When Is Retrieved Knowledge Helpful? Towards Explainable and Efficient Memory for kNN-MT Domain Adaptation

Anonymous ACL submission

Abstract

Retrieval-based kNN-MT presents a new paradigm for domain adaptation by building a large, redundant datastore to save all target language token occurrences in the parallel corpus. In this paper, we investigate the interpretability issue: when is the retrieved datastore entry helpful for NMT? We explore this issue from the point of local correctness (LAC), including entry correctness and neighborhood correctness. Empirical study shows that our investigation successfully finds potential weak points for NMT model, where assistance from the datastore is needed. Based on the analysis, we propose a LAC pruning algorithm to cut off redundant datastore entries. Experiments on six diverse target domains and two language-pairs show that our method can prune 15% to 45% datastore entries while retaining comparable translation performance. Pruned datastore not only occupies less disk storage but also accelerate the inference speed, which makes kNN-MT models more practical.

1 Introduction

Domain adaptation is a popular topic that aims at adapting pre-trained NMT models to a target domain (Chu et al., 2017; Thompson et al., 2019; Hu et al., 2019; Zhao et al., 2020; Zheng et al., 2021a). Fine-tuning (Luong and Manning, 2015) has been the de facto way for adaptation, which implicitly memorizes target domain training data in the neural network parameters. However, fine-tuning suffers from the catastrophic forgetting problem (McCloskey and Cohen, 1989; French, 1999).

Recently, Khandelwal et al. (2021) propose kNN-MT, a new paradigm for non-parametric domain adaptation. kNN-MT explicitly extracts translation knowledge in target domain training data into a *key-value* datastore. With the datastore, the pre-trained NMT model can generate translations better matching the target domain. The kNN-MT framework circumvents the necessity to update the parameters of the pre-trained NMT model and enables quick adaptation by switching datastores.

Basically, kNN-MT incorporates the symbolic datastore to assist the NMT model (Khandelwal et al., 2021; Zheng et al., 2021a; Jiang et al., 2021). However, the construction of datastore simply stores all the target tokens in the parallel data, without considering the capability of the neural model. As a result, the datastore is usually huge in size and possibly redundant.

This paper conducts investigation on the interpretability issue: *when is the retrieved knowledge helpful for NMT adaptation?* Intuitively, this external knowledge is only needed when the pre-trained NMT model fails. We explore this issue from the point of *local correctness*, which include two aspects, the entry itself and the entry's neighborhood.

We split the datastore entries into two categories, namely *known* and *unknown*, according to whether the NMT could make correct translation for the entry itself. Empirical study shown that, on average, 66.7% of the entries are known to the NMT model (on OPUS dataset, four domains (Section 3)). We propose a *knowledge margin* metric to evaluate the translation correctness of the NMT model in a neighborhood in the representation space, and find that the NMT model often fails when the *knowledge margin* is small.

Based on the analyses, we propose a simple and explainable datastore pruning algorithm LAC with the help of *knowledge margin*, which simply remove *known* entries with a higher margin value (Section 4). These entries are less useful, because the NMT model could already translate them correctly.

For evaluation, we conduct experiments on six diverse target domains in two language pairs. The experiment results show that our method can prune 15%∼45% entries in different domains' datastore while retaining comparable translation performance. In contrast, pruning random *known* entries

or pruning low *knowledge margin* entries will harm the translation performance (Section 6). Meanwhile, the pruned datastore not only occupies less disk storage, but also has less latency during inference, making kNN-MT models more practical.

2 Background

Recent years have witnessed the great thrive in kNN-MT (Khandelwal et al., 2021; Zheng et al., 2021a,b; Jiang et al., 2021). Basically, kNN-MT expects the constructed datastore to provide helpful translation knowledge for the pre-trained NMT model M_θ, to perform target domain translation. In this section, we briefly introduce kNN-MT and its state-of-the-art variant, adaptive kNN-MT (Zheng et al., 2021a).

2.1 Building a Domain Specific Datastore

Given target domain bilingual corpus $\mathcal{C}$, kNN-MT first creates a datastore $\mathcal{D}$ to save translation knowledge contained in $\mathcal{C}$. More specifically, all translation pairs of $\mathcal{C}$ are fed into the frozen pre-trained NMT model for decoding with teacher-forcing (Williams and Zipser, 1989). At decoding time step t, the hidden state from the last decoder layer $h(\mathbf{x}, \mathbf{y}_{<t})$ is taken as *key* and the t-th target token y_t is taken as *value*, resulting a *key-value* pair. For the entire corpus, the datastore $\mathcal{D}$ is consist of *key-value* pairs:

$$\mathcal{D} = \{(h(\mathbf{x}, \mathbf{y}_{<t}), y_t) \mid \forall y_t \in \mathbf{y}, (\mathbf{x}, \mathbf{y}) \in \mathcal{C}\}, \quad (1)$$

where $\mathbf{y}_{<t}$ denotes previous tokens in the sequence $\mathbf{y}$. Each entry in the datastore explicitly memorizes the following translation knowledge: generating the *value* token at the decoder hidden state *key*. And the datastore covers all target language token occurrences.

