Nearest Neighbor Knowledge Distillation for Neural Machine Translation

Anonymous ACL submission

Abstract

k-nearest-neighbor machine translation (kNN-MT), proposed by Khandelwal et al. (2021), has achieved many state-of-the-art results in machine translation tasks. Although effective, kNN-MT requires conducting kNN searches through the large datastore for each decoding step during inference, prohibitively increasing the decoding cost and thus leading to the difficulty for the deployment in real-world applications. In this paper, we propose to move the time-consuming kNN search forward to the preprocessing phase, and then introduce k Nearest Neighbor Knowledge Distillation (kNN-KD) that trains the base NMT model to directly learn the knowledge of kNN. Distilling knowledge retrieved by kNN can encourage the NMT model to take more reasonable target tokens into consideration, thus addressing the overcorrection problem. Extensive experimental results show that, the proposed method achieves consistent improvement over the state-of-the-art baselines including kNN-MT, while maintaining the same training and decoding speed as the standard NMT model[1].

1 Introduction

Neural machine translation (NMT) has shown impressive progress with the prevalence of deep neural networks (Vaswani et al., 2017; Zhang et al., 2019; Chen et al., 2020). Recently, Khandelwal et al. (2021) have proposed k-nearest-neighbor machine translation (kNN-MT) that first stores context representations and target tokens into a large datastore, and then retrieves k possible target tokens by conducting nearest search from the datastore to help with the final next-token decision. The results show that kNN-MT can significantly improve the performance over the base NMT model.

Despite the outstanding performance, kNN-MT will drastically increase the testing runtime since each decoding step needs to conduct kNN searches (Meng et al., 2021). How to speed up the decoding of kNN-MT without degrading performance still remains an open question. Several recent works (Meng et al., 2021; Wang et al., 2021b) introduce some elaborate strategies to compress the datastore in which kNN searches are conducted, thus improving decoding efficiency to some extent. However, we argue that, where there is a time-consuming kNN search in the decoding phase, there is the prohibitive decoding cost, which makes it hard to be deployed on real-world applications.

In order to address the aforementioned issue more thoroughly, it is necessary to figure out why kNN-MT performs so well. The standard NMT models are typically trained with cross-entropy (CE) loss with teacher forcing technique, which requires a strict word-by-word matching between the model prediction and the ground-truth. In natural language, a sentence usually has more than one expression. However, even when the model predicts a word that is reasonable but deviates from the ground-truth, the CE loss will treat it as an error and punish the model. This phenomenon is called *overcorrection* (Zhang et al., 2019), which often seriously harms the generalizability of NMT models. We conclude that kNN-MT can alleviate the problem of overcorrection by retrieving more reasonable target tokens in the decoding phase.

One natural question can be raised: can we train the model to directly learn the knowledge of kNN in the training phase, thus maintaining the standard decoding process without any additional decoding cost? To answer this question, we propose k Nearest Neighbor Knowledge Distillation (kNN-KD) to distill the knowledge of the non-parametric model, i.e., kNN, into the base NMT model in the training phase. In detail, we first construct the datastore and then conduct kNN searches immediately. These two steps can be done offline in the preprocessing phase. During training, a teacher distribution p^T_{kNN} can be easily computed using the pre-stored results

[1]We will release the source code upon acceptance

of kNN searches to train the NMT model to directly learn the knowledge of kNN. At inference time, kNN searches are not required, so the decoding speed is as fast as the base NMT model. Therefore, kNN-KD can achieve two desirable goals simultaneously: preventing overcorrection (effectiveness) and reducing decoding cost (efficiency).

We conduct experiments on two widely acknowledged NMT benchmarks: IWSLT'14 German-English and IWSLT'15 English-Vietnamese. Experimental results show that our kNN-KD maintains the same training and decoding speed as the standard NMT model, while it outperforms vanilla kNN-MT and all the other KD methods, and yields an improvement of $+2.14$ and $+1.51$ BLEU points over the Transformer baseline. We further verify that kNN-KD can be adapted to diverse domains by performing experiments on multi-domains translation datasets (Aharoni and Goldberg, 2020) and achieving 2.56 BLEU improvement over vanilla kNN-MT on average.

In summary, the contributions of our work are as follows:

- We propose kNN-KD that considers the kNN distribution as a teacher to guide the training of the base NMT model (Section 3.1).
- We prove that our proposed kNN-KD can help to address the overcorrection issue with theoretical analysis (Section 3.2).
- Quantitative and qualitative results on different translation tasks validate the effectiveness and efficiency of our method (Section 4).

2 Background

2.1 Neural Machine Translation

The goal of the standard NMT model is to learn the conditional probability $p_{\text{MT}}(\mathbf{y} \mid \mathbf{x})$ for translating a sentence $\mathbf{x} = \{x_1, \cdots, x_m\}$ in source language to a sentence $\mathbf{y} = \{y_1, \cdots, y_n\}$ in target language. Translation is usually performed in a autoregressive manner, and its probability can be factored as $p_{\text{MT}}(\mathbf{y} \mid \mathbf{x}) = \Pi_{i=1}^{n} p(y_i \mid \mathbf{x}, \mathbf{y}_{<i})$. When predicting i-th token in the target sentence given $(\mathbf{x}, \mathbf{y}_{<i})$ as the *translation context*, the NMT model encodes the translation context into a hidden state h_{i-1}, and outputs a probability distribution over vocabulary $\mathcal{V}$ as follows:

$$p_{\text{MT}}(y_i \mid \mathbf{x}, \mathbf{y}_{<i}) = \frac{\exp(\mathbf{o}_{y_i}^{\top} \mathbf{h}_{i-1})}{\sum_{w \in \mathcal{V}} \exp(\mathbf{o}_w^{\top} \mathbf{h}_{i-1})}, \quad (1)$$

where $\mathbf{o}_y$ is the output embedding for $w \in \mathcal{V}$.

