DIDST: Disentangling Datastore and Translation Model for Nearest Neighbor Machine Translation

Anonymous ACL submission

Abstract

Nearest Neighbor Machine Translation (kNN-MT) is a simple and effective method of augmenting neural machine translations with a token-level nearest neighbor retrieval mechanism. However, original kNN-MT build datastores based on machine translation (MT) models, which would hinder the performance gain when the MT models are not good enough. In this paper, we propose DIDST, a framework that DIstangles DIasstores and TIranslation models in kNN-MT. We build datastores by leveraging large scale pretrained models, and design a novel contrastive objective to allow MT models to retrieve from external datastores. Experiments on bilingual and multilingual translation benchmarks demonstrate the effectiveness of DIDST. The disentanglement of datastores and MT models also brings unique advantages in practice: datastore creation from monolingual corpus and potentially better user privacy preservation.

1 Introduction

Retrieval-Enhanced Neural Machine Translation (RE-NMT) aims to augment parametric neural models with external datastores, and has been proved to be effective in many works (Gu et al., 2018; Bapna and Firat, 2019; Khandelwal et al., 2019; Cai et al., 2021). Being able to access parallel corpus at the inference time, RE-NMT shows more expressiveness than pure parametric methods.

kNN-MT is a representative work of RE-NMT, and attracts much attention due to its conceptual simplicity and impressive performance (Zheng et al., 2021a; Jiang et al., 2021; Zheng et al., 2021b; Meng et al., 2021). kNN-MT stores all decoding states on the training corpus in the datastore, where the decoding state is obtained by encoding the source sentence and target sentence prefix using an off-the-shelf parametric NMT model. At the inference time, token-level neighborhood examples are retrieved based on the similarity measurement, which is calculated based on the NMT model representations, to help the translation.

Despite the success of kNN-MT, it is worthy of discussing whether using MT models to create datastores is the optimal choice. Since the retrieval procedure needs to find the k-nearest examples to the current decoding state in the datastore, the quality of the datastore strongly influences the final performance. When MT model cannot learn good representations, this would result in sub-optimal performance.

In this paper, we propose DIDST, a framework that disentangles datastores and translation models in kNN-MT. Datastores in DIDST can be generated from arbitrary neural models. This enables us to leverage more powerful models, e.g., pretrained language models, to obtain better datastores, thus leads to better translation. To bridge representation discrepancy between MT models and datastores generated by other models, we design an auxiliary contrastive objective function to help the model to retrieve meaningful neighborhood examples.

Empirically, DIDST is effective and efficient. Experiments on bilingual and multilingual translation
benchmarks demonstrate that DIDST achieves substantial improvements over baselines. Compared to using PLMs in the MT models in kNN-MT, our DIDST does not increase the MT model size, which makes the inference more efficient. Practically, DIDST also has desirable properties for industrial use. The ability to retrieve from monolingual datastores allows DIDST to fast adapt to low-resource domains. Moreover, by creating datastores based on pretrained models, not only can the translation quality be improved, but also user data privacy (Shokri and Shmatikov, 2015) can be potentially better preserved by transmitting datastores encoded by public pretrained models instead of raw data.

2 kNN-MT and Its Limitation

2.1 kNN-MT

kNN-MT proposes to augment the NMT predictions with retrieved examples from a pre-built datastore. Given a parallel corpus \( D = \{X_i, Y_i\}_i \), where \( N \) is the total number of sentences, and a pretrained NMT model, kNN-MT build a datastore by collecting all token-level examples in \( D \), and each example is a (key, value) pair in the form of

\[
(k, v) = (f(X, Y_{<t}), Y_t)
\]

where \( f(X, Y_{<t}) \) is contextual representation from MT decoder by teacher forcing decoding on the sentence pair \((X, Y)\), and \( Y_t \) is corresponding \( t \)-th target token.