2.2 Translating with the Datastore

During inference, given a source language sentence $\mathbf{x}$, kNN-MT simultaneously leverages M_θ and $\mathcal{D}$ to generate target language translation $\hat{\mathbf{y}}$=$\{\hat{y}_1, \hat{y}_2, \cdots, \hat{y}_{|\hat{\mathbf{y}}|}\}$. More specifically, at decoding time step t, kNN-MT queries the datastore with the decoder hidden state $h(\mathbf{x}, \hat{\mathbf{y}}_{<t})$ generated by M_θ. The nearest neighbors of the query $\mathcal{N}_k = \{(h^j, y^j)\}_1^k$ are retrieved, which are k entries with keys closest to the query. These retrieved knowledge are converted into a distribution over the vocabulary:

$$p_{k\text{NN}}(y_t|\mathbf{x}, \hat{\mathbf{y}}_{<t}) \propto \sum_{(h^j, y^j)\in\mathcal{N}_k} \mathbb{1}_{y_t=y^j} \exp\left(\frac{-d(h^j, h(\mathbf{x}, \hat{\mathbf{y}}_{<t}))}{T}\right), \quad (2)$$

Then, kNN-MT interpolates $p_{k\text{NN}}$ with the pre-trained NMT model's output distribution as the final translation distribution:

$$p(y_t|\mathbf{x}, \hat{\mathbf{y}}_{<t}) = \lambda\, p_{k\text{NN}}(y_t|\mathbf{x}, \hat{\mathbf{y}}_{<t}) + (1-\lambda)\, p_{\text{NMT}}(y_t|\mathbf{x}, \hat{\mathbf{y}}_{<t}) \quad (3)$$

The complete translation $\hat{\mathbf{y}}$ can be generated by beam search.

2.3 Adaptive kNN-MT

For vanilla kNN-MT, the selection of hyper-parameters, such as k or λ, highly affect the final translation performance, which is less stable across languages or domains. Adaptive kNN-MT retrieves a fixed number of entries, k_a, and uses a lightweight meta-k neural network to dynamically determine the usage of retrieved entries, which avoids the tuning of hyper-parameters and achieves a more stable performance (Zheng et al., 2021a).

3 When is the Retrieved Knowledge Helpful?

Although less accurate, the pretrained NMT model could perform translation without the datastore. This fact suggests that the pre-trained NMT model knows at least some bilingual knowledge of the target domain. Intuitively, the datastore is only helpful when NMT fails, otherwise the NMT itself will be sufficient to generate correct translation. However, it is hard to predict when the NMT will fail. We conduct investigation from two perspectives, namely the entry itself and entries in the neighborhood. Empirical analyses are made to reveal the facts.

3.1 Settings for Empirical Analyses

We follow Zheng et al. (2021a) and consider four domains in German-English OPUS dataset (Tiedemann, 2012) as target domains[1]. Table 1 lists statistics of four domains[2]. For pre-trained NMT model,

[1]Detailed description of OPUS domains can be found in Appendix A.

[2]We use the dataset re-split by Aharoni and Goldberg (2020), which removes the overlap between training, development and test sets.

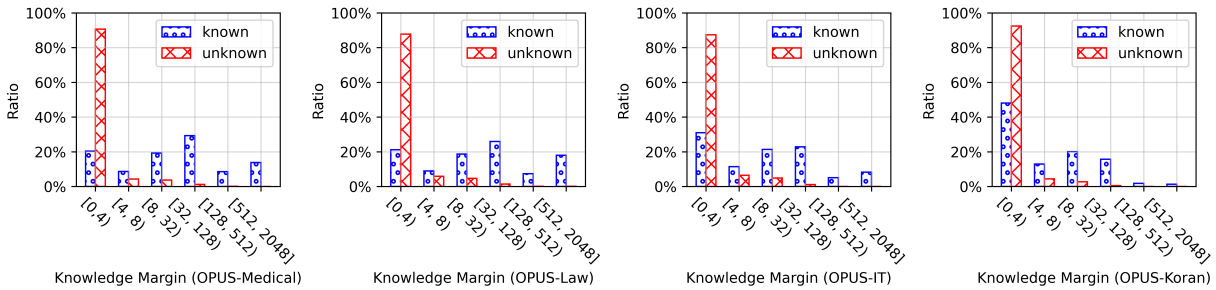

Figure 1: The ratio distribution on different *knowledge margin* for *known* and *unknown* entries on four OPUS domains

	OPUS-Medical	OPUS-Law	OPUS-IT	OPUS-Koran
Train	248,099	467,309	222,927	17,982
Dev	2,000	2,000	2,000	2,000
Test	2,000	2,000	2,000	2,000

Table 1: Number of sentences of the OPUS dataset. "Train", "Dev", "Test" denote training, development, test set, respectively.

	OPUS-Medical	OPUS-Law	OPUS-IT	OPUS-Koran
known	5,070,607	14,803,149	2,514,757	294,094
unknown	1,844,966	4,287,906	1,093,974	230,677
$\|\mathcal{D}\|$	6,915,573	19,091,055	3,608,731	524,771
known ratio	73.32%	66.74%	69.69%	56.04%

Table 2: The statistics of the *known* and *unknown* entries for the pre-trained NMT model on four OPUS domains' training set. The number of entries and the ratio of *known* entries are listed.

we use the winner model of WMT'19 German-English news translation task [3] (Ng et al., 2019) . The datastore for each domain is constructed on the corresponding training set with the pre-trained NMT model.

