Subset Retrieval Nearest Neighbor Machine Translation

Hiroyuki Deguchi[1,2] **Taro Watanabe**[1] **Yusuke Matsui**[3]
Masao Utiyama[2] **Hideki Tanaka**[2] **Eiichiro Sumita**[2]
[1]Nara Institute of Science and Technology [3]The University of Tokyo
[2]National Institute of Information and Communications Technology
{deguchi.hiroyuki.db0, taro}@is.naist.jp
matsui@hal.t.u-tokyo.ac.jp
{mutiyama, hideki.tanaka, eiichiro.sumita}@nict.go.jp

Abstract

k-nearest-neighbor machine translation (kNN-MT) (Khandelwal et al., 2021) boosts the translation performance of trained neural machine translation (NMT) models by incorporating example-search into the decoding algorithm. 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 this paper, 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. Our subset kNN-MT achieved a speed-up of up to 132.2 times and an improvement in BLEU score of up to 1.6 compared with kNN-MT in the WMT'19 De-En translation task and the domain adaptation tasks in De-En and En-Ja.

1 Introduction

Neural machine translation (NMT) (Sutskever et al., 2014; Bahdanau et al., 2015; Luong et al., 2015; Wu et al., 2016; Vaswani et al., 2017) has achieved state-of-the-art performance and become the focus of many studies. Recently, kNN-MT (Khandelwal et al., 2021) has been proposed, which addresses the problem of performance degradation 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 performance 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) (Jégou et al., 2011) in subset neighbor search.

Our subset kNN-MT achieved a speed-up of up to 132.2 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 and the domain adaptation tasks in German-to-English and English-to-Japanese with open-domain settings.

2 kNN-MT

kNN-MT (Khandelwal et al., 2021) 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

Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics
Volume 1: Long Papers, pages 174–189

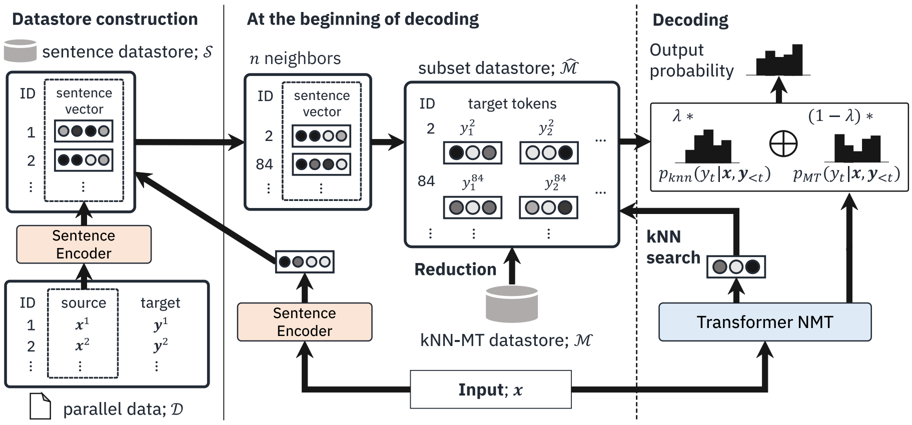


Figure 1: Overview of our subset kNN-MT.

of the cached translation memory.

Datastore Construction A typical NMT model is composed of an encoder that encodes a source sentence $\boldsymbol{x} = (x_1, x_2, \ldots, x_{|\boldsymbol{x}|}) \in \mathcal{V}_X^{|\boldsymbol{x}|}$ and a decoder that generates target tokens $\boldsymbol{y} = (y_1, y_2, \ldots, y_{|\boldsymbol{y}|}) \in \mathcal{V}_Y^{|\boldsymbol{y}|}$ where $|\boldsymbol{x}|$ and $|\boldsymbol{y}|$ are the lengths of sentences $\boldsymbol{x}$ and $\boldsymbol{y}$, respectively, and $\mathcal{V}_X$ and $\mathcal{V}_Y$ are the vocabularies of the source language and target language, respectively. 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 $\boldsymbol{y}_{<t}$. 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 follows:

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

where $\mathcal{D}$ is parallel data and $f : \mathcal{V}_X^{|\boldsymbol{x}|} \times \mathcal{V}_Y^{t-1} \to \mathbb{R}^D$ 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 (Khandelwal et al., 2021), the key is the intermediate representation before it is passed to the final feed-forward network.

Generation 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:

$$\begin{aligned} P(y_t|\boldsymbol{x}, \boldsymbol{y}_{<t}) = \lambda p_{k\mathrm{NN}}(y_t|\boldsymbol{x}, \boldsymbol{y}_{<t}) \\ + (1-\lambda) p_{\mathrm{MT}}(y_t|\boldsymbol{x}, \boldsymbol{y}_{<t}), \end{aligned} \tag{2}$$

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 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:

$$\begin{aligned} & p_{k\mathrm{NN}}(y_t|\boldsymbol{x}, \boldsymbol{y}_{<t}) \\ & \propto \sum_{i=1}^{k} \mathbb{1}_{y_t = v_i} \exp\left(\frac{-\|\boldsymbol{k}_i - f(\boldsymbol{x}, \boldsymbol{y}_{<t})\|_2^2}{\tau}\right), \end{aligned} \tag{3}$$

where τ is the temperature for $p_{k\mathrm{NN}}$, and we set $\tau = 100$. Note that this kNN search is seriously time-consuming[1] (Khandelwal et al., 2021).

