155$. Note that the difference in speeds between neural encoders and non-neural encoders is caused by operations, computing devices, and implementations.

Compared with other models, chunk-based kNN-MT and fast kNN-MT generated translations 4 and 15 times faster than the original kNN-MT, respectively. Chunk-based kNN-MT [76] caches the n-grams of neighboring tokens and reduces the time complexity from $\mathcal{O}(D|\mathcal{M}||\boldsymbol{y}|)$ to $D|\mathcal{M}|R$ where $R(<|\boldsymbol{y}|)$ is the number of retrieval in the generation. However, the computational bottleneck is usually the size of datastore $|\mathcal{M}|$, not the output length $|\boldsymbol{y}|$, it only improved the speed by 4 times. Fast kNN-MT [81] pre-constructed $|\mathcal{V}_X|$ datastores for each source token type. During decoding, it first retrieves n-nearest-neighbor source tokens for each input token and maps them into the target-side tokens by using word alignment, then it finds the k-nearest-neighbor target tokens from the reduced search space. It addresses the issue of datastore size and achieves faster decoding speed than chunk-based kNN-MT. However, the source-side token-level retrieval is computationally expensive compared with the sentence retrieval used in our model. Additionally, the search space is n' key vectors, where n' is the size of the kNN search space, but the distances between a query and n' key vectors are calculated directly, whereas subset kNN-MT employed ADC; thus, our subset kNN-MT is much faster than fast kNN-MT. Note that chunk-based kNN-MT does not use any additional resources and fast kNN-MT needs to create additional source side datastores that consume large memory and storage, and requires a source-to-target word alignment tool. Our subset kNN-MT uses a sentence encoder and creates the sentence datastore which has $|\mathcal{D}| \ll |\mathcal{M}|$ sentence representations in addition to the kNN-MT datastore.

In summary, this experiment showed that our subset kNN-MT is two orders of magnitude faster than kNN-MT and has the same translation quality.

3.5.3 Domain Adaptation

German-to-English We evaluated subset kNN-MT on out-of-domain translation in the IT, Koran, Law, Medical, and Subtitles domains [65, 1] with open-domain settings. The datastore was constructed from parallel data by merging all target domains and the general domain (WMT'19 De–En) assuming that the domain of the input sentences is unknown. The datastore contained 895.9M tokens obtained from 30.8M sentence pairs shown in Table 3.3(a). The NMT model is

Domain	#sentences	#tokens
General	29,540,337	862,648,422
IT	184,872	3,154,174
Koran	15,300	455,398
Law	450,870	18,430,516
Medical	209,828	5,741,839
Subtitles	442,653	5,461,071
Total	30,843,860	895,891,42

(a) De–En

Domain	#sentences	#tokens
General	21,911,738	685,820,792
ASPEC	2,000,000	68,305,379
KFTT	440,288	15,185,034
Total	24,352,026	769,311,205

(b) En–Ja

Table 3.3: Datastore statistics in the domain adaptation task.

the same as that used in Section 3.5.2 trained from WMT'19 De–En. The subset size was set to $n = 256$, and the batch size was set to 12,000 tokens, i.e., B_∞.

Table 3.4 shows the results. Compared with base MT, kNN-MT improves the translation quality in all domains but the decoding speed is much slower. In contrast, our subset kNN-MT generates translations faster than kNN-MT. However, in the domain adaptation task, there are differences in translation quality between those using neural sentence encoders and those using non-neural sentence encoders. The table shows that the use of non-neural sentence encoders (TF-IDF and BM25) causes drop in translation quality, whereas the use of neural sentence encoders (LaBSE and AvgEnc) do not. In addition, compared with kNN-MT, our subset kNN-MT with neural encoders achieves an improvement of up to 1.6 BLEU points on some datasets.

Then, we compared what examples are retrieved by LaBSE and TF-IDF, respectively. As mentioned in Section 3.3.3, TF-IDF may not retrieve semantically similar sentences by being susceptible to surface expressions. Table 3.5 shows that the top-3 neighboring sentences retrieved by LaBSE and TF-IDF, respectively. In the case of the table, TF-IDF retrieved sentences that are not related to the input sentence. For example, in TF-IDF-2, "dieser", "ist", "die", and "ca (CA)" match the input sentence; however, the meaning of the sentence is quite different. On the other hand, LaBSE-3 contains "Plasmaproteine" which semantically matches "Protein" and "Plasma" in the input sentence. From the table, we ob-

	IT			Koran			Law		
Model	BL	CM	tok/s	BL	CM	tok/s	BL	CM	tok/s
Base MT	38.7	83.1	4433.2	17.1	72.5	5295.0	46.1	85.8	4294.0
kNN-MT	<u>41.0</u>	<u>83.9</u>	22.3	19.5	<u>73.3</u>	19.3	52.6	**86.8**	18.6
Subset kNN-MT									
h: LaBSE	**41.9**	**84.2**	2362.2	**20.1**	**73.4**	2551.3	**53.6**	**86.8**	2258.0
h: AvgEnc	**41.9**	**84.2**	2197.8	<u>19.9</u>	**73.4**	2318.4	<u>53.2</u>	**86.8**	1878.8
h: TF-IDF	40.0	81.7	2289.0	19.3	72.7	2489.5	51.4	<u>86.0</u>	2264.3
h: BM25	40.0	81.2	1582.4	19.1	72.6	2089.5	50.8	85.8	1946.3
	Medical			**Subtitles**			**Avg.**		
Model	BL	CM	tok/s	BL	CM	tok/s	BL	CM	tok/s
Base MT	42.1	83.3	4392.1	29.4	<u>79.9</u>	6310.5	34.7	80.9	4945.0
kNN-MT	48.2	<u>84.6</u>	19.8	29.6	**80.0**	30.3	38.2	<u>81.7</u>	22.1
Subset kNN-MT									
h: LaBSE	**49.8**	<u>84.6</u>	2328.3	<u>29.9</u>	79.8	3058.4	**39.1**	**81.8**	2511.6
h: AvgEnc	<u>49.2</u>	**84.8**	2059.9	**30.0**	79.8	3113.0	<u>38.8</u>	**81.8**	2313.6
h: TF-IDF	47.5	83.4	2326.6	29.3	79.5	2574.4	37.5	80.7	2388.8
h: BM25	47.4	83.2	1835.6	29.4	79.4	1567.7	37.3	73.3	1804.3

Table 3.4: Results of out-of-domain translation with open-domain settings. "Avg." denotes the average scores. "BL" and "CM" denote BLEU and COMET scores, respectively.

served differences in retrieved subsets between non-neural and neural encoders. Note that this result could be caused by the sentence-level translation models because a single sentence makes it harder for non-neural encoders to obtain the sufficient statistics, e.g., term frequency and inversed document frequency.

In summary, these results show that neural sentence encoders are effective in retrieving domain-specific nearest neighbor sentences from a large datastore.

English-to-Japanese We also evaluated our model on English-to-Japanese translation. We used a pre-trained Transformer big model trained from JParaCrawl

Input	Dieser Anteil ist ca. um das 3fache höher als die nicht an Protein gebundene (freie) Efavirenz-Fraktion in Plasma.
LaBSE-1	Der Trypsininhibitorgehalt lag in der Ration der Versuchsgruppe mit 4,38 TIU / mg fast um das 5-fache höher als für die Kontrollgruppe.
LaBSE-2	Die Dosierung der Hyaluronsäure im vorliegenden Präparat beträgt das 2,5-fache des nichtliposomalen Hyaluronsäure-Konzentrats.
LaBSE-3	Verteilung Die Bindung von Telbivudin an menschliche Plasmaproteine ist in vitro gering (3,3%).
TF-IDF-1	Die Frolikha ist an dieser Stelle ca. 65 m breit und mehrere Meter tief.
TF-IDF-2	Anbieter dieser Dienste ist die Google Inc., 1600 Amphitheatre Parkway, Mountain View, CA 94043, USA.
TF-IDF-3	Širbegovic Enden Montage Stahlbetonkonstruktion, die Business-Lager Plamingo in Gracanica Fläche von ca. 6000 m2. Nutzlast Dielenböden ist 3000 kg / m2, die das Gebäude extrem anspruchsvollen macht. ”In dieser Anlage...

Table 3.5: Top-3 retrieved examples of LaBSE and TF-IDF in a case of the medical domain. "LaBSE-n" and "TF-IDF-n" denote the top-n neighboring sentences retrieved by LaBSE and TF-IDF, respectively.

v3 [83] and evaluated its translation quality on Asian Scientific Paper Excerpt Corpus (ASPEC) [85] and Kyoto Free Translation Task (KFTT; created from Wikipedia's Kyoto articles) [87]. The datastore was constructed from parallel data by merging ASPEC, KFTT, and the general domain (JParaCrawl v3), shown in Table 3.3(b). Note that ASPEC contains 3M sentence pairs, but we used only the first 2M pairs for the datastore to remove noisy data, following Neubig [88]. The datastore contained 735.9M tokens obtained from 24.4M sentence pairs. The subset size was set to $n = 512$, and the batch size was set to 12,000 tokens.

Table 3.6 shows the results. These show that kNN-MT improves out-of-domain translation quality compared with base MT on other language pairs other than German-to-English. On English-to-Japanese, subset kNN-MT improves the decoding speed, but subset kNN-MT with TF-IDF and BM25 degrades the translation quality compared with kNN-MT. However, subset kNN-MT still achieves

	ASPEC			KFTT		
Model	BLEU	COMET	tok/s	BLEU	COMET	tok/s
Base MT	26.7	88.55	5541.6	20.3	83.52	3714.4
kNN-MT	**32.8**	**89.13**	23.5	**27.8**	**85.32**	28.0
Subset kNN-MT						
h: LaBSE	<u>32.5</u>	<u>88.77</u>	2031.8	25.8	84.11	1436.6
h: AvgEnc	32.4	88.75	1775.6	<u>26.4</u>	<u>84.45</u>	1471.3
h: TF-IDF	29.5	88.24	1763.9	22.3	82.37	1559.3
h: BM25	29.4	88.04	1810.7	21.8	82.21	1533.8

Table 3.6: Results of out-of-domain translation in English-to-Japanese. The speed is measured with the B_∞ setting.

higher BLEU scores than base MT without any additional training steps, and it is two orders of magnitude faster than kNN-MT.

In summary, subset kNN-MT can achieve better translation quality than base MT in exchange for a slowdown to roughly 40% of the base MT in open-domain settings, while the original kNN-MT slows down the decoding speed to less than 1% of the base MT.

3.5.4 Multilingual Translation

We also evaluated multilingual translation quality across 11 translation directions using the Flores-101 dataset [40], which is created from English Wikipedia. We used the Flores101-M2M100[8] model with 615M parameters, which is extended from M2M [34] to support languages that are included in Flores-101 by training from OPUS data. The datastore of each language pair was constructed from CCMatrix [103] extracted from Common Crawl. Note that each datastore is created from parallel data of the language pair to be translated. We employed LaBSE and AvgEnc for the sentence encoder in this experiment. We tuned the subset size n to maximize BLEU among $\{256, 512, 1024, 2048\}$ in the validation sets of En–Ja and Ja–En translations, and set to $n = 2048$. The batch size was

[8] https://dl.fbaipublicfiles.com/flores101/pretrained_models/flores101_mm100_615M.tar.gz

set to 12,000 tokens, i.e., B_∞. We used the `flores101` tokenizer implemented in sacreBLEU[9] to calculate the BLEU score.

[9] Signature: `nrefs:1|case:mixed|eff:no|tok:flores101|smooth:exp|version:2.3.1`

	Zh–En (1.5B)			Ja–En (610.4M)			Fi–En (640.7M)		
Model	BL	CM	tok/s	BL	CM	tok/s	BL	CM	tok/s
Base MT	20.9	81.7	2030.0	19.5	82.1	2127.4	27.1	84.8	1976.3
*k*NN-MT	**25.9**	**84.3**	11.9	**24.6**	**84.3**	33.3	**31.3**	**87.0**	32.0
Subset kNN-MT									
h: LaBSE	25.0	83.5	869.1	22.4	83.4	916.2	29.5	86.1	880.8
h: AvgEnc	24.3	83.4	629.0	22.5	83.6	713.1	29.6	86.2	672.9

	Lt–En (440.4M)			En–Zh (1.5B)			En–Ja (714.3M)		
Model	BL	CM	tok/s	BL	CM	tok/s	BL	CM	tok/s
Base MT	27.0	81.8	2145.2	19.4	78.4	1892.1	22.8	83.8	2177.2
*k*NN-MT	**31.0**	**83.7**	44.2	**25.1**	**82.3**	14.1	**27.6**	**86.2**	31.0
Subset kNN-MT									
h: LaBSE	29.1	83.0	904.3	22.9	81.1	837.3	26.1	85.5	912.4
h: AvgEnc	29.5	83.0	676.1	22.7	80.9	597.8	25.8	85.4	627.0

	En–Fi (724.7M)			En–Lt (534.5M)			De–Ja (204.1M)		
Model	BL	CM	tok/s	BL	CM	tok/s	BL	CM	tok/s
Base MT	24.2	84.5	2167.4	27.6	82.8	2032.0	21.1	82.9	2093.2
*k*NN-MT	**29.0**	**87.2**	35.9	**32.1**	**85.4**	44.4	**24.0**	**84.2**	67.8
Subset kNN-MT									
h: LaBSE	27.0	86.4	899.4	30.7	84.7	840.8	23.2	83.4	866.2
h: AvgEnc	26.8	85.9	639.9	30.6	84.5	603.4	22.6	83.6	702.1

	Ru–Ja (149.5M)			Uk–Ja (28.3M)			Avg.		
Model	BL	CM	tok/s	BL	CM	tok/s	BL	CM	tok/s
Base MT	20.3	82.4	2166.4	20.2	81.0	1825.9	22.7	82.4	2057.6
*k*NN-MT	**23.3**	**83.5**	91.6	**22.1**	**81.7**	108.9	**26.9**	**84.5**	46.8
Subset kNN-MT									
h: LaBSE	22.0	83.1	825.2	20.9	80.7	909.3	25.3	83.7	878.3
h: AvgEnc	21.9	82.7	638.3	20.9	80.8	615.2	25.2	83.6	646.8

Table 3.7: Results of multilingual translation. The speed is evaluated with B_∞.

Input	Eine gemeinsame Anwendung von Nifedipin und Rifampicin ist daher kontraindiziert.
Reference	*Co-administration* of nifedipine with rifampicin is therefore contraindicated.
Base MT	A joint use of nifedipine and rifampicin is therefore contraindicated.
kNN-MT	A joint use of nifedipine and rifampicin is therefore contraindicated.
Subset kNN-MT	*Co-administration* of nifedipine and rifampicin is therefore contraindicated.

Table 3.8: Translation examples in the medical domain.

Table 3.7 shows the results of the multilingual translation. In the table, "BL" and "CM" denote BLEU and COMET scores, respectively, and "Avg." denotes the average of the scores. The number next to a language name pair indicates the size of the datastore, i.e., the number of target tokens in the parallel data. The results show that both kNN-MT and subset kNN-MT improve translation quality in multilingual translation. In Avg., subset kNN-MT degrades the translation quality compared with kNN-MT, but 19 times faster measured by tokens per second. Comparing the decoding speed for each language pair with kNN-MT, subset kNN-MT is 16.8 times faster in Uk–Ja with the smallest datastore and 134.2 times faster in the En–Zh with the largest datastore. In summary, this experiment shows that subset kNN-MT is more effective when the datastore is larger because the larger datastore will be reduced more examples from the search space by our subset retrieval.

3.6 Discussion

3.6.1 Case Study: Effects of Subset Search

Effective Cases of Subset kNN-MT Translation examples in the medical domain are shown in Table 3.8 and the search results of the top-3 nearest neighbor sentences are shown in Table 3.9. In the table, the subset kNN-MT results are obtained using a LaBSE encoder. Table 3.8 shows that subset kNN-MT correctly generates the medical term "Co-administration". The results of the

S-1	Die gemeinsame Anwendung von Ciprofloxacin und Tizanidin ist kontraindiziert.
S-2	Rifampicin und Nilotinib sollten nicht gleichzeitig angewendet werden.
S-3	Die gleichzeitige Anwendung von Ribavirin und Didanosin wird nicht empfohlen.
T-1	*Co-administration* of ciprofloxacin and tizanidine is contra-indicated.
T-2	Rifampicin and nilotinib should not be used concomitantly.
T-3	*Co-administration* of ribavirin and didanosine is not recommended.

Table 3.9: Top-3 neighbor sentences of our subset kNN-MT in Table 3.8. "S-" and "T-" denote the top-n neighbor source sentences and their translations, respectively.

timestep t	Base MT	kNN-MT	Subset kNN-MT
1	A: 0.80	A: 1.26	Co: 1.49
2	joint: 1.18	joint: 1.12	- (hyphen): 0.05
3	use: 0.83	use: 0.42	administration: 0.59
Avg	0.94	0.93	0.71

Table 3.10: Negative log-likelihood (NLL) of the first three tokens and their average in the case of Table 3.8. Note that a smaller NLL means a larger probability.

nearest neighbor sentence search (Table 3.9) show that "Co-administration" is included in the subset. In detail, there are 30 cases of "Co-administration" and no case of "A joint use" in the whole subset consisting of $n = 256$ neighbor sentences. Base MT and kNN-MT have the subwords of "Co-administration" in the candidates; however, the subwords of "A joint use" have higher scores. Table 3.10 shows the negative log-likelihood (NLL) of the first three tokens and their average for each model. The second token of subset kNN-MT, "-" (hyphen), has a significantly lower NLL than the other tokens. The number of "joint" and "-" in the subset were 0 and 101, respectively, and the k-nearest-neighbor tokens were all "-" in subset kNN-MT. Therefore, the NLL was low because $p_{k\mathrm{NN}}(\text{“-”}) = 1.0$, so the joint probability of a beam that generates the sequence "Co-administration" is higher than "A joint use".

Input	一方、動物性食物 (アリ、シロアリ、卵) は消化しやすいうえに、必須アミノ酸をすべて含む良質なタンパク質源です。
Reference	In contrast, animal foods (ants, *termites*, eggs) not only are easily digestible, but they provide high-quantity proteins that contain all the essential amino acids.
Base MT	On the other hand, animal food (algae, syrup, eggs) is a good source of protein, which contains all essential amino acids, to be easily digested.
kNN-MT	On the other hand, animal foods (ants, *termites*, eggs) are a good source of protein that contains all the essential amino acids.
Subset kNN-MT	On the other hand, animal food (soyks, cereals, eggs) is a good source of protein that contains all of the essential amino acids to be easily digested.

Table 3.11: Japanese-to-English translation examples in the Flores-101 multilingual translation task.

In summary, the proposed method can retrieve more appropriate words by searching a subset that consists only of neighboring cases when the translation examples of the target domain are contained in the datastore.

Ineffective Cases of Subset kNN-MT Japanese-to-English translation examples in the Flores-101 multilingual translation task are shown in Table 3.11 and the search results of the top-3 nearest neighbor sentences are shown in Table 3.12. In the table, the subset kNN-MT results are obtained using a LaBSE encoder. Table 3.11 shows that subset kNN-MT incorrectly generates the animal names, "アリ" → "ants" and "シロアリ" → "termites". The results of the nearest neighbor sentence search (Table 3.12) show that both words were not included in the subset. In detail, there are no cases of "ants" and "termites" in the whole subset consisting of $n = 2048$ neighbor sentences. Table 3.13 shows translation examples containing "termites" in the datastore. Compared to the input sentences in Table 3.11, the topics of the sentences containing "termites" were not

S-1	植物性タンパク質 (35%): すべての必須アミノ酸が含まれているヘンプミルクは、肉、牛乳、卵などの動物性タンパク質源とほぼ同じ割合のタンパク質が摂取できるといわれています。
S-2	しかし、米と組み合わせると、これは体に必要なすべてのアミノ酸を含む完全なタンパク質です。
S-3	しかし、あなたが必要とするヨウ素を得るためのさらに良い方法は、この栄養素の主要な天然の食物源である海藻や海産物などのヨウ素に富んだ食品です。
T-1	High percentage of vegetable proteins (35%): It contains all of the essential amino acids and in similar percentages to that of animal proteins sources like meat, milk, or eggs.
T-2	However, combined with rice, this is a complete protein with all the amino acids necessary to the body.
T-3	But an even better way to get the iodine you need is from iodine-rich foods like sea veggies and seafood, the major natural dietary sources of this nutrient.