We denote the ground-truth target sentence as $\mathbf{y}^\star = \{y_1^\star, \cdots, y_n^\star\}$, and for each $y_i^\star$ in the training set, the CE loss is usually used for optimizing NMT models:

$$\mathcal{L}_{\text{CE}} = -\sum_{y_i \in \mathcal{V}} \mathbb{1}_{y_i = y_i^\star} \log p_{\text{MT}}(y_i \mid \mathbf{x}, \mathbf{y}_{<i}^\star), \quad (2)$$

where $\mathbb{1}$ is the indicator function, and the ground-truth target sequence $\mathbf{y}_{<i}^\star$ is used in the conditions of p_{MT} due to the teacher forcing technique.

2.2 Nearest Neighbor Machine Translation

kNN-MT applies the nearest neighbor retrieval mechanism to the decoding phase of a NMT model, which allows the model direct access to a large-scale datastore for better inference. Specifically, kNN-MT includes two following steps:

Datastore Building Given a bilingual sentence pair in the training set $(\mathbf{x}, \mathbf{y}^\star) \in (\mathcal{X}, \mathcal{Y}^\star)$, kNN-MT first constructs a datastore $\mathcal{D}$ as follows:

$$(\mathcal{K}, \mathcal{V}) = \bigcup_{(\mathbf{x}, \mathbf{y}^\star) \in (\mathcal{X}, \mathcal{Y}^\star)} \{(f(\mathbf{x}, \mathbf{y}_{<i}^\star), y_i^\star), \forall y_i^\star \in \mathbf{y}^\star\}, \quad (3)$$

where the keys are the mapping representations of all the translation contexts in the training set using the projection $f(\cdot)$, and the values are corresponding ground-truth tokens.

Decoding During inference, kNN-MT aims to interpolate the base NMT model's probability in Equation 1 with a kNN model. At each decoding step i, kNN-MT maps the current translation context to a representation $f(\mathbf{x}, \mathbf{y}_{<i})$, which is used to query the datastore for k nearest neighbors according to the l_2 distances. Denote the retrieved neighbors as $\mathcal{N}^i = \{(\mathbf{k}_j, v_j), j \in \{1, 2, \ldots, k\}\}$, and then a kNN distribution over vocabulary $\mathcal{V}$ can be computed as:

$$p_{\text{kNN}}(y_i \mid \mathbf{x}, \mathbf{y}_{<i}) \propto \sum_{(\mathbf{k}_j, v_j) \in \mathcal{N}^i} \mathbb{1}_{y_i = v_j} \exp\left(\frac{-d(\mathbf{k}_j, f(\mathbf{x}, \mathbf{y}_{<i}))}{\tau}\right), \quad (4)$$

where τ is the temperature, and $d(\cdot, \cdot)$ is the l_2 distance function. The final probability for the next token in kNN-MT is the interpolation of $p_{\text{MT}}(y_i \mid \mathbf{x}, \mathbf{y}_{<i})$ and $p_{\text{kNN}}(y_i \mid \mathbf{x}, \mathbf{y}_{<i})$ with a tunable weight λ:

$$\begin{aligned} p(y_i \mid \mathbf{x}, \mathbf{y}_{<i}) = (1-\lambda) & p_{\text{MT}}(y_i \mid \mathbf{x}, \mathbf{y}_{<i}) \\ & + \lambda p_{\text{kNN}}(y_i \mid \mathbf{x}, \mathbf{y}_{<i}). \end{aligned} \quad (5)$$

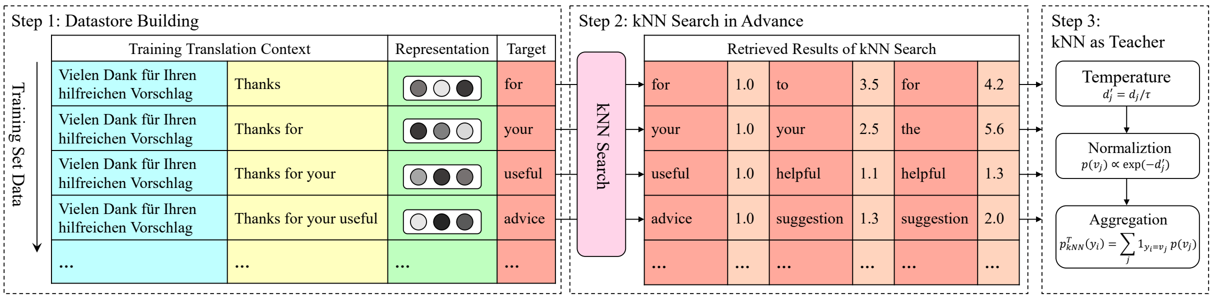
Step 1: Datastore Building
Training Set Data
Training Translation Context
Representation
Target
Vielen Dank für Ihren hilfreichen Vorschlag
Thanks
for
Thanks for
your
Thanks for your
useful
Thanks for your useful
advice
...
Step 2: kNN Search in Advance
kNN Search
Retrieved Results of kNN Search
for 1.0 to 3.5 for 4.2
your 1.0 your 2.5 the 5.6
useful 1.0 helpful 1.1 helpful 1.3
advice 1.0 suggestion 1.3 suggestion 2.0
Step 3: kNN as Teacher
Temperature $d'_j = d_j/\tau$
Normaliztion $p(v_j) \propto \exp(-d'_j)$
Aggregation $p^T_{kNN}(y_i) = \sum_j 1_{y_i=v_j}\, p(v_j)$

Figure 1: Illustration of kNN-KD. In the preprocessing phase, we finish the datastore building in Step 1, and conduct kNN search in advance in Step 2. These two steps can be done offline before training and inference. During training, we compute the kNN distribution as a teacher to train the base NMT model in Step 3. During inference, the model performs Step 4 to decode text in the standard Seq2Seq manner, which is omitted in this figure.

Note that each decoding step of each beam requires a kNN search over the whole datastore $\mathcal{D}$, whose time complexity is $\mathcal{O}(|\mathcal{D}|Bn)$ where B is the beam size, and n is the target length. The prohibitive decoding cost makes it hard for kNN-MT to be deployed on real-world applications.