3 Proposed Model: Subset kNN-MT

Our *Subset kNN-MT* (Figure 1) drastically accelerates vanilla kNN-MT by reducing the kNN search space by using sentence information (Section 3.1) and efficiently computing the distance between a query and key by performing table lookup (Section 3.2).

3.1 Subset Retrieval

Sentence Datastore Construction In our method, we construct a sentence datastore that stores pairs comprising a source sentence vector

[1]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. Note the kNN search is executed for each timestep in generating a target sentence.

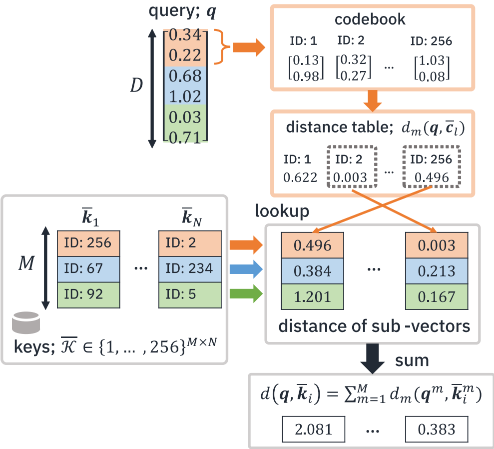


Figure 2: Distance computation using asymmetric distance computation (ADC).

and a target sentence. Concretely, a sentence datastore $\mathcal{S}$ is defined as follows:

$$\mathcal{S} = \{(h(\boldsymbol{x}), \boldsymbol{y}) \mid (\boldsymbol{x}, \boldsymbol{y}) \in \mathcal{D}\}, \tag{4}$$

where $h : \mathcal{V}_X^{|\boldsymbol{x}|} \rightarrow \mathbb{R}^{D'}$ represents a sentence encoder, which is a function that returns a D'-dimensional vector representation of a source sentence.

Decoding At the beginning of decoding, the model retrieves the n-nearest-neighbor sentences of the given input sentence from the sentence datastore $\mathcal{S}$. Let $\hat{\mathcal{S}} \subset \mathcal{S}$ be the subset comprising n-nearest-neighbor sentences. The nearest neighbor search space for target tokens in kNN-MT is then drastically reduced by constructing the datastore corresponding to $\hat{\mathcal{S}}$ as follows:

$$\begin{aligned}\hat{\mathcal{M}} = \{(f(\boldsymbol{x}, \boldsymbol{y}_{<t}), y_t) \mid \\ (h(\boldsymbol{x}), \boldsymbol{y}) \in \hat{\mathcal{S}}, 1 \leq t \leq |\boldsymbol{y}|\},\end{aligned} \tag{5}$$

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.

3.2 Efficient Distance Computation Using Lookup 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) (Jégou et al., 2011) instead of the usual distance computation between floating-point vectors. In ADC, the number of table lookup 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 [2], and this would be hard to load into RAM. To solve this memory problem, product quantization (PQ) (Jégou et al., 2011) 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:

$$\mathcal{C}^m = \{\boldsymbol{c}_1^m, \ldots, \boldsymbol{c}_L^m\},\ \boldsymbol{c}_l^m \in \mathbb{R}^{\frac{D}{M}}. \tag{6}$$

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

$$\bar{\boldsymbol{q}} = [\bar{q}^1, \ldots, \bar{q}^M]^\top \in \{1, \ldots, L\}^M, \tag{7}$$

$$\bar{q}^m = \operatorname*{argmin}_l \|\boldsymbol{q}^m - \boldsymbol{c}_l^m\|_2^2,\ \boldsymbol{q}^m \in \mathbb{R}^{\frac{D}{M}}. \tag{8}$$

[2] 3.2 TiB $\simeq$ 862.6M tokens $\times$ 1024 dimension $\times$ 32 bits (float size)/8 bits (byte size)/1024^4

Asymmetric Distance Computation (ADC) Our method efficiently computes the distance between a query vector and quantized key vectors using ADC (Jégou et al., 2011) (Figure 2). ADC computes the 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, as follows:

$$A_l^m = \|\boldsymbol{q}^m - \boldsymbol{c}_l^m\|_2^2. \tag{9}$$

Second, the distance between a query and each key $d(\boldsymbol{q}, \bar{\boldsymbol{k}}_i)$ is obtained by looking up the distance table as follows:

$$d(\boldsymbol{q}, \bar{\boldsymbol{k}}_i) = \sum_{m=1}^{M} d_m(\boldsymbol{q}^m, \bar{k}_i^m) = \sum_{m=1}^{M} A_{\bar{k}_i^m}^m. \tag{10}$$

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[3] (Equation 9) 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 $\hat{\mathcal{M}}$.

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 (Reimers and Gurevych, 2019). 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.