Table 3.12: Top-3 neighbor sentences of our subset kNN-MT in Table 3.11. "S-" and "T-" denote the top-n neighbor source sentences and their translations, respectively.

matched. In contrast, since kNN-MT searches on a token-by-token basis, it is also possible to retrieve target tokens from translation examples that have different topics. In summary, subset kNN-MT degrades the translation quality compared to kNN-MT when the neighboring sentences contain no correct word.

3.6.2 Diversity of Subset Sentences

We hypothesize that the noise introduced by sentence encoders causes the difference in accuracy. For example, if the sentence search is not accurate enough, it cannot retrieve translation examples related to the input sentence. In addition, we can expect consistency of translations by retrieving based on not only semantic similarity of sentences but also style and other aspects. From the results of

Japanese	English
実際に、シロアリの存在は、気候変動に対してこれらの生態系を守っている。	Indeed, the presence of termites buffers these ecosystems against climate change.
しかし、再びアリやシロアリの進化に目を移すと、もう一つ決定的なステップがあるのです。	But looking at the evolution of ants and termites again, there is another crucial step.
図 2: シロアリをどのように駆除するか: 上の写真: 以前。	Fig. 2: How to get rid of termites: Top photo: Before.

Table 3.13: Translation examples containing "termites" in the Japanese-to-English datastore constructed from CCMatrix.

		unique ratio %	
Model h	BLEU	source	target
LaBSE	49.8	19.6	18.5
AvgEnc	49.2	20.4	19.2
TF-IDF	47.5	33.3	32.3
BM25	47.4	34.2	32.9

Table 3.14: BLEU score and unique token ratio in the subset obtained by each sentence encoder in the medical domain.

Section 3.6.1 and Table 3.10, one characteristic subword that frequently occurs in the kNN changed the order of the beams, which contributed to the improvement of the translation quality of the subset kNN-MT. Thus, if the subset includes only the vocabulary that is more relevant to the translation, translation accuracy may be improved.

This section investigates whether a better sentence encoder would reduce the noise injected into the subset. We investigated the relationship between vocabulary diversity in the subset and translation quality in the medical domain. Because an output sentence is affected by the subset, we measured the unique token

n-selection	BLEU	unique ratio % source	unique ratio % target
Top	49.8	19.6	18.5
Random of $2n$	47.7	21.7	20.3
Bottom of $2n$	44.9	22.7	21.1

Table 3.15: BLEU score and unique token ratio in the subset obtained by different n-selection methods in the medical domain.

ratio of both source and target languages in the subset as the diversity as follows:

$$\frac{\text{number of unique tokens}}{\text{number of subset tokens}}. \tag{3.17}$$

Table 3.14 shows the BLEU score and unique token ratio for the various sentence encoders, in which "source" and "target" indicate the diversity of the neighbor sentences on the source-side and target-side, respectively. The results show that the more diverse the source-side is, the more diverse the target-side is. It also shows that the less diversity in the vocabulary of both the source and target languages in the subset, the higher BLEU score.

We also investigated the relationship between sentence encoder representation and BLEU scores. In particular, we evaluated translation quality when noise was injected into the subset by retrieving n sentences from outside the nearest neighbor. To clarify our hypothesis, we experimented with two artificially created subsets. One is "Bottom of $2n$", the n furthest sentences of the $2n$ neighbor sentences, which simulates the n nearest neighbor sentences cannot be retrieved. The other is "Random of $2n$", n sentences randomly selected from the $2n$ neighbor sentences, i.e., it can be regarded as a subset which mixed roughly half of "Top" subset and noise examples from half of "Bottom of $2n$". Thus, "Random of $2n$" uses more similar examples than "Bottom of $2n$".

Table 3.15 shows the results of various n-selection methods when LaBSE was used as the sentence encoder. In the table, "Top" indicates the n-nearest-neighbor sentences. The "Bottom of $2n$" and "Random of $2n$" have higher diversity than the "Top" on both the source and target sides, and the BLEU scores are correspondingly lower. In addition, "Random of $2n$" achieved higher BLEU score than

	ADC	
Model h	w/	w/o
LaBSE	2191.4	446.4 ($\times$0.20)
AvgEnc	1816.8	365.1 ($\times$0.20)
TF-IDF	2199.1	531.0 ($\times$0.24)
BM25	1903.9	471.6 ($\times$0.25)

Table 3.16: Efficiency of ADC in WMT'19 De–En. The results show the number of tokens generated per second, i.e., ↑tok/s, with the B_∞ setting.

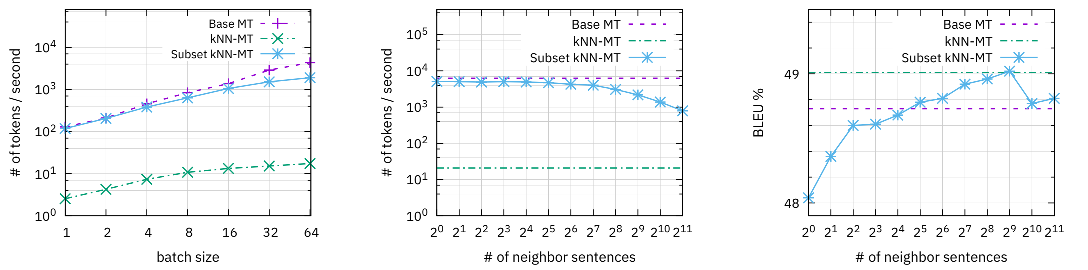
"Bottom of $2n$" with lower unique ratio. resulting in lower translation quality than "Top". These experiments showed that a sentence encoder that calculates similarity appropriately can reduce noise and prevent the degradation of translation quality because the subset consists only of similar sentences.

3.6.3 Analysis of Decoding Speed

Efficiency of ADC Subset kNN-MT computes the distance between a query vector and key vectors using ADC as described in Section 3.3.2. The efficiency of ADC in WMT'19 De–En is demonstrated in Table 3.16. The results show that "w/ ADC" is roughly 4 to 5 times faster than "w/o ADC".

Effect of Parallelization The method and implementation of our subset kNN-MT are designed for parallel computing. We measured the translation speed for different batch sizes in WMT'19 De–En. Figure 3.5(a) shows that subset kNN-MT (h: LaBSE) is two orders of magnitude faster than kNN-MT even when the batch size is increased.

Subset Size We measured the translation speed for different subset sizes, i.e., the number of n-nearest-neighbor sentences in WMT'19 De–En. Figure 3.5(b) shows the translation speed of subset kNN-MT (h: LaBSE). Subset kNN-MT is two orders of magnitude faster than kNN-MT even when the subset size is increased. The results also show that the speed becomes slower from $n = 256$



(a) Translation speed for different batch sizes.

(b) Translation speed for different subset sizes.

(c) Translation quality for different subset sizes in the development set.

Figure 3.5: Translation speed for different batch sizes, and subset sizes and translation quality for different subset sizes in WMT'19 De–En.

compared with base MT. We also found that 71.7% of the time was spent searching for the kNN tokens from the subset when $n = 2048$. Although ADC look-up search is slow for a large datastore, it is fast for kNN search when the subset size n is not large [78], e.g., $n = 512$.

Figure 3.5(c) shows the results for translation quality on the development set (newstest2018). The results show that a larger n improves BLEU up to $n = 512$, but decreases for greater values of n. In terms of both the translation quality and translation speed, we set $n = 512$ for WMT'19 De–En.

3.6.4 Relationship Between Neural/Non-neural Encoders and Translation Quality

From Section 3.5.2 and 3.5.3, the translation quality of the non-neural sentence encoder was almost the same as that of the neural sentence encoder in the WMT'19 translation task, while the non-neural sentence encoder degraded the translation quality compared with the neural sentence encoder in the domain adaptation task. We hypothesize that one of the causes of this phenomenon is that calculating TF-IDF and BM25 on a sentence, rather than on a document, would not extract sufficient statistics, especially in short sentences.

To verify this, we measured the total difference in sentence BLEU when using LaBSE and TF-IDF for each length bucket of the source sentences. Note that the

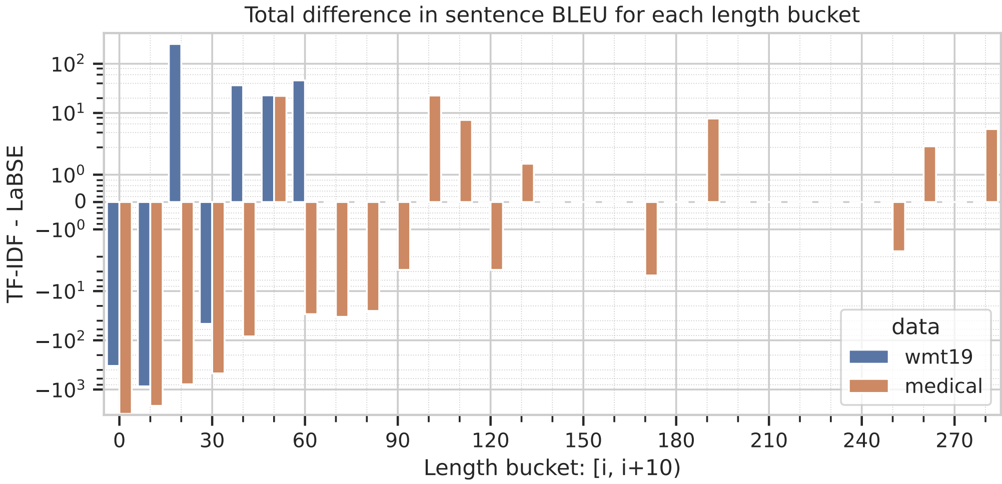


Figure 3.6: Total difference in sentence BLEU for each length bucket.

length bucket means the range from i (inclusive) to $i+10$ (exclusive). Figure 3.6 shows the results in the WMT'19 translation task and the medical domain adaptation task. It can be seen that TF-IDF often degraded the translation quality in short sentences, and the degradation is suppressed as the sentence length increases in both datasets. From Figure 3.7, the medical domain task has more short source sentences than the WMT'19 translation task. Therefore, the score difference between TF-IDF and LaBSE in the medical domain could have been larger than that in the WMT'19 translation task due to sentence lengths.

To summarize, we found that the non-neural encoder, TF-IDF, degraded the translation quality, especially for short sentences, while the neural encoder, LaBSE, retrieved similar sentences robustly and prevented the degradation even for short sentences.

3.7 Related Work

The first type of example-based machine translation method was analogy-based machine translation [84]. Zhang et al. [130], Gu et al. [42] incorporated example-based methods into NMT models, which retrieve examples according to edit distance. Bulte and Tezcan [10] and Xu et al. [122] concatenated an input sentence

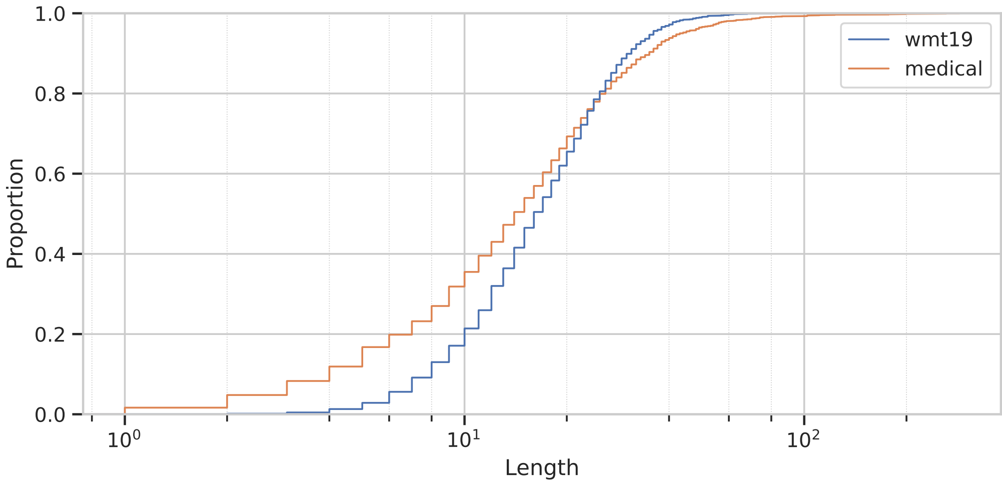


Figure 3.7: Cumulative distribution of the lengths of source sentences.

and translations of sentences similar to it. Both kNN-MT and subset kNN-MT retrieve kNN tokens according to the distance of intermediate representations and interpolate the output probability.

To improve the decoding speed of kNN-MT, fast kNN-MT [81] constructs additional datastores for each source token, and reduces the kNN search space using their datastores and word alignment. Subset kNN-MT requires a sentence datastore that is smaller than source token datastores and does not require word alignment. Martins et al. [76] cached n-gram tokens adjacent to the retrieved tokens and reduced the number of querys for the entire datastore; their model led to a speed-up of up to 4 times, compared with kNN-MT. In contrast, subset kNN-MT does not search for the entire datastore during decoding. Dai et al. [22] reduced the kNN search space by retrieving the neighbor sentences of the input sentence. They searched for neighboring sentences by BM25 scores with ElasticSearch[10], so our subset kNN-MT with BM25 can be regarded as an approximation of their method. They also proposed "adaptive lambda", which dynamically computes the weights of the lambda of linear interpolation in Equation 3.2 from the distance between the query and the nearest neighbor key vectors. However, adaptive lambda requires an exact distance and cannot employ data-

[10] https://github.com/elastic/elasticsearch

store quantization and the ADC look-up. To improve the translation quality of kNN-MT, Zheng et al. [131] computed the weighted average of kNN probabilities $p_{k\mathrm{NN}}$ over multiple values of k. Each weight is predicted by "meta-k network", trained to minimize cross-entropy in the training data. Their adaptive kNN-MT only improved the translation quality and its decoding speed is almost the same as that of kNN-MT[131]. In contrast, we focused on the improvement of the decoding speed. Additionally, our subset kNN-MT outperformed kNN-MT in some domain adaptation tasks as a positive side effect of subset retrieval. For the other tasks, kNN-LM [60], Efficient kNN-LM [46], and RETRO [7] used kNN search for language modeling (LM). Our subset search method may be applied to LM regarding the prompt text as the query, but the way to construct the sentence datastore from monolingual data is non-trivial, and we leave this issue for future work.

Some work used sentence similarity to improve the translation quality of NMT models. Wieting et al. [120] showed that minimum risk training using the cosine similarity between the generated hypothesis and the reference translation improved the translation quality of NMT models. Another approach uses the sentence similarity between the output sentence and the reference as the reward of reinforcement learning [128] to prevent excessive penalty due to cross-entropy that does not take into account the semantics of the sentence. Both of their methods used sentence similarity to put the semantics of the output sentence close to the reference translation, whereas our method uses sentence similarity to search for translation examples. They used the sentence similarity in the target side, while we use the similarity between the input sentence and the source sentences in the parallel data.

Quality estimation models and metric models, which use similarity between the source sentence and the hypothesis, or the hypothesis and the reference, have been proposed to evaluate the translation quality [99, 100, 104]. They use the similarity on the target side, whereas our model uses it on the source side. Sellam et al. [104] augmented the training data of the metric model by mask-filling with BERT [26], back-translation, and dropping words to allow the model to capture the various errors. In our model, we may improve the accuracy of the similar sentence search by fine-tuning the sentence encoder to retrieve better subsets that improve the translation quality.

In the field of kNN search, Matsui et al. [78] allowed search in dynamically created subsets, whereas conventional search methods assume only full search. Subset kNN-MT retrieves kNN tokens from a subset depending on a given input. In our subset kNN-MT, the decoding speed is slow when the subset size n is large. The bottleneck is the look-up in the distance table, and this can be improved by efficient look-up methods that use SIMD [2, 79].

3.8 Limitations

This study focuses only on improving the speed of kNN-MT during decoding; other problems with kNN-MT remain. For example, it still demands large amounts of memory and disk space for the target token datastore. In addition, our subset kNN-MT requires to construct a sentence datastore; therefore, the memory and disk requirements are increased. For example, the quantized target token datastore has 52GB ($|\mathcal{M}|$ = 862,648,422) and our sentence datastore has 2GB ($|S|$ = 29,540,337) in the experiment of WMT'19 De–En (Section 3.5.2). Although subset kNN-MT is faster than the original kNN-MT in inference, datastore construction is still time-consuming. The decoding latency of our subset kNN-MT is still several times slower than base MT for large batch sizes. The experiments reported in this study evaluated the inference speed of the proposed method on a single computer and single run only; the amount of speed improvement may differ when different computer architectures are used.

3.9 Conclusion

We proposed "Subset kNN-MT", which improves the decoding speed of kNN-MT by two methods: (1) retrieving neighbor tokens from only the neighbor sentences of the input sentence, not from all sentences, and (2) efficient distance computation technique that is suitable for subset neighbor search using a look-up table. Our subset kNN-MT achieved a speed-up of up to 134.2 times and an improvement in BLEU of up to 1.6 compared with kNN-MT in the WMT'19 De–En translation task, the domain adaptation tasks in De–En and En–Ja, and the Flores101 multilingual translation task. From the experiments, we found that the translation quality varied depending on sentence encoders. For future work, we would like to compare them with other pre-trained models and also fine-tune sentence encoders maximizing the metrics. In addition, we would like to apply our method to other text generation tasks, such as not only single-modal tasks like text simplification but also multi-modal tasks like speech-to-text translation.

Chapter 4

Detector–Corrector: Edit-Based Automatic Post Editing for Human Post-Editing

4.1 Introduction

Neural machine translation (NMT) [112, 3, 72, 121, 113] sometimes make errors [90], and post-editing is crucial in the real world to correct the mis-translations. Automatic post-editing (APE) attempts to correct and refine the translations generated by MT models (MT sentences) for better translation quality. However, many APE models are based on sequence generation [58, 20, 106, 11, 12, 5], and their decision for correction is harder to interpret due to the black-box nature of the generation models.

Some prior work [75, 43, 89, 109, 73, 74] showed that edit-based models improve interpretability in monolingual text editing, e.g., grammatical error correction (GEC), compared with sequence-to-sequence models. The APE task can be regarded as a text edit task in terms of rewriting MT sentences, but differs from general monolingual text editing tasks in that it uses cross-lingual information from source sentences, such as inserting untranslated words and reordering translation words. For example, if an edit-based model cannot perform reordering, it is represented as deletion and insertion, which increases the number of edit operations and makes it harder for humans to interpret the edit.

In this paper, we propose "detector–corrector", an edit-based post-editing

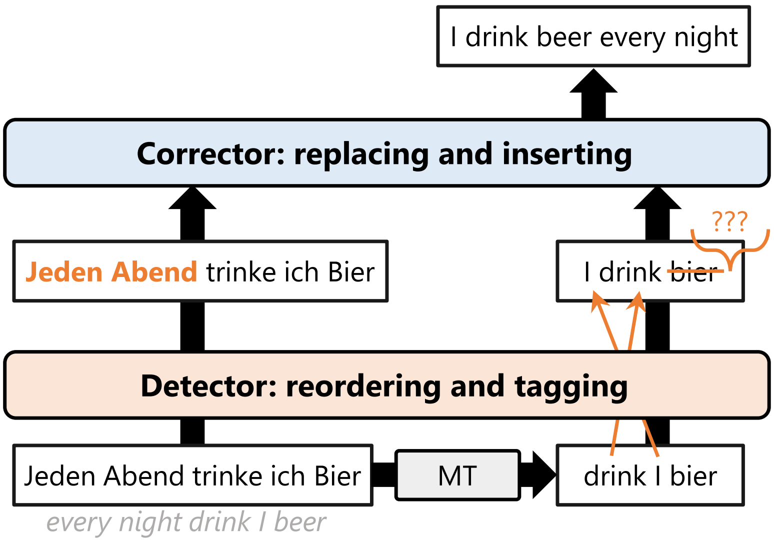


Figure 4.1: Overview of the post-editing process of our detector–corrector model. The detector tags as "Jeden Abend" is untranslated, "drink" and "I" should be reordered, etc. The corrector generates the word sequence for replacement and insertion.

model, in which the post-editing process is broken into two steps for assisting human post-editing: error detection and error correction. We designed our model after interviewing with professional translators regarding the post-editing process; specifically, they first spot errors and then make corrections, and omission errors are crucial for the editing process. The overview of our detector–corrector model is shown in Figure 4.1. The detector model, which extends a word-level quality estimation (QE) model, tags each MT output token as whether it should be corrected and/or reordered and identifies which source tokens are not translated in the MT sentence. Then, the corrector model receives the annotated source and MT sentences and corrects words for each span identified as incorrect in the detector model. Our corrector model can insert any number of spans of variable length. In addition, we propose data augmentation methods especially designed for the detector and corrector models to enhance each model, and lightweight iterative refinement to improve the inference speed.