2.3 Knowledge Distillation

Knowledge Distillation (KD) (Hinton et al., 2015b) refers to the transfer of knowledge from one neural network T (called "teacher model") to another network S (called "student model").

For convenience, we introduce the details of KD from the perspective of machine translation. Let $\mathbf{z} \in \mathcal{R}^{|\mathcal{V}|}$ denote the logits over $\mathcal{V}$. Student model S outputs the probability:

$$p^{\mathrm{S}}\left(y_i \mid \mathbf{x}, \mathbf{y}_{<i}\right) = \frac{\exp\left(\mathbf{z}_{y_i}\right)}{\sum_{w \in \mathcal{V}} \exp\left(\mathbf{z}_w\right)}, \tag{6}$$

where $\mathbf{z}_w$ is the logit for token w. Correspondingly, teacher model T also predicts the probability in the same way, and a temperature factor τ can be introduced to soften the teacher's outputs as:

$$p^{\mathrm{T}}\left(y_i \mid \mathbf{x}, \mathbf{y}_{<i}\right) = \frac{\exp\left(\mathbf{z}_{y_i}/\tau\right)}{\sum_{w \in \mathcal{V}} \exp\left(\mathbf{z}_w/\tau\right)}. \tag{7}$$

When $\tau \to \infty$, p^{T} degenerates into a uniform distribution, and when $\tau \to 0$, p^{T} becomes an one-hot vector. Specifically, KD defines the objective as:

$$\begin{aligned} \mathcal{L}_{\mathrm{KD}} = - \sum_{y_i \in \mathcal{V}} & p^{\mathrm{T}}\left(y_i \mid \mathbf{x}, \mathbf{y}^{\star}_{<i}\right) \\ & \times \log p^{\mathrm{S}}\left(y_i \mid \mathbf{x}, \mathbf{y}^{\star}_{<i}\right). \end{aligned} \tag{8}$$

When we apply KD to improve the performance of machine translation, student model S is usually the NMT model that will be used for testing. And then, the overall training procedure is to minimize the summation of Equation 2 and Equation 8:

$$\mathcal{L} = (1-\alpha)\mathcal{L}_{\mathrm{CE}} + \alpha\mathcal{L}_{\mathrm{KD}}, \tag{9}$$

where α is a weight to balance two losses.

3 Methodology

The core idea of our work is to enhance the NMT model with a nearest neighbor retrieval mechanism in a training manner, and thus quantitatively evaluated, the model can perform as well or better than vanilla kNN-NMT without any additional decoding cost. In Section 3.1, we first introduce k Nearest Neighbor Knowledge Distillation (kNN-KD) to distill the knowledge of kNN into a base NMT model. And then, we provide the theoretical analysis in Section 3.2 to support that our method can help to address the overcorrection issue.

3.1 Nearest Neighbor Knowledge Distillation

When we apply vanilla kNN-MT for testing using beam search with B, the time complexity of it is $\mathcal{O}(|\mathcal{D}|Bn)$. Compared with the standard beam search whose time complexity is $\mathcal{O}(|\mathcal{V}|Bn)$, the decoding speed of vanilla kNN-MT is prohibitively slow. This is mainly because vanilla kNN-MT has to conduct a kNN search over an extremely large datastore $\mathcal{D}$ for each decoding step of each beam. We propose to move this time-consuming search process forward to the preprocessing phase which can be done offline before training and inference. As shown in Figure 1, our proposed kNN-KD can be described as follows:

Step 1: Datastore Building We build the datastore $\mathcal{D}$ in the same way as vanilla kNN-MT (Khandelwal et al., 2021) which has been described in Section 2.2, so we omit it here.

Step 2: kNN Search in Advance For all the translation contexts $\left(\mathbf{x}, \mathbf{y}_{<i}^{\star}\right)$ in the training set, we conduct a kNN search using $f\left(\mathbf{x}, \mathbf{y}_{<i}^{\star}\right)$ as a query to search through the datastore $\mathcal{D}$ built in Step 1, and then we obtain the retrieved results $\mathcal{N}^i = \{(\mathbf{k}_j, v_j), j \in \{1, 2, \ldots, k\}\}$. Note that we are performing kNN search for **training set** translation contexts on the datastore built with the **training set**, which is equivalent to extending the training data by adding k reasonable target tokens for every translation context. Formally, by conducting kNN search in advance, we extend the target sentence in the training set from $\mathbf{y}^\star = \{y_1^\star, \cdots, y_n^\star\}$ to $\mathbf{y}^\star = \left\{\left(y_1^\star, \mathcal{K}^1\right), \cdots, \left(y_n^\star, \mathcal{K}^n\right)\right\}$, where $\mathcal{K}^i = \left\{\left(d\left(\mathbf{k}_j, f\left(\mathbf{x}, \mathbf{y}_{<i}^{\star}\right)\right), v_j\right), j \in \{1, 2, \ldots, k\}\right\}$.

Step 3: kNN as a Teacher In the training phase, a kNN distribution can be formulated as:

$$p_{\mathrm{kNN}}^{\mathrm{T}}\left(y_i \mid \mathbf{x}, \mathbf{y}_{<i}^{\star}\right) \propto \sum_{(d_j, v_j) \in \mathcal{K}^i} \mathbb{1}_{y_i = v_j} \exp\left(\frac{-d_j}{\tau}\right), \tag{10}$$

We then use $p_{\mathrm{kNN}}^{\mathrm{T}}$ as a teacher to train the base NMT model, and the knowledge distillation objective in Equation 8 can be rewritten as:

$$\mathcal{L}_{\mathrm{kNN-KD}} = -\sum_{y_i \in \mathcal{V}} p_{\mathrm{kNN}}^{\mathrm{T}}\left(y_i \mid \mathbf{x}, \mathbf{y}_{<i}^{\star}\right) \times \log p_{\mathrm{MT}}\left(y_i \mid \mathbf{x}, \mathbf{y}_{<i}^{\star}\right). \tag{11}$$

And the final training objective in Equation 9 can be rewritten as:

$$\mathcal{L} = (1-\alpha)\mathcal{L}_{\mathrm{CE}} + \alpha\mathcal{L}_{\mathrm{kNN-KD}}, \tag{12}$$

where $\mathcal{L}_{\mathrm{CE}}$ can be calculated as Equation 2.