4 Experiments

4.1 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 (Post, 2018) and COMET (Rei et al., 2020). The speed was evaluated on a single NVIDIA V100 GPU. We varied the batch size settings: either 12,000 tokens (B_∞), to simulate the document 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.

k-Nearest-Neighbor Search In kNN-MT, we set the number of nearest neighbor tokens to $k = 16$. We used FAISS (Johnson et al., 2019) to retrieve the kNN tokens in kNN-MT and for neighbor sentence search in subset kNN-MT. The subset search and ADC were implemented in PyTorch. We use approximate distance computed from quantized keys instead of full-precision keys in Equation 3, following the original kNN-MT (Khandelwal et al., 2021) implementation. The kNN-MT datastore and our sentence datastore used IVF and optimized PQ (OPQ) (Ge et al., 2014). 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

[3] The direct distance computation requires N times calculations according to $\|\boldsymbol{q} - \boldsymbol{k}\|^2$. ADC computes the distance only $L \ll N$ times and just looks-up the table N times.

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 (Feng et al., 2022) 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 (Jones, 1972) and BM25 (Jones et al., 2000) 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 set $k_1 = 2.0, b = 0.75$ in BM25 (Jones et al., 2000). Both TF-IDF and BM25 vectors were normalized by their L2-norm and their dimensionality was reduced to 256-dimensions by singular value decomposition.

4.2 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 kNN-MT baselines (Khandelwal et al., 2021; Meng et al., 2022). We used a trained Transformer big implemented in FAIRSEQ (Ott et al., 2019) as the base MT model. We constructed the datastore from the parallel data of the WMT'19 De-En news translation task with subword lengths of 250 or less and a sentence length ratio of 1.5 or less between the source and target sentences. The datastore contained 862.6M target tokens obtained from 29.5M sentence pairs. The subset size was set to $n = 512$.

Table 1 shows our experimental results. In the table, "tok/s" denotes the number of tokens generated per second. 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 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 summary, this experiment showed that our subset kNN-MT is two orders of magnitude faster than kNN-MT and has the same translation performance.

Model	↑BLEU	↑COMET	↑tok/s	
			B_∞	B_1
Base MT	39.2	84.56	6375.2	129.14
kNN-MT	40.1	84.73	19.6	2.5
Fast kNN-MT	40.3	84.70	286.9	27.1
Ours: Subset kNN-MT				
h: LaBSE	40.1	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 1: Results of translation quality and decoding speed in the WMT'19 De-En translation task. "h:" shows the type of sentence encoder used.

4.3 Domain Adaptation

German-to-English We evaluated subset kNN-MT on out-of-domain translation in the IT, Koran, Law, Medical, and Subtitles domains (Koehn and Knowles, 2017; Aharoni and Goldberg, 2020) 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. The NMT model is the same as that used in Section 4.2 trained from WMT'19 De-En. The subset size was set to $n = 256$, and the batch size was set to 12,000 tokens.

Table 2 shows the results. Compared with base MT, kNN-MT improves the translation performance 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

	IT		Koran		Law		Medical		Subtitles	
Model	BLEU	tok/s	BLEU	tok/s	BLEU	tok/s	BLEU	tok/s	BLEU	tok/s
Base MT	38.7	4433.2	17.1	5295.0	46.1	4294.0	42.1	4392.1	29.4	6310.5
kNN-MT	41.0	22.3	19.5	19.3	52.6	18.6	48.2	19.8	29.6	30.3
Subset kNN-MT										
h: LaBSE	**41.9**	2362.2	**20.1**	2551.3	**53.6**	2258.0	**49.8**	2328.3	29.9	3058.4
h: AvgEnc	**41.9**	2197.8	19.9	2318.4	53.2	1878.8	49.2	2059.9	**30.0**	3113.0
h: TF-IDF	40.0	2289.0	19.3	2489.5	51.4	2264.3	47.5	2326.6	29.3	2574.4
h: BM25	40.0	1582.4	19.1	2089.5	50.8	1946.3	47.4	1835.6	29.4	1567.7

Table 2: Results of out-of-domain translation with open-domain settings. The speed is evaluated with B_∞. **Bold scores** show the best translation performance in each domain. The COMET scores are listed in the appendix due to space limitations.

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. 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 v3 (Morishita et al., 2022) and evaluated its performance on Asian Scientific Paper Excerpt Corpus (ASPEC) (Nakazawa et al., 2016) and Kyoto Free Translation Task (KFTT; created from Wikipedia's Kyoto articles) (Neubig, 2011). The datastore was constructed from parallel data by merging ASPEC, KFTT, and the general domain (JParaCrawl v3). Note that ASPEC contains 3M sentence pairs, but we used only the first 2M pairs for the datastore to remove noisy data, following Neubig (2014). 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 shows the results. These show that kNN-MT improves out-of-domain translation performance 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 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 performance than base MT in exchange for a small slowdown in open-domain settings.

5 Discussion

5.1 Case Study: Effects of Subset Search

Translation examples in the medical domain are shown in Table 4 and the search results of the top-3 nearest neighbor sentences are shown in Table 5. In the table, the subset kNN-MT results are obtained using a LaBSE encoder. Table 4 shows that subset kNN-MT correctly generates the medical term "Co-administration". The results of the nearest neighbor sentence search (Table 5) 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 $k = 256$ 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 6 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".

In summary, the proposed method can retrieve more appropriate words by searching a subset that consists only of neighboring cases.