Experiments on the WMT'20 English–German (En–De) and English–Chinese (En–Zh) APE tasks showed that our detector–corrector improved translation edit rate (TER) [107] compared to not only an edit-based model [43] but also a black-box sequence-to-sequence model by 0.7 points in En–De and En–Zh. Moreover, our model is more explainable than sequence-to-sequence models because it is based on edit operations and it can be integrated into computer-aided translation tools [47].

4.2 Background and Related Work

4.2.1 Edit-Based Model

Chen et al. [16] have built an edit-based GEC system that detects erroneous spans and then corrects the words within the detected erroneous spans. GECToR [89] is also an edit-based GEC mode, in which the model predicts the error type tag for each word, and then words identified as errors are corrected according to the rules for each tag type.

Levenshtein Transformer [43], a non-autoregressive Transformer encoder-decoder model, predicts deletion, placeholder insertion, and word filling. It can be used for the APE task by rewriting an MT sentence, but it cannot represent reordering and detecting untranslated words. Seq2Edits [109] edits an input text by span tagging and replacement prediction to improve interpretability for text-editing tasks. However, it is not suitable for the APE task because it only monotonically edits an MT output from left to right according to the tags and cannot perform reordering of spans or inserting missing words which often occur in erroneous translations. FELIX [73] breaks down text editing into three components: tagging, reordering, and word in-filling. It performs tagging using a pre-trained encoder model like BERT, reordering using a pointer network, and predicting words of replacement and insertion using a masked language model. However, it does not explicitly use source information. In addition, word insertion is predicted non-autoregressively; thus, the number of words to be inserted must be given in advance for the insertion operation, which is not trivial. EdiT5 [74] uses the T5 [95] encoder-decoder and decomposes the editing process into (1) tagging that decides which tokens are kept, (2) reordering the input tokens, and (3) insertion that infills the missing tokens. Unlike FELIX, Edit5 uses the autoregressive T5 decoder for word prediction, allowing for variable length insertion. However, the positions that can be inserted depend on the special tokens used in pre-training of T5 for filling masked spans, e.g., <extra_id_6> as <pos6>; thus, the number of positions that can be inserted is limited to those observed in pre-training.

4.2.2 Word-Level Quality Estimation

The word-level quality estimation task estimates the word-level quality of MT sentences, which is closely related to the post-editing task. It is divided into three binary classifications [108]: MT-tag, MT-gap, and SRC-tag. MT-tag detects erroneous words in MT sentences. MT-gap predicts where to insert untranslated words in MT sentences, and SRC-tag detects untranslated source words.

Predictor-estimator model [62, 63] is a well-known architecture for the word-level quality estimation task, in which the predictor is used for feature extraction from translation results while the estimator estimates the translation quality based on the features from the predictor. Ding et al. [28] used Levenshtein Transformer [43] for the word-level quality estimation task. Their method uses the edit probabilities of deletion and insertion of Levenshtein Transformer as tag prediction probabilities instead of explicitly predicting OK/BAD tags. DirectQE [21] is a pre-training method designed for the QE task, which consists of two components: generator and detector. In pre-training, The generator rewrites words by a cross-lingual masked language model, then the detector detects the replaced words. After pre-training, the detector model is fine-tuned with real QE data. SiameseTransQuest [96] employed the word-level QE architecture using XLM-R for the sentence-level quality estimation task, and they showed that using XLM-R is effective in the QE task. Ranasinghe et al. [97] demonstrated that the fine-tuned XLM-R predicts word-level QE on other language pairs than a language pair that is trained explicitly, i.e., the model can perform zero-shot QE.

4.2.3 Automatic Post Editing

The automatic post-editing (APE) task aims to improve the translation quality by editing translations generated from black-box MT models [12]. The APE system receives the source and MT sentences and generates the post-edited (PE) sentence. This task mainly evaluates correction performance using translation edit rate (TER) [107] based on the edit distance between the human-revised translation and the corrected sentence.

Correia and Martins [20] built a sequence-to-sequence APE system by only fine-tuning pre-trained BERT models, in which weight initialization is carefully designed to employ pre-trained weights for both encoder and decoder. In the APE shared task, the high-ranked systems often employ Transformer encoder-decoder architectures with pre-trained models [12, 5, 124, 118, 68, 24, 51]. The sequence-to-sequence model, which learns post-editing in an end-to-end manner, can achieve high translation quality; however, it cannot explicitly expose the editing process, making it hard to utilize the model in scenarios that require manual checking. The copy mechanism [41] can be used for APE tasks by copying words in MT sentences that do not need to be modified [50]. This model can show us edited and non-edited words using the copy probability. Neural Programmer-Interpreter (NPI) [117] generates PE sentences by predicting the edit actions and the target tokens comprising three editing operations: keep, delete, and insert. Although NPI is more interpretable than the sequence-to-sequence models, it cannot represent reordering nor differentiate replacement and insertion. Deoghare et al. [25] incorporated the word-level quality estimation into an APE model. Their model predicts which word should be edited through multi-task learning; however, it cannot use human-annotated QE tags because the information of QE tags, which is passed to the decoder, is represented as hidden vectors.

4.3 Proposed Model: Detector–Corrector

4.3.1 Edit Operations

We first discuss edit operations that our model treats. In previous work, the most widely used operations are deletion, replacement, and insertion [117, 43, 16, 73, 74]. Note that some models support only a few operations. For example, Levenshtein Transformer does not perform the replacement operation explicitly.

In the GEC task, GECToR [89] and Seq2Edits [109] predict error type tags for each input token or span. Their models provide more human-interpretable

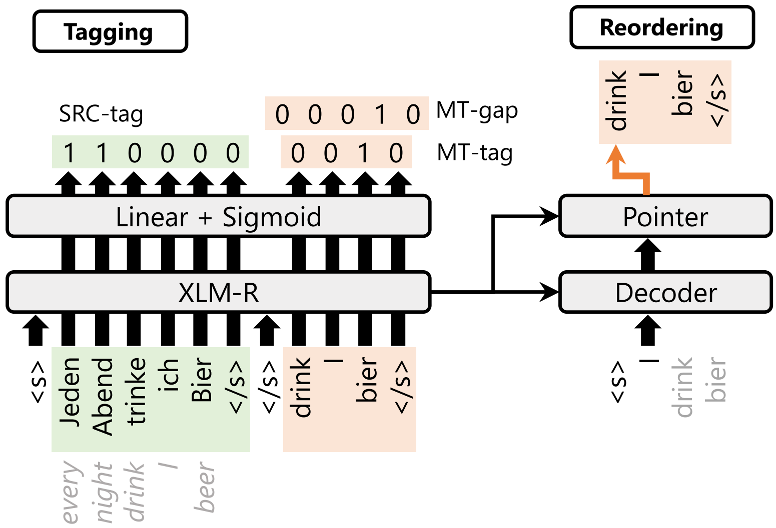


Figure 4.2: Overview of our detector model. The model detects OK and BAD tags as 0 and 1, respectively.

outputs by predefining many types of edit operations based on the human tendency to make grammatical errors. This study attempts to correct translations generated from any MT systems and we do not care about a particular model; thus, it is difficult to predefine specific error types.

Since the above-mentioned general operations, i.e., deletion, insertion, and replacement, are designed for monolingual text editing tasks, these operations may lack the edits required for the translation post-editing. For instance, word reordering might be helpful for translations of language pairs that have different word orders [129, 15]. If a translation model generates n-gram repetition, n-gram deduplication will be needed [45]. In industrial translation, lexical substitution by matching to the bilingual dictionary is necessary to deal with terminology translation [6].

In this study, we focus on the operations of deletion, insertion, replacement, and word reordering, which are employed in the several evaluation metrics of the translation quality, e.g., TER [107], CDER [69], and extended edit distance (EED) [110].

4.3.2 Detector

Our detector model (Figure 4.2) predicts shift and edit operations based on translation edit rate (TER) [107]. TER iteratively reorders an input sequence to minimize the edit distance from the target sequence, called "shift" operation, then calculates edit distance between the reordered input sequence and the target sequence, called "edit" operations. To represent this TER behavior, our detector model performs tagging to predict whether edits are need needed ("Tagging" in Figure 4.2), and reordering of the given MT sentence with a pointer network [114] ("Reordering" in Figure 4.2). Let $\boldsymbol{x} = (x_1, \ldots, x_{|\boldsymbol{x}|}) \in \mathcal{V}^*$ and $\boldsymbol{y} = (y_1, \ldots, y_{|\boldsymbol{y}|}) \in \mathcal{V}^*$ denote the given source sentence and its translation generated by machine translation (MT sentence), respectively, where $\mathcal{V}^*$ is the Kleene closure of the vocabulary[1] $\mathcal{V}$. Note that both $\boldsymbol{x}$ and $\boldsymbol{y}$ always have the end-of-sentence symbol "`</s>`" as the last tokens, i.e., $x_{|\boldsymbol{x}|} = y_{|\boldsymbol{y}|} =$ "`</s>`". Let $\boldsymbol{x} \circ \boldsymbol{y}$ be the concatenated sequence, where $\circ$ represents the join operation with a separator token between the sequences[2]. XLM-RoBERTa (XLM-R) encoder [19] encodes the concatenated sequence $\boldsymbol{x} \circ \boldsymbol{y}$ into D-dimensional hidden vectors through L layers $\boldsymbol{H}^{(L)} = (\boldsymbol{h}_1^{(L)}, \ldots, \boldsymbol{h}_{|\boldsymbol{x} \circ \boldsymbol{y}|}^{(L)})^\top \in \mathbb{R}^{|\boldsymbol{x} \circ \boldsymbol{y}| \times D}$.

Tagging To perform tagging, we train a word-level quality estimation model. In particular, the detector model performs three binary classifications as defined by Specia et al. [108]: MT-tag, MT-gap, and SRC-tag.

Let $\boldsymbol{o}^T \in \{0, 1\}^{|\boldsymbol{y}|}$ denote the MT-tag which represents whether an MT token would be edited, i.e., $o_i^T = 1$ if y_i is deletion or replacement in a TER edit sequence, e.g., "bier" in Figure 4.2. The MT-tag classification identifies whether an MT token should be edited based on the bad probabilities:

$$p_i^T \coloneqq p(o_i^T = 1 | \boldsymbol{x}, \boldsymbol{y}) = \sigma(\boldsymbol{w}_T^\top \boldsymbol{h}_{y_i}^{(l_T)}), \tag{4.1}$$

where $\boldsymbol{w}_T \in \mathbb{R}^D$ is a learned parameter for MT-tag prediction, $1 \leq l_T \leq L$ denotes the layer used for MT-tag prediction, and $\sigma : \mathbb{R} \to [0, 1]$ is a sigmoid function.

[1] We employ XLM-R, a multilingual encoder; thus, the vocabulary is shared between the source and target languages.

[2] In XLM-R, the class token is represented by "`<s>`", and two sentences are joined by "`</s>`" symbols, like "`<s>` a b c `</s>` `</s>` A B `</s>`". We regard the first symbol as the end-of-sentence symbol of the first sentence, i.e., $x_{|\boldsymbol{x}|}$, and the second one as the separator token.

Note that $\boldsymbol{h}_{y_i}^{(l)}$ is a row of $\boldsymbol{H}^{(l)}$, which is the hidden vector corresponding to the token y_i in the l-th layer.

Similarly, MT-gap classification predicts whether some words need to be inserted at a token boundary in the MT sentence based on the insertion probabilities:

$$p_i^G \coloneqq p(o_i^G = 1|\boldsymbol{x}, \boldsymbol{y}) = \sigma(\boldsymbol{w}_G^\top[\boldsymbol{h}_{y_{i-1}}^{(l_G)}; \boldsymbol{h}_{y_i}^{(l_G)}]), \tag{4.2}$$

where $\boldsymbol{o}^G \in \{0, 1\}^{|\boldsymbol{y}|}$ represents insertion in a TER edit sequence, e.g., the token boundary between "bier" and "</s>" in Figure 4.2. $\boldsymbol{w}_G \in \mathbb{R}^{2D}$ is a learned parameter for MT-gap prediction, $1 \leq l_G \leq L$ denotes the layer used for MT-gap prediction, and $[\cdot;\cdot]$ denotes the concatenation of two vectors. Note that y_0 is the separator token between the source and MT sentences.

Likewise, the SRC-tag $\boldsymbol{o}^S \in \{0, 1\}^{|\boldsymbol{x}|}$ is constructed from a source-target word alignment as $x_i = 1$ if x_i is not aligned to any target token like "Jeden" and "Abend" in Figure 4.2. In this paper, we used AWESOME-ALIGN [30] to obtain the gold alignment. The SRC-tag classification predicts whether a source token is untranslated or not using the probabilities:

$$p_i^S \coloneqq p(o_i^S = 1|\boldsymbol{x}, \boldsymbol{y}) = \sigma(\boldsymbol{w}_S^\top \boldsymbol{h}_{x_i}^{(l_S)}), \tag{4.3}$$

where $\boldsymbol{w}_S \in \mathbb{R}^D$ is a learned parameter for SRC-tag prediction and $1 \leq l_S \leq L$ denotes the layer used for SRC-tag prediction.

During inference, each tag $\boldsymbol{o}^T$, $\boldsymbol{o}^G$, and $\boldsymbol{o}^S$ are respectively predicted to be "BAD" when each probability p_i is greater than 0.5, and "OK" otherwise.

Reordering Our detector also predicts reordering by generating the reordered sequence $\bar{\boldsymbol{y}} = (\bar{y}_1, \ldots, \bar{y}_{|\bar{\boldsymbol{y}}|})$ using the pointer network [114] at the top of the decoder. It autoregressively selects the next token for each timestep from the MT sentence according to the probability p^R, as follows:

$$\bar{\boldsymbol{y}}^* = \operatorname*{argmax}_{(\bar{y}_1, \ldots, \bar{y}_{|\bar{\boldsymbol{y}}|})} \prod_{i=1}^{|\boldsymbol{y}|} p^R(\bar{y}_i|\boldsymbol{x}, \boldsymbol{y}, \bar{\boldsymbol{y}}_{<i}), \tag{4.4}$$

$$p^R(\bar{y}_i = y_j|\boldsymbol{x}, \boldsymbol{y}, \bar{\boldsymbol{y}}_{<i}) \propto \exp(\boldsymbol{k}_{y_j}^\top \boldsymbol{q}_{\bar{y}_i}), \tag{4.5}$$

$$\boldsymbol{k}_{y_j} = \boldsymbol{W}_k \boldsymbol{h}_{y_j}, \tag{4.6}$$

$$\boldsymbol{q}_{\bar{y}_i} = \boldsymbol{W}_q \text{Decoder}(\bar{\boldsymbol{y}}_{<i}, \boldsymbol{H}^{(L)}), \tag{4.7}$$

where Decoder: $\mathcal{V}^* \times \mathbb{R}^{|\boldsymbol{x} \circ \boldsymbol{y}| \times D} \rightarrow \mathbb{R}^D$ is a Transformer decoder that computes a hidden vector of the i-th step $\boldsymbol{q}_{\bar{y}_i}$ from the given encoder hidden vectors and the prefix of reordered sequence. $\boldsymbol{W}_q \in \mathbb{R}^{D \times D}$ and $\boldsymbol{W}_k \in \mathbb{R}^{D \times D}$ are the learned parameters, and $\bar{\boldsymbol{y}}^*$ is the reordered sequence predicted by the model. Note that the hidden vectors $\boldsymbol{H}^{(L)}$ are computed using the same encoder as used in tagging.

During inference, the tokens of the MT sentence and their corresponding MT-tag and MT-gap are reordered according to the order of $\bar{\boldsymbol{y}}^*$. Note that the MT-gap tags are reordered in accordance with the order of their right-side tokens of boundaries. For example, in Figure 4.2, the MT-gap model predicts that some words need to be inserted at the token boundary between "bier" and "`</s>`", and the boundary position is attached to the left of "`</s>`" after reordering.

Objective function We trained the MT-tag, MT-gap, and SRC-tag classifications by minimizing their objective functions, $\mathcal{L}_T$, $\mathcal{L}_G$, and $\mathcal{L}_S$, computed by the binary cross-entropy, as follows:

$$-\sum_i \left(o_i \log p_i + (1 - o_i) \log(1 - p_i)\right), \tag{4.8}$$

where $o_i \in \{0, 1\}$ is the ground truth label of the probability p_i. The model is also trained to generate reordered MT sentences by minimizing the following cross-entropy:

$$\mathcal{L}_R = -\sum_{i=1}^{|\boldsymbol{y}|} \log p^R(\bar{y}_i | \boldsymbol{x}, \boldsymbol{y}, \bar{\boldsymbol{y}}_{<i}), \tag{4.9}$$

where the gold reordered sequence is created from the TER shift alignment. Finally, our detector model is trained by minimizing the following objective $\mathcal{L}$ through multi-task learning:

$$\mathcal{L} = \mathcal{L}_T + \mathcal{L}_G + \mathcal{L}_S + \mathcal{L}_R. \tag{4.10}$$

Note that all loss functions in $\mathcal{L}$ are computed during a single forward pass since the encoder parameters are shared between all tagging and reordering predictions.

4.3.3 Corrector

The corrector model (Figure 4.3) corrects the reordered MT sentence by generating tokens corresponding to the erroneous spans identified by MT-tag and

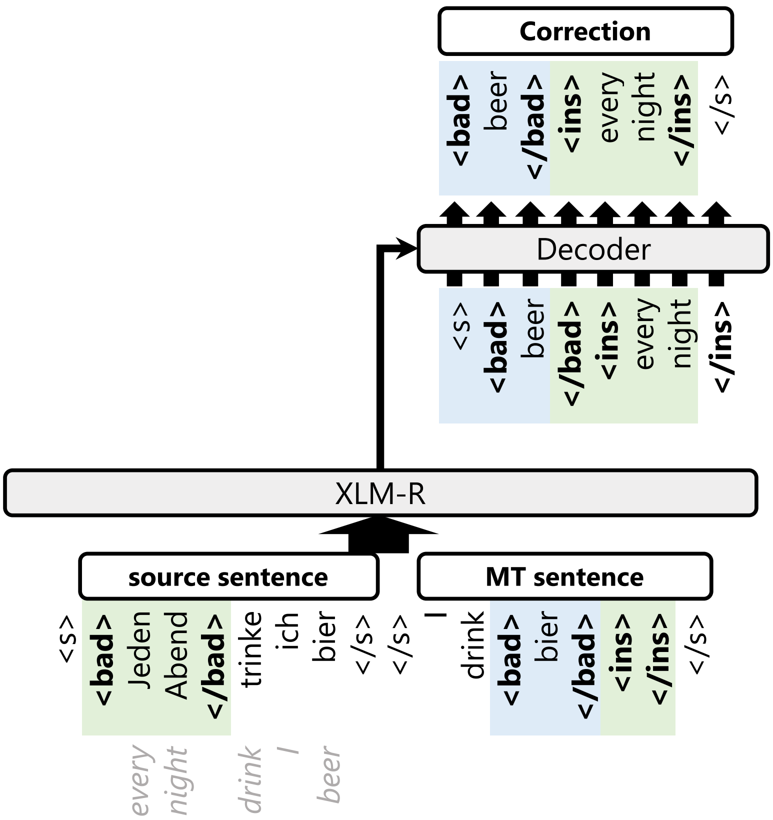


Figure 4.3: Token generation within each tagged span by our corrector model.

MT-gap predictions. The corrector represents edit operations by predicting zero words in a bad span for deletion, one or more words in a bad span for replacement, and one or more words in an insertion span for insertion, as shown on the output of the decoder in Figure 4.3.

First, the tags predicted by the detector model are used to annotate the source sentence and its corresponding reordered MT output as span tags. In the source sentence, `<bad>` and `</bad>` tags are inserted to the beginning and end of untranslated spans, respectively, using the SRC-tag $\boldsymbol{o}^S$, as shown on the left side of the input of the XLM-R encoder in Figure 4.3. Similarly, `<bad>` and `</bad>` tags are inserted into reordered MT output where identified by the MT-tag tagging $\boldsymbol{o}^T$ in addition to the `<ins>` and `</ins>` tags to the positions that need to be inserted words, as shown on the right side of the input of the XLM-R encoder in Figure 4.3.

Next, the annotated source and reordered MT sentences are concatenated with the separator token and fed into the encoder. We initialize the corrector encoder with XLM-R as well as the detector model in order to preserve consistency with the subword unit tags used in the detector. Then, the decoder generates tokens for all tagged spans in the left-to-right manner until the number of corrected spans satisfies the number of bad and insertion spans in the annotated reordered MT sentence. Finally, our detector–corrector outputs a corrected target sentence by replacing each tagged span of the MT sentence with a token sequence predicted by the corrector decoder.