3.2 Known v.s. Unknown for Entry Correctness

The relationship between the NMT model and the datastore is difficult to describe directly. However, as the datastore is consist of entries from the parallel corpus, it is easier to check whether the NMT model knows a specific entry. We call a entry as a *known* entry for the NMT model, if the NMT model could predict it correctly; and *unknown*, otherwise.

This can be efficiently accomplished by an extra evaluation during decoding with teacher-forcing. More specifically, at each time step t of the teacher-forcing process, we not only record the hidden states $h(\mathbf{x}, \mathbf{y}_{<t})$ and the correct target token y_t, but also evaluate the prediction of the NMT model $\hat{y}_t$, which is the target token with highest probability $p_{\text{NMT}}(y_t|\mathbf{x}, \hat{\mathbf{y}}_{<t})$ (Equation 4).

$$(h(\mathbf{x}, \mathbf{y}_{<t}), y_t) \text{ is } \begin{cases} \text{known}, & \text{if } \hat{y}_t = y_t \\ \text{unknown}, & \text{o.w.} \end{cases} \quad (4)$$

Thus, the datastore entries are divided into two categories: *known entries* and *unknown entries*. We collect statistics about the two categories of entries and report results in Table 2. The results show that 56%∼73% (averaging 66.7%) of datastore entries is implicitly contained in the pre-trained NMT model. This high ratio indicates that a large amount of datastore entries may be redundant.

It is straight forward that, these *unknown* entries are more helpful for the NMT model, because it cannot make correct translation about these entries. However, *known* entries may still be usefull. Simply removing those known entries lowers the over-all performance of kNN-MT (Table 7 and 8). One possible explanation is the model's ability for generalization. Although the NMT model makes correct prediction on the target domain training data, it may still fail during inference, where the context could be similar but different.

3.3 The Knowledge Margin Metric for Neighborhood Correctness

To further examine the NMT model's generalization ability of translating a datastore entry, we propose a metric called *knowledge margin*, denoted as km, which evaluates the NMT model's prediction on a neighborhood in the representation space.

Given an entry (h, y), its neighborhood is defined by its k nearest neighbors in the datastore $\mathcal{N}_k(h) = \{(h^j, y^j)\}_1^k$. The *knowledge margin* of

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

	OPUS-Medical	OPUS-Law	OPUS-IT	OPUS-Koran
Spearman's ρ	0.7764	0.7782	0.7337	0.6497
Kendall's τ	0.5824	0.5826	0.5475	0.4987

Table 3: Correlation between *knowledge margin* and correct target token prediction probability on four OPUS domains.

	OPUS-Medical	OPUS-Law	OPUS-IT	OPUS-Koran
$km \geq 8$	97.11%	97.28%	95.01%	93.81%
$km < 8$	44.74%	52.34%	50.52%	44.16%

Table 4: The NMT model's prediction accuracy at positions with different knowledge margin in OPUS domains' development set.

the entry is defined as:

$$km(h) = \arg\max_t \forall (h^j, y^j) \in \mathcal{N}_t(h) \text{ is known} \quad (5)$$

Intuitively, km is the maximum neighborhood of the entry where the NMT could make correct translation. In the extreme case, all the neighbor entries are known by NMT, the margin is k. Please note that, the definition of *knowledge margin* applies for any point in the representation space, because for each point (e.g. an actual query q during inference), its neighborhood could be defined by querying the datastore $\mathcal{N}_k(q)$.

For analysis, we plot the distribution of *knowledge margin* for *known* and *unknown* entries with $k = 2048$ (in Figure 1). The distributions on four OPUS domains show the same trends. Most *unknown* entries has a very low *knowledge margin*, e.g. more than 80% of *unknown* entries have a margin value between 0 and 4. In contrast, the distribution for *known* entries is more diverse.

We further measure the correlation between *knowledge margin* value and the probability of correct target tokens from the NMT model. Results in Table 3 show a similar positive correlation on all four domains, indicating that for entries with higher *knowledge margin* value, the NMT model is more likely to generate the correct translation. On the other hand, for entries with a smaller *knowledge margin* value, the NMT model is more likely to fail, making the entries more important to obtain correct translation.

3.4 NMT Generalizes Better on Higher Knowledge Margin Regions

To verify the relation between *knowledge margin* and NMT performance, we conduct experiments on the development set for each domain. For each token y_t in the dev set, we perform teacher-forcing until time step $t-1$ and query the datastore for the neighborhood at time step t[4]. We evaluate the *knowledge margin* of the neighborhood and the prediction accuracy of the NMT model.

Table 4 shows the results. For tokens in a higher margin neighborhood, i.e. $km \geq 8$, the prediction accuracy of the NMT model is higher than 93%. In contrast, for tokens in a lower margin neighborhood, the accuracy is only around 50%. This is a strong evidence that the NMT model generalizes better on regions with higher margin.