At the inference time, kNN-MT predicts the target token \( y_t \) relying on not only the MT model probability, but also retrieved examples from the datastore. Specifically, given the already generated tokens \( y_{<t} \), the contextual representation \( f(X, y_{<t}) \) is computed as query to retrieve \( k \) neighbors from the datastore w.r.t Euclidean distance, which is denoted as \( N = \{(k_i, v_i), i = 1, 2, \ldots, k\} \), the probability distribution from \( k \)-NN is computed as:

\[
p_{k\text{NN}}(y|x, y_{<t}) \propto \sum_{(k_i, v_i)} \mathbb{I}_{y = v_i} \cdot e^{-d(f(X, y_{<t}), k_i)^2}
\]

where \( T \) is the temperature hyperparameter, and \( d(\cdot, \cdot) \) is distance function between two representations. The final probability is an interpolation of MT model probability \( p_{MT} \) and \( k \)-NN probability \( p_{k\text{NN}} \):

\[
p(y|x, y_{<t}) = \lambda p_{MT}(y|x, y_{<t}) + (1 - \lambda)p_{k\text{NN}}(y|x, y_{<t})
\]

2.2 Analysis On Datastores of kNN-MT

The core assumption of kNN-MT is that similar representation from an off-the-shelf MT model tends to share the same (or similar) target tokens. However, this assumption may not hold when the MT model does not produce a good representation space. To study this problem, we randomly select subsets of different sizes from WMT14 English-German dataset, and evaluate the \( k \)-NN accuracy of datastores built from NMT systems trained using these subsets. The \( k \)-NN accuracy is defined as the average proportions of top- \( k \) neighbors that share the same target tokens for each entry in the datastore:

\[
\text{Acc}@k(S) = \frac{1}{|S|} \sum_{y \in S} \sum_{i=1}^{k} \mathbb{I}_{y(q) = y(N_i^q)}
\]

Figure 1 shows the results. We can see that when the training corpus size is small, the MT model cannot learn a good representation space, making \( k \)-NN accuracy of the datastore relatively low (less than 50%). Although the quality of the datastore grows with the training corpus’s size, it generally lies at a low level. This motivates us to disentangle datastores’ representation with MT representation and seek better ways to build datastores for retrieval-based neural machine translation.

3 Method

In this section, we introduce DIDST , a framework that enables neural machine translation models to retrieve from any given external datastore. We start by introducing our unified model architecture that performs translation and retrieval simultaneously. We then propose an auxiliary training objective to bridge the representation gap between the translation model and external datastores. Finally, we describe how to make use of stronger machine translation models or pretrained language models to build datastores.

3.1 Model Architecture

Our model is based on Transformer (Vaswani et al., 2017) architecture. Given an input sentence \( x \) and the translation history \( y_{<t} \), we extract two representations from the Transformer decoder: translation state \( h \) and retrieval query \( q \), where \( h \) is the output of the final decoder layer, and \( q \) is an intermediate representation from one of the decoder layers.

Given query \( q \), we retrieve \( K \) neighbors from the datastore \( D \), which is denoted as
Figure 2: The training paradigm of our methods. Translation of "我爱桃子" is "I love peaches".

(k_1, y_1), (k_2, y_2), ..., (k_K, y_K). Following Yogatama et al. (2021); Cai et al. (2021), we integrate the retrieval results at the representation level instead of probability level. This avoids manually tuning of the mixing weight $\lambda$ and temperature $T$ in the original kNN-MT. Concretely, we embed each $y_i$ as $e_i$ using the target-side word embedding matrix $W_e$. Then we use a simple attention mechanism to aggregate $e_1, e_2, ..., e_k$ to a single vector $m$, which is fused to the translation state in a gated fashion:

$$m = \sum_{k=1}^{K} \frac{\exp(e_k^T h)}{\sum_{i=1}^{K} \exp(e_i^T h)} \cdot e_k$$

$$g = \sigma(w_1 h + w_2 m + b)$$

$$z = g \odot m + (1 - g) \odot h$$

where $W_1, W_2$ are transforming matrices, and $b$ is the bias term. The final generation probability is computed as

$$p(y_t|x, y_{<t}) = \text{softmax}(z; W_e)$$

3.2 Learning

Ideally, query $h_t^{Q}$ should retrieve neighbors that are semantically similar to the target token $y_t$, so that they would be helpful for prediction. However, since the datastore has been disentangled from the MT model, directly using query $h_t^{Q}(x, y_{<t})$ to retrieve from this external datastore would result in meaningless retrieval results. To alleviate this problem, we propose Neighborhood Contrastive Aligning (NCA), an auxiliary training objective that explicitly aligns the retrieval query representation to its corresponding key representation in the datastore. Figure 2 illustrates the training process.