Our corrector can be regarded as a translation suggestion (TS) model [126, 127], in which better alternative translations are suggested phrase-by-phrase by replacing incorrect translation spans. Our model differs from TS models in that untranslated spans in source sentences are explicitly identified and incorrect translations and/or insertions are clearly differentiated by the bad and insertion tags, respectively. Furthermore, MT sentences are reordered and multiple spans are corrected in our model, which are out of the scope of the TS task[3].

[3] The TS task assumes only a single incorrect span for each sentence and does not treat reordering.

4.3.4 Data Augmentation

Data Augmentation for Detector

Since the detector–corrector is trained to correct only erroneous spans identified by the detector, improving the tagging accuracy will directly lead to improved translation quality. For this purpose, we create the synthetic data from the reference translations of the training data and let the detector learn the editing operations of deletion, replacement, and insertion. We randomly delete tokens with a probability of 5%, insert tokens with a probability of 10%, and replace tokens with a probability of 30%. We employ XLM-R to fill the masked tokens for the replacement and insertion decision.

Data Augmentation for Corrector

The training data for the corrector model is created from the tokens for each span identified as an error using the oracle annotated source and MT sentences. However, the detector might make wrong decision during inference, which might cause a large discrepancy between the training and inference for the corrector. In addition, the performance of the corrector might suffer from the limited coverage of the vocabulary in the training data when compared with a conventional sequence-to-sequence MT model. For these reasons, we employ two simple data augmentation methods for the corrector model without additional computational cost: MT training and PE training. These two augmentation methods are orthogonal with each other; thus, they can be combined.

MT Training In MT training, the corrector model is trained to predict the PE sentence from only the source sentence without the corresponding MT sentence. To preserve the model consistency, an MT output is treated as an empty text by augmenting with "`<ins> </ins>`" so that the model learns to insert the whole PE sentence from the empty MT sentence. The encoder input sequence of MT training is formulated as follows:

$$\texttt{<bad>}\ \boldsymbol{x}\ \texttt{</bad>} \circ \texttt{<ins>}\ \texttt{</ins>}, \tag{4.11}$$

and the corrector is trained to generate the post-edited sentence with the insertion, i.e., `<ins>` $\boldsymbol{y}^{\mathrm{PE}}$ `</ins>`, where $\boldsymbol{y}^{\mathrm{PE}} \in \mathcal{V}^*$ is the post-edited sentence.

PE Training PE training differs from MT training in that the MT sentences are given. The corrector model is trained to generate the whole PE sentence from the given source and MT sentences. This is the same setting as the standard sequence-to-sequence APE model training, except that the MT sentence is explicitly annotated as "`<bad>`". To maintain model consistency, the whole MT sentence is treated as a bad span to be corrected:

$$\boldsymbol{x} \circ \texttt{<bad>}\ \boldsymbol{y}\ \texttt{</bad>}, \tag{4.12}$$

and the model learns to replace the MT sentence with the PE sentence, i.e., the model is trained to generate `<bad>` $\boldsymbol{y}^{\mathrm{PE}}$ `</bad>`.

4.3.5 Lightweight Iterative Refinement

The detector model detects each erroneous span in a non-autoregressive manner; thus, a single inference may not generate sufficiently correct PE sentences that are consistent across the entire sentence. To address such issues, some prior non-autoregressive models [43, 59, 89] decode sequences by iteratively feeding the output into the model. We follow the practice by iteratively refining an MT sentence by treating the post-edited sentence corrected by our model as an MT output, i.e., the corrected sentence in the $k-1$-th iteration is used as the input of the detector model in the k-th iteration. However, the iterative refinement approach demands huge computation in particular for our approach, in which an end-to-end inference predicts three edit operations in the following order: tagging, reordering, and correcting.

Tagging can be predicted with only a single forward pass of the detector encoder, and correcting can be finished very quickly since it generates only a few words for each erroneous span. In contrast, reordering is relatively slower than the other operations because the decoder runs for the length of the MT sentence in an auto-regressive manner.

In order to overcome such bottleneck, we propose lightweight refinement, in which inference is carried out only by predicting tags and generating correct tokens without reordering after the second time in the iterative refinement.

4.4 Experiments

4.4.1 Setup

We compared the translation quality of our detector–corrector with that of the sequence-to-sequence (seq2seq) APE model and Levenshtein Transformer (LevT) [43]. We evaluated TER (↓T), BLEU (↑B), and COMET (↑C) using sacreBLEU [94] and COMET[4] [98, 99] in the WMT'20 English–German (En–De) and English–Chinese (En–Zh) automatic post-editing tasks.

Datasets Training data came from WMT'20 APE tasks, which were created from wikipedia articles that contain 7,000 sentences, and we applied upsampling by 20 times to them. In addition to the provided data, we created additional training data that consists of ⟨source sentence, MT sentence, PE sentence⟩ triplets using a parallel corpus following the idea from Negri et al. [86]. In particular, we randomly sampled 2 million sentences from the training data of the WMT'19 En–De and En–Zh translation tasks and translated them with MT models, which were used to generate the data for the APE tasks [37]. As described in Section 4.3.4, the training data for the detector and corrector were further augmented. The data statistics are shown in the appendix (Table A.2).

Models The seq2seq APE model, LevT, and our detector–corrector comprise the XLM-R large encoder and Transformer decoder. The seq2seq, LevT, and corrector models were trained in 60,000 steps, and the detector model was trained in 40,000 steps. All models were optimized by Adam optimizer ($\beta_1 = 0.9, \beta_2 = 0.98, \epsilon = 10^{-8}$). The learning rate was linearly increased up to 4,000 steps and then decayed proportional to the inverse square root of the training steps. The beam size was set to 5, and the length penalty was set to $\alpha = 1.0$. We saved checkpoints of all models for every 1,000 steps and took an average of the last 5 checkpoints. The LevT edited the MT sentences 5 times iteratively, and the detector–corrector edited 4 times, i.e., $k = 4$, by tuning on the development set. For tagging, we used the intermediate representations of the 20th layer, i.e.,

[4] https://huggingface.co/Unbabel/wmt22-comet-da

Dataset	Model	↓T	↑B	↑C
En–De	do nothing (MT)	31.3	50.2	77.1
	seq2seq	28.4	53.3	77.7
	LevT [43]	31.9	49.4	75.6
	detector–corrector	**27.7**†	**53.6**	**79.6**†
En–Zh	do nothing (MT)	58.3	24.3	86.3
	seq2seq	56.7	26.0	**89.4**†
	LevT [43]	59.3	23.6	86.0
	detector–corrector	**56.0**	**26.1**	89.2

Table 4.1: Comparison of post-editing performance in the WMT'20 En–De and En–Zh APE tasks. Do nothing (MT) does not edit MT sentences and the scores are calculated between MT and PE sentences. The best scores of each dataset are emphasized by the **bold** font. The symbol † indicates that the score difference is statistically significant ($p < 0.05$) between seq2seq and detector–corrector.

$l_T = l_G = l_S = 20$ in En–De, and the 24th layer, i.e., $l_T = l_G = l_S = 24$ in En–Zh. The details of each model are shown in the appendix (Table A.1).

4.4.2 Results

Our main results are shown in Table 4.1. Our detector–corrector model improved TER and BLEU from both LevT and seq2seq models. Especially in TER, detector–corrector outperforms the black-box seq2seq model by 0.7 % in En–De and En–Zh while providing the editing process.

Table 4.2 shows the ablation study of our proposed methods. In the table, "light-iter" denotes the lightweight iterative refinement, and "DAug" denotes data augmentation. The results show that both lightweight iterative refinement and data augmentation for the detector and corrector are effective, which improve the TER scores by 3.5 % in En–De and 5.2 % in En–Zh compared to the vanilla detector–corrector.

Our data augmentation for the detector can be used for other baseline models,

Model	En–De ↓T	↑B	↑C	En–Zh ↓T	↑B	↑C
ours	**27.7**†	**53.6**†	**79.6**†	**56.0**†	**26.1**†	**89.2**†
- light-iter	28.9	52.1	77.7	56.6	25.5	88.0
-- MT training	29.3	51.5	77.7	56.6	25.4	88.3
-- PE training	29.2	51.8	77.7	56.6	25.2	88.3
-- DAug for corrector	30.2	50.1	77.6	57.0	24.9	88.6
--- DAug for detector	31.2	49.0	77.1	61.2	22.7	86.7

Table 4.2: Ablation study of our methods in the WMT'20 En–De and En–Zh APE tasks. The symbol † indicates that the score difference is statistically significant ($p < 0.05$) between "ours" and "- light-iter".

Dataset	Model	↓T w/o	↓T w	↑B w/o	↑B w	↑C w/o	↑C w
En–De	seq2seq	28.4	28.4	53.3	52.9	77.7	78.0
	LevT	31.9	32.1	49.4	49.0	75.6	75.8
En–Zh	seq2seq	56.7	57.0	26.0	26.0	89.4	89.5
	LevT	59.3	59.9	23.6	23.4	86.0	86.1

Table 4.3: Translation quality of baseline models trained using our data augmentation for the detector.

seq2seq and LevT[5]. To confirm that the data augmentation is effective for our model, we also trained the baseline models using the augmented data. Table 4.3 shows that the translation quality of baseline models trained on the augmented data. Unlike the "DAug for detector" row in Table 4.2, there is no improvement in all metrics of more than 1 % even if the augmented data is used. This is because the data augmentation for the detector is designed to enhance word-level quality estimation.

To summarize, we confirmed that our model outperformed LevT and a black-

[5] The data augmentation for corrector cannot be applied to other models because they have been already trained to generate the whole target sentence.

Tagging	Dataset	DAug	MCC	F1-OK	F1-BAD
Target	En–De	w/o	0.468	0.935	0.523
		w/	0.475	0.937	0.526
	En–Zh	w/o	0.505	0.893	0.602
		w/	0.537	0.902	0.619
Source	En–De	w/o	0.782	0.985	0.794
		w/	0.791	0.985	0.805
	En–Zh	w/o	0.641	0.943	0.695
		w/	0.676	0.948	0.724

Table 4.4: Word-level quality estimation performance of our detector model.

box seq2seq model, and our approaches mitigate the translation quality degradation issue caused by predicting tags in a non-autoregressive manner and being trained from only a vocabulary limited to correction words.

4.5 Discussion

4.5.1 Accuracy of the Detector

We evaluated the tagging performance of our detector model and investigated the effectiveness of data augmentation for the detector. Since tags are predicted on subword units, we assigned a BAD tag to a word if one of the subwords in the word was assigned a BAD tag. The gold tags are calculated from the TER edit sequence after applying the shift operations in the same way as described in Section 4.3.2.

Table 4.4 shows the results of the word-level quality estimation. In the table, "MCC" denotes Matthews correlation coefficient [80]. "Target" and "Source" are the target-side tagging, i.e., MT-tag and MT-gap without distinction, and the source-side tagging, i.e., SRC-tag, respectively. We only compared our models with and without data augmentation. This is because in the WMT'20 word-level QE task, the target-side tags are produced from TER edit operations without shift

Dataset	Model	↓T	↑B	↑C
En–De	do nothing (MT)	31.3	50.2	77.1
	detector–corrector	27.7	53.6	79.6
	w/ oracle tags	13.8	74.6	82.9
		(-13.9)	(+21.0)	(+3.3)
En–Zh	do nothing (MT)	58.3	24.3	86.3
	detector–corrector	56.0	26.1	89.2
	w/ oracle tags	33.2	46.6	90.1
		(-22.8)	(+20.5)	(+0.9)

Table 4.5: Correction performance in the WMT'20 En–De and En–Zh APE tasks when the erroneous spans are given manually.

operations, and the source-side tags are produced by FAST_ALIGN[6] [31], while in our model the target-side tags include the shift operation and the source-side tags are produced by AWESOME-ALIGN. The results show that the data augmentation for the detector improved the all MCC scores, which has the direct impact to the improvements measured by BLEU and TER for our detector–corrector as shown in Table 4.2.

We also observed that the F1-BAD scores of the target-side tagging are not high in both language pairs. In particular, the accuracy of erroneous span detection is 0.526 and 0.619 in En–De and En–Zh, respectively. This low accuracy could be the reason why the correction performance is only improved by 0.7% TER compared with the seq2seq model. Because the corrector model only corrects the detected spans, the F1-BAD scores are closely linked to the correction performance of our detector–corrector. The problem of the error detection performance is one of the remaining challenges in this study.

4.5.2 Correction Performance of Oracle Tagged Sentences

We evaluated the performance of the corrector model for oracle tags, assuming a setting in which error spans are given manually. Oracle tags were given from the

[6] SimAlign [52] is employed since the WMT'21 word-level QE task.

	En–De			En–Zh		
Reordering	↓T	↑B	↑C	↓T	↑B	↑C
w/	28.9	52.1	77.7	56.6	25.5	88.0
w/o	28.9	52.4	78.2	57.4	24.9	88.1

Table 4.6: Translation quality of detector–corrector with and without reordering. Note that we evaluated translation quality on the results of the first iteration in iterative refinement.

	En–De		En–Zh	
Reordering	# of edits	TER_{MT}	# of edits	TER_{MT}
w/	2,506	17.6	5,603	31.6
w/o	2,614	18.5	7,410	38.0

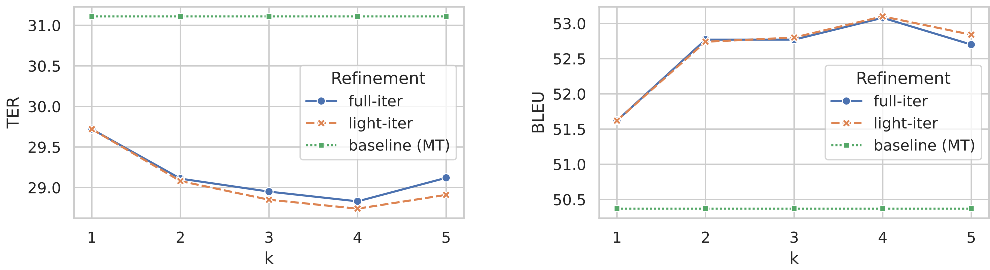
Table 4.7: The total number of spans tagged by the detector and TER scores that measured the amount of editing from the MT sentence to the post-edited sentence corrected by the corrector in the WMT'20 APE En–De and En–Zh tasks.

TER alignment between the MT sentence and the reference translation as well as the supervision in the training data.

In Table 4.5, "w/ oracle tags" shows the result of oracle correction in the WMT'20 En–De and En–Zh APE tasks. The results showed that when given the ideal tags, the correction performance significantly improved by -13.9 and -22.8 % TER, +21.0 and +20.5 % BLEU, and +3.3 and +0.9 % COMET in En–De and En–Zh, respectively. This means that the corrector model has been successfully trained, and a further improvement in post-editing performance can be achieved by improving the accuracy of the detector model.

4.5.3 Ablation Study of Reordering

We also investigated the effectiveness of using the reordering operation. The training data for the model without reordering was created from the edit align-



(a) Comparison of TER scores for each iteration.

(b) Comparison of BLEU scores for each iteration.

Figure 4.4: Comparison of various iterations in iterative refinement. The scores were evaluated on the development set in the WMT'20 En–De APE task.

ments based on the edit distance. We compared the translation quality in the first iteration. Table 4.6 shows the experimental results of detector–corrector with and without reordering. In TER, which indicates the number of edits to the reference translation, detector–corrector without reordering resulted in the same score as detector–corrector with reordering in En–De and degraded in En–Zh.

To investigate this gap in TER scores, we counted the total number of spans tagged by the detector and evaluated the TER score that measured the number of edits from the MT sentence to the post-edited sentence corrected by our detector–corrector (TER_{MT}). Table 4.7 shows that the number of edited spans was decreased by reordering, especially in En–Zh. In addition, the reordering operation reduces the TER_{MT} by 0.9% and 6.4% in En–De and En–Zh, respectively. This means that the number of edits from the MT sentence and the number of edits to the reference translation decreases by using the reordering operation; hence, the editing process becomes easier for humans to interpret.

In summary, we confirmed that reordering is effective in reducing the number of edits, as shown by the TER scores in Table 4.6 and Table 4.7.

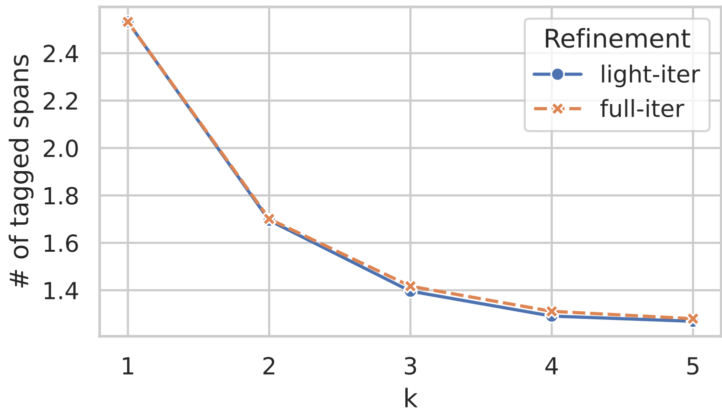


Figure 4.5: Number of tagged spans per sentence in the WMT'20 En–De APE task.

4.5.4 Effectiveness of Iterative Refinement

To verify the effectiveness of iterative refinement, we evaluated BLEU and TER scores in the WMT'20 En–De APE task at various numbers of inference iterations $k \in \{1, 2, 3, 4, 5\}$ on the development set. We also compared the difference between including ("full-iter") and not including ("light-iter") reordering when $k \geq 2$. Figure 4.4(a) and 4.4(b) shows that the first iterative refinement ($k = 2$) significantly improved the TER and BLEU scores from the first inference ($k = 1$). From $k = 2$ to 4, we see a slight improvement in both TER and BLEU. Comparing the iterative refinement methods, light-iter was slightly more accurate than full-iter, but the difference is lower than 0.1 % in both metrics.

Figure 4.5 shows the average number of bad- and insertion-tagged spans of MT sentences, which was corrected by the corrector. The figure shows that the number of corrected spans decreases in each iteration, especially when it significantly decreases in the second refinement, i.e., $k = 2$, which corresponds to the decrease of TER and BLEU in Figure 4.5.

We also measured the cumulative time for each inference step. Figure 4.6 shows the total inference time in seconds for full-iter and light-iter when processing 1,000 sentences. In the figure, "k-D" and "k-C" denote the k-th inference step of the detector model and corrector model, respectively. It can be seen that light-iter

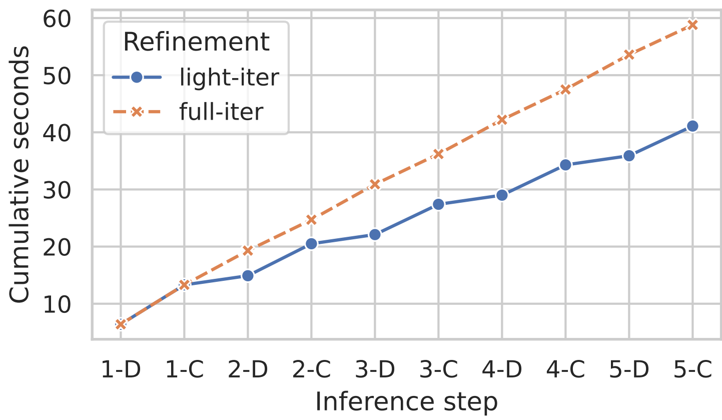


Figure 4.6: Cumulative time taken for each inference step. "k-D" and "k-C" denote the k-th inference step of the detector model and corrector model, respectively.

infers faster than full-iter because light-iter does not predict reordering, which is time-consuming, in the detector inference at each iteration in $k \geq 2$.

From the results, our detector–corrector is further improved by using iterative refinement at least twice, and the inference speed is reduced by two-thirds using our lightweight iterative refinement without losing qualities.