Model	ASPEC			KFTT		
	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	32.5	88.77	2031.8	25.8	84.11	1436.6
h: AvgEnc	32.4	88.75	1775.6	26.4	84.45	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: Results of out-of-domain translation in English-to-Japanese. 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 contra-indicated.
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 4: Translation examples in the medical domain.

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 5: Top-3 neighbor sentences of our subset kNN-MT in Table 4. “S-” and “T-” denote the top-n neighbor source sentences and their translations, respectively.

5.2 Diversity of Subset Sentences

We hypothesize that the noise introduced by sentence encoders causes the difference in accuracy. In this section, we investigate whether a better sentence encoder would reduce the noise injected into the subset. In particular, 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 ratio of both source and target languages in the subset as the diversity as follows:

$$\frac{\text{number of unique tokens}}{\text{number of subset tokens}}. \tag{11}$$

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 6: Negative log-likelihood (NLL) of the first three tokens and their average in the case of Table 4. Note that a smaller NLL means a larger probability.

Model h	BLEU	unique ratio %	
		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 7: BLEU score and unique token ratio in the subset obtained by each sentence encoder in the medical domain.

Table 7 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. We found that using a model more accurately represents sentence similarity improves the BLEU score. In particular, we evaluated translation quality when noise was injected into the subset by retrieving n sentences from outside the nearest neighbor. Table 8 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, “Bottom

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

Table 8: BLEU score and unique token ratio in the subset obtained by different n-selection methods in the medical domain.

Model h	↑tok/s (B_∞) w/ ADC	w/o ADC
LaBSE	2191.4	446.4 (×0.20)
AvgEnc	1816.8	365.1 (×0.20)
TF-IDF	2199.1	531.0 (×0.24)
BM25	1903.9	471.6 (×0.25)

Table 9: Efficiency of ADC in WMT'19 De-En.

of $2n$" the n furthest sentences of $2n$ neighbor sentences, and "Random of $2n$" n sentences randomly selected from $2n$ 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. These experiments showed that a sentence encoder that calculates similarity appropriately can reduce noise and prevent the degradation of translation performance because the subset consists only of similar sentences.

5.3 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 3.2. The efficiency of ADC in WMT'19 De-En is demonstrated in Table 9. 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(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 (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 $n = 256$ 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 lookup search is slow for a large datastore, it is fast for kNN search when the subset size n is not large (Matsui et al., 2018), e.g., $n = 512$.

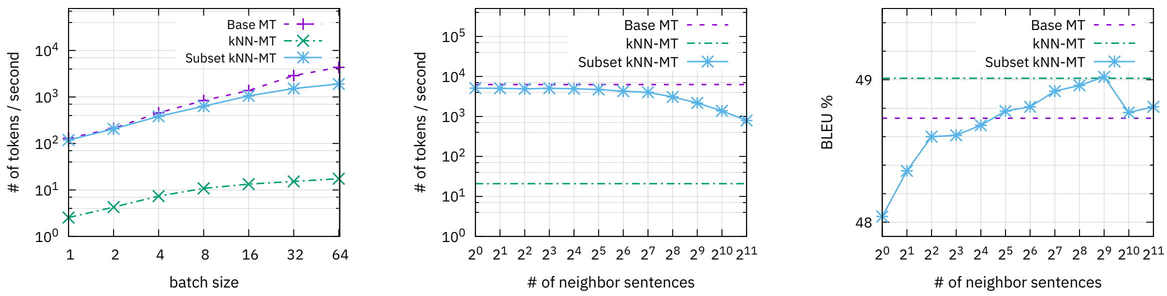
Figure 3(c) shows the results for translation quality on the development set (newstest2018). The results show that a larger n improves BLEU up to $n = 512$, but decreases for greater values of n. In terms of both the translation quality and translation speed, we set $n = 512$ for WMT'19 De-En.

6 Related Work

The first type of example-based machine translation method was analogy-based machine translation (Nagao, 1984). Zhang et al. (2018); Gu et al. (2018) incorporated example-based methods into NMT models, which retrieve examples according to edit distance. Bulte and Tezcan (2019) and Xu et al. (2020) concatenated an input sentence 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 (Meng et al., 2022) 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. (2022) decreased the number of query times by retrieving chunked text; their model led to a speed-up of up to 4 times, compared with kNN-MT. In contrast, subset kNN-MT reduces the search space. Dai et al. (2023) reduced the kNN search space by retrieving the neighbor sentences of the input sentence. They searched for neighboring sentences by BM25 scores with ElasticSearch[4], 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 2 from the distance between the query and the nearest neighbor

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



(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: Translation speed for different batch sizes, and subset sizes and translation quality for different subset sizes in WMT'19 De-En.

key vectors. However, adaptive lambda requires an exact distance and cannot employ datastore quantization and the ADC lookup. To improve the translation performance of kNN-MT, Zheng et al. (2021) computed the weighted average of kNN probabilities $p_{k\text{NN}}$ over multiple values of k. Each weight is predicted by "meta-k network", trained to minimize cross-entropy in the training data. For the other tasks, kNN-LM (Khandelwal et al., 2020), Efficient kNN-LM (He et al., 2021), and RETRO (Borgeaud et al., 2022) used kNN search for language modeling (LM). Our subset search method cannot be applied to LM because the entire input cannot be obtained.