Specifically, we treat each query as the anchor example, and its corresponding key representation as the positive example. The set of negative examples $\mathcal{N}$ consist of all retrieved keys in the mini-batch, summing up to $B \times K$ examples, where $B$ is the number of tokens in the mini-batch, and $K$ is the number of retrieved neighbors of each query. We then minimize a contrastive objective as follows:

$$L_{NCA}(x, y) = -\frac{1}{|y|} \sum_{t=1}^{|y|} \log \frac{\exp(q_t^T k / \tau)}{\sum_{k \in \mathcal{N}} \exp(q_t^T k / \tau)}$$

where $q_t$ is the retrieval query at the $t$ timestep, $\tau$ is the temperature parameter.

Overall Objective

We train our model end to end by optimizing the translation objective and alignment objective simultaneously:

$$L_{MT}(x, y) = -\frac{1}{|y|} \sum_{i=1}^{|y|} \log p(y_t|x, y_{<t})$$

$$L_{overall} = L_{MT} + \alpha L_{NCA}$$

where $\alpha$ is the hyperparameter that balancing translation objective and alignment objective.

Unlike previous works (Cai et al., 2021), we do not update representations in datastores since they
usually come from powerful pretrained models, and further finetuning would cause catastrophic forgetting of previously acquired language knowledge.

3.3 Disentangled Datastore Creation

The disentanglement between MT models and datastores allows us to explore more novel ways for datastore creation. Theoretically, any neural models producing token-level representation can be used to build datastores. In this section, we introduce two ways to build better datastores using pretrained models: MT model based datastore creation and PLM based datastore creation.

3.3.1 MT model based Datastore Creation

For MT model based datastore creation, we follow the procedure $k$NN-MT used, except that representations come from another more powerful MT model, such as models that are specifically pretrained for machine translation (Lin et al., 2020; Pan et al., 2021). Our method enables us to retrieve from better datastores while avoiding computing through the large MT model at inference time.

One may think our method resembles with knowledge distillation (Hinton et al., 2015; Kim and Rush, 2016). However, we would like to point out that our method is different from knowledge distillation in terms of both motivation and technical details. On the one hand, KD aims to learn a lightweight student model under the supervision of a heavy teacher model, while our goal is to enable MT models to retrieve from better datastores. On the other hand, traditionally KD directly minimizes the mean squared error (MSE) between representations of the student model and teacher model, while we propose a contrastive objective beside the MSE objective, where we show empirically the former is better than the latter.

3.3.2 PLM based Datastore Creation

Another viable choice is to utilize large-scale pretrained language models for datastore creation. Existing PLMs can be divided into three categories according to their training objectives: (1) masked language models (MLM) (2) denoising autoencoders (DAE) (3) causal language models (CLM). We describe different strategies for each kind of model, respectively.

Datastore Creation for MLM models

For MLM models $f_{MLM}$, e.g. BERT (Devlin et al., 2019), Roberta (Liu et al., 2019), XLM-R (Conneau et al., 2020), we feed target-side sentences to them, and obtain the contextual representation at the top layer as keys, with the input tokens as values:

$$H = f_{MLM}(Y)$$

$$(k, v) = (H_t, y_t)$$

Datastore Creation for DAE models

For DAE models $f_{DAE}$, e.g. BART (Lewis et al., 2020), MBART (Liu et al., 2020), we generate the (key,value) pair in a way similar to vanilla kNN-MT, except that the source and target side are both target language sentences:

$$(k, v) = (f_{DAE}(Y, Y_{<t}), y_t)$$

Datastore Creation for CLM Models

For CLM models $f_{CLM}$, e.g. GPT (Radford et al., 2018), GPT-2 (Radford et al., 2019), we generate the (key,value) pair directly using teacher forcing decoding:

$$(k, v) = (f_{CLM}(Y_{<t}), y_t)$$

4 Experiments

4.1 Datasets

We experiment on bilingual and multilingual machine translation benchmarks.