Step 4: Decoding During inference, our model remains in the standard Seq2Seq manner (Vaswani et al., 2017), so we omit it here.

3.2 Theoretical Analysis

In this section, we show that our proposed kNN-KD can help address the overcorrection issue from the perspective of gradients. The gradient of the final objective in Equation 12 with respect to the logit $\mathbf{z}_{y_i}, y_i \in \mathcal{V}$ is:

$$\begin{aligned}\frac{\partial \mathcal{L}}{\partial \mathbf{z}_{y_i}} &= (1-\alpha)\left(p(y_i) - \mathbb{1}_{y_i = y_i^\star}\right) + \alpha\left(p(y_i) - p^{\mathrm{T}}(y_i)\right) \\ &= \begin{cases} p(y_i) - \alpha p^{\mathrm{T}}(y_i), & \text{if } y_i \neq y_i^\star \text{ and } y_i \in \mathcal{K}^i \\ p(y_i), & \text{if } y_i \neq y_i^\star \text{ and } y_i \notin \mathcal{K}^i \\ p(y_i) - \left(1 - \alpha + \alpha p^{\mathrm{T}}(y_i)\right), & \text{if } y_i = y_i^\star \end{cases}\end{aligned} \tag{13}$$

where we abbreviate $p_{\mathrm{MT}}\left(y_i \mid \mathbf{x}, \mathbf{y}_{<i}^{\star}\right)$ to $p(y_i)$ and $p_{\mathrm{kNN}}^{\mathrm{T}}\left(y_i \mid \mathbf{x}, \mathbf{y}_{<i}^{\star}\right)$ to $p^{\mathrm{T}}(y_i)$.

For every gradient update in the training phase, the model is trained to decrease the gradient norm to 0 to reach a local minimum (Lin et al., 2021). Therefore, for the tokens that are reasonable but not ground-truth (i.e., $y_i \neq y_i^\star$ and $y_i \in \mathcal{K}^i$), the model has to learn to increase the probability $p(y_i)$ by the degree of $\alpha p^{\mathrm{T}}(y_i)$ so that the gradient norm $|p(y_i) - \alpha p^{\mathrm{T}}(y_i)|$ can reach 0. For the other non-ground-truth token (i.e., $y_i \neq y_i^\star$ and $y_i \notin \mathcal{K}^i$), $p^{\mathrm{T}}(y_i)$ is equal to 0 since y_i is not included in the retrieved results of kNN search, and the model will learn to assign much lower probability $p(y_i)$ to reduce $|p(y_i)|$. Besides, since we build the datastore and conduct kNN search on the same training set data, for any translation context, its nearest neighbor over the datastore must be itself, which means if $y_i = y_i^\star$, then $y_i \in \mathcal{K}^i$. Then, for the ground-truth token (i.e., $y_i = y_i^\star$), the model is trained to increase the probability $p(y_i)$ by the degree of $\left(1 - \alpha + \alpha p^{\mathrm{T}}(y_i)\right)$. Note that, the gradient norm of the standard CE loss is $|p(y_i) - 1|$ for $y_i = y_i^\star$, and thus that standard CE increases the probability $p(y_i)$ by the degree of 1. This demonstrates that our kNN-KD still makes the model learn to predict the ground-truth but with a relatively lower strength than the standard CE.

Taking the case in Figure 1 as an example, given the translation context *"Vielen Dank für Ihren hilfreichen Vorschlag* || *Thanks for your"*, its ground-truth target token is *"useful"*, while *"helpful"* is also reasonable for this translation. Assuming that we have conducted the kNN search with $k = 3$ in advance as shown in Figure 1, and set τ to 1, we can then compute the kNN teacher distribution as:

$$p^T(y_4) = \begin{cases} 0.378, & \text{if } y_4 \text{ is } \textit{“useful”} \\ 0.622, & \text{if } y_4 \text{ is } \textit{“helpful”} \\ 0, & \text{otherwise} \end{cases} \tag{14}$$

According to Equation 13, the gradient norms are $|p(\textit{“helpful”}) - 0.622\alpha|$ for *"helpful"*, and $|p(\textit{“useful”}) - (1 - 0.622\alpha)|$ for *"useful"*. Therefore, our kNN-KD can train the model to learn from the kNN model to increase the probability of *"helpful"* that is reasonable but not ground-truth, thus addressing the overcorrection issue.

4 Experiments

4.1 Datasets

We conduct experiments on IWSLT'14 German-English (De-En, $160k$ training samples), IWSLT'15 English-Vietnamese (En-Vi, $113k$ training samples), and multi-domains translation datasets (Aharoni and Goldberg, 2020) (De-En, $733k$ training samples). For IWSLT'14 De-En, we follow the preprocessing steps provided by fairseq[2] (Ott et al., 2019) to split the data, and process the text into bytepair encoding (BPE) (Sennrich et al., 2016). For IWSLT'15 En-Vi, we use the pre-processed dataset[3] provided by Luong and Manning (2015). We use tst2012 as the validation set and tst2013 as the test set, which contains $1,553$ and $1,268$ sentences respectively. For multi-domains translation datasets, we use the pre-processed dataset[4] provided by Zheng et al. (2021), and consider domains including *Koran*, *Medical*, and *Law* in our experiments.

4.2 Competitive Models

The proposed kNN-KD is an architecture-free method that can be applied to arbitrary Seq2Seq models, which is orthogonality to previous works that design delicate structures to improve performance. Therefore, we mainly compare kNN-KD with vanilla kNN-MT and some typical KD methods:

- **Word-KD** (Hinton et al., 2015b). As described in Section 2.3, Word-KD is the standard KD that distills knowledge equally for each word.