In the field of kNN search, Matsui et al. (2018) 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 lookup in the distance table, and this can be improved by efficient look-up methods that uses SIMD (André et al., 2015; Matsui et al., 2022).

7 Conclusion

In this paper, 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 132.2 times and an improvement in BLEU of up to 1.6 compared with kNN-MT in the WMT'19 De-En translation task and the domain adaptation tasks in De-En and En-Ja. For future work, we would like to apply our method to other tasks.

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 4.2). 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 paper 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.

Ethical Consideration

We construct both kNN-MT and subset kNN-MT datastores from open datasets; therefore, if their datasets have toxic text, kNN-MT and our subset kNN-MT may have the risk of generating toxic contents.

Acknowledgements

This work was partially supported by JSPS KAKENHI Grant Number JP22J1127 and JP22KJ2286.

References

Roee Aharoni and Yoav Goldberg. 2020. Unsupervised domain clusters in pretrained language models. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7747–7763, Online. Association for Computational Linguistics.

Fabien André, Anne-Marie Kermarrec, and Nicolas Le Scouarnec. 2015. Cache locality is not enough: High-performance nearest neighbor search with product quantization fast scan. *Proc. VLDB Endow.*, 9(4):288–299.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. In *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*.

Sebastian Borgeaud, Arthur Mensch, Jordan Hoffmann, Trevor Cai, Eliza Rutherford, Katie Millican, George Bm Van Den Driessche, Jean-Baptiste Lespiau, Bogdan Damoc, Aidan Clark, Diego De Las Casas, Aurelia Guy, Jacob Menick, Roman Ring, Tom Hennigan, Saffron Huang, Loren Maggiore, Chris Jones, Albin Cassirer, Andy Brock, Michela Paganini, Geoffrey Irving, Oriol Vinyals, Simon Osindero, Karen Simonyan, Jack Rae, Erich Elsen, and Laurent Sifre. 2022. Improving language models by retrieving from trillions of tokens. In *Proceedings of the 39th International Conference on Machine Learning*, volume 162 of *Proceedings of Machine Learning Research*, pages 2206–2240. PMLR.

Bram Bulte and Arda Tezcan. 2019. Neural fuzzy repair: Integrating fuzzy matches into neural machine translation. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 1800–1809, Florence, Italy. Association for Computational Linguistics.

Yuhan Dai, Zhirui Zhang, Qiuzhi Liu, Qu Cui, Weihua Li, Yichao Du, and Tong Xu. 2023. Simple and scalable nearest neighbor machine translation. In *The Eleventh International Conference on Learning Representations*.

Fangxiaoyu Feng, Yinfei Yang, Daniel Cer, Naveen Arivazhagan, and Wei Wang. 2022. Language-agnostic BERT sentence embedding. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 878–891, Dublin, Ireland. Association for Computational Linguistics.

Tiezheng Ge, Kaiming He, Qifa Ke, and Jian Sun. 2014. Optimized product quantization. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 36(4):744–755.

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

Junxian He, Graham Neubig, and Taylor Berg-Kirkpatrick. 2021. Efficient nearest neighbor language models. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 5703–5714, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Hervé Jégou, Matthijs Douze, and Cordelia Schmid. 2011. Product quantization for nearest neighbor search. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 33(1):117–128.

Jeff Johnson, Matthijs Douze, and Hervé Jégou. 2019. Billion-scale similarity search with GPUs. *IEEE Transactions on Big Data*, 7(3):535–547.

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

Karen Sparck Jones. 1972. A statistical interpretation of term specificity and its application in retrieval. *Journal of documentation*.

Urvashi Khandelwal, Angela Fan, Dan Jurafsky, Luke Zettlemoyer, and Mike Lewis. 2021. Nearest neighbor machine translation. In *International Conference on Learning Representations (ICLR)*.

Urvashi Khandelwal, Omer Levy, Dan Jurafsky, Luke Zettlemoyer, and Mike Lewis. 2020. Generalization through memorization: Nearest neighbor language models. In *International Conference on Learning Representations*.

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

Thang Luong, Hieu Pham, and Christopher D. Manning. 2015. Effective approaches to attention-based neural machine translation. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1412–1421, Lisbon, Portugal. Association for Computational Linguistics.

Pedro Henrique Martins, Zita Marinho, and André FT Martins. 2022. Chunk-based nearest neighbor machine translation. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 4228–4245, Abu Dhabi, United Arab Emirates.

Yusuke Matsui, Ryota Hinami, and Shin'ichi Satoh. 2018. Reconfigurable inverted index. In *ACM International Conference on Multimedia (ACMMM)*, pages 1715–1723.

Yusuke Matsui, Yoshiki Imaizumi, Naoya Miyamoto, and Naoki Yoshifuji. 2022. 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.

Yuxian Meng, Xiaoya Li, Xiayu Zheng, Fei Wu, Xiaofei Sun, Tianwei Zhang, and Jiwei Li. 2022. Fast nearest neighbor machine translation. In *Findings of the Association for Computational Linguistics: ACL 2022*, pages 555–565, Dublin, Ireland. Association for Computational Linguistics.

Makoto Morishita, Katsuki Chousa, Jun Suzuki, and Masaaki Nagata. 2022. JParaCrawl v3.0: A large-scale English-Japanese parallel corpus. In *Proceedings of the Thirteenth Language Resources and Evaluation Conference*, pages 6704–6710, Marseille, France. European Language Resources Association.