Bilingual Machine Translation

For bilingual machine translation, we consider low, medium, and high resource settings. For low resource setting, we use IWSLT14 German-English dataset, a beam size of 5, and report case-insensitive detokenized BLEU. For medium resource setting, we use WMT16 English-Romanian dataset, a beam size of 5 and report case-sensitive detokenized BLEU. We remove diacritics of Romanian text following previous work (Sennrich et al., 2016). For high resource setting, we use WMT14 English-German dataset, a beam size of 4, and report case-sensitive detokenized BLEU computed by sacrebleu.

Multilingual Machine Translation

For multilingual machine translation, we use IWSLT14 multilingual dataset, which consists of parallel sentences between English and seven languages. We use a beam size of 5 and report case-sensitive detokenized BLEU.

4.2 Implementation Details

Tokenization

For vanilla Transformer baselines, we learn a joint-bpe vocab using sentencepiece.

1https://github.com/google/sentencepiece
### Table 1: Bilingual translation results on WMT14 En-De, WMT16 En-Ro and IWSLT14 De-En datasets.

<table>
<thead>
<tr>
<th></th>
<th>WMT14</th>
<th>WMT16</th>
<th>IWSLT14</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>En De</td>
<td>De En</td>
<td>En Ro</td>
</tr>
<tr>
<td>No PLM</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Transformer</td>
<td>27.32</td>
<td>31.47</td>
<td>31.53</td>
</tr>
<tr>
<td>kNN-MT</td>
<td>27.61</td>
<td>31.62</td>
<td>32.69</td>
</tr>
<tr>
<td>with XLMR.base</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Transformer</td>
<td>27.87</td>
<td>32.45</td>
<td>32.89</td>
</tr>
<tr>
<td>kNN-MT</td>
<td>27.99</td>
<td>32.68</td>
<td>33.77</td>
</tr>
<tr>
<td>DIDST</td>
<td><strong>28.67</strong></td>
<td><strong>33.01</strong></td>
<td><strong>34.18</strong></td>
</tr>
<tr>
<td>with mRASP2</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Transformer</td>
<td>28.10</td>
<td>32.69</td>
<td>33.14</td>
</tr>
<tr>
<td>kNN-MT</td>
<td>28.31</td>
<td>32.86</td>
<td><strong>33.90</strong></td>
</tr>
<tr>
<td>DIDST</td>
<td><strong>28.79</strong></td>
<td><strong>33.14</strong></td>
<td><strong>33.81</strong></td>
</tr>
</tbody>
</table>

### Table 2: Multilingual translation results on IWSLT14 multilingual datasets. Average BLEU scores on seven languages are reported. Models with † are access to XLMR as external resources.

<table>
<thead>
<tr>
<th></th>
<th>X → En</th>
<th>En → X</th>
</tr>
</thead>
<tbody>
<tr>
<td>Transformer</td>
<td>25.54</td>
<td>28.90</td>
</tr>
<tr>
<td>kNN-MT</td>
<td>25.67</td>
<td>29.08</td>
</tr>
<tr>
<td>Transformer †</td>
<td>25.73</td>
<td>29.25</td>
</tr>
<tr>
<td>kNN-MT †</td>
<td>25.97</td>
<td>29.61</td>
</tr>
<tr>
<td>DIDST †</td>
<td><strong>26.39</strong></td>
<td><strong>29.91</strong></td>
</tr>
</tbody>
</table>

### 4.3 Results on Bilingual Translation

We compare DIDST to vanilla transformers and the original kNN-MT. For PLM based datastore creation, we use XLMR.base, as it achieves the best performance in preliminary experiments. To make a more fair comparison, we also list the results of MT models that utilize pretrained language models, where the output of PLMs are fed into MT models as inputs. Previous works (Xu et al., 2021; Sun et al., 2021) show this is an effective way to incorporate PLMs.

