Advances in Artificial Intelligence and Machine Learning; Research 4 (1) 1943-1958 Received 21-12-2023; Accepted 09-02-2024; Published 16-02-2024

Towards Faster k-Nearest-Neighbor Machine Translation

Xiangyu Shi 22120416@bjtu.edu.cn
School of Computer and Information Technology
Beijing JiaoTong University
BeiJing, No.3 Shangyuancun, China

Yunlong Liang yunlonliang@gmail.com
School of Computer and Information Technology
Beijing JiaoTong University
BeiJing, No.3 Shangyuancun, China

Jinan Xu jaxu@bjtu.edu.cn
School of Computer and Information Technology
Beijing JiaoTong University
BeiJing, No.3 Shangyuancun, China

Yufeng Chen chenyf@bjtu.edu.cn
School of Computer and Information Technology
Beijing JiaoTong University
BeiJing, No.3 Shangyuancun, China

Corresponding Author: Jinan Xu

Abstract

Recent works have proven the effectiveness of k-nearest-neighbor machine translation(a.k.a kNN-MT) approaches to produce remarkable improvement in cross-domain translations. However, these models suffer from heavy retrieve overhead on the entire datastore when decoding each token. We observe that during the decoding phase, about 67% to 84% of tokens are unvaried after searching over the corpus datastore, which means most of the tokens cause futile retrievals and introduce unnecessary computational costs by initiating k-nearest-neighbor searches. We consider this phenomenon is explainable in linguistics and propose a simple yet effective multi-layer perceptron (MLP) network to predict whether a token should be translated jointly by the neural machine translation model and probabilities produced by the kNN or just by the neural model. The results show that our method succeeds in reducing redundant retrieval operations and significantly reduces the overhead of kNN retrievals by up to 53% at the expense of a slight decline in translation quality. Moreover, our method could work together with all existing kNN-MT systems.

Keywords: Neural Machine Translation, K-Nearset Neighbor

Citation: Xiangyu Shi, et al. Towards Faster k-Nearest-Neighbor Machine Translation. Advances in Artificial Intelligence and Machine Learning. 2024;4(1):111.

1. INTRODUCTION

Neural machine translation has achieved great success but also faces huge challenges. Stacked-Transformers [1], based neural machine translation(NMT) have shown promising performance and evolved various methodologies for further improvements. When neural translation models try to learn representations of the semantics on the training corpus and generalize to downstream tasks by classifying representations to some tokens, therefore out-of-domain issues occur. The recent concerned k nearest neighbor machine translation(kNN-MT) [2], proposes a non-parametric method to enhance NMT systems. At each decoding phase kNN-MT retrieves by current latent representation to potential target tokens in a pre-built key-value datastore, it jointly considers the probabilities distribution of each token given by the neural model and the k nearest reference tokens. When using an out-of-domain neural translation model, the datastore could be created from an in-domain corpus, thus kNN-MT significantly rises the performance on cross-domain machine translations. Besides, it could also slightly boost the results of in-domain translations.

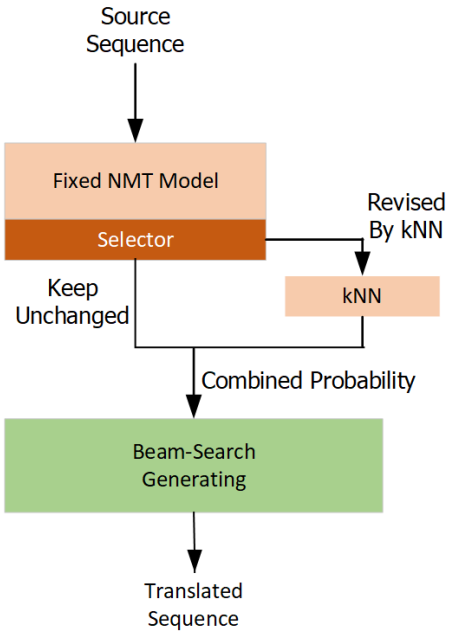

Figure 1: Overview of our method. The selector predicts all the tokens to decide whether to revise the probability distribution. kNN retrievals only occur on parted tokens instead of all tokens, thus we speed up the kNN-MT systems.

However, massive retrievals from entire datastore by high-dimension representations bring non-negligible computational costs to kNN-MT systems. According to the primitive paper of kNN-MT [2], kNN introduces two-orders slower to original NMT models. This overhead limits the applications of kNN-MT systems to some practical scenarios, such as simultaneous translation [3], tasks.

Previous works for accelerating kNN-MT systems suggest shrinking the size of the datastore [4–6], reducing the dimension of representation for retrieval [5, 6]. We observe that about 67%-84% predicted tokens are kept unchanged on its datasets after being revised by the vanilla kNN-MT

Table 1: Redundant ratio tested on multi-domain dataset [7, 8]. The results are measured on test set, and beam searching has been considered in.