On the other hand, the NMT model needs to rely on the retrieved knowledge from datastore to make right predictions in the regions with lower margin. Table 5 shows an example of such cases. The retrieved entries are *unknown*, so *knowledge margin* is 0. The NMT model actually fails to generate the last subword of "Cyankoit"[5] as expected. With the help of entries memorized in the datastore, the correct translation "it" is provided to improve the translation of the NMT model.

4 Datastore Pruning by Local Correctness

Existing kNN-MT researches construct datastore for each target token in the bilingual corpus, resulting very huge datastores. In the previous section, we show that local correctness on the entry and the neighborhood are good indicators for the performance of the NMT model. Therefore, it is natural to prune datastore entries where the NMT model may have good performance. We propose LAC, a pruning algorithm based on **L**oc**A**l **C**orrectness, to cut off *known* entries with a high *knowledge margin*.

The procedure is described in Algorithm 1. There are two steps. In the first step, each entry (h, y) in the datastore $\mathcal{D}$ is checked. If (h, y) is known and the *knowledge margin* of (h, y) is greater than the threshold k_p, the entry is collected as the pruning candidates. In the second step, these pruning candidates are randomly selected and get

[4]Following Zheng et al. (2021a), we set number of retrieved entries, k_a, as 8.

[5]"Cyankoit" is a drug name, which contains the active substance hydroxocobalamin (vitamin B12).

Source sentence (x): Wie ist Cy@@ an@@ ko@@ it anzu@@ wenden ?
Previous translation ($y_{<t}$): How to use Cy@@ an@@ ko@@
Correct target token ($\hat{y}_t$): it
NMT's prediction (y_t): ite

No.	Retrieved Keys: source (x)	Retrieved Keys: target ($y_{<t}$)	Retrieved Values	Known
1	Wie wird Cy@@ an@@ ok@@ it ange@@ wendet ?	How is Cy@@ an@@ ko@@	it	✗
2	Sie erhalten Cy@@ an@@ ok@@ it als In@@ fusion in eine V@@ ene .	You will have Cy@@ an@@ ok@@	it	✗
3	Wo@@ für wird Cy@@ an@@ ok@@ it ange@@ wendet ?	What is Cy@@ an@@ ok@@	it	✗
4	Wel@@ ches Risiko ist mit Cy@@ an@@ ok@@ it verbundenn ?	What is the risk associated with Cy@@ an@@ ok@@	it	✗
5	Die folgenden Neben@@ wirkungen wurden in Verbindung mit der Anwendung von Cy@@ an@@ ok@@ it berichtet .	The following un@@ desi@@ rable effects have been reported in association with Cy@@ an@@ ok@@	it	✗
6	Warum wurde Cy@@ an@@ ok@@ it zugelassen ?	Why has Cy@@ an@@ ok@@	it	✗
7	Beson@@ dere Vorsicht bei der Anwendung von Cy@@ an@@ ok@@ it ist erforderlich	Take special care with Cy@@ an@@ ok@@	it	✗
8	Wie wirkt Cy@@ an@@ ok@@ it ?	How does Cy@@ an@@ ok@@	it	✗

Table 5: An example of kNN-MT where the retrieved entries do help the NMT model (sentence are tokenized into subwords). At the current time step, all retrieved entries are *unknown*, so *knowledge margin* is 0. The prediction of NMT is highly likely to be wrong and relying on the retrieved knowledge is preferable.

Algorithm 1 Datastore Pruning by LAC

Input: datastore $\mathcal{D}$, the *knowledge margin* threshold k_p, the pruning ratio r
Output: pruned datastore $\mathcal{D}$

$candidates \leftarrow \emptyset$ ▷ step 1: collect
for each entry (h, y) in $\mathcal{D}$ **do**
  if (h, y) is known **and** $km(h) \geq k_p$ **then**:
    $candidates \leftarrow candidates \cup (h, y)$
  end if
end for
repeat ▷ step 2: drop
  randomly select entry (h, y) from $candidates$
  remove (h, y) from $\mathcal{D}$
until pruning ratio r is satisfied
return $\mathcal{D}$

removed from $\mathcal{D}$ until the required pruning ration is reached.

Since our method does not need to train any additional neural networks, it can be easily implemented. The pruned datastore can be directly used in any kNN-MT framework, such as adaptive kNN-MT.

5 Experiment Setup

We conduct datastore pruning for 6 different domains from 2 language pairs.

5.1 Data and Processing

Same as previous sections, we take 4 domains in the OPUS dataset for German-English experiments (Table 1). We also take "Law" and "Thesis"

	UM-Law	UM-Thesis
Train	216,000	296,000
Dev	2,000	2,000
Test	2,000	2,000

Table 6: Detailed statistics of UM dataset. We report the sentence number of each subset. "Train", "Dev", "Test" denote training, development, test set respectively.

domains in the UM dataset[6] for Chinese-English experiments (Tian et al., 2014). Statistics can be found in Table 6.

For preprocessing, we use *moses*[7] toolkit to tokenize German and English corpus and *jieba*[8] to tokenize Chinese corpus. For German-English corpus, we apply byte pair encoding[9] (BPE) (Sennrich et al., 2016) with Ng et al. (2019)'s released BPE-code. For Chinese-English corpus, we learn and apply BPE with 32k merge operations.