For MT model based datastore creation, we finetune mRASP2, a recently proposed multilingual sequence-to-sequence model pretrained for machine translation, on each dataset and treat it as the external powerful MT model. For kNN-MT, we train MT models with knowledge distilling objectives under the supervision of finetuned mRASP2 for a fair comparison. Table 1 shows the results.

**DIDST is effective.** It can be seen DIDST performs better than baselines in 11 out of 12 settings, e.g., when using XLMR as the external pretrained model, DIDST outperforms kNN-MT by a large margin, for example +0.68 BLEU on WMT14 En De translation, +0.71 BLEU on IWSLT14 De En translation. This indicates DIDST is a more effective way to leverage pretrained language models.

**DIDST is also efficient.** The superior performance of DIDST also comes with a lower computational cost at the inference time: kNN-MT needs to compute through the large XLMR model to get the encoding. In contrast, DIDST only needs to compute through the original transformer model.

### 4.4 Results on Multilingual Translation

Table 2 lists experimental results on IWSLT14 multilingual datasets. We can see DIDST also achieves improvements over kNN-MT on multilingual settings, when using XLMR as external resources. It is worth mentioning that for many to one translation, kNN-MT needs to keep $L$ bilingual datastores.
Table 3: Comparison of MSE and NCA training objective for translation performance.

<table>
<thead>
<tr>
<th></th>
<th>IWSLT14</th>
<th>WMT14</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>De-En</td>
<td>En-De</td>
</tr>
<tr>
<td>MSE</td>
<td>34.91</td>
<td>28.80</td>
</tr>
<tr>
<td>NCA</td>
<td>36.36</td>
<td>30.12</td>
</tr>
</tbody>
</table>

Table 4: Comparison of kNN-MT and DIDST under different volumes of parallel data.

<table>
<thead>
<tr>
<th></th>
<th>kNN-MT</th>
<th>DIDST</th>
<th>Δ BLEU</th>
</tr>
</thead>
<tbody>
<tr>
<td>100k</td>
<td>12.43</td>
<td>13.95</td>
<td>+1.52</td>
</tr>
<tr>
<td>200k</td>
<td>17.20</td>
<td>18.31</td>
<td>+1.11</td>
</tr>
<tr>
<td>500k</td>
<td>21.50</td>
<td>22.49</td>
<td>+0.99</td>
</tr>
<tr>
<td>1M</td>
<td>23.89</td>
<td>24.84</td>
<td>+0.95</td>
</tr>
<tr>
<td>2M</td>
<td>25.52</td>
<td>26.41</td>
<td>+0.89</td>
</tr>
<tr>
<td>4M</td>
<td>27.35</td>
<td>28.05</td>
<td>+0.70</td>
</tr>
</tbody>
</table>

In contrast, DIDST only needs to keep one monolingual datastore of the target language, which can substantially save the cost of disk storage.

5 Discussion

5.1 Effectiveness of Contrastive Training

For the alignment objective, it is also possible to directly minimize the mean squared error (MSE) distance between queries and keys as in knowledge distillation (Hinton et al., 2015) and previous works (Zheng et al., 2021b). We experiment with such an objective and compare it with the proposed NCA objective in Table 3. As can be seen, NCA objective surpasses MSE objective by a large margin across four translation directions, demonstrating the necessity of contrastive alignment.

5.2 Impact of Choices of PLMs

We have shown DIDST brings substantial improvements over baselines with the help of XLMR, a multilingual language model pretrained with an MLM objective. In this section, we present a further study on the relationship between choices of PLMs and translation performance.

Concretely, we investigate two factors that may affect PLM’s performance when used to build datastores: training corpus and training objective. We consider monolingual/multilingual setting for training corpus, and MLM/CLM/DAE for training objectives. For each combination of factors, we select a representative pretrained model, summing up to 6 models: Roberta (monolingual MLM), XLM-R (multilingual + MLM), BART (monolingual DAE), mBART (multilingual DAE), GPT-2 (multilingual CLM), and mGPT (multilingual CLM). We build a datastore from each model and report translation results on IWSLT14 De En dataset in Figure 3a.