4.5.5 Case Study: Editing Process

We analyzed examples of the editing processes of detector–corrector. Table 4.8 shows an example of the editing process of an MT sentence. In the table, the "Annotated source" line is the source sentences annotated with SRC-tag by the detector, and the "Annotated MT" line is the reordered MT sentences annotated with MT-tag and MT-gap by the detector. The "Correction" and "Output" lines are the correction sequence generated by the corrector and the outputs of the detector–corrector, respectively. The table shows that our model detects and corrects the erroneous spans iteratively, and outputs the sentence with 17.7 TER in the second iteration. Note that the detector did not detect any erroneous spans in this example when $k \geq 3$. The table also shows that our model swaps two spans, "89 岁" and "佐治亚州 李", which makes the word order align with

Source	Georgia Lee , 89 , Australian jazz and blues singer .
Reference	乔治亚 · 李 (Georgia Lee) , 89 岁 , 澳大利亚 爵士 和 蓝调 歌手 。
MT (TER=64.7)	89 岁 的 佐治亚州 李 , 澳大利亚 爵士乐 和 布鲁斯 歌手 .
Reordered MT	的 佐治亚州 李 89 岁 , 澳大利亚 爵士乐 和 布鲁斯 歌手 .
$k = 1$	
Annotated source	Georgia Lee <bad>, </bad> 89 , Australian jazz and blues singer .
Annotated MT	<bad>的</bad> 佐治亚 <bad>州</bad> 李 <ins></ins> 89 岁 , 澳大利亚 爵士乐 和 <bad> 布鲁斯</bad> 歌手 <bad>.</bad>
Correction	<bad></bad> <bad> · </bad> <ins>,</ins> <bad>蓝调</bad> <bad>。</bad>
Output (TER=35.3)	佐治亚 · 李 , 89 岁 , 澳大利亚 爵士乐 和 蓝调 歌手 。
$k = 2$	
Annotated source	Georgia Lee , 89 , Australian jazz and blues singer .
Annotated MT	佐治亚 · 李 <ins></ins> , 89 岁 , 澳大利亚 爵士乐 和 蓝调 歌手 。
Correction	<ins> (George Lee) </ins>
Output (TER=17.7)	佐治亚 · 李 (George Lee) , 89 岁 , 澳大利亚 爵士乐 和 蓝调 歌手 。

Table 4.8: An example of the editing process.

the source sentence and reference translation. In this case, the person name "Georgia" is mistranslated to "George", but the output of $k = 2$ has a lower TER score (TER=17.7) than the MT output (TER=64.7); thus, the editing cost was reduced. In the future, we need to improve the detection performance to detect "George" detect as a mistranslation.

4.6 Limitations

Our study focuses on correcting translation errors, and thus our model cannot detect and correct non-factual information when including them in a source sentence. In addition, our model only corrects the erroneous spans detected by the detector; thus, spans that the detector fails to detect may remain uncorrected.

In addition, in our method, multiple editing processes can be considered for the same translation, but we trained models from a single editing process. It may improve the correction performance by training models from multiple editing processes.

This study only focuses on edit operations based on TER calculation: deletion, insertion, replacement, and word reordering. However, as mentioned in Section 4.3.1, there are other edit operations, e.g., lexical substitution by matching to the dictionary and n-gram deduplication. In addition, if the model-specific errors are classified, it would be possible to train a detector with their error type tags.

4.7 Conclusion

We proposed "detector–corrector", the edit-based automatic post-editing (APE) model, which explains which words are wrong in MT sentences and how to correct them for human post-editors. Experiments on the WMT' 20 English–German and English–Chinese APE tasks showed that our detector–corrector model provides the editing process and outperformed the previous edit-based model, Levenshtein Transformer, and a black-box sequence-to-sequence APE model in TER.

In the future, we will further investigate what is needed to reduce the workload of human post-editors. In addition, the corrector model can generate multiple correction candidates. Specifically, the use of diverse beam search and sampling-based decoding methods could be helpful to provide diverse translation suggestions. We would like to confirm that whether the corrector model can be utilized for the translation suggestion task in future work.

Chapter 5

Conclusion

5.1 Summary

This dissertation improved the efficiency of the translation process from both translation and post-editing aspects.

For domain adaptation, our subset kNN-MT improves the decoding speed of kNN-MT by two methods: (1) retrieving neighbor tokens from only the neighbor sentences of the input sentence, not from all sentences, and (2) efficient distance computation technique that is suitable for subset neighbor search using a look-up table. Our subset kNN-MT achieved a speed-up of up to 134.2 times and an improvement in BLEU of up to 1.6 compared with kNN-MT in the WMT'19 De–En translation task, the domain adaptation tasks in De–En and En–Ja, and the Flores101 multilingual translation task.

In addition, we proposed "detector–corrector", the edit-based automatic post-editing (APE) model, which explains which words are wrong in MT sentences and how to correct them for human post-editors. Experiments on the WMT' 20 English-to-German and English-to-Chinese APE tasks showed that our detector–corrector model provides the editing process and outperforming a black-box sequence-to-sequence APE model and an edit-based model, Levenshtein Transformer.

To summarize, we tackled problems from translation to post-editing that are assumed in the real world translation processes, and confirmed that our subset kNN-MT is effective for domain adaptation and our detector–corrector can present an editing process without degrading the translation quality for post-editing.

5.2 Limitations and Future Work

In this section, we discuss the limitations and future work of this dissertation. We hope to address these issues in the future.

5.2.1 Detection and Correction Performance of Detector–Corrector

The performance of error detection and error correction of detector–corrector is still not enough. Especially, the MCC and F1-BAD scores in the target side tagging are about 50% shown in Table 4.4; the improvements of these scores are one of the challenges in future work. Table 4.4 also shows that data augmentation improves tagging accuracy, so we would like to investigate more effective methods of pseudo-data creation.

In addition, the error correction might be improved by other approaches. As above-mentioned, the performance of erroneous span detection is not enough; thus, the corrector is susceptible to the detection errors because detector–corrector is a cascade model and the detection errors directly propagate to the corrector. To address the issue and make the model more robust, we will attempt to use an end-to-end detector–corrector model, where the detector and corrector are connected as a single model in future work. In that model, the erroneous span detection is predicted as a sub-task, and the predicted tags are regarded as latent variables. This approach might not only mitigate the error propagation from the detector to the corrector but also it allows to marginalize multiple edit paths from the MT output sentences to the post-edited sentences.

5.2.2 Bridging Subset kNN-MT and Detector–Corrector

Integrating the two proposed models, subset kNN-MT and detector–corrector, is one of the future directions. Dinh et al. [29] proposed kNN-QE, which uses kNN-MT to estimate the translation quality. In particular, the approaches that use kNN retrieval may potentially improve the performance of error detection, especially in the out-of-domain without additional training. In addition, kNN-based error detection can work with only parallel corpora; in other words, triplet data, i.e., source, MT output, and PE sentences, is not necessary. Thus, it can use existing resources effectively.

Another aspect of the kNN-based error detection is an improvement of interpretability. Users can see translation examples to understand the reason why the spans are detected as errors. For instance, kNN-based error detection and correction using translation memory will be an improvement of this dissertation. As with the kNN-MT, the issue of computational complexity will be a challenge in the kNN-based error detection and correction; thus, we hope that our subset retrieval reduces the computational cost and makes it more efficient.

5.2.3 Introduction Our Methods to Actual Translation Scene

One of the future work is to incorporate both subset kNN-MT and detector–corrector into the actual translation process and evaluate how much the workload of human translators is reduced.

5.2.4 Applying Our Methods to Large Language Models

Both the subset kNN-MT and the detector–corrector are designed for the encoder-decoder model. Recently, decoder-only models like large language model (LLM) have been successful in various NLP tasks; thus, we would like to apply our methods to such models. In subset kNN-MT, it is necessary to create a sentence datastore from monolingual data. Since the input and output sentences are not explicitly separated in the language model, what we should use for the key vector of the sentence datastore is not trivial. For example, the user prompt could be the query and key vector. However, in a QA task, even if neighboring questions of the given input question can be retrieved, the answer or its related facts are not retrieved.

In detector–corrector, it is necessary to represent tagging and reordering using generation models. Tagging needs constraints to generate the tag sequence which has the same length as the input sentence, and reordering needs constraints to generate the reordered sentence which contains only the input words. These constraints could be realized by using constrained decoding [49, 13].

5.2.5 Extension to Multimodal Models

The proposed method could be extended to use modalities other than text. Subset kNN-MT can be applied to the speech-to-text translation, in which input sentences are given by speech, by building a sentence datastore with the speech vector [102] as the key.

5.2.6 Human-Computer Interaction

Detector–corrector can be combined with interfaces other than keyboard input to reduce the human workload further. For example, a touchscreen can be used for reordering and deletion [47]. By presenting edit candidates with an interface that is suitable for each edit operation, post-editing can be performed more intuitively.

5.2.7 Interpretable Neural Machine Translation

While there are other aspects of the interpretability of NMT, this dissertation focused only on generation based on translation examples and providing the post-editing processes. As long as users of machine translation and reader of translated documents are human, we still need to improve the interpretability of machine translation. There are other problems in the field; for example, the influence of training data and input tokens on the generation [111, 116], understanding of the role of each parameter and layer [27, 115], combination of previous interpretable approaches with neural models [60, 61], employing more interpretable model architectures [48].

In the studies of this dissertation, subset kNN-MT can provide which tokens in the datastore are useful for generating each target token by using interpretable kNN method; however, it is hard to understand what the key and query vectors represent in the feature space. To disentagle the high-dimensional contextualized embeddings, independent component analysis (ICA) might be helpful [123]. ICA has a PCA transformation internally; hence, we can aim for both dimensionality reduction to reduce the computational complexity and improvement of the interpretability of the vector representations.

In the study of detector–corrector, the detector model learned the error detection capabilities by being trained to predict the tags using the middle layer of the XLM-R encoder. We empirically observed that it is not always better to predict in the last layer, but better to tune which layer should be used. This phenomenon has been also observed in the other tasks like cross-lingual word alignment [52, 30, 119]. If we can understand what information each layer of the pretrained model captures, it would be possible to design high performance models.

Acknowledgements

はじめに，本論文の執筆にあたり，奈良先端科学技術大学院大学 (NAIST) 教授 渡辺太郎先生に深く感謝いたします．渡辺先生とお会いしたのは，私が初めて参加した学会である，2019 年の言語処理学会年次大会のことでした．当時まだ学部生だった私の研究発表を最前列で聞いていただき，大変参考になる助言や質問をいただきました．学会中のお昼休みにお話していただいたことも鮮明に覚えています．実は私が博士進学を心に決めたのはそのときのことでした．その後，博士の進学先を考え始めたころにちょうど NAIST に移られた旨を耳にし，迷うことなく渡辺研への進学を志願しました．先生から学んだことはここに書ききれないくらい多く，研究についてはもちろん，研究以外のことまで含めて大変お世話になりました．先生のもとで機械翻訳の研究に取り組めた経験は私の宝です．

同研究室准教授 上垣外英剛先生は着任されてすぐに気さくにお話していただき，何度も研究相談に乗っていただきました．いつもどんなに小さなアイデアでも上垣外先生に相談するとどんどん議論が進み，研究のモチベーションも高まりました．助教 大内啓樹先生はユニークな研究テーマの発想や惹きつけられるプレゼンテーションなど，研究活動において大切なことを示していただきました．また，同じ部屋で日々の雑談や挨拶を通して，肩の力を抜いて作業することができました．

NAIST 教授 中村哲先生には，お忙しい中公聴会にご参加いただきました．中村先生には ACL2023 の開催地トロントにてお話していただき，機械翻訳において直積量子化の技術を活用していることに興味を持っていただいたことを覚えています．

NAIST 教授 Sakti 先生には，着任されてすぐのご多忙な時期にもかかわらず，快く副査を引き受けていただき，大変感謝しております．

研究室秘書の北川裕子さんには，入学してから 3 年半，多岐にわたる事務でお世話になりました．変則的な要望に対しても明るく迅速に対応していただいたことに感謝いたします．

研究室の同期には他愛ない話から進路や研究の相談までしていただき，いつも元気をもらいました．先輩・後輩は皆さん非常に優秀で，また多様性に満ちており，刺激的な研究生活を送ることができました．博士生活を支えていただいた皆さんに感謝いたします．

大学外では，情報通信研究機構 (NICT) 翻訳研がもう一つの研究生活の場でした．副査も引き受けていただきました内山将夫さんには，NAIST に来る前の修士インターンのときからお世話になり，いつも一つ先を見据えた助言をいただきました．内山将夫さん，田中英輝さん，隅田英一郎さんには，NICT にお誘いいただきました．業務面のみならず，潤沢な計算機資源や給与の面も含め，何一つ不自由なくのびのびと研究できる環境を作っていただいたことに感謝いたします．翻訳研のアシスタントの皆さまには日々の研究生活の支援から海外出張のための手続きなど，大変お世話になりました．翻訳研の研究員の皆さまには進捗報告などを通して研究の議論をしていただきました．NICT 翻訳研に所属しながら博士生活を送れたことに感謝いたします．

東京大学 講師 松井勇佑先生には近傍探索やベクトル量子化について，深く相談に乗っていただきました．この博士課程で奥深い近傍探索の世界を知れたこと，機械翻訳 × 近傍探索の研究で ACL に論文が採択されたこと，近傍探索を新たな武器にして活動の幅を広げられたことなど，松井先生との出会いはこの博士課程において重要のものとなりました．先生とご一緒に研究できたことに感謝いたします．

NTT コミュニケーション科学基礎研究所 (CS 研) 永田昌明さんには，博士 1 年のときにインターンに受け入れていただき，その後長期に渡り共同研究していただきました．永田さんの翻訳・機械翻訳に対する考え方は，長年第一線でこの分野に取り組まれてきたからこそのエッセンスが詰まっているように感じました．ご一緒に共同研究できたことに感謝いたします．

修士課程においてご指導いただきました，愛媛大学 教授 二宮崇先生，助教 梶原智之先生，同志社大学 准教授 田村晃裕先生に感謝いたします．二宮先生とは，大学に入学してすぐに LISP の話で打ち解けたのを覚えています．自然言語処理に足を踏み入れるきっかけになったのは二宮先生でした．梶原先生には赴任されてすぐのお忙しい中，研究だけでなく博士進学における不安などの相談にも乗っていただきました．田村先生にはいつも気さくに話しかけていただき，研究活動の楽しさを学びました．また，博士課程でもお世話になった NICT に修士でインターンに行かせていただきましたが，これは田村先生にご提案いただいたことで

実現し，大変貴重な経験を積むことができました．二宮研の先生方やスタッフの皆さま，当時の同期や先輩・後輩の皆さんに感謝いたします．

最後に，ここまで支えていただいた家族と友達に感謝いたします．

Bibliography

[1] R. Aharoni and Y. Goldberg. Unsupervised domain clusters in pretrained language models. In D. Jurafsky, J. Chai, N. Schluter, and J. Tetreault, editors, *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7747–7763, Online, July 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.acl-main.692. URL https://aclanthology.org/2020.acl-main.692.

[2] F. André, A.-M. Kermarrec, and N. Le Scouarnec. Cache locality is not enough: High-performance nearest neighbor search with product quantization fast scan. *Proc. VLDB Endow.*, 9(4):288â299, dec 2015. ISSN 2150-8097. doi: 10.14778/2856318.2856324. URL https://doi.org/10.14778/2856318.2856324.

[3] D. Bahdanau, K. Cho, and Y. Bengio. Neural machine translation by jointly learning to align and translate. In Y. Bengio and Y. LeCun, editors, *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, 2015. URL http://arxiv.org/abs/1409.0473.

[4] S. Bengio, O. Vinyals, N. Jaitly, and N. Shazeer. Scheduled sampling for sequence prediction with recurrent neural networks. In *Proceedings of the 28th International Conference on Neural Information Processing Systems - Volume 1*, NIPS'15, page 1171 – 1179, Cambridge, MA, USA, 2015. MIT Press.

[5] P. Bhattacharyya, R. Chatterjee, M. Freitag, D. Kanojia, M. Negri, and M. Turchi. Findings of the WMT 2022 shared task on automatic post-editing. In P. Koehn, L. Barrault, O. Bojar, F. Bougares, R. Chatterjee, M. R. Costa-jussà, C. Federmann, M. Fishel, A. Fraser, M. Freitag,

Y. Graham, R. Grundkiewicz, P. Guzman, B. Haddow, M. Huck, A. Jimeno Yepes, T. Kocmi, A. Martins, M. Morishita, C. Monz, M. Nagata, T. Nakazawa, M. Negri, A. Névéol, M. Neves, M. Popel, M. Turchi, and M. Zampieri, editors, *Proceedings of the Seventh Conference on Machine Translation (WMT)*, pages 109–117, Abu Dhabi, United Arab Emirates (Hybrid), Dec. 2022. Association for Computational Linguistics. URL https://aclanthology.org/2022.wmt-1.5.

[6] N. Bogoychev and P. Chen. Terminology-aware translation with constrained decoding and large language model prompting. In P. Koehn, B. Haddow, T. Kocmi, and C. Monz, editors, *Proceedings of the Eighth Conference on Machine Translation*, pages 890–896, Singapore, Dec. 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.wmt-1.80. URL https://aclanthology.org/2023.wmt-1.80.

[7] S. Borgeaud, A. Mensch, J. Hoffmann, T. Cai, E. Rutherford, K. Millican, G. B. Van Den Driessche, J.-B. Lespiau, B. Damoc, A. Clark, D. De Las Casas, A. Guy, J. Menick, R. Ring, T. Hennigan, S. Huang, L. Maggiore, C. Jones, A. Cassirer, A. Brock, M. Paganini, G. Irving, O. Vinyals, S. Osindero, K. Simonyan, J. Rae, E. Elsen, and L. Sifre. Improving language models by retrieving from trillions of tokens. In K. Chaudhuri, S. Jegelka, L. Song, C. Szepesvari, G. Niu, and S. Sabato, editors, *Proceedings of the 39th International Conference on Machine Learning*, volume 162 of *Proceedings of Machine Learning Research*, pages 2206–2240. PMLR, 17–23 Jul 2022.

[8] P. F. Brown, J. Cocke, S. A. Della Pietra, V. J. Della Pietra, F. Jelinek, J. D. Lafferty, R. L. Mercer, and P. S. Roossin. A statistical approach to machine translation. *Computational Linguistics*, 16(2):79–85, 1990. URL https://aclanthology.org/J90-2002.

[9] E. Bugliarello and N. Okazaki. Enhancing machine translation with dependency-aware self-attention. In D. Jurafsky, J. Chai, N. Schluter, and J. Tetreault, editors, *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 1618–1627, Online, July

2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.acl-main.147. URL https://aclanthology.org/2020.acl-main.147.

[10] B. Bulte and A. Tezcan. Neural fuzzy repair: Integrating fuzzy matches into neural machine translation. In A. Korhonen, D. Traum, and L. Màrquez, editors, *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 1800–1809, Florence, Italy, July 2019. Association for Computational Linguistics. doi: 10.18653/v1/P19-1175. URL https://aclanthology.org/P19-1175.

[11] R. Chatterjee, C. Federmann, M. Negri, and M. Turchi. Findings of the WMT 2019 shared task on automatic post-editing. In O. Bojar, R. Chatterjee, C. Federmann, M. Fishel, Y. Graham, B. Haddow, M. Huck, A. J. Yepes, P. Koehn, A. Martins, C. Monz, M. Negri, A. Névéol, M. Neves, M. Post, M. Turchi, and K. Verspoor, editors, *Proceedings of the Fourth Conference on Machine Translation (Volume 3: Shared Task Papers, Day 2)*, pages 11–28, Florence, Italy, Aug. 2019. Association for Computational Linguistics. doi: 10.18653/v1/W19-5402. URL https://aclanthology.org/W19-5402.

[12] R. Chatterjee, M. Freitag, M. Negri, and M. Turchi. Findings of the WMT 2020 shared task on automatic post-editing. In L. Barrault, O. Bojar, F. Bougares, R. Chatterjee, M. R. Costa-jussà, C. Federmann, M. Fishel, A. Fraser, Y. Graham, P. Guzman, B. Haddow, M. Huck, A. J. Yepes, P. Koehn, A. Martins, M. Morishita, C. Monz, M. Nagata, T. Nakazawa, and M. Negri, editors, *Proceedings of the Fifth Conference on Machine Translation*, pages 646–659, Online, Nov. 2020. Association for Computational Linguistics. URL https://aclanthology.org/2020.wmt-1.75.

[13] G. Chen, Y. Chen, Y. Wang, and V. O. Li. Lexical-constraint-aware neural machine translation via data augmentation. In C. Bessiere, editor, *Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence, IJCAI-20*, pages 3587–3593. International Joint Conferences on Artificial Intelligence Organization, 7 2020. doi: 10.24963/ijcai.2020/496. URL https://doi.org/10.24963/ijcai.2020/496. Main track.