5.2 Pre-trained NMT Model

For German-English tasks, we use the winner model (270M) of WMT'19 German-English news

[6]Detailed description of UM domains can be found in Appendix A. We split the original training set into training, development, test set because there is no development set provided in the original dataset and there also exists an overlap between original training and test sets.

[7]https://github.com/moses-smt/mosesdecoder

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

[9]https://github.com/rsennrich/subword-nmt

	OPUS-Medical			OPUS-Law			OPUS-IT			OPUS-Koran		
	Ratio	BLEU↑	COMET↑	Ratio	BLEU↑	COMET↑	Ratio	BLEU↑	COMET↑	Ratio	BLEU↑	COMET↑
Base	-	39.73	0.4665	-	45.68	0.5761	-	37.94	0.3862	-	16.37	-0.0097
Finetune	-	58.09	0.5725	-	62.67	0.6849	-	49.08	0.6343	-	22.40	0.0551
Adaptive kNN	0%	57.98	0.5801	0%	63.53	0.7033	0%	48.39	0.5694	0%	20.67	0.0364
Random	45%	54.08*	0.5677*	45%	58.69*	0.6690*	40%	45.54*	0.5314*	25%	20.36	0.0434
Known	45%	56.44	0.5691*	45%	61.61*	0.6885*	40%	45.93*	0.5563*	25%	20.35	0.0338
All Known	73%	42.73*	0.4926*	66%	51.90*	0.6200*	69%	40.93*	0.4604*	56%	17.76*	0.0008*
LAC (ours)	45%	57.66	0.5773	45%	63.22	0.6953*	40%	48.22	0.5560	25%	20.96	0.0442

Table 7: Pruning Effect on four OPUS domains. "Ratio" denotes the pruning ratio. Higher "BLEU" and "COMET" scores indicates better translation quality. "*" means that performance decline is statistically significant ($p < 0.05$).

translation task, which is based on the big Transformer architecture (Vaswani et al., 2017) with 8192 FFN size. For Chinese-English tasks, we train the base Transformer model (88M) on CWMT'17 Chinese-English Dataset[10] (9 million sentence pairs) from scratch because there is no publicly available Chinese-English pre-train NMT model.

5.3 Baselines and Pruning Methods

For translation adaptation, we report performance for the following systems: the general NMT model (Base), the general model finetuned on each domain (Finetune), adaptive kNN with full datastores built for each domain on their training set (Adaptive kNN), respectively.

The following pruning methods are applied and compared: randomly pruning entries (**Random**), randomly pruning known entries (**Known**), pruning all known entries (**All Known**), **LAC**.

5.4 Implementation Details

We implement adaptive kNN-MT with Zheng et al. (2021a)'s released code and script[11] based on *fairseq*[12] (Ott et al., 2019). Due to the large space of hyper-parameters, we follow Zheng et al. (2021a) to set k_a as 8 when training adaptive kNN-MT models for most experiments, and report pruning performance under different k_a in Section 6.5. During inference, we set beam size as 5 and length penalty as 1.0.

For implementing LAC, the hyper-parameter k_p in Algorithm 1 implicitly determines the maximum number of entries that are allowed to be pruned. So we tune k_p among the subset of {4, 8, 16, 32} when given different pruning ratio r.

After buiding the datastore, we follow previous kNN-MT works (Khandelwal et al., 2021; Zheng et al., 2021a) and use Faiss[13] index (Johnson et al., 2019) to represent the datastore and accelerate nearest neighbors search.

All models are trained on a single NVIDIA Titan RTX for comparison of performance.

To evaluate translation performance, we report case-sensitive detokenized BLEU (Papineni et al., 2002) calculated by *sacrebleu*[14] and COMET (Rei et al., 2020) calculated by publicly available *wmt20-comet-da*[15] model. Statistical significance test (Koehn, 2004) are conducted as well.

6 Experiment Results and Analysis

6.1 How Much Datastore Can We Prune?

We perform datastore pruning for different domain and report the largest pruning ratio without significant performance degradation on the test set.

We present experiment results on OPUS domains in Table 7. For the reference, the general domain NMT model does not translate well on target domains (Base). Finetuning and Adaptive kNN are two alternatives for adaptation, which achieve comparable performances. All pruning experiments are performed based on Adaptive kNN.

As shown in Table 7, compared with using full datastore (Adaptive kNN), our method (**LAC**) achieves comparable performance while cutting off at least 25% entries of the datastore. On the two largest "OPUS-Medical" and "OPUS-Law" domains, our method successfully prune 45% datastore (millions of *key-value* pairs). Excellent pruning performance also validates our analysis about *local correctness*.