Multilingual pretraining is more beneficial than monolingual pretraining. In Figure 3a, multilingual pretrained models show consistent advantages over their monolingual counterparts. This is reasonable since translation requires both source and target language knowledge, while monolingual models can only provide knowledge for target side.

MLM objective is better than DAE and CLM objective. Another interesting trend is that models trained with MLM datastores perform the best among the three training objectives, while models trained with CLM datastore offer nearly no improvements over vanilla kNN-MT.

To gain more insight into this phenomenon, we compare the kNN accuracy of datastores in Figure 3b, and the retrieval accuracy of final models with these datastores in Figure 3c. From Figure 3b, we can see the kNN accuracy of CLM datastores is very low compared to MLM and DAE datastores, which may explain the inferiority of final translation models trained with them. This is because the target token $y_t$ does not exist explicitly in the input of CLM models, making them unable to correlate representation $h_t$ with $y_t$, very well. In contrast, for MLM and DAE models, $y_t$ is explicitly fed to encoders, thus substantially reducing the uncertainty.

Comparing MLM and DAE objectives, we can see that although datastores built from MLM and DAE models can achieve similar kNN accuracy, models trained with DAE datastores achieve relatively lower retrieval accuracy than models with MLM datastores, indicating it is more difficult to align MT representation to DAE models’ representation than to MLM models’ representation. We leave the investigation of reasons to future works.

5.3 Translation Quality Under Different Volumes of Parallel Data

Original kNN-MT builds datastores from the MT model’s representation, making its performance correlate strongly with the quality of MT representation. In contrast, DIDST utilizes datastores coming from external powerful pretrained models. Assuming that representation quality grows with the size of the training corpus, DIDST should bring more improvements when there are fewer parallel
Figure 3: (a) Translation performance, (b) $k$-NN accuracy of datastores and (c) retrieval accuracy of DIDST with datastores built from different PLMs on IWSLT14 De-En task. Red bars denotes the PLM is trained on monolingual corpus, while blue bars denotes the PLM is trained on multilingual corpus.

<table>
<thead>
<tr>
<th>Src</th>
<th>Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>eine befreundete journalistin</td>
<td>a journalist friend has been</td>
</tr>
<tr>
<td>hatte ... .</td>
<td>a friend journalist has been</td>
</tr>
<tr>
<td>kNN-MT</td>
<td>a liberated journalist had</td>
</tr>
<tr>
<td>DIDST</td>
<td>a friend journalist had been</td>
</tr>
</tbody>
</table>

$\begin{array}{c}
\text{Src} \\
\text{Reference} \\
\text{kNN-MT Retr.} \\
\text{DIDST Retr.}
\end{array}
\begin{array}{c}
\text{liberated journalist had ...} \\
\text{friend friend friend friend}
\end{array}

Table 5: Translation and retrieval examples of $k$NN-MT and DIDST. Retr. means the retrieved top 4 examples.

<table>
<thead>
<tr>
<th>Src</th>
<th>Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>wenn dan eine marke wäre ...</td>
<td>if dan were a brand, ...</td>
</tr>
<tr>
<td>kNN-MT</td>
<td>if dan was a brand, ...</td>
</tr>
<tr>
<td>DIDST</td>
<td>if dan were a brand, ...</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Src</th>
<th>Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>was was found was</td>
<td>were were were</td>
</tr>
</tbody>
</table>

we can see that by retrieving more accurate results, DIDST mitigates the negative effect that $k$NN-MT brings, i.e., incorrect words that $k$NN-MT generates due to its retrieval error, despite the translation of the vanilla MT model is correct.

5.4 Case Study

We present several translation examples of baseline model and DIDST in Table 5. We also list the retrieval neighbors at the timestep where models generate incorrect words. From the first example, we can see that by retrieving more accurate results, DIDST mitigates the negative effect that $k$NN-MT brings, i.e., incorrect words that $k$NN-MT generates due to its retrieval error, despite the translation of the vanilla MT model is correct.

We can also observe that DIDST is better at distinguishing similar but not exchangeable words, e.g., personal pronouns in the second example and verbs in different tenses in the third example. We take the idea of contrastive translation evaluation (Rios Gonzales et al., 2017) to quantify this phenomenon better.