[14] K. Chen, R. Wang, M. Utiyama, L. Liu, A. Tamura, E. Sumita, and T. Zhao. Neural machine translation with source dependency representation. In M. Palmer, R. Hwa, and S. Riedel, editors, *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 2846–2852, Copenhagen, Denmark, Sept. 2017. Association for Computational Linguistics. doi: 10.18653/v1/D17-1304. URL https://aclanthology.org/D17-1304.

[15] K. Chen, R. Wang, M. Utiyama, and E. Sumita. Neural machine translation with reordering embeddings. In A. Korhonen, D. Traum, and L. Màrquez, editors, *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 1787–1799, Florence, Italy, July 2019. Association for Computational Linguistics. doi: 10.18653/v1/P19-1174. URL https://aclanthology.org/P19-1174.

[16] M. Chen, T. Ge, X. Zhang, F. Wei, and M. Zhou. Improving the efficiency of grammatical error correction with erroneous span detection and correction. In B. Webber, T. Cohn, Y. He, and Y. Liu, editors, *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 7162–7169, Online, Nov. 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.emnlp-main.581. URL https://aclanthology.org/2020.emnlp-main.581.

[17] C. Chu and R. Wang. A survey of domain adaptation for neural machine translation. In E. M. Bender, L. Derczynski, and P. Isabelle, editors, *Proceedings of the 27th International Conference on Computational Linguistics*, pages 1304–1319, Santa Fe, New Mexico, USA, Aug. 2018. Association for Computational Linguistics. URL https://aclanthology.org/C18-1111.

[18] C. Chu, R. Dabre, and S. Kurohashi. An empirical comparison of domain adaptation methods for neural machine translation. In R. Barzilay and M.-Y. Kan, editors, *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 385–391, Vancouver, Canada, July 2017. Association for Computational Linguistics. doi: 10.18653/v1/P17-2061. URL https://aclanthology.org/P17-2061.

[19] A. Conneau, K. Khandelwal, N. Goyal, V. Chaudhary, G. Wenzek, F. Guzmán, E. Grave, M. Ott, L. Zettlemoyer, and V. Stoyanov. Unsupervised cross-lingual representation learning at scale. In D. Jurafsky, J. Chai, N. Schluter, and J. Tetreault, editors, *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 8440–8451, Online, July 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.acl-main.747. URL https://aclanthology.org/2020.acl-main.747.

[20] G. M. Correia and A. F. T. Martins. A simple and effective approach to automatic post-editing with transfer learning. In A. Korhonen, D. Traum, and L. Màrquez, editors, *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 3050–3056, Florence, Italy, July 2019. Association for Computational Linguistics. doi: 10.18653/v1/P19-1292. URL https://aclanthology.org/P19-1292.

[21] Q. Cui, S. Huang, J. Li, X. Geng, Z. Zheng, G. Huang, and J. Chen. Directqe: Direct pretraining for machine translation quality estimation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pages 12719–12727, 2021.

[22] Y. Dai, Z. Zhang, Q. Liu, Q. Cui, W. Li, Y. Du, and T. Xu. Simple and scalable nearest neighbor machine translation. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=uu1GBD9SlLe.

[23] H. Deguchi, A. Tamura, and T. Ninomiya. Dependency-based self-attention for transformer NMT. In R. Mitkov and G. Angelova, editors, *Proceedings of the International Conference on Recent Advances in Natural Language Processing (RANLP 2019)*, pages 239–246, Varna, Bulgaria, Sept. 2019. INCOMA Ltd. doi: 10.26615/978-954-452-056-4_028. URL https://aclanthology.org/R19-1028.

[24] S. Deoghare and P. Bhattacharyya. IIT Bombay's WMT22 automatic post-editing shared task submission. In P. Koehn, L. Barrault, O. Bojar, F. Bougares, R. Chatterjee, M. R. Costa-jussà, C. Federmann, M. Fishel,

A. Fraser, M. Freitag, Y. Graham, R. Grundkiewicz, P. Guzman, B. Haddow, M. Huck, A. Jimeno Yepes, T. Kocmi, A. Martins, M. Morishita, C. Monz, M. Nagata, T. Nakazawa, M. Negri, A. Névéol, M. Neves, M. Popel, M. Turchi, and M. Zampieri, editors, *Proceedings of the Seventh Conference on Machine Translation (WMT)*, pages 682–688, Abu Dhabi, United Arab Emirates (Hybrid), Dec. 2022. Association for Computational Linguistics. URL https://aclanthology.org/2022.wmt-1.67.

[25] S. Deoghare, D. Kanojia, F. Blain, T. Ranasinghe, and P. Bhattacharyya. Quality estimation-assisted automatic post-editing. In H. Bouamor, J. Pino, and K. Bali, editors, *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 1686–1698, Singapore, Dec. 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.findings-emnlp.115. URL https://aclanthology.org/2023.findings-emnlp.115.

[26] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova. BERT: Pre-training of deep bidirectional transformers for language understanding. In J. Burstein, C. Doran, and T. Solorio, editors, *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, June 2019. Association for Computational Linguistics. doi: 10.18653/v1/N19-1423. URL https://aclanthology.org/N19-1423.

[27] K. Dhamdhere, M. Sundararajan, and Q. Yan. How important is a neuron. In *International Conference on Learning Representations*, 2019. URL https://openreview.net/forum?id=SylKoo0cKm.

[28] S. Ding, M. Junczys-Dowmunt, M. Post, and P. Koehn. Levenshtein training for word-level quality estimation. In M.-F. Moens, X. Huang, L. Specia, and S. W.-t. Yih, editors, *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 6724–6733, Online and Punta Cana, Dominican Republic, Nov. 2021. Association for Computational Linguistics. doi: 10.18653/v1/2021.emnlp-main.539. URL https://aclanthology.org/2021.emnlp-main.539.

[29] T. A. Dinh, T. Palzer, and J. Niehues. Quality estimation with k-nearest neighbors and automatic evaluation for model-specific quality estimation, 2024. URL https://arxiv.org/abs/2404.18031.

[30] Z.-Y. Dou and G. Neubig. Word alignment by fine-tuning embeddings on parallel corpora. In P. Merlo, J. Tiedemann, and R. Tsarfaty, editors, *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume*, pages 2112–2128, Online, Apr. 2021. Association for Computational Linguistics. doi: 10.18653/v1/2021.eacl-main.181. URL https://aclanthology.org/2021.eacl-main.181.

[31] C. Dyer, V. Chahuneau, and N. A. Smith. A simple, fast, and effective reparameterization of IBM model 2. In L. Vanderwende, H. Daumé III, and K. Kirchhoff, editors, *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 644–648, Atlanta, Georgia, June 2013. Association for Computational Linguistics. URL https://aclanthology.org/N13-1073.

[32] B. Eikema and W. Aziz. Is MAP decoding all you need? the inadequacy of the mode in neural machine translation. In D. Scott, N. Bel, and C. Zong, editors, *Proceedings of the 28th International Conference on Computational Linguistics*, pages 4506–4520, Barcelona, Spain (Online), Dec. 2020. International Committee on Computational Linguistics. doi: 10.18653/v1/2020.coling-main.398. URL https://aclanthology.org/2020.coling-main.398.

[33] A. Eriguchi, Y. Tsuruoka, and K. Cho. Learning to parse and translate improves neural machine translation. In R. Barzilay and M.-Y. Kan, editors, *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 72–78, Vancouver, Canada, July 2017. Association for Computational Linguistics. doi: 10.18653/v1/P17-2012. URL https://aclanthology.org/P17-2012.

[34] A. Fan, S. Bhosale, H. Schwenk, Z. Ma, A. El-Kishky, S. Goyal, M. Baines, O. Celebi, G. Wenzek, V. Chaudhary, N. Goyal, T. Birch, V. Liptchinsky,

S. Edunov, E. Grave, M. Auli, and A. Joulin. Beyond english-centric multilingual machine translation. *J. Mach. Learn. Res.*, 22(1), jan 2021. ISSN 1532-4435.

[35] F. Feng, Y. Yang, D. Cer, N. Arivazhagan, and W. Wang. Language-agnostic BERT sentence embedding. In S. Muresan, P. Nakov, and A. Villavicencio, editors, *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 878–891, Dublin, Ireland, May 2022. Association for Computational Linguistics. doi: 10.18653/v1/2022.acl-long.62. URL https://aclanthology.org/2022.acl-long.62.

[36] P. Fernandes, A. Farinhas, R. Rei, J. G. C. de Souza, P. Ogayo, G. Neubig, and A. Martins. Quality-aware decoding for neural machine translation. In M. Carpuat, M.-C. de Marneffe, and I. V. Meza Ruiz, editors, *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1396–1412, Seattle, United States, July 2022. Association for Computational Linguistics. doi: 10.18653/v1/2022.naacl-main.100. URL https://aclanthology.org/2022.naacl-main.100.

[37] M. Fomicheva, S. Sun, L. Yankovskaya, F. Blain, F. Guzmán, M. Fishel, N. Aletras, V. Chaudhary, and L. Specia. Unsupervised quality estimation for neural machine translation. *Transactions of the Association for Computational Linguistics*, 8:539–555, 2020. doi: 10.1162/tacl_a_00330. URL https://aclanthology.org/2020.tacl-1.35.

[38] T. Ge, K. He, Q. Ke, and J. Sun. Optimized product quantization. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 36(4):744–755, 2014. doi: 10.1109/TPAMI.2013.240.

[39] J. Gehring, M. Auli, D. Grangier, and Y. Dauphin. A convolutional encoder model for neural machine translation. In R. Barzilay and M.-Y. Kan, editors, *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 123–135,

Vancouver, Canada, July 2017. Association for Computational Linguistics. doi: 10.18653/v1/P17-1012. URL https://aclanthology.org/P17-1012.

[40] N. Goyal, C. Gao, V. Chaudhary, P.-J. Chen, G. Wenzek, D. Ju, S. Krishnan, M. Ranzato, F. Guzmán, and A. Fan. The Flores-101 evaluation benchmark for low-resource and multilingual machine translation. *Transactions of the Association for Computational Linguistics*, 10:522–538, 2022. doi: 10.1162/tacl_a_00474. URL https://aclanthology.org/2022.tacl-1.30.

[41] J. Gu, Z. Lu, H. Li, and V. O. Li. Incorporating copying mechanism in sequence-to-sequence learning. In K. Erk and N. A. Smith, editors, *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1631–1640, Berlin, Germany, Aug. 2016. Association for Computational Linguistics. doi: 10.18653/v1/P16-1154. URL https://aclanthology.org/P16-1154.

[42] J. Gu, Y. Wang, K. Cho, and V. O. K. Li. Search engine guided neural machine translation. *AAAI*, 2018.

[43] J. Gu, C. Wang, and J. Zhao. Levenshtein transformer. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019.

[44] S. Gu and Y. Feng. Investigating catastrophic forgetting during continual training for neural machine translation. In D. Scott, N. Bel, and C. Zong, editors, *Proceedings of the 28th International Conference on Computational Linguistics*, pages 4315–4326, Barcelona, Spain (Online), Dec. 2020. International Committee on Computational Linguistics. doi: 10.18653/v1/2020.coling-main.381. URL https://aclanthology.org/2020.coling-main.381.

[45] N. M. Guerreiro, E. Voita, and A. Martins. Looking for a needle in a haystack: A comprehensive study of hallucinations in neural machine translation. In A. Vlachos and I. Augenstein, editors, *Proceedings of the 17th Conference of the European Chapter of the Association for Computational Linguistics*, pages 1059–1075, Dubrovnik, Croatia, May 2023. Association

for Computational Linguistics. doi: 10.18653/v1/2023.eacl-main.75. URL https://aclanthology.org/2023.eacl-main.75.

[46] J. He, G. Neubig, and T. Berg-Kirkpatrick. Efficient nearest neighbor language models. In M.-F. Moens, X. Huang, L. Specia, and S. W.-t. Yih, editors, *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 5703–5714, Online and Punta Cana, Dominican Republic, Nov. 2021. Association for Computational Linguistics. doi: 10.18653/v1/2021.emnlp-main.461. URL https://aclanthology.org/2021.emnlp-main.461.

[47] N. Herbig, T. Düwel, S. Pal, K. Meladaki, M. Monshizadeh, A. Krüger, and J. van Genabith. MMPE: A Multi-Modal Interface for Post-Editing Machine Translation. In D. Jurafsky, J. Chai, N. Schluter, and J. Tetreault, editors, *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 1691–1702, Online, July 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.acl-main.155. URL https://aclanthology.org/2020.acl-main.155.

[48] J. Hewitt, J. Thickstun, C. Manning, and P. Liang. Backpack language models. In A. Rogers, J. Boyd-Graber, and N. Okazaki, editors, *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 9103–9125, Toronto, Canada, July 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.acl-long.506. URL https://aclanthology.org/2023.acl-long.506.

[49] C. Hokamp and Q. Liu. Lexically constrained decoding for sequence generation using grid beam search. In R. Barzilay and M.-Y. Kan, editors, *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1535–1546, Vancouver, Canada, July 2017. Association for Computational Linguistics. doi: 10.18653/v1/P17-1141. URL https://aclanthology.org/P17-1141.

[50] X. Huang, Y. Liu, H. Luan, J. Xu, and M. Sun. Learning to copy for automatic post-editing. In K. Inui, J. Jiang, V. Ng, and X. Wan, editors, *Proceedings of the 2019 Conference on Empirical Methods in Nat-*

ural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP), pages 6122–6132, Hong Kong, China, Nov. 2019. Association for Computational Linguistics. doi: 10.18653/v1/D19-1634. URL https://aclanthology.org/D19-1634.

[51] X. Huang, X. Lou, F. Zhang, and T. Mei. LUL's WMT22 automatic post-editing shared task submission. In P. Koehn, L. Barrault, O. Bojar, F. Bougares, R. Chatterjee, M. R. Costa-jussà, C. Federmann, M. Fishel, A. Fraser, M. Freitag, Y. Graham, R. Grundkiewicz, P. Guzman, B. Haddow, M. Huck, A. Jimeno Yepes, T. Kocmi, A. Martins, M. Morishita, C. Monz, M. Nagata, T. Nakazawa, M. Negri, A. Névéol, M. Neves, M. Popel, M. Turchi, and M. Zampieri, editors, *Proceedings of the Seventh Conference on Machine Translation (WMT)*, pages 689–693, Abu Dhabi, United Arab Emirates (Hybrid), Dec. 2022. Association for Computational Linguistics. URL https://aclanthology.org/2022.wmt-1.68.

[52] M. Jalili Sabet, P. Dufter, F. Yvon, and H. Schütze. SimAlign: High quality word alignments without parallel training data using static and contextualized embeddings. In T. Cohn, Y. He, and Y. Liu, editors, *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 1627–1643, Online, Nov. 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.findings-emnlp.147. URL https://aclanthology.org/2020.findings-emnlp.147.

[53] H. Jégou, M. Douze, and C. Schmid. Product quantization for nearest neighbor search. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 33(1):117–128, 2011.

[54] L. Jin, J. He, J. May, and X. Ma. Challenges in context-aware neural machine translation. In H. Bouamor, J. Pino, and K. Bali, editors, *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 15246–15263, Singapore, Dec. 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.emnlp-main.943. URL https://aclanthology.org/2023.emnlp-main.943.

[55] J. Johnson, M. Douze, and H. Jégou. Billion-scale similarity search with GPUs. *IEEE Transactions on Big Data*, 7(3):535–547, 2019.

[56] K. S. Jones. A statistical interpretation of term specificity and its application in retrieval. *Journal of documentation*, 1972.

[57] K. S. Jones, S. Walker, and S. E. Robertson. A probabilistic model of information retrieval: development and comparative experiments: Part 2. *Information processing & management*, 36(6):809–840, 2000.

[58] M. Junczys-Dowmunt and R. Grundkiewicz. MS-UEdin submission to the WMT2018 APE shared task: Dual-source transformer for automatic post-editing. In O. Bojar, R. Chatterjee, C. Federmann, M. Fishel, Y. Graham, B. Haddow, M. Huck, A. J. Yepes, P. Koehn, C. Monz, M. Negri, A. Névéol, M. Neves, M. Post, L. Specia, M. Turchi, and K. Verspoor, editors, *Proceedings of the Third Conference on Machine Translation: Shared Task Papers*, pages 822–826, Belgium, Brussels, Oct. 2018. Association for Computational Linguistics. doi: 10.18653/v1/W18-6467. URL https://aclanthology.org/W18-6467.

[59] J. Kasai, J. Cross, M. Ghazvininejad, and J. Gu. Non-autoregressive machine translation with disentangled context transformer. In H. D. III and A. Singh, editors, *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pages 5144–5155. PMLR, 13–18 Jul 2020.

[60] U. Khandelwal, O. Levy, D. Jurafsky, L. Zettlemoyer, and M. Lewis. Generalization through memorization: Nearest neighbor language models. In *International Conference on Learning Representations*, 2020. URL https://openreview.net/forum?id=HklBjCEKvH.

[61] U. Khandelwal, A. Fan, D. Jurafsky, L. Zettlemoyer, and M. Lewis. Nearest neighbor machine translation. In *International Conference on Learning Representations (ICLR)*, 2021. URL https://openreview.net/forum?id=7wCBOfJ8hJM.

[62] H. Kim, H.-Y. Jung, H. Kwon, J.-H. Lee, and S.-H. Na. Predictor-estimator: Neural quality estimation based on target word prediction for machine translation. *ACM Trans. Asian Low-Resour. Lang. Inf. Process.*, 17(1), sep 2017. ISSN 2375-4699. doi: 10.1145/3109480. URL https://doi.org/10.1145/3109480.

[63] H. Kim, J.-H. Lee, and S.-H. Na. Predictor-estimator using multilevel task learning with stack propagation for neural quality estimation. In O. Bojar, C. Buck, R. Chatterjee, C. Federmann, Y. Graham, B. Haddow, M. Huck, A. J. Yepes, P. Koehn, and J. Kreutzer, editors, *Proceedings of the Second Conference on Machine Translation*, pages 562–568, Copenhagen, Denmark, Sept. 2017. Association for Computational Linguistics. doi: 10.18653/v1/W17-4763. URL https://aclanthology.org/W17-4763.

[64] T. Kocmi, R. Bawden, O. Bojar, A. Dvorkovich, C. Federmann, M. Fishel, T. Gowda, Y. Graham, R. Grundkiewicz, B. Haddow, R. Knowles, P. Koehn, C. Monz, M. Morishita, M. Nagata, T. Nakazawa, M. Novák, M. Popel, and M. Popović. Findings of the 2022 conference on machine translation (WMT22). In P. Koehn, L. Barrault, O. Bojar, F. Bougares, R. Chatterjee, M. R. Costa-jussà, C. Federmann, M. Fishel, A. Fraser, M. Freitag, Y. Graham, R. Grundkiewicz, P. Guzman, B. Haddow, M. Huck, A. Jimeno Yepes, T. Kocmi, A. Martins, M. Morishita, C. Monz, M. Nagata, T. Nakazawa, M. Negri, A. Névéol, M. Neves, M. Popel, M. Turchi, and M. Zampieri, editors, *Proceedings of the Seventh Conference on Machine Translation (WMT)*, pages 1–45, Abu Dhabi, United Arab Emirates (Hybrid), Dec. 2022. Association for Computational Linguistics. URL https://aclanthology.org/2022.wmt-1.1.

[65] P. Koehn and R. Knowles. Six challenges for neural machine translation. In T. Luong, A. Birch, G. Neubig, and A. Finch, editors, *Proceedings of the First Workshop on Neural Machine Translation*, pages 28–39, Vancouver, Aug. 2017. Association for Computational Linguistics. doi: 10.18653/v1/W17-3204. URL https://aclanthology.org/W17-3204.

[66] S. Läubli, C. Amrhein, P. Düggelin, B. Gonzalez, A. Zwahlen, and M. Volk.

Post-editing productivity with neural machine translation: An empirical assessment of speed and quality in the banking and finance domain. In M. Forcada, A. Way, B. Haddow, and R. Sennrich, editors, *Proceedings of Machine Translation Summit XVII: Research Track*, pages 267–272, Dublin, Ireland, Aug. 2019. European Association for Machine Translation. URL https://aclanthology.org/W19-6626.

[67] A. Lee, M. Auli, and M. Ranzato. Discriminative reranking for neural machine translation. In C. Zong, F. Xia, W. Li, and R. Navigli, editors, *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 7250–7264, Online, Aug. 2021. Association for Computational Linguistics. doi: 10.18653/v1/2021.acl-long.563. URL https://aclanthology.org/2021.acl-long.563.