	IT	Koran	Law	Medical
Redundant	0.80	0.67	0.84	0.81

system, which means retrievals for these tokens bring redundant computational overhead, as shown in TABLE 1. Towards faster kNN-MT, we proposed a simple yet effective MLP to predict the necessity to retrieve each token. The translation procedure is shown as FIGURE 1. We only add an MLP (so-called selector) after an off-the-shelf NMT model.

We propose our contributions are:

- We observe that most of the tokens do not actually require retrievals. This is our concerned issues of current kNN-MT models, which opens up a new idea for kNN-MT and may inspire valuable academical research.
- We provide a simple yet effective baseline model and public the source codes[1] for this idea, the code may help industrial applications.
- Our method and potential similarity methods could work together with any existing kNN-MT system for further accelerating.

2. BACKGROUND

This section briefly introduces the background of kNN-MT, including its previous works, methodologies, and diverse kNN-MT variants.

2.1 Neural Machine Translation

Currently, neural machine translation is implemented under sequence to sequence [9], frameworks with attention mechanism [10, 11]. Given a source sequence $\mathbf{x} = \{x_1, \cdots, x_n\}$, the NMT model $\mathcal{M}$ translates $\mathbf{x}$ to the target sequence $\mathbf{y} = \{y_1, \cdots, y_m\}$ in another language. At each step of decoding, the NMT model produces the probability distribution over the vocabulary $p_{MT}(\hat{y_i}|\mathbf{x}, \hat{\mathbf{y}}_{1:i-1})$. This probability will be used in beam search for text generation. NMT models trained on particular domains perform deterioration when translating out-domain sentences [7, 12].

2.2 Domain Adaptation for NMT

The most concerned approach to adapt a general domain NMT model to a partial domain is to continue training the neural model on the in-domain corpus. However, this finetuning method demands

[1] https://github.com/sxysxy/Less-Retrieve-KNN-MT

large computational resources and suffers from the notorious catastrophic forgetting issue [13, 14]. Moreover, in real applications scenarios, the domains of translating sentences are rarely known ahead of the time. So multi-domain neural machine translation in one architecture is proposed [15–17].

2.3 Retrieval-Argmented Text Generation

Retrieval-augmented methods are concerned in text generation tasks recently. RetNRef [18], concatenate the representations of a generative LSTM with attention model and the embeddings of retrieved over dialogue history, then use it to generate as usual. Knowledge-intensive question answering is augmented by retrieving similar documents from Wikipedia as contexts [19]. BERT-kNN [20], adds a kNN searching over a large datastore after original BERT model [21], and excels the BERT model on question answering by large margin. BERT-kNN model gives more precise answers than the baseline model and can learn new knowledge from the datastore without training. As for machine translation, Gu et al. [22], propose to retrieve several parallel sentence pairs based on edit distance with source sequence to perform the translation.

The common pattern is to retrieve similar texts or embeddings in a datastore and use retrieving results to enhance generation. Training a tuned neural translation model requires enormous computing power and a parallel corpus. Non-parametric or few-parametric retrieval-based approaches offer opportunities to learn new knowledge and adapt to new domains at a low cost.

2.4 kNN-MT

The basis of kNN-MT is datastore creation and retrieving for probability distributions.

2.4.1 Datastore

kNN-MT first creates a datastore on corpus C, for each parallel sequence pair $(\mathbf{x}, \mathbf{y})$, it adds several key-value pair $\{(f(\mathbf{x}, \mathbf{y}_{1:i-1}), y_i)|i = 1, \cdots, m\}$ into the datastore $\mathcal{D}$, where $f(\mathbf{x}, \mathbf{y}_{1:i-1})$ is the intermediate representation given by the decoder of $\mathcal{M}$. Therefore, the datastore could be created on any parallel corpus without any training step.

2.4.2 Estimation of probability distribution

At decoding time given input $(\mathbf{x}, \hat{\mathbf{y}}_{i-1})$, it makes a query $q = f(\mathbf{x}, \hat{\mathbf{y}}_{i-1})$ and retrieves k nearest neighbor $k_1, \cdot, k_K$ with their corresponding target tokens. Thus, p_{kNN} is estimated by L_2 distances d and a temperature T, normalizing retrieved set into a probability distribution over vocabulary by softmax:

$$p_{KNN}(\hat{y_i}|\mathbf{x}, \hat{\mathbf{y}}_{i-1}) = \text{sofmtax}(\frac{-d(q, k_j)}{T}), j = 1, \cdots, K \tag{1}$$

Temperature T greater than one flattens the distribution and prevents overfitting to the only nearest retrieval [2].