- **Seq-KD** (Kim and Rush, 2016). In this method, teacher model T first generates an extra dataset by running beam search and taking the highest-scoring sequence. Then student model S is trained on this teacher-generated data, and the training objective can be formulated as:

$$\mathcal{L}_{\text{Seq-KD}} = -\sum_{i=1}^{n} \sum_{y_i \in \mathcal{V}} \mathbb{1}_{y_i = \hat{y}_i} \times \log p_{\text{MT}}\left(y_i \mid \mathbf{x}, \hat{\mathbf{y}}_{<j}\right), \tag{15}$$

where $\hat{\mathbf{y}}$ is the target sequence generated by teacher model, and n is the length of it.

- **BERT-KD** (Chen et al., 2020). This method distills knowledge learned in BERT (Devlin et al., 2019) to the student NMT model to improve translation quality.

- **Selective-KD** (Wang et al., 2021a). This work finds that some of the teacher's knowledge will hurt the effect of KD, and then address this issue by introducing Selective-KD to select suitable samples for distillation.

[2] https://github.com/pytorch/fairseq/blob/main/examples/translation/prepare-iwslt14.sh

[3] https://nlp.stanford.edu/projects/nmt/

[4] https://github.com/zhengxxn/adaptive-knn-mt

Datasets	$\lvert\mathcal{D}\rvert$	k	τ
IWSLT'14 De-En	3,949,106	64	100
IWSLT'15 En-Vi	3,581,500	64	100
Koran	524,374	16	100
Medical	6,903,141	4	10
Law	19,062,738	4	10

Table 1: Hyper-parameter settings for different datasets.

4.3 Implementation Details

All the algorithms are implemented in Pytorch with fairseq toolkit (Ott et al., 2019), and all the experiments are conducted on a machine with 8 NVIDIA GTX 1080Ti GPUs. Other details of the experimental setup can be seen in Appendix A.

Model Configuration We choose Transformer (Vaswani et al., 2017) as our base NMT model. For IWSLT'14 De-En and IWSLT'15 En-Vi, we use $transformer\_iwslt\_de\_en$ configuration, which has 6 layers in both encoder and decoder, embedding size 512, feed-forward size $1,024$ and attention heads 4. For multi-domains translation datasets, we follow Khandelwal et al. (2021) to adopt $transformer\_wmt19\_de\_en$ configuration, which has 6 layers in both encoder and decoder, embedding size $1,024$, feed-forward size $8,192$ and attention heads 8.

Preprocessing Details When building the datastores, we use the context vectors input to the final output layer as keys in the datastore $\mathcal{D}$. For IWSLT datasets, the base NMT model is used to obtain the context vectors, while for multi-domains translation datasets, we follow Khandelwal et al. (2021) to build datastores by the pre-trained model[5]. According to the model

[5] https://github.com/pytorch/fairseq/tree/main/examples/wmt19

Models	De-En			En-Vi		
	BLEU	upd/s	token/s	BLEU	upd/s	token/s
Transformer	34.11	2.02(1.00×)	3148.10(1.00×)	30.76	2.55(1.00×)	2870.07(1.00×)
Word-KD	34.26	1.77(0.88×)	3291.28(1.06×)	30.98	2.14(0.84×)	2782.53(0.97×)
Seq-KD	34.60	2.14(1.06×)	3409.86(1.08×)	31.20	2.80(1.10×)	2855.77(1.00×)
BERT-KD	35.63	1.70(0.84×)	3275.43(1.04×)	31.51	2.14(0.84×)	2785.69(0.97×)
Selective-KD	34.38	1.72(0.85×)	3365.68(1.07×)	31.48	2.09(0.82×)	3044.68(1.06×)
kNN-MT	36.17	-	920.72(0.29×)	32.08	-	617.88(0.22×)
kNN-KD	**36.30**	2.14(1.06×)	3321.24(1.05×)	**32.27**	2.60(1.02×)	2879.68(1.00×)

Table 2: Experimental results on IWSLT'14 De-En and IWSLT'15 En-Vi translation tasks. "-" means "not applicable" since vanilla kNN-MT can only be adopted in the decoding phase.

Models	Koran		Medical		Law	
	BLEU	token/s	BLEU	token/s	BLEU	token/s
Pre-trained Model	16.26	1038.97(1.00×)	39.91	1765.56(1.00×)	45.71	2404.31(1.00×)
kNN-MT	19.45	246.17(0.24×)	54.35	701.29(0.40×)	61.78	853.66(0.36×)
Transformer	13.84	1297.45(1.25×)	27.51	1073.53(0.61×)	60.77	1689.89(0.70×)
kNN-KD	**24.86**	1236.23(1.19×)	**56.50**	1853.58(1.05×)	**61.89**	2456.62(1.02×)

Table 3: Experimental results on multi-domains translation datasets. We leave out the metric for training efficiency (upd/s) since it is only applicable for Transformer and kNN-KD, and the training efficiency of these two models are basically the same.

configuration, the keys are 512-dimensional and 1024-dimensional for IWSLT datasets and multi-domains translation datasets, respectively. We use FAISS (Johnson et al., 2017) for the nearest neighbor search. And we conduct grid searches over $k \in \{4, 8, 16, 32, 64, 128, 256, 512, 1024\}$ and $\tau \in \{1, 10, 50, 100, 200, 500, 1000\}$, and choose the final settings according to the best BLEU score on the validation set. The final hyper-parameter settings are shown in Table 1.

Evaluation For all the datasets, we use the beam search with beam size 5. We evaluate the translation in terms of quality and efficiency.

- **Quality.** For IWSLT'14 De-En and IWSLT'15 En-Vi, following the common practice, we measure case sensitive BLEU by *multi-bleu.perl*[6]. For multi-domains translation datasets, we closely follow Khandelwal et al. (2021) to evaluate the results by SacreBLEU (Post, 2018) for a fair comparison.

- **Efficiency.** We evaluate the efficiency of training and inference by the training updates per second (upd/s) and the generated tokens per second (token/s), respectively.