A contrastive translation pair contains a source, a reference, and one or more contrastive translations. Contrastive translations are constructed by substituting words in the reference according to specific rules. NMT systems are used to score reference and contrastive translations. If the reference is scored higher than all contrastive translations, then the NMT system passes the contrastive translation test.

Specifically, we consider four kinds of words for substitution: person pronouns ($I$/you/he/she), indicative pronouns ($this/that/these/those$), linking...
verbs (is/are/was/were) and prepositions (in/on/at). The accuracy of contrastive translation test of baseline and DIDST are plot in Figure 4. DIDST consistently outperforms baseline on all word categories, demonstrating the effectiveness of the contrastive training objective.

5.5 Bonus of Disentanglement Between MT Models and Datastores

The disentanglement of MT models and datastores not only allows us to utilize datastores built from more powerful pretrained models to improve translation performance, but also brings several desirable properties in real-world scenarios, which we will discuss in this section.

Personalization Being able to adapt to new domains rapidly is very important for industrial applications. Although original $k$NN-MT shows strong performance for domain adaptation by directly switching to an in-domain datastore, the creation of such datastore requires a large amount of in-domain parallel sentences, which may not be available in many scenarios. In contrast, DIDST only requires in-domain monolingual sentences for building datastore. This significantly eases the data demand and paves the way to more fine-grained domain adaptation — in the extreme case, personalizing the translation system for every single user.

Data privacy The deployment of original $k$NN-MT needs to encode the user’s private data as datastores using the serving MT model. This either requires the server to transmit its model to the user or requires the user to transmit their data to the server, which can potentially harm the privacy of the server of the user. DIDST avoids this problem by enabling the user to encode their data using publicly available pretrained models, and then transmit the encoded datastore to the server.

6 Related Work

6.1 Retrieval-Enhanced Neural Machine Translation

RE-NMT, which enhances neural machine translation system with retrieval mechanism, has been shown effective in many previous works. Most RE-NMT methods are based on sentence or $n$-gram level retrieval (Zhang et al., 2018; Gu et al., 2018; Xia et al., 2019; He et al., 2021; Cai et al., 2021). Our work differs from theirs in that we retrieve at the token level, which significantly eases the data sparsity problem and enables our method to retrieve more relevant examples. There is also a line of work that retrieves at the token level. Khandelwal et al. (2019) firstly propose to retrieve similar tokens based on the similarity of decoding representation. Zheng et al. (2021a) learn a Meta-$k$ network to select the optimal $k$ at different timestep. Jiang et al. (2021) propose to smooth the distribution of retrieved results and learn the interpolation weight automatically. Meng et al. (2021) propose to prune the datastore firstly to accelerate the retrieval process. All these works build their datastores using the original MT representation, while we explore ways to leverage powerful pretrained models for datastore creation.

The most relevant work to ours is Zheng et al. (2021b), which adapts the original MT model to generate monolingual datastores for retrieving. However, the datastore generator in their work is still learned from the limit size of MT corpus, thus facing the same representation quality problem as $k$NN-MT.

6.2 Pretrained Language Models for Neural Machine Translation

Many works attempts to make use of large-scale pretrained language models for NMT. Conneau and Lample (2019); Conneau et al. (2020); Liu et al. (2020); Xu et al. (2021) directly initialize NMT models with parameters of multilingual PLMs, and achieve substantial improvement in both supervised and unsupervised machine translation. Zhu et al. (2019); Yang et al. (2020) fuse BERT’s representation to NMT models in a dynamic way. Chen et al. (2020) first finetune BERT to be a conditional MLM model, and then distill the knowledge of it to NMT models. Unlike previous works, PLMs in our work are used to create better datastores.

7 Conclusion

We introduce a framework that disentangle datastores and translation models in $k$NN-MT. Pretrained models can be efficiently leveraged to create datastores, and a unified model that conduct retrieving and translating is trained via a novel contrastive objective. Experiments demonstrate the effectiveness of our method. Our work opens the gate to make use of pretrained models for RE-NMT, and investigating on how to better create datastores from PLMs is a promising future work.
References