The probability distribution given by the NMT model and kNN is finally interpolated with a hyper-parameter λ:

$$\begin{aligned} p_{combined}(\hat{y}_i|\mathbf{x}, \hat{\mathbf{y}}_{i-1}) = \lambda p_{KNN}(\hat{y}_i|\mathbf{x}, \hat{\mathbf{y}}_{i-1}) \\ + (1-\lambda) p_{MT}(\hat{y}_i|\mathbf{x}, \hat{\mathbf{y}}_{1:i-1}) \end{aligned} \tag{2}$$

To summarize, compared with other methods of cross-domain transfer learning of neural machine translation models, kNN based approach stands out The original method is non-parametric, which means that only more and more in-doman data is needed to work well without any training. In theory, it's possible to update machine translation systems in real time to keep up with the rapid development of The Times. While other effective methods always require heavy training work and hard to update by new data. And derived methods, for example our work and other kNN-MT variants presented later, although the introduction of new neural network structures require training, they typically only require only a few thousands parallel sentence pairs, whereas retraining or fine-tuning a neural machine translation model requires hundreds of thousands or even millions data.

Our work focus on improving the performance of the kNN-MT models with a large datastore. Previous accelerating works consider reducing the dimension of features, and trimming the size of datastore, we do by reducing the number of retrievals, this is a new research idea.

2.5 Adaptive-kNN-MT

Adaptive variant [23], adds a light-weight feed-forward network named *Meta-k* network to dynamically assign weights for probability distributions given by the neural model and k nearest reference sample, the final prediction is obtained by

$$\begin{aligned} p_{combined}(\hat{y}_i|\mathbf{x}, &\hat{\mathbf{y}}_{i-1}) \\ &= p_{\text{Meta}}(f(\mathbf{x}, \mathbf{y}_{1:i-1})) \cdot p_{MT}(\hat{y}_i|\mathbf{x}, \hat{\mathbf{y}}_{1:i-1}) \\ &+ \sum_{j=1}^{K} p_{\text{Meta}}(k_j) \cdot p_{k_i NN}(\hat{y}_i|\mathbf{x}, \hat{\mathbf{y}}_{1:i-1}) \end{aligned}$$

Here p_{Meta} is calculated by a simple feed-forward network with distances and distribution of target tokens of its k nearest neighbors, $p_{k_i NN}$ is also given by Eq. 1. The Adaptive approach introduces almost no inference latency and achieves 1.44 ~ 2.97 BLEU score improvements than the vanilla kNN-MT.

2.6 PCK-kNN-MT

PCK variant [5], extends the Adaptive method and trains a compact network to reduce the dimension of key vectors. It also makes margins between keys with different target tokens more discriminable. Then it prunes the size of the datastore by deleting redundant key-value pairs. It is a typical and effective accelerating method of reducing the query dimension and shrinking the datastore. Compared with our method, which accelerates kNN-MT by eliminating unnecessary retrievals and is simple also highly interpretable.

We will show that our method could work with the Adaptive and the PCK kNN-MT.

2.7 Some Other kNN-MT Variants

- **SK-MT** [24], a distance-ware adapter is introduced to adaptively incorporate retrieval results.
- **Revised Key KNN-MT** [25], a simple feed-forward netowrk is trained to add a revising vector to original query. It improves the retrieving accuarcy and enhances the adaptation.
- **Robust KNN-MT** [26], introducing NMT condfidence to alleviate deterioration casued by noisy key-value pairs.

2.8 kNN-BOX

kNN-BOX [27], assembles various implementations of kNN-MT systems, together with the baseline NMT model. It provides a unified framework for modeling and evaluations. It uses FAISS [28], for efficient high-dimensional vector retrievals implementations. Our implementation and experiments are based on this framework. It is a remarkable fact that kNN-BOX further optimizes the codes and significantly speeds up retrievals, and we do not observe two-orders slower in speed as the original paper reports.

3. METHODOLOGY

We consider that the reason why most tokens do not require kNN retrieval in cross-domain machine translation is that the frequent words always keep unvaried in diverse domains, for example, in English, they may be punctuations and prepositions. Meanwhile, the out-of-domain issue often occurs in nouns and verbs. We count these types of words, and the ratios keep unchanged after kNN retrievals, for example, as shown in FIGURE 2. Thus, the idea is highly explainable.

3.1 Our Mehtod

We train a simple 3-layer MLP network called selector to predict whether each token to retrieve or not. We want the selector directly distinguish the latent representations of in-domain and out-domain semantics. The MLP network contains one hidden layer with $\mathrm{ReLU}(x) = \max(0, x)$ as

the activation function, and a $\mathrm{softmax}$ activation function after the output layer, thus It gives the probabilities for the two options by

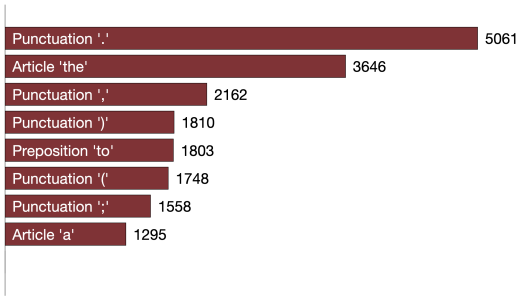

Figure 2: Top-8 tokens which cause futile retrievals on multi-domain IT dataset. These tokens are always unchanged after kNN retrievals.