Simply pruning all *known* entries results in a significant drop of performance (**All Known**). Pruning *known* entries to the same ratio as LAC also

[10] http://nlp.nju.edu.cn/cwmt-wmt

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

[12] https://github.com/pytorch/fairseq

[13] https://github.com/facebookresearch/faiss

[14] https://github.com/mjpost/sacrebleu

[15] https://github.com/Unbabel/COMET

	UM-Law			UM-Thesis		
	Ratio	BLEU↑	COMET↑	Ratio	BLEU↑	COMET↑
Base	-	30.36	0.3857	-	13.13	-0.0442
Finetune	-	58.55	0.6019	-	17.46	-0.0262
Adaptive kNN	0%	58.64	0.6017	0%	17.49	-0.0146
Random	30%	53.78*	0.5661*	15%	16.14*	-0.0280*
Known	30%	56.92*	0.5762*	15%	17.25	-0.0143
All Known	63%	46.45*	0.4720*	47%	15.33*	-0.0525*
LAC (ours)	30%	58.65	0.6056	15%	17.52	-0.0122

Table 8: Pruning Effect on two UM domains. "Ratio" denotes the pruning ratio. Higher "BLEU" and "COMET" scores indicate better translation quality. "*" means that performance decline is statistically significant ($p < 0.05$).

lead to degradation (**Known**). These comparisons indicates that the entry correctness alone is not enough for identifying NMT performance, demonstrating the necessity of the neighborhood correctness analysis with *knowledge margin*.

The results on UM domains are presented in Table 8. The datastore could be pruned by 30% for "UM-Law" and 15% for "UM-Thesis" Datastore without any sacrifice in translation performance. The other findings are similar with those in German-English experiments.

6.2 Hyper-parameter Tuning of LAC

We also examine the effect of hyper-parameter in LAC. Figure 2 plots BLEU scores of adaptive kNN-MT models under different pruning ratios on different domains' development sets. We can observe that trends are mostly similar in different domains. Pruning by LAC achieves the best performance over the other two baselines and the performance is more stable even with a higher pruning ratio. The performance of **Known** is also worse in most of the settings, demonstrating the necessity of considering *knowledge margin* when we identify each entry's importance.

When tuning the hyperparameter k_p in our proposed method, we can see a trade-off between BLEU score and the pruning ratio. Large k leads to a small sacrifice of BLEU score but a lower pruning ratio. Small k allows us to prune more entries but causes significant BLEU scores decline after a specific threshold ratio. For example, when $k = 4$, it is allowed to prune 55% "OPUS-Medical" datastore, but translation performance declines drastically after the pruning ratio reaches 50%.

We choose the top-right point[16] in each subfigure as the best-performed setting for each domain, which are used in other experiments.

[16]hyper-parameter values of these points are reported in Appendix C.

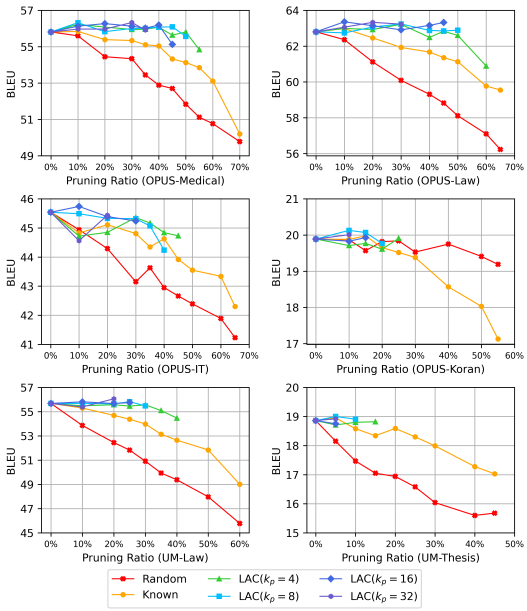

Figure 2: BLEU scores of adaptive kNN-MT models with different scale datastore on four OPUS domains' developments set. Different symbols represent different ways of pruning datastore.

6.3 Pruned Datastore Occupies Less Disk Storage

We compare the disk storage difference between our pruned datastore and full datastore. Table 9 lists the disk storage of both key file and index file. Compared with full datastore, pruned datastore occupies less disk storage. For the largest "OPUS-Law" datastore, disk storage of key file and index file can be reduced by 16782 MB and 589 MB respectively. Pruned datastore makes kNN-MT models more practical.

6.4 Pruned Datastore Causes Less Inference Latency

We compare the inference speed of different models on "OPUS-Law" domain's test set (Table 10). Full datastore of "OPUS-Law" domain is the largest among six domains, which introduces highest computational cost during inference. We repeat each experiment three times on a single NVIDIA Titan-RTX and report averaged inference speed. We can see that the adaptive kNN-MT model with pruned datastore decodes faster than the model with full datastore. However, due to the acceleration mechanism of Faiss index, the speed improvement is minor. There is still a large speed gap

	OPUS-Medical		OPUS-Law		OPUS-IT		OPUS-Koran		UM-Law		UM-Thesis	
	Key	Index	Key	Index	Key	Index	Key	Index	Key	Index	Key	Index
Full Datastore	13,507	492	37,290	1,328	7,069	265	1,025	54	9,546	680	11,388	810
LAC (ours)	7,429	279	20,508	739	4,229	166	769	45	6,682	479	9,680	690

Table 9: Disk storage (MB) comparsion between pruned datastore and full datastore. "Key" and "Index" column denotes the disk storage of key file and index file respectively.