[68] J. Lee, W. Lee, J. Shin, B. Jung, Y.-K. Kim, and J.-H. Lee. POSTECH-ETRI's submission to the WMT2020 APE shared task: Automatic post-editing with cross-lingual language model. In L. Barrault, O. Bojar, F. Bougares, R. Chatterjee, M. R. Costa-jussà, C. Federmann, M. Fishel, A. Fraser, Y. Graham, P. Guzman, B. Haddow, M. Huck, A. J. Yepes, P. Koehn, A. Martins, M. Morishita, C. Monz, M. Nagata, T. Nakazawa, and M. Negri, editors, *Proceedings of the Fifth Conference on Machine Translation*, pages 777–782, Online, Nov. 2020. Association for Computational Linguistics. URL https://aclanthology.org/2020.wmt-1.82.

[69] G. Leusch, N. Ueffing, and H. Ney. CDER: Efficient MT evaluation using block movements. In D. McCarthy and S. Wintner, editors, *11th Conference of the European Chapter of the Association for Computational Linguistics*, pages 241–248, Trento, Italy, Apr. 2006. Association for Computational Linguistics. URL https://aclanthology.org/E06-1031.

[70] E. K.-Y. Liu. Low-resource neural machine translation: A case study of Cantonese. In Y. Scherrer, T. Jauhiainen, N. Ljubešić, P. Nakov, J. Tiedemann, and M. Zampieri, editors, *Proceedings of the Ninth Workshop on NLP for Similar Languages, Varieties and Dialects*, pages 28–40, Gyeongju,

Republic of Korea, Oct. 2022. Association for Computational Linguistics. URL https://aclanthology.org/2022.vardial-1.4.

[71] M.-T. Luong and C. Manning. Stanford neural machine translation systems for spoken language domains. In M. Federico, S. Stüker, and J. Niehues, editors, *Proceedings of the 12th International Workshop on Spoken Language Translation: Evaluation Campaign*, pages 76–79, Da Nang, Vietnam, Dec. 3-4 2015. URL https://aclanthology.org/2015.iwslt-evaluation.11.

[72] T. Luong, H. Pham, and C. D. Manning. Effective approaches to attention-based neural machine translation. In L. Màrquez, C. Callison-Burch, and J. Su, editors, *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1412–1421, Lisbon, Portugal, Sept. 2015. Association for Computational Linguistics. doi: 10.18653/v1/D15-1166. URL https://aclanthology.org/D15-1166.

[73] J. Mallinson, A. Severyn, E. Malmi, and G. Garrido. FELIX: Flexible text editing through tagging and insertion. In T. Cohn, Y. He, and Y. Liu, editors, *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 1244–1255, Online, Nov. 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.findings-emnlp.111. URL https://aclanthology.org/2020.findings-emnlp.111.

[74] J. Mallinson, J. Adamek, E. Malmi, and A. Severyn. EdiT5: Semi-autoregressive text editing with t5 warm-start. In Y. Goldberg, Z. Kozareva, and Y. Zhang, editors, *Findings of the Association for Computational Linguistics: EMNLP 2022*, pages 2126–2138, Abu Dhabi, United Arab Emirates, Dec. 2022. Association for Computational Linguistics. doi: 10.18653/v1/2022.findings-emnlp.156. URL https://aclanthology.org/2022.findings-emnlp.156.

[75] E. Malmi, S. Krause, S. Rothe, D. Mirylenka, and A. Severyn. Encode, tag, realize: High-precision text editing. In K. Inui, J. Jiang, V. Ng, and X. Wan, editors, *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 5054–5065, Hong

Kong, China, Nov. 2019. Association for Computational Linguistics. doi: 10.18653/v1/D19-1510. URL https://aclanthology.org/D19-1510.

[76] P. H. Martins, Z. Marinho, and A. F. T. Martins. Chunk-based nearest neighbor machine translation. In Y. Goldberg, Z. Kozareva, and Y. Zhang, editors, *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 4228–4245, Abu Dhabi, United Arab Emirates, Dec. 2022. Association for Computational Linguistics. doi: 10.18653/v1/2022.emnlp-main.284. URL https://aclanthology.org/2022.emnlp-main.284.

[77] S. Maruf, A. F. T. Martins, and G. Haffari. Selective attention for context-aware neural machine translation. In J. Burstein, C. Doran, and T. Solorio, editors, *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 3092–3102, Minneapolis, Minnesota, June 2019. Association for Computational Linguistics. doi: 10.18653/v1/N19-1313. URL https://aclanthology.org/N19-1313.

[78] Y. Matsui, R. Hinami, and S. Satoh. Reconfigurable inverted index. In *ACM International Conference on Multimedia (ACMMM)*, pages 1715–1723, 2018.

[79] Y. Matsui, Y. Imaizumi, N. Miyamoto, and N. Yoshifuji. Arm 4-bit pq: Simd-based acceleration for approximate nearest neighbor search on arm. In *ICASSP 2022 - 2022 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 2080–2084, 2022. doi: 10.1109/ICASSP43922.2022.9746589.

[80] B. W. Matthews. Comparison of the predicted and observed secondary structure of t4 phage lysozyme. *Biochimica et Biophysica Acta (BBA)-Protein Structure*, 405(2):442–451, 1975.

[81] Y. Meng, X. Li, X. Zheng, F. Wu, X. Sun, T. Zhang, and J. Li. Fast nearest neighbor machine translation. In S. Muresan, P. Nakov, and A. Villavicencio, editors, *Findings of the Association for Computational Linguistics: ACL 2022*, pages 555–565, Dublin, Ireland, May 2022. Association for

Computational Linguistics. doi: 10.18653/v1/2022.findings-acl.47. URL https://aclanthology.org/2022.findings-acl.47.

[82] T. Mihaylova and A. F. T. Martins. Scheduled sampling for transformers. In F. Alva-Manchego, E. Choi, and D. Khashabi, editors, *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics: Student Research Workshop*, pages 351–356, Florence, Italy, July 2019. Association for Computational Linguistics. doi: 10.18653/v1/P19-2049. URL https://aclanthology.org/P19-2049.

[83] M. Morishita, K. Chousa, J. Suzuki, and M. Nagata. JParaCrawl v3.0: A large-scale English-Japanese parallel corpus. In N. Calzolari, F. Béchet, P. Blache, K. Choukri, C. Cieri, T. Declerck, S. Goggi, H. Isahara, B. Maegaard, J. Mariani, H. Mazo, J. Odijk, and S. Piperidis, editors, *Proceedings of the Thirteenth Language Resources and Evaluation Conference*, pages 6704–6710, Marseille, France, June 2022. European Language Resources Association. URL https://aclanthology.org/2022.lrec-1.721.

[84] M. Nagao. A framework of a mechanical translation between japanese and english by analogy principle. In *Proc. of the International NATO Symposium on Artificial and Human Intelligence*, pages 173–180, 1984.

[85] T. Nakazawa, M. Yaguchi, K. Uchimoto, M. Utiyama, E. Sumita, S. Kurohashi, and H. Isahara. ASPEC: Asian scientific paper excerpt corpus. In N. Calzolari, K. Choukri, T. Declerck, S. Goggi, M. Grobelnik, B. Maegaard, J. Mariani, H. Mazo, A. Moreno, J. Odijk, and S. Piperidis, editors, *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC'16)*, pages 2204–2208, Portorož, Slovenia, May 2016. European Language Resources Association (ELRA). URL https://aclanthology.org/L16-1350.

[86] M. Negri, M. Turchi, R. Chatterjee, and N. Bertoldi. ESCAPE: a large-scale synthetic corpus for automatic post-editing. In N. Calzolari, K. Choukri, C. Cieri, T. Declerck, S. Goggi, K. Hasida, H. Isahara, B. Maegaard, J. Mariani, H. Mazo, A. Moreno, J. Odijk, S. Piperidis, and T. Tokunaga, editors, *Proceedings of the Eleventh International Conference on*

Language Resources and Evaluation (LREC 2018), Miyazaki, Japan, May 2018. European Language Resources Association (ELRA). URL https://aclanthology.org/L18-1004.

[87] G. Neubig. The Kyoto free translation task. http://www.phontron.com/kftt, 2011.

[88] G. Neubig. Forest-to-string SMT for Asian language translation: NAIST at WAT 2014. In T. Nakazawa, H. Mino, I. Goto, S. Kurohashi, and E. Sumita, editors, *Proceedings of the 1st Workshop on Asian Translation (WAT2014)*, pages 20–25, Tokyo, Japan, Oct. 2014. Workshop on Asian Translation. URL https://aclanthology.org/W14-7002.

[89] K. Omelianchuk, V. Atrasevych, A. Chernodub, and O. Skurzhanskyi. GECToR – grammatical error correction: Tag, not rewrite. In J. Burstein, E. Kochmar, C. Leacock, N. Madnani, I. Pilán, H. Yannakoudakis, and T. Zesch, editors, *Proceedings of the Fifteenth Workshop on Innovative Use of NLP for Building Educational Applications*, pages 163–170, Seattle, WA, USA → Online, July 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.bea-1.16. URL https://aclanthology.org/2020.bea-1.16.

[90] M. Ott, M. Auli, D. Grangier, and M. Ranzato. Analyzing uncertainty in neural machine translation. In J. G. Dy and A. Krause, editors, *Proceedings of the 35th International Conference on Machine Learning, ICML 2018, Stockholmsmässan, Stockholm, Sweden, July 10-15, 2018*, volume 80 of *Proceedings of Machine Learning Research*, pages 3953–3962. PMLR, 2018.

[91] M. Ott, S. Edunov, A. Baevski, A. Fan, S. Gross, N. Ng, D. Grangier, and M. Auli. fairseq: A fast, extensible toolkit for sequence modeling. In W. Ammar, A. Louis, and N. Mostafazadeh, editors, *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics (Demonstrations)*, pages 48–53, Minneapolis, Minnesota, June 2019. Association for Computational Linguistics. doi: 10.18653/v1/N19-4009. URL https://aclanthology.org/N19-4009.

[92] K. Papineni, S. Roukos, T. Ward, and W.-J. Zhu. Bleu: a method for automatic evaluation of machine translation. In P. Isabelle, E. Charniak, and D. Lin, editors, *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, pages 311–318, Philadelphia, Pennsylvania, USA, July 2002. Association for Computational Linguistics. doi: 10.3115/1073083.1073135. URL https://aclanthology.org/P02-1040.

[93] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12: 2825–2830, 2011.

[94] M. Post. A call for clarity in reporting BLEU scores. In O. Bojar, R. Chatterjee, C. Federmann, M. Fishel, Y. Graham, B. Haddow, M. Huck, A. J. Yepes, P. Koehn, C. Monz, M. Negri, A. Névéol, M. Neves, M. Post, L. Specia, M. Turchi, and K. Verspoor, editors, *Proceedings of the Third Conference on Machine Translation: Research Papers*, pages 186–191, Brussels, Belgium, Oct. 2018. Association for Computational Linguistics. doi: 10.18653/v1/W18-6319. URL https://aclanthology.org/W18-6319.

[95] C. Raffel, N. Shazeer, A. Roberts, K. Lee, S. Narang, M. Matena, Y. Zhou, W. Li, and P. J. Liu. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of Machine Learning Research*, 21(140): 1–67, 2020. URL http://jmlr.org/papers/v21/20-074.html.

[96] T. Ranasinghe, C. Orasan, and R. Mitkov. TransQuest: Translation quality estimation with cross-lingual transformers. In D. Scott, N. Bel, and C. Zong, editors, *Proceedings of the 28th International Conference on Computational Linguistics*, pages 5070–5081, Barcelona, Spain (Online), Dec. 2020. International Committee on Computational Linguistics. doi: 10.18653/v1/2020.coling-main.445. URL https://aclanthology.org/2020.coling-main.445.

[97] T. Ranasinghe, C. Orasan, and R. Mitkov. An exploratory analysis of multilingual word-level quality estimation with cross-lingual transformers. In C. Zong, F. Xia, W. Li, and R. Navigli, editors, *Proceedings of the*

59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 2: Short Papers), pages 434–440, Online, Aug. 2021. Association for Computational Linguistics. doi: 10.18653/v1/2021.acl-short.55. URL https://aclanthology.org/2021.acl-short.55.

[98] R. Rei, C. Stewart, A. C. Farinha, and A. Lavie. COMET: A neural framework for MT evaluation. In B. Webber, T. Cohn, Y. He, and Y. Liu, editors, *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 2685–2702, Online, Nov. 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.emnlp-main.213. URL https://aclanthology.org/2020.emnlp-main.213.

[99] R. Rei, J. G. C. de Souza, D. Alves, C. Zerva, A. C. Farinha, T. Glushkova, A. Lavie, L. Coheur, and A. F. T. Martins. COMET-22: Unbabel-IST 2022 submission for the metrics shared task. In P. Koehn, L. Barrault, O. Bojar, F. Bougares, R. Chatterjee, M. R. Costa-jussà, C. Federmann, M. Fishel, A. Fraser, M. Freitag, Y. Graham, R. Grundkiewicz, P. Guzman, B. Haddow, M. Huck, A. Jimeno Yepes, T. Kocmi, A. Martins, M. Morishita, C. Monz, M. Nagata, T. Nakazawa, M. Negri, A. Névéol, M. Neves, M. Popel, M. Turchi, and M. Zampieri, editors, *Proceedings of the Seventh Conference on Machine Translation (WMT)*, pages 578–585, Abu Dhabi, United Arab Emirates (Hybrid), Dec. 2022. Association for Computational Linguistics. URL https://aclanthology.org/2022.wmt-1.52.

[100] R. Rei, M. Treviso, N. M. Guerreiro, C. Zerva, A. C. Farinha, C. Maroti, J. G. C. de Souza, T. Glushkova, D. Alves, L. Coheur, A. Lavie, and A. F. T. Martins. CometKiwi: IST-unbabel 2022 submission for the quality estimation shared task. In P. Koehn, L. Barrault, O. Bojar, F. Bougares, R. Chatterjee, M. R. Costa-jussà, C. Federmann, M. Fishel, A. Fraser, M. Freitag, Y. Graham, R. Grundkiewicz, P. Guzman, B. Haddow, M. Huck, A. Jimeno Yepes, T. Kocmi, A. Martins, M. Morishita, C. Monz, M. Nagata, T. Nakazawa, M. Negri, A. Névéol, M. Neves, M. Popel, M. Turchi, and M. Zampieri, editors, *Proceedings of the Seventh Conference on Machine Translation (WMT)*, pages 634–645, Abu Dhabi, United Arab Emi-

rates (Hybrid), Dec. 2022. Association for Computational Linguistics. URL https://aclanthology.org/2022.wmt-1.60.

[101] N. Reimers and I. Gurevych. Sentence-BERT: Sentence embeddings using Siamese BERT-networks. In K. Inui, J. Jiang, V. Ng, and X. Wan, editors, *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 3982–3992, Hong Kong, China, Nov. 2019. Association for Computational Linguistics. doi: 10.18653/v1/D19-1410. URL https://aclanthology.org/D19-1410.

[102] S. Schneider, A. Baevski, R. Collobert, and M. Auli. wav2vec: Unsupervised pre-training for speech recognition. In G. Kubin and Z. Kacic, editors, *Interspeech 2019, 20th Annual Conference of the International Speech Communication Association, Graz, Austria, 15-19 September 2019*, pages 3465–3469. ISCA, 2019. doi: 10.21437/INTERSPEECH.2019-1873.

[103] H. Schwenk, G. Wenzek, S. Edunov, E. Grave, A. Joulin, and A. Fan. CCMatrix: Mining billions of high-quality parallel sentences on the web. In C. Zong, F. Xia, W. Li, and R. Navigli, editors, *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 6490–6500, Online, Aug. 2021. Association for Computational Linguistics. doi: 10.18653/v1/2021.acl-long.507. URL https://aclanthology.org/2021.acl-long.507.

[104] T. Sellam, D. Das, and A. Parikh. BLEURT: Learning robust metrics for text generation. In D. Jurafsky, J. Chai, N. Schluter, and J. Tetreault, editors, *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7881–7892, Online, July 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.acl-main.704. URL https://aclanthology.org/2020.acl-main.704.

[105] R. Sennrich, B. Haddow, and A. Birch. Improving neural machine translation models with monolingual data. In K. Erk and N. A. Smith, editors, *Proceedings of the 54th Annual Meeting of the Association for Com-*

putational Linguistics (Volume 1: Long Papers), pages 86–96, Berlin, Germany, Aug. 2016. Association for Computational Linguistics. doi: 10.18653/v1/P16-1009. URL https://aclanthology.org/P16-1009.

[106] A. Sharma, P. Gupta, and A. Nelakanti. Adapting neural machine translation for automatic post-editing. In L. Barrault, O. Bojar, F. Bougares, R. Chatterjee, M. R. Costa-jussa, C. Federmann, M. Fishel, A. Fraser, M. Freitag, Y. Graham, R. Grundkiewicz, P. Guzman, B. Haddow, M. Huck, A. J. Yepes, P. Koehn, T. Kocmi, A. Martins, M. Morishita, and C. Monz, editors, *Proceedings of the Sixth Conference on Machine Translation*, pages 315–319, Online, Nov. 2021. Association for Computational Linguistics. URL https://aclanthology.org/2021.wmt-1.35.

[107] M. Snover, B. Dorr, R. Schwartz, L. Micciulla, and J. Makhoul. A study of translation edit rate with targeted human annotation. In *Proceedings of the 7th Conference of the Association for Machine Translation in the Americas: Technical Papers*, pages 223–231, Cambridge, Massachusetts, USA, Aug. 8-12 2006. Association for Machine Translation in the Americas. URL https://aclanthology.org/2006.amta-papers.25.

[108] L. Specia, F. Blain, M. Fomicheva, E. Fonseca, V. Chaudhary, F. Guzmán, and A. F. T. Martins. Findings of the WMT 2020 shared task on quality estimation. In L. Barrault, O. Bojar, F. Bougares, R. Chatterjee, M. R. Costa-jussà, C. Federmann, M. Fishel, A. Fraser, Y. Graham, P. Guzman, B. Haddow, M. Huck, A. J. Yepes, P. Koehn, A. Martins, M. Morishita, C. Monz, M. Nagata, T. Nakazawa, and M. Negri, editors, *Proceedings of the Fifth Conference on Machine Translation*, pages 743–764, Online, Nov. 2020. Association for Computational Linguistics. URL https://aclanthology.org/2020.wmt-1.79.

[109] F. Stahlberg and S. Kumar. Seq2Edits: Sequence transduction using span-level edit operations. In B. Webber, T. Cohn, Y. He, and Y. Liu, editors, *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 5147–5159, Online, Nov. 2020. Associ-

ation for Computational Linguistics. doi: 10.18653/v1/2020.emnlp-main.418. URL https://aclanthology.org/2020.emnlp-main.418.

[110] P. Stanchev, W. Wang, and H. Ney. EED: Extended edit distance measure for machine translation. In O. Bojar, R. Chatterjee, C. Federmann, M. Fishel, Y. Graham, B. Haddow, M. Huck, A. J. Yepes, P. Koehn, A. Martins, C. Monz, M. Negri, A. Névéol, M. Neves, M. Post, M. Turchi, and K. Verspoor, editors, *Proceedings of the Fourth Conference on Machine Translation (Volume 2: Shared Task Papers, Day 1)*, pages 514–520, Florence, Italy, Aug. 2019. Association for Computational Linguistics. doi: 10.18653/v1/W19-5359. URL https://aclanthology.org/W19-5359.

[111] M. Sundararajan, A. Taly, and Q. Yan. Axiomatic attribution for deep networks. In *Proceedings of the 34th International Conference on Machine Learning - Volume 70*, ICML'17, page 3319 – 3328. JMLR.org, 2017.

[112] I. Sutskever, O. Vinyals, and Q. V. Le. Sequence to sequence learning with neural networks. In *Proceedings of the 27th International Conference on Neural Information Processing Systems - Volume 2*, NIPS'14, page 3104 – 3112, Cambridge, MA, USA, 2014. MIT Press.

[113] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin. Attention is all you need. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems 30*, pages 5998–6008. Curran Associates, Inc., 2017.

[114] O. Vinyals, M. Fortunato, and N. Jaitly. Pointer networks. In C. Cortes, N. Lawrence, D. Lee, M. Sugiyama, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 28. Curran Associates, Inc., 2015. URL https://proceedings.neurips.cc/paper/2015/file/29921001f2f04bd3baee84a12e98098f-Paper.pdf.