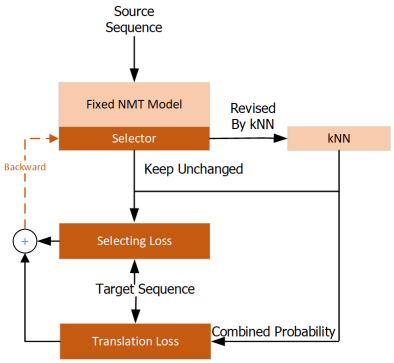

Figure 3: The training procedure of our method.

$$p(\mathcal{A}_i|f(\mathbf{x},\mathbf{y}_{1:i-1})) = \mathrm{softmax}(W_2^T[\mathrm{ReLU}(W_1^T \cdot f(\mathbf{x},\mathbf{y}_{1:i-1}))]) \tag{3}$$

where $W_1 \in \mathcal{R}^{d\times d'}$ and $W_2 \in \mathcal{R}^{d'\times 2}$. d' is the inner hidden dimension of the MLP. $\mathcal{A}_i \in \{0,1\}$ is the decision of whether to retrieve for y_i, 0 is to retrieve and 1 is not.

3.2 Training Procedures

We make labels l for the selector by observing the predictions of the neural translation model $\mathcal{M}$ and the target sequences $\mathbf{y}$. If $\hat{y}_i = \mathcal{M}(f(\mathbf{x},\mathbf{y}_{1:i-1}))$ is equal to y_i, we mark the label of $f(\mathbf{x},\mathbf{y}_{1:i-1})$ to be 1, otherwise 0. Weighted cross-entropy loss is used as the criterion to train the network, the loss is measured by

$$\begin{aligned}\mathcal{L}_1 = \sum_i [-\frac{N}{B}[l_i = 0]\log p(\mathcal{A}_i = 0|f(\mathbf{x},\mathbf{y}_{1:i-1})) \\ -(1-\frac{N}{B})[l_i = 1]\log p(\mathcal{A}_i = 1|f(\mathbf{x},\mathbf{y}_{1:i-1}))]\end{aligned} \tag{4}$$

In Eq. 4 B is the number of the tokens in the training batch, and N is the number of negative samples where $\mathcal{A}_i = 1$

The selector predicts each token and impacts the decoding results, so we also want to train the network by translation loss. For each sample, the selector selects those tokens requiring retrievals then revises their probability distributions by Eq. 2, the rest tokens keep their probability distributions unvaried. The final probability distribution over vocabulary V for $\hat{y}_i$ is:

$$p(\hat{y}_i|\mathbf{x}, \hat{\mathbf{y}}_{1:i-1}) = \begin{cases} p_{MT}(\hat{y}_i|\mathbf{x}, \hat{\mathbf{y}}_{1:i-1}) & \mathcal{A}_i = 0 \\ p_{combined}(\hat{y}_i|\mathbf{x}, \hat{\mathbf{y}}_{1:i-1}) & \mathcal{A}_i = 1 \end{cases} \tag{5}$$

Thus we could train the parameters by translation loss:

$$\mathcal{L}_2 = \sum_i [- \sum_{\hat{y}_i \in V} [y_i == \hat{y}_i] \log p(\hat{y}_i|\mathbf{x}, \hat{\mathbf{y}}_{1:i-1})] \tag{6}$$

And the final loss:

$$\mathcal{L} = \frac{\mathcal{L}_1}{B} + \frac{\mathcal{L}_2}{B} \tag{7}$$

Where B is the total number of tokens in the training batch, the entire training procedure is shown in FIGURE 3.

Another problem is to determine

$$\mathcal{A}_i = \text{argmax}_{\mathcal{A}} p(\mathcal{A}|f(\mathbf{x}, \mathbf{y}_{1:i-1})), \mathcal{A} \in \{0, 1\} \tag{8}$$

Undifferentiable argmax operation is used to address gradient intercept issues, we use the gumbel-softmax [29], trick to approximate a gradient for the argmax operation as follows:

$$G \approx \nabla_W \frac{\exp((\log p(\mathcal{A}|f(\mathbf{x}, \mathbf{y}_{1:i-1}); W)) + g_m(\mathcal{A}'))/\tau)}{\sum_{\mathcal{A}' \in \{0,1\}} \exp((\log p(\mathcal{A}'|f(\mathbf{x}, \mathbf{y}_{1:i-1}); W)) + g_m(\mathcal{A}'))/\tau)}$$

Where W is the inner trainable parameters of the selector, and $g_m = -\log(-\log(u))$ with $u \sim U(0, 1)$, τ is the temperature hyper-parameter.

Finally, the model could be optimized by this loss described in 7 with Adam algorithm [30].

3.3 Metrics

To evaluate our method, we use the following metrics. The time is measured three times and averaged.

- **Sacre-BLEU** [31], is the recommended by WMT to be the evaluating metrics.
- **Inference time**: The total time consumed for the neural model, selector(if exists) and kNN to translate all testing sequences.