[6]https://github.com/moses-smt/mosesdecoder/blob/master/scripts/generic/multi-bleu.perl

4.4 Main Results

Results of IWSLT Datasets We first compare kNN-KD with vanilla kNN-MT and other KD methods on the two IWSLT translation tasks. Note that there are several hyper-parameters in vanilla kNN-MT: tunable weight (λ), number of neighbors per query (k), and temperature (τ). These hyper-parameters have great effects on the translation results. We also conduct grid searches over these hyper-parameters, and find the best settings according to BLEU score on the validation set.

As shown in Table 2, kNN-KD outperforms all the other strong baselines on both IWSLT datasets, e.g., an improvement of +2.14 and +1.51 BLEU score over Transformer. Moreover, we observe that our proposed kNN-KD can even perform better than vanilla kNN-MT, while gaining a significant speedup. On the one hand, kNN-KD, like other KD methods, maintains the standard Seq2Seq manner at inference time, thus keeping the same decoding speed as Transformer. On the other hand, kNN-KD also keeps the same training speed as Transformer, and it is more efficient than Word-KD, BERT-KD and Selective-KD. This is because the calculation of the teacher model distribution $p^{\mathrm{T}}_{k\mathrm{NN}}\left(y_i \mid \mathbf{x}, \mathbf{y}^{\star}_{<i}\right)$ only needs to be performed on a relatively small kNN retrieved set $\mathcal{K}^i$, while word-

Models	Law→Medical	Medical→Law
Transformer	18.73	2.07
kNN-KD	22.31	14.82

Table 4: Generalizability Evaluation. "Law→Medical" means that we train the model on the Law domain and directly apply it to Medical domain, and vice versa.

level KD have to compute the teacher distribution over the whole vocabulary $\mathcal{V}$.

Results of Multi-domains Datasets Apart from IWSLT datasets, we further compare our kNN-KD with kNN-MT on multi-domains translation datasets. First, we follow Khandelwal et al. (2021) to conduct inference with the pre-trained model and vanilla kNN-MT. Then, we train the base NMT model using standard CE and kNN-KD on each domain's training data, and report the results in Table 3 as a comparison. In all domains, kNN-KD again outperforms all the baselines. Most importantly, our proposed kNN-KD can achieve a consistent improvement over vanilla kNN-MT (+2.56 BLEU score on average) with a significant speedup. This further confirms the effectiveness and efficiency of our method.

Generalizability To verify the generalizability of our method, we further conduct experiments on the scenario that we train a NMT model on a specific domain and evaluate it on the out-of-domain test set. As shown in Table 4, our kNN-KD performs significantly better than Transformer trained by standard CE. It proves the statement in Section 1 that compared with standard CE, kNN-KD can improve the generalizability of NMT models.

4.5 Analysis

There are two key hyper-parameters in our kNN-KD: number of neighbors per query (k), and temperature (τ). In this section, we investigate the effects of these two hyper-parameters on the validation set of IWSLT'14 De-En.

Effect of k We fix the temperature τ to 100, and train the model using kNN-KD with different k. As shown in Figure 2, the BLEU score first rises with the increase of k, and reaches the best performance peak when $k = 64$. And then, performance deteriorates with a larger k. This suggests that, the retrieved results of kNN search can substantially improve training when k is relatively small, but it will also introduce more noise when k gets larger.

Effect of τ We train the model using kNN-KD with different τ and fixed k ($k = 64$). As shown

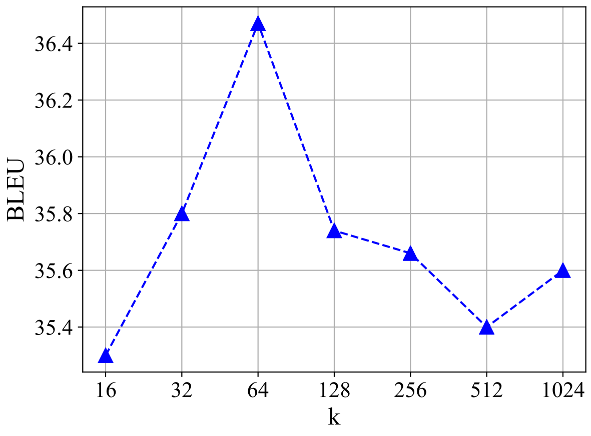

Figure 2: BLEU scores with different k and fixed τ ($\tau = 100$) on the validation set of IWSLT'14 De-En dataset.

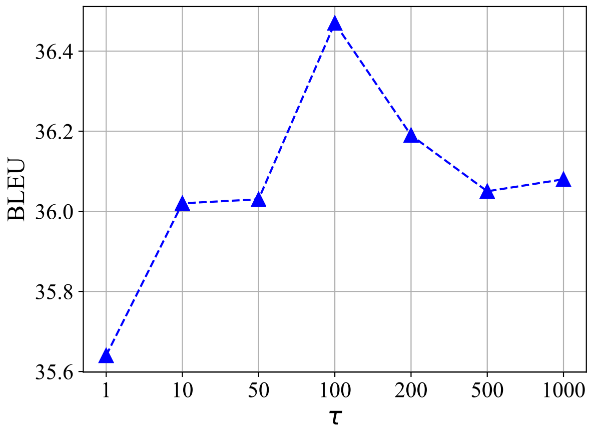

Figure 3: BLEU scores with different τ and fixed k ($k = 64$) on the validation set of IWSLT'14 De-En dataset.

in Figure 3, a temperature of 1 results in a significantly lower BLEU score than those greater than 1. This is because a large temperature value can flatten the kNN teacher distribution in Equation 10 to prevent assigning most of the probability mass to a single neighbor. The results show that for $k = 64$, the optimal temperature is 100.