Makoto Nagao. 1984. 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.

Toshiaki Nakazawa, Manabu Yaguchi, Kiyotaka Uchimoto, Masao Utiyama, Eiichiro Sumita, Sadao Kurohashi, and Hitoshi Isahara. 2016. ASPEC: Asian scientific paper excerpt corpus. In *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC'16)*, pages 2204–2208, Portorož, Slovenia. European Language Resources Association (ELRA).

Graham Neubig. 2011. The Kyoto free translation task. http://www.phontron.com/kftt.

Graham Neubig. 2014. Forest-to-string SMT for Asian language translation: NAIST at WAT 2014. In *Proceedings of the 1st Workshop on Asian Translation (WAT2014)*, pages 20–25, Tokyo, Japan. Workshop on Asian Translation.

Myle Ott, Sergey Edunov, Alexei Baevski, Angela Fan, Sam Gross, Nathan Ng, David Grangier, and Michael Auli. 2019. fairseq: A fast, extensible toolkit for sequence modeling. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics (Demonstrations)*, pages 48–53, Minneapolis, Minnesota. Association for Computational Linguistics.

Matt Post. 2018. A call for clarity in reporting BLEU scores. In *Proceedings of the Third Conference on Machine Translation: Research Papers*, pages 186–191, Brussels, Belgium. Association for Computational Linguistics.

Ricardo Rei, Craig Stewart, Ana C Farinha, and Alon Lavie. 2020. COMET: A neural framework for MT evaluation. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 2685–2702, Online. Association for Computational Linguistics.

Nils Reimers and Iryna Gurevych. 2019. Sentence-BERT: Sentence embeddings using Siamese BERT-networks. In *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. Association for Computational Linguistics.

Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. 2014. 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. MIT Press.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In I Guyon, U V Luxburg, S Bengio, H Wallach, R Fergus, S Vishwanathan, and R Garnett, editors, *Advances in Neural Information Processing Systems 30*, pages 5998–6008. Curran Associates, Inc.

Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V. Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, Jeff Klingner, Apurva Shah, Melvin Johnson, Xiaobing Liu, Lukasz Kaiser, Stephan Gouws, Yoshikiyo Kato, Taku Kudo, Hideto Kazawa, Keith Stevens, George Kurian, Nishant Patil, Wei Wang, Cliff Young, Jason Smith, Jason Riesa, Alex Rudnick, Oriol Vinyals, Greg Corrado, Macduff Hughes, and Jeffrey Dean. 2016. Google's neural machine translation system: Bridging the gap between human and machine translation. *CoRR*, abs/1609.08144.

Jitao Xu, Josep Crego, and Jean Senellart. 2020. Boosting neural machine translation with similar translations. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 1580–1590, Online. Association for Computational Linguistics.

Jingyi Zhang, Masao Utiyama, Eiichro Sumita, Graham Neubig, and Satoshi Nakamura. 2018. Guiding neural machine translation with retrieved translation pieces. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 1325–1335, New Orleans, Louisiana. Association for Computational Linguistics.

Xin Zheng, Zhirui Zhang, Junliang Guo, Shujian Huang, Boxing Chen, Weihua Luo, and Jiajun Chen. 2021. Adaptive nearest neighbor machine translation. In *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. Association for Computational Linguistics.

A Datasets, Tools, Models

Datasets Parallel data of the WMT'19 De-En translation task can be used for research purposes as described in https://www.statmt.org/wmt19/translation-task.html. The five domain adaptation datasets in De-En can be used for research purposes as described in the paper (Aharoni and Goldberg, 2020). ASPEC can be used for research purposes as described in https://jipsti.jst.go.jp/aspec/. KFTT is licensed by Creative Commons Attribution-Share-Alike License 3.0.

Tools FAIRSEQ and FAISS are MIT-licensed.

Models We used the following pre-trained NMT models implemented in FAIRSEQ.

- De-En: https://dl.fbaipublicfiles.com/fairseq/models/wmt19.de-en.ffn8192.tar.gz
- En-Ja: http://www.kecl.ntt.co.jp/icl/lirg/jparacrawl/release/3.0/pretrained_models/en-ja/big.tar.gz

The De-En model is included in FAIRSEQ and it is MIT-licensed. The Ja-En model is licensed by Nippon Telegraph and Telephone Corporation (NTT) for research use only as described in http://www.kecl.ntt.co.jp/icl/lirg/jparacrawl/.

We used the pre-trained LaBSE model licensed by Apache-2.0.

B Pseudo Code for ADC lookup

Algorithm 1 shows the pseudo code for the ADC lookup described in Section 3.2. The function COMPUTE_DISTANCES calculates the squared Euclidean distances between a query vector and each quantized key vector by looking up the distance table.