Algorithm 1 Integrating the selector into kNN-MT systems

Input: Source sequnece $\mathbf{x}$ and partial translated $\hat{\mathbf{y}}_{1:i-1}$
Parameter: NMT Model $\mathcal{M}$, Selector $\mathcal{S}$, Datastore $\mathcal{D}$, K
Output: $p(\hat{y}_i|\mathbf{x}, \hat{\mathbf{y}}_{1:i-1})$

Let $z = f_{\mathcal{M}}(\mathbf{x}, \hat{\mathbf{y}}_{1:i-1})$
Let $p_{MT}(\hat{y}_i|\mathbf{x}, \hat{\mathbf{y}}_{1:i-1}) = \mathcal{M}(z)$
if $\mathcal{S}(z) == 0$ **then**
    Retrieve K nearest neighbors from $\mathcal{D}$
    Estimated $p_{kNN}(\hat{y}_i|\mathbf{x}, \hat{\mathbf{y}}_{1:i-1})$ by Eq. 1
    Caculate $p_{combined}(\hat{y}_i|\mathbf{x}, \hat{\mathbf{y}}_{1:i-1})$ by Eq. 2
    return return $p_{combined}(\hat{y}_i|\mathbf{x}, \hat{\mathbf{y}}_{1:i-1})$
else
    return $p_{MT}(\hat{y}_i|\mathbf{x}, \hat{\mathbf{y}}_{1:i-1})$
end if

- **KNN overhead time**: A time composed of selector prediction, kNN retrievals and probability distribution revision. It is also a part contained in the total inference time. The reason why we do not only use inference time or tokens per second metrics but also measure this indicator is discussed later.
- **Tokens per second** The average tokens translated per second.
- **Precision** and **Recall**: We dictate that samples requiring retrievals are positive and others are negative. Therefore higher precision index indicates less futile retrievals and is closer to the speed of the pure NMT model. A high recall index means approaching full retrievals on kNN and better translation quality.
- **Ratrieving ratio**: Besides, we introduce a new metric retrieving ratio for the selector, it is the ratio of tokens predicted not to retrieve to the total number of tokens.

3.4 Integration

Our selector could be integrated into any other kNN-MT systems by training the selector and save its weights on vanilla kNN-MT then loading it in any kNN-MT variants. The procedure could be described as pseudo shown in algorithm 1.

Table 2: These values come from the original kNN-MT paper FIGURE 5 [2].

	IT	Koran	Law	Medical
λ	0.7	0.8	0.8	0.8
T	10	100	10	10

Table 3: Time comparison in seconds. For each dataset we measured the inference time(Total) and kNN overhead time(KNN) metrics. To measure the exact time, the global CUDA stream is synchronized before the boundary of the relevant code snippets, slightly reducing operational efficiency. KNN overhead time comprises kNN retrieving, selector predicting, and probability distribution revising. The server equips with an Intel Core i9-12900K CPU and an Nvidia Quadro RTX8000 GPU.

Systems	IT		Koran		Law		Medical		Averge	
	Total	KNN	Total	KNN	Total	KNN	Total	KNN	Total	KNN
Vanilla KNN-MT	18.83	4.04	28.90	5.02	64.53	22.22	39.33	9.43	37.90	10.18
Vanilla + Selector	16.77	2.41 (-40.3%)	27.77	3.74 (-34.2%)	52.6	10.38 (-53.3%)	34.63	5.97 (-36.7%)	32.94	5.63 (-44.7%)

Table 4: Domain adaption performance on the test dataset. For the PCK method, we do not perform a datastore prune to keep the same datastore scale. So our selector remarkably improves translation speed for all these kNN-MT variants. The larger the size of the datastore and the number of neighbors K, there is the more pronounced the advantage of reducing redundant retrievals.

Systems	IT		Koran		Law		Medical		Averge	
	BLEU	Tokens/s	BLEU	Tokens/s	BLEU	Tokens/s	BLEU	Tokens/s	BLEU	Tokens/s
Pure NMT Model	38.35	2171.8	16.26	2199.0	45.48	1922.6	40.06	1919.6	35.04	2053.3
Vanilla, $K = 8$										
Vanilla KNN-MT	45.52	1638.3	20.54	1720.8	61.08	1243.0	53.51	1347.2	45.16	1487.3
Vanilla + Selector	43.90	1843.1	18.55	1820.2	57.18	1509.1	48.70	1540.4	42.08	1678.2
Adaptive, maximum $K = 4$										
Adaptive KNN-MT	47.78	1676.0	20.23	1749.3	63.00	1230.1	56.31	1371.4	46.83	1506.7
Adaptive + Selector	46.12	1857.2	18.64	1847.0	58.71	1489.8	50.26	1619.5	43.43	1703.4
PCK, maximum $K = 4$										
PCK KNN-MT	47.61	1774.5	19.74	1788.0	62.95	1376.0	56.62	1510.1	46.73	1612.2
PCK + Selector	45.77	1891.8	18.67	1869.5	57.96	1552.2	50.38	1583.0	43.20	1724.1

4. EXPERIMENTS

4.1 Hyper-parameters

Adopting the WMT19 En-De model [32], and freezing its parameters during training, We employ the crucial hyper-parameters interpolating index λ and temperature T the same values in the vanilla kNN-MT as TABLE 2.