4.6 Case Study

In this section, we show how our proposed method works by presenting a real case. There exists an example in the test set of IWSLT'14 De-En that the source sentence is *"es gibt eine menge geschichten darüber , warum wir dies getan haben ."* and the corresponding target sentence is *"there are a lot of stories about why we did this ."*. Given the source sentence and target subsequence *"there are"* as the translation context, *"many..."*, *"lots of..."*, and *"a lot of..."* are all correct translations. We input

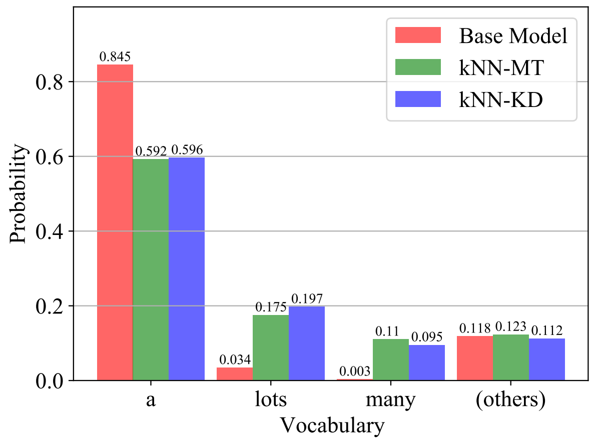

Figure 4: Predicted probabilities output from the base NMT model, kNN-MT and our kNN-KD, given the translation context *"es gibt eine menge geschichten darüber , warum wir dies getan haben . || there are"*

this translation context to the base NMT model, kNN-MT, and our model, and observe the predicted probabilities over the vocabulary. As shown in Figure 4, all the models predict *"a"* with the maximal probability that matches the ground-truth. However, since the base model is trained by CE loss using one-hot vector as supervision, it suffers from a serious overcorrection problem that the model assigns an overconfident probability to the token *"a"* and almost none to other reasonable target tokens such as *"lots"* and *"many"*. On the contrary, both kNN-MT and our kNN-KD increase the probabilities of the reasonable target tokens, and these two models have similar predicted probabilities. Note that kNN-MT obtains this probability distribution by interpolating the base NMT probability with a kNN search probability at decoding time, while our kNN-KD directly outputs this distribution without any additional operations. This further confirms that kNN-KD can train the model to learn the knowledge of kNN that prevents the overconfidence of the model on the one-hot label, thus leading to the better generalizability for inference.

5 Related Works

5.1 Neural Machine Translation

Machine translation has developed rapidly in recent years. The early models were mainly based on statistical machine learning (Brown et al., 1990; Och, 2003; Koehn et al., 2007). Then, with the development of deep learning technology, many models used RNN(Sutskever et al., 2014; Bahdanau et al., 2015), CNN(Gehring et al., 2017), or Transformer(Vaswani et al., 2017) as their backbones.

Recently, a few studies have combined k nearest neighbors algorithm closely with NMT models to improve performance. Khandelwal et al. (2021) used a nearest neighbor classifier to predict tokens on a large datastore of cached examples and proposed kNN-MT. However, Meng et al. (2021) pointed out that kNN-MT is two-order slower than vanilla MT models, which limits the deployment for real-world applications. They proposed Fast kNN-MT to solve this problem. Wang et al. (2021b) also noticed the low-efficiency problem of kNN-MT. Thus, they used a hierarchical clustering strategy and proposed Faster kNN-MT. Although the above studies have made feasible fixes, kNN search is still required in the decoding phase, which dramatically increases the difficulty of practical applications compared to standard MT models.

5.2 Knowledge Distillation

Knowledge distillation (KD) introduces teacher network and student network to help knowledge transfer and it was widely used in NMT (Hinton et al., 2015a). Kim and Rush (2016) introduced two sequence-level KD methods to improve the performance of NMT. Miceli-Barone et al. (2017) used KD to address the problem of catastrophic forgetting in the fine-tuning stage. Tan et al. (2019) used KD to enable the multilingual model to fit the training data and to match the outputs of the teacher models. Clark et al. (2019) distilled single-task models into one multi-task model. Chen et al. (2020) used BERT as the teacher model after fine-tuning on the target generation tasks to improve the conventional Seq2Seq models. Wang et al. (2021a) proposed batch-level and global-level selection strategies to choose appropriate samples for knowledge distillation. We focus on using KD to leverage the knowledge retrieved by kNN search to enhance a base NMT model.

6 Conclusion

In this paper, we introduce kNN-KD that distills the knowledge retrieved by kNN search to prevent the base NMT model from overcorrection. Experiments show that kNN-KD can improve over vanilla kNN-MT and other baselines without any additional cost for training and decoding. In the future, we will apply kNN-KD to many other tasks. We will also explore the effect of kNN-KD on improving the diversity of text generation.

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, ACL 2020, Online, July 5-10, 2020*, pages 7747–7763. Association for Computational Linguistics.

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*.

Peter F Brown, John Cocke, Stephen A Della Pietra, Vincent J Della Pietra, Frederick Jelinek, John Lafferty, Robert L Mercer, and Paul S Roossin. 1990. A statistical approach to machine translation. *Computational linguistics*, 16(2):79–85.

Yen-Chun Chen, Zhe Gan, Yu Cheng, Jingzhou Liu, and Jingjing Liu. 2020. Distilling knowledge learned in BERT for text generation. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, ACL 2020, Online, July 5-10, 2020*, pages 7893–7905. Association for Computational Linguistics.

Kevin Clark, Minh-Thang Luong, Urvashi Khandelwal, Christopher D Manning, and Quoc Le. 2019. Bam! born-again multi-task networks for natural language understanding. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 5931–5937.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2019, Minneapolis, MN, USA, June 2-7, 2019, Volume 1 (Long and Short Papers)*, pages 4171–4186.

Jonas Gehring, Michael Auli, David Grangier, Denis Yarats, and Yann N Dauphin. 2017. Convolutional sequence to sequence learning. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 1243–1252.

Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. 2015a. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*.

Geoffrey E. Hinton, Oriol Vinyals, and Jeffrey Dean. 2015b. Distilling the knowledge in a neural network. *CoRR*, abs/1503.02531.

Jeff Johnson, Matthijs Douze, and Hervé Jégou. 2017. Billion-scale similarity search with gpus. *CoRR*, abs/1702.08734.

Urvashi Khandelwal, Angela Fan, Dan Jurafsky, Luke Zettlemoyer, and Mike Lewis. 2021. Nearest neighbor machine translation. In *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*. OpenReview.net.