	Batch=1		Batch=16		Batch=32		Batch=64	
	sents/s	tokens/s	sents/s	tokens/s	sents/s	tokens/s	sents/s	tokens/s
Base	3.45	135.40	35.29	1385.47	50.58	1985.70	60.38	2374.29
Full Datastore	2.25	89.72	11.59	461.83	13.84	551.70	15.10	601.70
LAC (ours)	2.27	90.51	11.85	472.39	14.37	572.89	15.77	628.89

Table 10: Inference speed of different models under different batch size. "sents/s" and "tokens/s" denotes the number of generated sentences and tokens per second respectively.

OPUS-Law	$k_a = 4$	$k_a = 8$	$k_a = 16$	$k_a = 32$
Full Datastore	63.31	63.53	63.56	63.33
LAC (ours)	62.93	63.22	63.18	63.22

Table 11: Pruning performance under different k_a.

OPUS-Law	10%	20%	30%	40%	45%
reverse pruning	-1.91	-4.00	-6.19	-8.71	-10.38
LAC (ours)	+0.00	-0.19	+0.18	-0.21	-0.31

Table 12: Translation performance difference compared with Adaptive kNN (BLEU) under different pruning ratios.

between kNN-MT and base models. It remains a challenging problem for researchers to optimize the inference speed of kNN-MT models in the future.

6.5 Pruning Effect is Insensitive to Hyperparameter k_a

To demonstrate the robustness of our pruned datastore, after pruning datastore, we train adaptive kNN-MT models with different hyperparameter k_a and evaluate their translation performance (BLEU) on "OPUS-Law" domain's test set (Table 11). Results show that our pruned datastore enjoys consistent performance under different k_a.

6.6 Remaining Knowledge in the Pruned Datastore Is Valuable

For ablation study, we prune entries with a reversed strategy, i.e. (h, y) is *unknown* or the knowledge margin of (h, y) is less than k_p. Table 12 shows comparison results. We can see that pruning entries with reverse strategy suffer more significant performance decline at different pruning ratios, demonstrating that our design is reasonable. Both unknown entries and small knowledge margin entries are valuable knowledge for domain adaptation.

7 Related Work

Less attention have been paid to the research of interpretability of kNN-MT. But we do notice a last-day release on arxiv. Wang et al. propose to train a cluster-based compact network to prune the datastore. On four OPUS domains, they prune 10% datastore at the cost of 0.9 BLEU in average. Our method not only safely prunes more entries, but also enjoys better interpretability.

8 Conclusion

It is interesting to explore how a neural model and a symbolic model works together. In this paper, we propose to analyze the local correctness of the neural model's predictions to identify the conditions where the neural model may fail. By introducing a *knowledge margin* metric to measure the local correctness, we find that the NMT model often fails when the *knowledge margin* is small. These results provide support for building a more explainable machine translation system.

Based on analyses, we can safely prune the datastore with the proposed LAC method. Empirically, the datastore could be successfully pruned up to 45% while retaining translation performance. This results validate our earlier findings about the local correctness and translation failures.

Our method is general to all kNN-MT variants and easy to implement. Future directions maybe adapting the neural models while compressing the symbolic datastore, so the two systems can better collaborate with each other.

9 Ethical Considerations

In kNN-MT works, the symbolic datastore helps adaptation but also introduce privacy concerns. Since kNN-MT explicitly saves all target language tokens in the datastore, there is a risk of privacy leakage. At the same time, kNN-MT allows developers to directly erase privacy entries in the datastore, which may make it easy to protect privacy.

References

Roee Aharoni and Yoav Goldberg. 2020. Unsupervised domain clusters in pretrained language models. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL)*.

Chenhui Chu, Raj Dabre, and Sadao Kurohashi. 2017. An empirical comparison of domain adaptation methods for neural machine translation. In *Annual Meeting of the Association for Computational Linguistics (ACL)*.

Robert M French. 1999. Catastrophic forgetting in connectionist networks. *Trends in cognitive sciences*.

Junjie Hu, Mengzhou Xia, Graham Neubig, and Jaime G Carbonell. 2019. Domain adaptation of neural machine translation by lexicon induction. In *Annual Meeting of the Association for Computational Linguistics (ACL)*.

Qingnan Jiang, Mingxuan Wang, Jun Cao, Shanbo Cheng, Shujian Huang, and Lei Li. 2021. Learning kernel-smoothed machine translation with retrieved examples. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

Jeff Johnson, Matthijs Douze, and Hervé Jégou. 2019. Billion-scale similarity search with gpus. *IEEE Transactions on Big Data*.

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

Philipp Koehn. 2004. Statistical significance tests for machine translation evaluation. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (ACL)*.

Minh-Thang Luong and Christopher D Manning. 2015. Stanford neural machine translation systems for spoken language domains. In *International Workshop on Spoken Language Translation (IWSLT)*.

Michael McCloskey and Neal J Cohen. 1989. Catastrophic interference in connectionist networks: The sequential learning problem. In *Psychology of learning and motivation*. Elsevier.

Nathan Ng, Kyra Yee, Alexei Baevski, Myle Ott, Michael Auli, and Sergey Edunov. 2019. Facebook FAIR's WMT19 news translation task submission. In *Proceedings of the Conference on Machine Translation (WMT)*.

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

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL)*.

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

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016. Neural machine translation of rare words with subword units. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL)*.