As for kNN-MT systems with adaptive *Meta-k* network, we set the maximum number of neighbors to 4, and as for others, we form to retrieve fixed 8 nearest neighbors.

We create datastore on the training corpus and train the selector on the valid dataset of each domain as kNN-BOX framework used to do. An Adam optimizer [30], is used for training 100 epochs with an initial learning rate to be 1e-4. Temperature τ for Gumbel-softmax is fixed at 0.1, more details are reported in the appendix.

Table 5: The metrics of the selector.

	Precision	Recall	Retrieving Ratio
IT	0.60	0.81	0.53
Koran	0.70	0.78	0.58
Law	0.53	0.81	0.48
Medical	0.64	0.80	0.49

Table 6: The metrics of the selector trained without translation loss. (+/-*) compares with counterparts in TABLE 5.

	Precision	Recall	Retrieving Ratio
IT	0.68(+0.08)	0.60(-0.21)	0.35(-0.18)
Koran	0.76(+0.06)	0.62(-0.16)	0.42(-0.16)
Law	0.58(+0.05)	0.73(-0.08)	0.39(-0.09)
Medical	0.67(+0.03)	0.77(-0.04)	0.44(-0.05)

4.2 Results

4.2.1 The selector succeeds in discriminating samples that need retrievals.

TABLE 5 shows the classifying quality of the selector. These metrics indicate that the selector selects about 50% tokens to retrieve, and about 80% of them genuinely require retrievals. By contrast, if each sample is randomly predicted, the Recall metric should be only about 50%. According to TABLE 1, only 16%-33% tokens entail retrievals, but the selector performs much higher Precision metrics, The results have proven the selector to be effective. Our approach is independent of both retrieving strategy and the datastore. Experiments only test the kNN overhead time reduction after adding the selector to the Vanilla kNN-MT, but in theory, there should also be speedups close to TABLE 3, for any other kNN-MT systems.

4.2.2 The selector significantly speeds up existing kNN-MT system.

TABLE 3 indicates the selector optmizes 36.7% to 53.3% kNN overhead time. TABLE 4, represents the translation qualities and inference speed. With a selector, all other kNN-MT systems gain accelerated.

4.2.3 Ablation for translation loss.

We also train selectors without translation losses described by Eq. 6, Note that Gumbel-softmax trick is turned off when translation losses are undesired. the results are present in TABLE 6 and TABLE 7. We observe considerably lower translation quality on all the datasets, however, remarkable variations of selector metrics occur on the IT and Koran dataset while slight changes in selector

IT	Koran	Law	Medical
41.53(-2.37)	17.91(-0.64)	54.90(-2.28)	47.59(-1.11)

Table 7: BLEU scores of vanilla *k*NN-MT involving a selector trained without translation loss. (-*) compares with counterparts in TABLE 4, on the row 'Vanilla + Selector'.

metrics occur on Law and Medical dataset. It is unclear how these metrics quantitatively affect translation quality, but we can empirically conclude that translation losses do not, at the very least, lead to model deterioration.

4.2.4 summary

Hyper-parameters T and λ do not affect the efficiency of the operation, and the best values for translation quality have been tested and given in previous related work and shown in TABLE 2, please refer to the references. All experiments can be completed within a few hours on an RTX8000 GPU, and training a selector only takes tens of minutes. The introduction of translation loss significantly improves the performance without causing a significant reduction in training speed. It's possible to introduce a factor to increase or decrease the confidence of the decision given by the selector, thus controlling the trade-off between translation quality and inference speed.

5. DISCUSSION

5.1 Real-World Applications

From the results, the Koran datasets is mostly composed of religious terms, The scarcity of data and the uncommon distribution of language features of this kind of corpus cause high perplexity of the model, we can see the selector performs far worse on koran dataset than other specialized datasets. We conduct that for this case, the upstream machine translation model should be trained on this kind of corpus and then enhanced by our method.

The selector only uses 3000 sentences to train. In practice, since it is learning and classifying the distribution of semantics, adding a small amount of new data to the datastore usually does not require retraining the selector. If necessary, retraining can be done very quickly, which is very beneficial for quickly updating machine translation applications.

5.2 Different Implementations of Baselines From Previous Works

It can be seen from TABLE 3, that *k*NN retrievals only take up a small part of the time in inference time, one reason is that we do not use too large datastore due to hardware limitations, the other reason is that we adopt the KNN-BOX framework, which remarkably optimizes the retrieving performances as the baseline code implementations, rather than comparing the results with the code of the original *k*NN-MT paper. Two-orders slower is reported in that paper, which means *k*NN

retrievals take up almost all the inference time, therefore, based on the code of the original kNN-MT, the improvements of metrics such as inference time or tokens per second can be completely considered as an improvement from their methods, but in our case, we could not do so. Therefore we measure and compare kNN overhead times.