Yoon Kim and Alexander M. Rush. 2016. Sequence-level knowledge distillation. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, EMNLP 2016, Austin, Texas, USA, November 1-4, 2016*, pages 1317–1327. The Association for Computational Linguistics.

Philipp Koehn, Hieu Hoang, Alexandra Birch, Chris Callison-Burch, Marcello Federico, Nicola Bertoldi, Brooke Cowan, Wade Shen, Christine Moran, Richard Zens, et al. 2007. Moses: Open source toolkit for statistical machine translation. In *Proceedings of the 45th annual meeting of the association for computational linguistics companion volume proceedings of the demo and poster sessions*, pages 177–180.

Xiang Lin, Simeng Han, and Shafiq R. Joty. 2021. Straight to the gradient: Learning to use novel tokens for neural text generation. In *Proceedings of the 38th International Conference on Machine Learning, ICML 2021, 18-24 July 2021, Virtual Event*, volume 139 of *Proceedings of Machine Learning Research*, pages 6642–6653. PMLR.

Minh-Thang Luong and Christopher D. Manning. 2015. Stanford neural machine translation systems for spoken language domains. In *Proceedings of the 12th International Workshop on Spoken Language Translation: Evaluation Campaign@IWSLT 2015, Da Nang, Vietnam, December 3-4, 2015*.

Yuxian Meng, Xiaoya Li, Xiayu Zheng, Fei Wu, Xiaofei Sun, Tianwei Zhang, and Jiwei Li. 2021. Fast nearest neighbor machine translation. abs/2105.14528.

Antonio Valerio Miceli-Barone, Barry Haddow, Ulrich Germann, and Rico Sennrich. 2017. Regularization techniques for fine-tuning in neural machine translation. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 1489–1494.

Franz Josef Och. 2003. Minimum error rate training in statistical machine translation. In *Proceedings of the 41st annual meeting of the Association for Computational Linguistics*, pages 160–167.

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

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, Belgium, Brussels. Association for Computational Linguistics.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016. Neural machine translation of rare words with subword units. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, ACL 2016, August 7-12, 2016, Berlin, Germany, Volume 1: Long Papers*. The Association for Computer Linguistics.

Ilya Sutskever, Oriol Vinyals, and Quoc V Le. 2014. Sequence to sequence learning with neural networks. In *Advances in neural information processing systems*, pages 3104–3112.

Xu Tan, Yi Ren, Di He, Tao Qin, Zhou Zhao, and Tie-Yan Liu. 2019. Multilingual neural machine translation with knowledge distillation. *arXiv preprint arXiv:1902.10461*.

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 *In Advances in Neural Information Processing Systems*, pages 5998–6008.

Fusheng Wang, Jianhao Yan, Fandong Meng, and Jie Zhou. 2021a. Selective knowledge distillation for neural 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, ACL/IJCNLP 2021, (Volume 1: Long Papers), Virtual Event, August 1-6, 2021*, pages 6456–6466. Association for Computational Linguistics.

Shuhe Wang, Jiwei Li, Yuxian Meng, Rongbin Ouyang, Guoyin Wang, Xiaoya Li, Tianwei Zhang, and Shi Zong. 2021b. Faster nearest neighbor machine translation. *CoRR*, abs/2112.08152.

Wen Zhang, Yang Feng, Fandong Meng, Di You, and Qun Liu. 2019. Bridging the gap between training and inference for neural machine translation. In *Proceedings of the 57th Conference of the Association for Computational Linguistics, ACL 2019, Florence, Italy, July 28- August 2, 2019, Volume 1: Long Papers*, pages 4334–4343. 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, ACL/IJCNLP 2021, (Volume 2: Short Papers), Virtual Event, August 1-6, 2021*, pages 368–374. Association for Computational Linguistics.

A Experimental Setup

A.1 Datasets

The dataset statistics for all the datasets are reported in Table 5. It is worth to mention that IWSLT datasets are under the Creative Commons BY-NC-ND license, and the multi-domains translation datasets are under the BSD license.

	Train	Valid	Test
IWLST‘14 De-En	160,239	7,283	6,750
IWLST‘15 En-Vi	133,166	1,553	1,268
Koran	17,982	2,000	2,000
Medical	248,099	2,000	2,000
Law	467,309	2,000	2,000

Table 5: The number of examples for different datasets.

A.2 Hyper-parameters Setting

All the algorithms are implemented in Pytorch with fairseq toolkit (Ott et al., 2019), and all the experiments are conducted on a machine with 8 NVIDIA GTX 1080Ti GPUs with the hyperparameters reported in Table 6.

Hyperparameters	IWSLT	Multi-domains
Max tokens	8192	1280
Learning rate	5e-4	5e-4
LR scheduler	Inverse sqrt	Inverse sqrt
Minimal LR	1e-9	1e-9
Warm-up LR	1e-7	1e-7
Warm-up steps	4000	4000
Gradient clipping	0.0	0.0
Weight decay	0.0	0.0001
Droupout	0.3	0.2
Attention dropout	0.0	0.1
Activation dropout	0.0	0.1
α in Equation 12	0.5	0.5
Optimizer	Adam	Adam
-β_1	0.9	0.9
-β_2	0.98	0.98
-ϵ	1e-8	1e-8

Table 6: Hyperparameter settings for different datasets.

Note that during training, we are using the dynamic batching provided by fairseq, and choose the max tokens according to the GPU memory constraint. We train the model for 200 epochs on IWSLT datasets, 250 epochs on Koran domain, 100 epochs on Medical domain, 120 epochs on Law domain, while the early-stop mechanism is also adopted with patience set to 20.

B Limitation and Potential Risks

Although kNN-KD is efficient in both training and inference, it will take a relatively long time for preprocessing to build the datastore and conduct kNN searches, and it also requires large disk space to store all these results. However, since the preprocessing can be done offline, it does not limit the deployment of kNN-KD in real-world applications.

Our model is trained on open source datasets, and thus if there exists toxic text in the training data, our model may have the risk of producing toxic content.