[115] E. Voita, D. Talbot, F. Moiseev, R. Sennrich, and I. Titov. Analyzing multi-head self-attention: Specialized heads do the heavy lifting, the rest can be pruned. In A. Korhonen, D. Traum, and L. Màrquez, editors, *Proceedings of*

the 57th Annual Meeting of the Association for Computational Linguistics, pages 5797–5808, Florence, Italy, July 2019. Association for Computational Linguistics. doi: 10.18653/v1/P19-1580. URL https://aclanthology.org/P19-1580.

[116] E. Voita, R. Sennrich, and I. Titov. Analyzing the source and target contributions to predictions in neural machine translation. In C. Zong, F. Xia, W. Li, and R. Navigli, editors, *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 1126–1140, Online, Aug. 2021. Association for Computational Linguistics. doi: 10.18653/v1/2021.acl-long.91. URL https://aclanthology.org/2021.acl-long.91.

[117] T.-T. Vu and G. Haffari. Automatic post-editing of machine translation: A neural programmer-interpreter approach. In E. Riloff, D. Chiang, J. Hockenmaier, and J. Tsujii, editors, *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 3048–3053, Brussels, Belgium, Oct.-Nov. 2018. Association for Computational Linguistics. doi: 10.18653/v1/D18-1341. URL https://aclanthology.org/D18-1341.

[118] J. Wang, K. Wang, K. Fan, Y. Zhang, J. Lu, X. Ge, Y. Shi, and Y. Zhao. Alibaba's submission for the WMT 2020 APE shared task: Improving automatic post-editing with pre-trained conditional cross-lingual BERT. In L. Barrault, O. Bojar, F. Bougares, R. Chatterjee, M. R. Costa-jussà, C. Federmann, M. Fishel, A. Fraser, Y. Graham, P. Guzman, B. Haddow, M. Huck, A. J. Yepes, P. Koehn, A. Martins, M. Morishita, C. Monz, M. Nagata, T. Nakazawa, and M. Negri, editors, *Proceedings of the Fifth Conference on Machine Translation*, pages 789–796, Online, Nov. 2020. Association for Computational Linguistics. URL https://aclanthology.org/2020.wmt-1.84.

[119] W. Wang, G. Chen, H. Wang, Y. Han, and Y. Chen. Multilingual sentence transformer as a multilingual word aligner. In Y. Goldberg, Z. Kozareva, and Y. Zhang, editors, *Findings of the Association for Com-*

putational Linguistics: EMNLP 2022, pages 2952–2963, Abu Dhabi, United Arab Emirates, Dec. 2022. Association for Computational Linguistics. doi: 10.18653/v1/2022.findings-emnlp.215. URL https://aclanthology.org/2022.findings-emnlp.215.

[120] J. Wieting, T. Berg-Kirkpatrick, K. Gimpel, and G. Neubig. Beyond BLEU:training neural machine translation with semantic similarity. In A. Korhonen, D. Traum, and L. Màrquez, editors, *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4344–4355, Florence, Italy, July 2019. Association for Computational Linguistics. doi: 10.18653/v1/P19-1427. URL https://aclanthology.org/P19-1427.

[121] Y. Wu, M. Schuster, Z. Chen, Q. V. Le, M. Norouzi, W. Macherey, M. Krikun, Y. Cao, Q. Gao, K. Macherey, J. Klingner, A. Shah, M. Johnson, X. Liu, L. Kaiser, S. Gouws, Y. Kato, T. Kudo, H. Kazawa, K. Stevens, G. Kurian, N. Patil, W. Wang, C. Young, J. Smith, J. Riesa, A. Rudnick, O. Vinyals, G. Corrado, M. Hughes, and J. Dean. Google's neural machine translation system: Bridging the gap between human and machine translation. *CoRR*, abs/1609.08144, 2016. URL http://arxiv.org/abs/1609.08144.

[122] J. Xu, J. Crego, and J. Senellart. Boosting neural machine translation with similar translations. In D. Jurafsky, J. Chai, N. Schluter, and J. Tetreault, editors, *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 1580–1590, Online, July 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.acl-main.144. URL https://aclanthology.org/2020.acl-main.144.

[123] H. Yamagiwa, M. Oyama, and H. Shimodaira. Discovering universal geometry in embeddings with ICA. In H. Bouamor, J. Pino, and K. Bali, editors, *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 4647–4675, Singapore, Dec. 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.emnlp-main.283. URL https://aclanthology.org/2023.emnlp-main.283.

[124] H. Yang, M. Wang, D. Wei, H. Shang, J. Guo, Z. Li, L. Lei, Y. Qin,

S. Tao, S. Sun, and Y. Chen. HW-TSC's participation at WMT 2020 automatic post editing shared task. In L. Barrault, O. Bojar, F. Bougares, R. Chatterjee, M. R. Costa-jussà, C. Federmann, M. Fishel, A. Fraser, Y. Graham, P. Guzman, B. Haddow, M. Huck, A. J. Yepes, P. Koehn, A. Martins, M. Morishita, C. Monz, M. Nagata, T. Nakazawa, and M. Negri, editors, *Proceedings of the Fifth Conference on Machine Translation*, pages 797–802, Online, Nov. 2020. Association for Computational Linguistics. URL https://aclanthology.org/2020.wmt-1.85.

[125] Y. Yang, L. Huang, and M. Ma. Breaking the beam search curse: A study of (re-)scoring methods and stopping criteria for neural machine translation. In E. Riloff, D. Chiang, J. Hockenmaier, and J. Tsujii, editors, *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 3054–3059, Brussels, Belgium, Oct.-Nov. 2018. Association for Computational Linguistics. doi: 10.18653/v1/D18-1342. URL https://aclanthology.org/D18-1342.

[126] Z. Yang, F. Meng, Y. Zhang, E. Li, and J. Zhou. Findings of the WMT 2022 shared task on translation suggestion. In P. Koehn, L. Barrault, O. Bojar, F. Bougares, R. Chatterjee, M. R. Costa-jussà, C. Federmann, M. Fishel, A. Fraser, M. Freitag, Y. Graham, R. Grundkiewicz, P. Guzman, B. Haddow, M. Huck, A. Jimeno Yepes, T. Kocmi, A. Martins, M. Morishita, C. Monz, M. Nagata, T. Nakazawa, M. Negri, A. Névéol, M. Neves, M. Popel, M. Turchi, and M. Zampieri, editors, *Proceedings of the Seventh Conference on Machine Translation (WMT)*, pages 821–829, Abu Dhabi, United Arab Emirates (Hybrid), Dec. 2022. Association for Computational Linguistics. URL https://aclanthology.org/2022.wmt-1.76.

[127] Z. Yang, F. Meng, Y. Zhang, E. Li, and J. Zhou. WeTS: A benchmark for translation suggestion. In Y. Goldberg, Z. Kozareva, and Y. Zhang, editors, *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 5278–5290, Abu Dhabi, United Arab Emirates, Dec. 2022. Association for Computational Linguistics. doi: 10.18653/v1/2022.emnlp-main.353. URL https://aclanthology.org/2022.emnlp-main.353.

[128] G. Yasui, Y. Tsuruoka, and M. Nagata. Using semantic similarity as reward for reinforcement learning in sentence generation. In F. Alva-Manchego, E. Choi, and D. Khashabi, editors, *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics: Student Research Workshop*, pages 400–406, Florence, Italy, July 2019. Association for Computational Linguistics. doi: 10.18653/v1/P19-2056. URL https://aclanthology.org/P19-2056.

[129] J. Zhang, M. Wang, Q. Liu, and J. Zhou. Incorporating word reordering knowledge into attention-based neural machine translation. In R. Barzilay and M.-Y. Kan, editors, *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1524–1534, Vancouver, Canada, July 2017. Association for Computational Linguistics. doi: 10.18653/v1/P17-1140. URL https://aclanthology.org/P17-1140.

[130] J. Zhang, M. Utiyama, E. Sumita, G. Neubig, and S. Nakamura. Guiding neural machine translation with retrieved translation pieces. In M. Walker, H. Ji, and A. Stent, editors, *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 1325–1335, New Orleans, Louisiana, June 2018. Association for Computational Linguistics. doi: 10.18653/v1/N18-1120. URL https://aclanthology.org/N18-1120.

[131] X. Zheng, Z. Zhang, J. Guo, S. Huang, B. Chen, W. Luo, and J. Chen. Adaptive nearest neighbor machine translation. In C. Zong, F. Xia, W. Li, and R. Navigli, editors, *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, pages 368–374, Online, Aug. 2021. Association for Computational Linguistics. doi: 10.18653/v1/2021.acl-short.47. URL https://aclanthology.org/2021.acl-short.47.

Appendices

A Detector–Corrector: Edit-Based Automatic Post Editing for Human Post-Editing – Supplementary Material

A.1 Tools, Models, and Datasets

Tools We implemented all models in FAIRSEQ which is published under the MIT-license.

Models We used the following pre-trained NMT models implemented in FAIRSEQ to create the training data.

- En-De: https://www.quest.dcs.shef.ac.uk/wmt20_files_qe/models_en-de.tar.gz
- En-Zh:https://www.quest.dcs.shef.ac.uk/wmt20_files_qe/models_en-zh.tar.gz

Our models were trained by using NVIDIA A6000 GPU. The training costs, "GPU hours", multiplied by the number of GPUs and computation time, are shown in Table A.1. Note that the translation performance for each model was evaluated with only a single training.

Datasets We evaluated all models using WMT'20 APE datasets published under the Creative Commons Zero v1.0 Universal license. Parallel data of the WMT'19 En-De and En-Zh translation tasks, used in our training data, can be used for research purposes as described in https://www.statmt.org/wmt19/

translation-task.html. In the En-Zh task, we tokenized the test set of the En-Zh APE task using JIEBA[1] to calculate the TER and BLEU scores.

The statistics of the training data are shown in Table A.2.

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

Seq2Seq		**Detector**	
Encoder	XLM-R large	Encoder	XLM-R large
#layers	24	#layers	24
Decoder	Transformer decoder	Decoder	Transformer decoder
#layers	6	#layers	4
Hidden size	1024	Hidden size	1024
FFN hidden size	4096	FFN hidden size	4096
Learning rate	1e-4	Learning rate	3e-5
Batch size	24,000 tokens	Batch size	6,000 tokens
Training steps	60,000	Training steps	40,000
Training cost	24.6 GPU hours	Training cost	8.0 GPU hours
LevT		**Corrector**	
Encoder	XLM-R large	Encoder	XLM-R large
#layers	24	#layers	24
Decoder	Transformer decoder	Decoder	Transformer decoder
#layers	6	#layers	6
Hidden size	1024	Hidden size	1024
FFN hidden size	4096	FFN hidden size	4096
Learning rate	1e-4	Learning rate	1e-4
Batch size	12,000 tokens	Batch size	24,000 tokens
Training steps	60,000	Training steps	60,000
Training cost	12.4 GPU hours	Training cost	29.0 GPU hours

Table A.1: Hyperparameters of the models.

	DAug for detector	
	w/o	w/
(1) APE task data	7,000	7,000
(2) Translation task data	2,000,000	2,000,000
Training data of detector		
Base data: (1)×20 + (2)	2,140,000	4,280,000
Training data of corrector		
Base data: (1)×20 + (2)	2,140,000	4,280,000
+ MT training	4,280,000	8,560,000
+ PE training	4,280,000	8,560,000
+ MT & PE training	6,420,000	12,840,000

Table A.2: Statistics of the training data. In the experiment, to make the difference in data size fair, we trained with the same number of parameter updates without using the number of epochs, i.e., the number of training epochs decreases as the data size increases.

Dataset	Development	Test
WMT'20 En-De APE	1,000	1,000
WMT'20 En-Zh APE	1,000	1,000

Table A.3: Statistics of the development and test sets.

List of Publications

Journals

1. Hiroyuki Deguchi, Taro Watanabe, Yusuke Matsui, Masao Utiyama, Hideki Tanaka, Eiichiro Sumita. "Subset Retrieval Nearest Neighbor Machine Translation", 自然言語処理, Vol.31, No.2, pp.374–406 , 2024 年 6 月.

2. 出口 祥之, 内山 将夫, 田村 晃裕, 二宮 崇, 隅田 英一郎. "ニューラル機械翻訳のためのバイリンガルなサブワード分割", 自然言語処理, Vol.28, No.2, pp.632–650, 2021 年 6 月.

3. 出口 祥之, 田村 晃裕, 二宮 崇. "係り受け構造に基づく Attention の制約を用いた Transformer ニューラル機械翻訳", 自然言語処理, Vol.27, No.3, pp.553–571, 2020 年 9 月.

International Conferences and Workshops

1. Hiroyuki Deguchi, Yusuke Sakai, Hidetaka Kamigaito, Taro Watanabe, Hideki Tanaka, Masao Utiyama. "Centroid-Based Efficient Minimum Bayes Risk Decoding", Findings of the Association for Computational Linguistics: ACL2024 (Findings of ACL2024), pp. 11009–11018, Bangkok, Thailand, August 2024.

2. Hiroyuki Deguchi, Masaaki Nagata, Taro Watanabe. "Detector–Corrector: Edit-Based Automatic Post Editing for Human Post Editing", Proceedings of the 25th Annual Conference of the European Association for Machine Translation (EAMT2024), pp. 191–206, Sheffield, United Kingdom, June 2024.

3. Hiroyuki Deguchi, Kenji Imamura, Yuto Nishida, Yusuke Sakai, Justin Vasselli, Taro Watanabe. "NAIST-NICT WMT' 23 General MT Task Submission", Proceedings of the Eighth Conference on Machine Translation (WMT'23), pp.110–118, Singapore, December 2023.

4. Hiroyuki Deguchi, Taro Watanabe, Yusuke Matsui, Masao Utiyama, Hideki Tanaka, Eiichiro Sumita, "Subset Retrieval Nearest Neighbor Machine Translation", Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers) (ACL2023), pp.174–189, Toronto, Canada, July 2023.

5. Hiroyuki Deguchi, Kenji Imamura, Masahiro Kaneko, Yuto Nishida, Yusuke Sakai, Justin Vasselli, Huy Hien Vu, Taro Watanabe. "NAIST-NICT-TIT WMT22 General MT Task Submission", Proceedings of the Seventh Conference on Machine Translation (WMT'22), pp.244–250, Abu Dhabi, United Arab Emirates (Hybrid), December 2022.

6. Hiroyuki Deguchi, Akihiro Tamura, Takashi Ninomiya. "Synchronous Syntactic Attention for Transformer NMT", Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing: Student Research Workshop (ACL-IJCNLP SRW 2021), pp.348–355, Bangkok, Thailand (Online), August 2021.

7. Hiroyuki Deguchi, Masao Utiyama, Akihiro Tamura, Takashi Ninomiya, Eiichiro Sumita. "Bilingual Subword Segmentation for Neural Machine Translation", Proceedings of the 28th International Conference on Computational Linguistics (COLING 2020), pp.4287 – 4297, Barcelona, Spain (Online), December 2020.

8. Hiroyuki Deguchi, Akihiro Tamura, Takashi Ninomiya. "Dependency-Based Self-Attention for Transformer NMT", Proceedings of International Conference Recent Advances in Natural Language Processing (RANLP 2019), pp.239 – 246, Varna, Bulgaria, September 2019.

Domestic Conferences

1. 出口 祥之, 鴨田 豪, 松下 祐介, 慶田 開, 和賀 正樹, 横井 祥, “柔らかい grep/KWIC に向けて：高速単語列マッチングの埋め込み表現による連続化”, NLP 若手の会 (YANS) 第 19 回シンポジウム (2024), 2024 年 9 月. (デモ賞, リクルート賞)

2. 夏見 昂樹, 出口 祥之, 上垣外 英剛, 渡辺 太郎, “知識蒸留モデルと合意をとる頑健な行列補完を用いた高速な確率的最小ベイズリスク復号法”, NLP 若手の会 (YANS) 第 19 回シンポジウム (2024), 2024 年 9 月.

3. 岩國 巧, 出口 祥之, 上垣外 英剛, 渡辺 太郎, “機械翻訳の評価指標における信頼度の評価”, NLP 若手の会 (YANS) 第 19 回シンポジウム (2024), 2024 年 9 月.

4. 五藤 巧, 出口 祥之, 上垣外 英剛, 渡辺 太郎, “k 近傍事例を用いたニューラルモデルの予測における定量的な解釈”, 情報処理学会研究報告, 自然言語処理研究会, 2024-NL-261 (15), pp.1–9, 2024 年 9 月.

5. 出口 祥之, 渡辺 太郎, 松井 勇佑, 内山 将夫, 田中 英輝, 隅田 英一郎, “サブセット探索を用いた高速な kNN ニューラル機械翻訳”, 第 1 回 AAMT 若手翻訳研究会, 2024 年 3 月. (最優秀賞)

6. 出口 祥之, 坂井 優介, 上垣外 英剛, 渡辺 太郎, “疑似参照訳文ベクトルの重心に基づく高速なニューラル最小ベイズリスク復号”, 言語処理学会 第 30 回年次大会, 2024 年 3 月. (シェルパ・アンド・カンパニー賞)

7. 西田 悠人, 森下 睦, 出口 祥之, 上垣外 英剛, 渡辺 太郎, “kNN 言語モデルは低頻度語の予測に役立つか？”, 言語処理学会 第 30 回年次大会, 2024 年 3 月. (第一著者若手奨励賞)

8. 出口 祥之, 平野 颯, 星野 智紀, 西田 悠人, Justin Vasselli, 渡辺 太郎, “knn-seq: 高速・拡張可能な kNN 機械翻訳フレームワーク”, NLP 若手の会 (YANS) 第 18 回シンポジウム (2023), 2023 年 8 月. (奨励賞)

9. 林 和樹, 出口 祥之, Xincan Feng, 上垣外 英剛, 林 克彦, 渡辺 太郎, “kNN-LM による知識グラフを用いた大規模言語モデルにおける知識の操作”, NLP

若手の会 (YANS) 第18回シンポジウム (2023), 2023年8月. (第一著者奨励賞)

10. 出口 祥之, 渡辺 太郎, 松井 勇佑, 内山 将夫, 田中 英輝, 隅田 英一郎, “近傍文検索を用いたサブセット kNN ニューラル機械翻訳”, 言語処理学会 第29回年次大会, pp.283 – 288, 2023年3月.

11. 芳賀 あかり, 平尾 努, 帖佐 克己, 本多 右京, 出口 祥之, 渡辺 太郎, “画像キャプショニングのための制約語の抽出法”, 言語処理学会 第29回年次大会, pp.2206 – 2210, 2023年3月.

12. 井手 佑翼, 出口 祥之, 五藤 巧, Armin Sarhangzadeh, 渡辺 太郎. “後続文脈の考慮が文法誤り訂正性能にもたらす影響の調査”, 情報処理学会研究報告, 自然言語処理研究会, 2022-NL-253, 2022年9月.

13. 出口 祥之, 田村 晃裕, 二宮 崇. “同期注意制約を与えた依存構造に基づく Transformer NMT”, 言語処理学会 第27回年次大会, pp.1369 – 1374, 2021年3月.

14. 佐々木 拓馬, 田村 晃裕, 出口 祥之, 二宮 崇, 加藤 恒夫. “逆順デコーダを用いた係り受け構造に基づく Transformer ニューラル機械翻訳”, 言語処理学会 第27回年次大会, pp.133 – 137, 2021年3月.

15. 出口 祥之, 内山 将夫, 田村 晃裕, 二宮 崇, 隅田 英一郎. “ニューラル機械翻訳のためのバイリンガルなサブワード分割”, 情報処理学会研究報告, 自然言語処理研究会, 2020-NL-246 (22), pp.1 – 8, 2020年12月. (優秀研究賞)

16. 出口 祥之, 田村 晃裕, 二宮 崇. “同期注意制約を与えた Transformer によるニューラル機械翻訳”, 言語処理学会 第26回年次大会, pp.1459 – 1462, 2020年3月.

17. 出口 祥之, 田村 晃裕, 二宮 崇. “係り受け構造に基づく Attention の制約を用いた NMT”, 言語処理学会 第25回年次大会, pp.13 – 16, 2019年3月.

Awards

1. 2024/09/06 デモ賞, リクルート賞, NLP 若手の会 (YANS) 第19回シンポジウム (2024)
2. 2024/03/22 最優秀賞, 第1回 AAMT 若手翻訳研究会
3. 2024/03/14 シェルパ・アンド・カンパニー賞, 言語処理学会 第30回年次大会
4. 2023/08/31 奨励賞, NLP 若手の会 (YANS) 第18回シンポジウム (2023)
5. 2022/01/14 優秀先端学生賞, 奈良先端科学技術大学院大学 創発的先端人材育成
6. 2020/12/03 優秀研究賞, 情報処理学会 第246回自然言語処理研究会