Algorithm 1 ADC lookup

Require:
  query; $\boldsymbol{q} \in \mathbb{R}^D$
  quantized keys; $\bar{\mathcal{K}} = \{\bar{\boldsymbol{k}}_i\}_{i=1}^N \subseteq \{1, \dots, L\}^M$
  codebook; $\mathcal{C} = \{\mathcal{C}^1, \dots, \mathcal{C}^M\}$,
    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(q, K̄, C)
    for m = 1, ..., M do
        for l = 1, ..., L do
            A_l^m ← ||q^m − c_l^m||_2^2
        end for
    end for
    for i = 1, ..., N do
        d_i ← Σ_{m=1}^M A^m_{k̄_i^m}
    end for
    return d
end function
```

$$A_l^m \leftarrow \|\boldsymbol{q}^m - \boldsymbol{c}_l^m\|_2^2, \qquad d_i \leftarrow \sum_{m=1}^M A^m_{\bar{k}_i^m}$$

C Tuning of the Subset Size in Domain Adaptation

Section 5.3 showed that $n = 256$ and 512 are in balance between speed and quality. To tune the subset size n in the domain adaptation task, we evaluated for $n = 256$ and 512 on the development set of each domain, and the choice of n was judged by the averaged BLEU. Table 10 and 11 show the results of the domain adaptation translation on each development set. We tuned the subset size by using LaBSE for the sentence encoder. Finally, we chose $n = 256$ for the German-to-English and $n = 512$ for the English-to-Japanese domain adaptation tasks.

n	IT	Koran	Law	Medical	Subtitles	Avg.
256	40.5	19.7	53.3	48.6	29.5	38.3
512	40.0	19.7	53.4	48.3	29.9	38.1

Table 10: Results of the German-to-English domain adaptation translation on the development set.

D Details of Translation Quality

We evaluated all experiments by BLEU, COMET, and chrF scores.

Table 12, 13, and 14 show the results of the WMT'19 De-En translation task, the domain adaptation task in De-En, and En-Ja, respectively. Note that Table 13 only shows COMET and chrF scores and the BLEU scores are shown in Table 2 due to space limitations.

E Details of kNN Indexes.

The details of the kNN indexes are shown in Table 15.

n	ASPEC	KFTT	Avg.
256	31.7	24.5	28.1
512	32.0	25.5	28.8

Table 11: Results of the English-to-Japanese domain adaptation translation on the development set.

Model	↑BLEU	↑chrF	↑COMET
Base MT	39.2	63.7	84.56
kNN-MT	40.1	64.2	84.73
Fast kNN-MT (Meng et al., 2022)	40.3	64.6	84.70
Ours: Subset kNN-MT			
h: LaBSE	40.1	64.1	84.66
h: AvgEnc	39.9	64.0	84.68
h: TF-IDF	40.0	64.2	84.63
h: BM25	40.0	63.9	84.60

Table 12: Details of translation quality in the WMT'19 De-En translation task. "h:" shows the type of sentence encoder used.

F Domain Adaptation with Closed Domain Settings

We carried out the German-to-English domain adaptation experiments faithful to the original kNN-MT settings. In this experiment, the datastore for each domain was created only from the parallel data of the target domain, assuming a scenario where the domain of the input sentences is known. Note that the general domain data, i.e., the training data of the WMT'19 De-En translation task, is not included in the datastores.

Table 16 shows the German-to-English domain adaptation translation results in closed-domain settings. The original kNN-MT is faster than that of open-domain settings because the datastore is smaller; however, our subset kNN-MT is still 10 times faster than the original kNN-MT.

	IT		Koran		Law		Medical		Subtitles	
Model	COMET	chrF	COMET	chrF	COMET	chrF	COMET	chrF	COMET	chrF
Base MT	83.09	58.9	72.50	40.0	85.79	66.2	83.31	61.6	79.85	48.6
kNN-MT	83.93	60.6	73.33	41.9	86.83	70.4	84.63	65.4	79.98	48.7
Subset kNN-MT										
h: LaBSE	84.17	60.7	73.43	42.3	86.82	70.9	84.60	66.4	79.82	48.7
h: AvgEnc	84.23	60.9	73.40	42.2	86.84	70.7	84.75	66.1	79.83	48.6
h: TF-IDF	81.70	59.2	72.65	41.4	85.96	69.2	83.38	64.6	79.50	48.3
h: BM25	81.16	58.9	72.60	41.3	85.79	68.6	83.17	64.4	79.35	48.1

Table 13: COMET and chrF scores in the German-to-English domain adaptation. BLEU scores are shown in Table 2.

	ASPEC			KFTT		
Model	BLEU	COMET	chrF	BLEU	COMET	chrF
Base MT	26.7	88.55	37.6	20.3	83.52	28.0
kNN-MT	32.8	89.13	41.5	27.8	85.32	33.9
Subset kNN-MT						
h: LaBSE	32.5	88.77	40.6	25.8	84.11	32.0
h: AvgEnc	32.4	88.75	40.5	26.4	84.45	32.1
h: TF-IDF	29.5	88.24	38.5	22.3	82.37	28.6
h: BM25	29.4	88.04	38.4	21.8	82.21	28.2

Table 14: Details of translation quality in the English-to-Japanese domain adaptation.

	kNN-MT	Subset kNN-MT	
	DS; $\mathcal{M}$	Sentence DS; $\mathcal{S}$	DS; $\hat{\mathcal{M}}$
Search Method	IVF	IVF	Linear ADC look-up
Vector Transform	OPQ (Ge et al., 2014)	OPQ (Ge et al., 2014)	PCA: 1024 $\rightarrow$ 256 dim
# of PQ Sub-vectors; M	64	64	64
# of Centroids; N_{list}	131,072	32,768	—
# of Probed Clusters	64 clusters	64 clusters	—
Size of Search Target	$\sum_{\boldsymbol{y}\in\mathcal{D}} \lvert\boldsymbol{y}\rvert$	$\lvert\mathcal{D}\rvert$	$\sum_{(h(\boldsymbol{x}),\boldsymbol{y})\in\hat{\mathcal{S}}} \lvert\boldsymbol{y}\rvert$

Table 15: Details of kNN indexes. "DS" indicates "Datastore".