5.3 Comments on Our Method

Although the selector performs pretty, there is still room for improvement. The idealized result is to distinguish all tokens that truly need retrievals and make no futile retrieval. This motivation and mechanism are clear and explainable, rule-based and statistics-based techniques may help.

Reducing undesired retrievals is a promising research direction for fast kNN-MT.

Due to unbalanced positive and negative samples, complex and dense latent representations, it is hard for a simple MLP network to learn strong discrimination, while more complex models take more time.

These efforts have been applied to further improve the selector, but they do not make a difference.

- Use the probability distribution $p_{MT}(\hat{y}_i|\mathbf{x}, \hat{\mathbf{y}}_{1:i-1})$ given by $\mathcal{M}$ as the input of the selector, which greatly extends the number of parameters of the selector(about 40 times) and does not work.
- Add more one hidden layer in the selector.
- Train the selector on the train set trying to enhance generalization, but it fails to improve metrics.
- Use DiceLoss [33], and FocalLoss [34], instead of weighted cross-entropy loss. They cause more fearful overfitting issues, the selector predicts nearly all the tokens to retrieve.

We do not rule out that the last two points are caused by inappropriate hyper-parameters or lack of training ticks.

6. CONCLUSION

This paper opens a new idea for faster kNN-MT. We propose a simple yet effective selector to reduce redundant retrievals. Experiment results on four benchmark datasets show that the selector remarkably speeds up other kNN-MT systems and keeps acceptable translation qualities. The limitation is that it could not outperform the depending kNN-MT system in translation quality. We note that there are methods to dynamically adjust the interpolation coefficient(λ) of the probability distribution between KNN and NMT model based on the token-level confidence produced by the NMT model [35], and we believe it could be further studied whether this confidence can be used to assist the decision of whether to perform retrieval. Besides, vector vector quantization and parallel retrieval techniques on GPU are also important directions for improvement.

References

[1] Vaswani A, Shazeer N, Parmar N, Uszkoreit J, Jones L, et al. Attention Is All You Need. Adv Neural Inf Process Syst. 2017.

[2] Khandelwal U, Fan A, Jurafsky D, Zettlemoyer L, Lewis M, et al. Nearest Neighbor Machine Translation International Conference on Learning Representations. 2021. ArXiv Preprint: https://arxiv.org/pdf/2010.00710.pdf

[3] Gu J, Neubig G, Cho K, Li VOK. Learning to Translate in Real-Time With Neural Machine Translation. In: Proceedings of the 15th conference of the European Chapter of the Association for Computational Linguistics, Long papers. Vol. 1. Spain: Association for Computational Linguistics. 2017:1053-1062.

[4] Meng Y, Li X, Zheng X, Wu F, Sun X, et al. Fast Nearest Neighbor Machine Translation. Findings of the Association for Computational Linguistics. ACL. 2022. ArXiv Preprint: https://arxiv.org/pdf/2105.14528.pdf

[5] DWang D, Fan K, Chen B, Xiong D. Efficient Cluster-Based K-Nearest-Neighbor Machine Translation. In: Proceedings of the 60th annual meeting of the Association for Computational Linguistics. 2022. ArXiv Preprint: https://arxiv.org/pdf/2204.06175.pdf

[6] Martins PH, Marinho Z, Martins AFT. Efficient Machine Translation Domain Adaptation. In: Proceedings of the 1st workshop on semiparametric methods in NLP: decoupling logic from knowledge. Dublin, Ireland and Online Association for Computational Linguistics. 2022:23-29.

[7] Koehn P, Knowles R. Six Challenges for Neural Machine Translation. 2017. ArXiv Preprint: https://arxiv.org/pdf/1706.03872.pdf

[8] Aharoni R, Goldberg Y. Unsupervised Domain Clusters in Pretrained Language Models. In: Proceedings of the 58th annual meeting of the Association for Computational Linguistics. Association for Computational Linguistics. 2022:7747-7763.

[9] Sutskever I, Vinyals O, Le QV. Sequence to Sequence Learning With Neural Networks. Adv Neural Inf Process Syst. 2014;27.

[10] Mnih V, Heess N, Graves A, kavukcuoglu K. Recurrent Models of Visual Attention. Adv Neural Inf Process Syst. 2014;27.

[11] Bahdanau D, Cho K, Bengio Y. Neural Machine Translation by Jointly Learning to Align and Translate. International Conference on Learning Representations. 2015.

[12] Chu C, Wang R. A Survey of Domain Adaptation for Neural Machine Translation. In: Proceedings of the 27th international conference on computational linguistics. 2018:1304-1319.

[13] McCloskey M, Cohen NJ. Catastrophic Interference in Connectionist Networks: The Sequential Learning Problem. Psychol Learn Motiv. 1989;24:109-165.

[14] Santoro A, Bartunov S, Botvinick M, Wierstra D, Lillicrap T, et al. 'One-Shot Learning With Memory-Augmented Neural Networks. 2016. ArXiv Preprint: https://arxiv.org/pdf/1605.06065.pdf.

[15] Farajian MA, Turchi M, Negri M, Federico M. Multi-Domain Neural Machine Translation Through Unsupervised Adaptation. In: Proceedings of the second conference on machine translation. Association for Computational Linguistics. 2017:127-137.

[16] Pham MQ, Crego JM, Yvon F. Revisiting Multi-Domain Machine Translation. Trans Assoc Comp Linguist. 2021;9:17-35.

[17] ZLin Z, Wu L, Wang M, Li L. Learning Language Specific Sub-Network for Multilingual 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 1: Long Papers). 2021:293-305.

[18] Weston J, Dinan E, Miller A. Retrieve and Refine: Improved Sequence Generation Models for Dialogue. In: Proceedings of the 2018 EMNLP workshop SCAI: The 2nd International Workshop on Search-Oriented Conversational AI. Association for Computational Linguistics. 2018:87-92.

[19] Lewis P, Perez E, Piktus A, Petroni F, Karpukhin V, et al. Retrieval-Augmented Generation for Knowledge intensive Nlp Tasks. Adv Neural Inf Process Syst. 2020;33:9459-9474.

[20] Kassner N, Schütze H. BERT-kNN: Adding a KNN Search Component to Pretrained Language Models for Better QA. In: Findings of the Association for Computational Linguistics: EMNLP. Association for Computational Linguistics. 2020:3424-3430.

[21] Devlin J, Chang MW, Lee K, Toutanova K. BERT: Pretraining of Deep Bidirectional Transformers for Language Understanding. In: Proceedings of the 2019 conference of the north American Chapter of the Association for Computational Linguistics: Human Language Technologies (Long and short papers). 2019;1:4171-4186.

[22] Gu J, Wang Y, Cho K, Li VOK. Search Engine Guided Neural Machine Translation. In Proceedings of the AAAI Conference on Artificial Intelligence . 2018;32.

[23] Zheng X, Zhang Z, Guo J, Huang S, Chen B, et al. 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). Association for Computational Linguistics. 2021:368-374.

[24] Dai Y, Zhang Z, Liu Q, Cui Q, Li W, et al, Simple and Scalable Nearest Neighbor Machine Translation. The Eleventh International Conference on Learning Representations, 2023.

[25] Cao Z, Yang B, Lin H, Wu S, Wei X, et al. Bridging the Domain Gaps in Context Representations for K-Nearest Neighbor Neural Machine Translation. In: Proceedings of the 61st annual meeting of the Association for Computational Linguistics (volume 1: long Papers). Canada: Association for Computational Linguistics. 2023:5841-5853,

[26] Jiang H, Lu Z, Meng F, Zhou C, Zhou J, et al. Towards Robust K-Nearest-neighbor Machine Translation. In: Proceedings of the 2022 conference on empirical methods in natural language processing. Abu Dhabi, United Arab Emirates: Association for Computational Linguistics. 2022: 5468-5477.

[27] Zhu W, Zhao Q, Lv Y, Huang S, Zhao S, et al. 'KNN-Box: A Unified Framework for Nearest Neighbor Generation. 2023. ArXiv Preprint: https://arxiv.org/pdf/2302.13574.pdf

[28] Johnson J, Douze M, Jégou H. Billion-Scale Similarity Search With Gpus. IEEE Trans Big Data. 2021;7:535-547.

[29] Jang E, Gu S, Poole B. Categorical Reparameterization With Gumbel-Softmax. 2016. ArXiv Preprint: https://arxiv.org/pdf/1611.01144.pdf

[30] Kingma DP, Ba J. Adam: A Method for Stochastic Optimization. 2014. ArXiv Preprint: https://arxiv.org/pdf/1412.6980.pdf.

[31] Post M. A Call for Clarity in Reporting BLEU Scores. In Proceedings of the Third Conference on Machine Translation: Research Papers, pages Brussels, Belgium. Association for Computational Linguistics. 2018:186–191.

[32] Ng N, Yee K, Baevski A, Ott M, Auli M, et al. Facebook Fair's WMT19 News Translation Task Submission. In: Proceedings of the fourth conference on machine translation (volume 2: shared task papers, Day 1). Florence, Italy: Association for Computational Linguistics. 2019:314-319.

[33] Li X, Sun X, Meng Y, Liang J, Wu F, et al. Dice Loss for Data-Imbalanced Nlp Tasks. 2019. ' ArXiv preprint: https://arxiv.org/pdf/1911.02855.pdf

[34] Lin TY, Goyal P, Girshick R, He K, Dollar P, et al. Focal Loss for Dense Object Detection. Proceedings of the IEEE international conference on computer vision(ICCV). 2017:2980-2988.

[35] Jiang H, Lu Z, Meng F, Zhou C, Zhou J, et al. Towards Robust K-Nearest Neighbor Machine Translation. In: Proceedings of the 2022 conference on empirical methods in natural language processing. Abu Dhabi, United Arab Emirates: Association for Computational Linguistics. 2022: 5468-5477.