	IT		Koran		Law		Medical		Subtitles	
Model	BLEU	tok/s	BLEU	tok/s	BLEU	tok/s	BLEU	tok/s	BLEU	tok/s
Base MT	38.7	4433.2	17.1	5295.0	46.1	4294.0	42.1	4392.1	29.4	6310.5
kNN-MT	43.2	143.9	21.6	146.8	54.1	142.2	49.7	144.0	30.9	142.3
Subset kNN-MT										
h: LaBSE	42.8	2232.7	21.2	2737.0	54.5	2175.6	50.2	2287.3	30.5	3554.6
h: AvgEnc	42.6	2423.3	20.7	2754.4	54.1	2259.5	50.0	2348.9	30.3	3569.7
h: TF-IDF	42.1	2464.1	20.7	3426.9	54.0	2137.0	49.8	2526.4	29.8	3916.0
h: BM25	42.7	2519.9	20.4	3370.1	53.8	2152.6	49.8	2510.5	29.9	3723.2

Table 16: Results of out-of-domain translation with closed-domain settings. The speed is evaluated with B_{∞}.

ACL 2023 Responsible NLP Checklist

A For every submission:

☑ A1. Did you describe the limitations of your work?
After Conclusion ("Limitations" section)

☑ A2. Did you discuss any potential risks of your work?
After Limitations ("Ethical Consideration" section)

☑ A3. Do the abstract and introduction summarize the paper's main claims?
Section 1

☑ A4. Have you used AI writing assistants when working on this paper?
We use tools that only assist with language: deepl, grammarly.

B ☑ Did you use or create scientific artifacts?

Section 4

☑ B1. Did you cite the creators of artifacts you used?
Section 4

☑ B2. Did you discuss the license or terms for use and / or distribution of any artifacts?
Appendix (Section A: Dataset, Tools, Models)

☑ B3. Did you discuss if your use of existing artifact(s) was consistent with their intended use, provided that it was specified? For the artifacts you create, do you specify intended use and whether that is compatible with the original access conditions (in particular, derivatives of data accessed for research purposes should not be used outside of research contexts)?
Appendix (Section A: Datasets, Tools, Models)

☒ B4. Did you discuss the steps taken to check whether the data that was collected / used contains any information that names or uniquely identifies individual people or offensive content, and the steps taken to protect / anonymize it?
We noted in the Ethical Consideration section that our used data may contain toxic contents.

☑ B5. Did you provide documentation of the artifacts, e.g., coverage of domains, languages, and linguistic phenomena, demographic groups represented, etc.?
Section 4

☑ B6. Did you report relevant statistics like the number of examples, details of train / test / dev splits, etc. for the data that you used / created? Even for commonly-used benchmark datasets, include the number of examples in train / validation / test splits, as these provide necessary context for a reader to understand experimental results. For example, small differences in accuracy on large test sets may be significant, while on small test sets they may not be.
Section 4

C ☑ Did you run computational experiments?

Section 4 and 5

☑ C1. Did you report the number of parameters in the models used, the total computational budget (e.g., GPU hours), and computing infrastructure used?
Section 4

The Responsible NLP Checklist used at ACL 2023 is adopted from NAACL 2022, with the addition of a question on AI writing assistance.

☑ C2. Did you discuss the experimental setup, including hyperparameter search and best-found hyperparameter values?
Section 4 and 5

☑ C3. Did you report descriptive statistics about your results (e.g., error bars around results, summary statistics from sets of experiments), and is it transparent whether you are reporting the max, mean, etc. or just a single run?
We report the experimental results of just a single run and that is noted in Limitations section.

☑ C4. If you used existing packages (e.g., for preprocessing, for normalization, or for evaluation), did you report the implementation, model, and parameter settings used (e.g., NLTK, Spacy, ROUGE, etc.)?
Section 4

D ☒ Did you use human annotators (e.g., crowdworkers) or research with human participants?
Left blank.

☐ D1. Did you report the full text of instructions given to participants, including e.g., screenshots, disclaimers of any risks to participants or annotators, etc.?
No response.

☐ D2. Did you report information about how you recruited (e.g., crowdsourcing platform, students) and paid participants, and discuss if such payment is adequate given the participants' demographic (e.g., country of residence)?
No response.

☐ D3. Did you discuss whether and how consent was obtained from people whose data you're using/curating? For example, if you collected data via crowdsourcing, did your instructions to crowdworkers explain how the data would be used?
No response.

☐ D4. Was the data collection protocol approved (or determined exempt) by an ethics review board?
No response.

☐ D5. Did you report the basic demographic and geographic characteristics of the annotator population that is the source of the data?
No response.