Brian Thompson, Jeremy Gwinnup, Huda Khayrallah, Kevin Duh, and Philipp Koehn. 2019. Overcoming catastrophic forgetting during domain adaptation of neural machine translation. In *Proceedings of the Conference of the North American Chapter of the Association for Computational Linguistics (NAACL)*.

Liang Tian, Derek F Wong, Lidia S Chao, Paulo Quaresma, Francisco Oliveira, Yi Lu, Shuo Li, Yiming Wang, and Longyue Wang. 2014. Um-corpus: A large english-chinese parallel corpus for statistical machine translation. In *Proceedings of international conference on language resources and evaluation (LREC)*.

Jörg Tiedemann. 2012. Parallel data, tools and interfaces in OPUS. In *Proceedings of the Eighth International Conference on Language Resources and Evaluation (LREC)*.

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 *Advances in Neural Information Processing Systems (NeurIPS)*.

Dexin Wang, Kai Fan, Boxing Chen, and Deyi Xiong. Efficient cluster-based k-nearest-neighbor machine translation. *arXiv preprint arXiv:2204.06175*.

Ronald J Williams and David Zipser. 1989. A learning algorithm for continually running fully recurrent neural networks. *Neural Computation*.

Yang Zhao, Jiajun Zhang, Yu Zhou, and Chengqing Zong. 2020. Knowledge graphs enhanced neural machine translation. In *International Joint Conference on Artificial Intelligence (IJCAI)*.

Xin Zheng, Zhirui Zhang, Junliang Guo, Shujian Huang, Boxing Chen, Weihua Luo, and Jiajun Chen. 2021a. Adaptive nearest neighbor machine translation. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL)*.

Xin Zheng, Zhirui Zhang, Shujian Huang, Boxing Chen, Jun Xie, Weihua Luo, and Jiajun Chen. 2021b. Non-parametric unsupervised domain adaptation for neural machine translation. In *Findings of the Association for Computational Linguistics: EMNLP*.

A Detailed descriptions of Target Domains

"OPUS-Medical" domain is made out of PDF documents from the European Medicines Agency. "OPUS-Law" domain is a collection of the legislative text of the European Union. "OPUS-IT" domain is constructed from localization files and documents of GNOME, KDE, PHP, Ubuntu, and OpenOffice. "OPUS-Koran" domain is a collection of Quran translations complied by the Tanzil project. "UM-Law" domain contains law statements from mainland China, Hong Kong, and Macau. "UM-Thesis" domain is composed of journal topics in the research area, including electronics, agriculture, biology, economy, etc.

B Involved Scientific Artifacts

In this section, we list the artifact used in our project:

Moses (LGPL-2.1-License): It is a statistical machine translation system that allows you to automatically train translation models for any language pair.

Jieba (MIT-License): it is a library for chinese word segmentation.

Subword-nmt (MIT-License): Subword-nmt is a package containing preprocessing scripts to segment text into subword units.

Fairseq (MIT-license): It is a sequence modeling toolkit that allows researchers and developers to train custom models for translation, summarization, language modeling and other text generation tasks.

Faiss (MIT-license): It is a library for efficient similarity search and clustering of dense vectors.

For the sake of ethic, our use of these artifacts is consistent with their intended use.

C Details of Training Adaptive kNN-MT models with Pruned Datastore

In Table 13, we report hyperparameters to reproduce our main results in Table 7 and 8. In our experiments, it takes at most 1.5 GPU hours to train adaptive kNN-MT models on a single NVIDIA Titan RTX.

Target Domain	k_p	r	k_a	T
OPUS-Medical	8	45%	8	10
OPUS-Law	16	45%	8	10
OPUS-IT	4	40%	8	10
OPUS-Koran	4	25%	8	100
UM-Law	4	30%	8	100
UM-Thesis	4	15%	8	100

Table 13: Hyperparameters for pruning datastore and training adaptive kNN-MT models.

D Implementation Details of Finetuning

We report our finetuning script in Figure 3. In our experiments, it takes at most 12 GPU hours to finetune a general domain NMT model to the target domain on a single NVIDIA Titan RTX.

```
fairseq-train \
  $DATA_PATH \
  --save-dir $SAVE_DIR \
  --tensorboard-logdir $TENSORBOARD_DIR \
  --source-lang $SRC --target-lang $TGT \
  --finetune-from-model $PRETRAIN_MODEL_PATH \
  --arch transformer_wmt_en_de_big --share-all-embeddings \
  --encoder-ffn-embed-dim 8192 \
  --dropout 0.1 --weight-decay 0.0001 \
  --optimizer adam --adam-betas '(0.9, 0.98)' --clip-norm 0.0 \
  --lr 5e-4 --lr-scheduler inverse_sqrt --warmup-updates 4000 \
  --criterion label_smoothed_cross_entropy --label-smoothing 0.1 \
  --max-tokens 4096 --patience 10 \
  --eval-bleu \
  --eval-bleu-args '{"beam": 5, "max_len_a": 1.2, "max_len_b": 10}' \
  --eval-bleu-detok moses \
  --eval-bleu-remove-bpe \
  --eval-bleu-print-samples \
  --best-checkpoint-metric bleu --maximize-best-checkpoint-metric \
  --no-epoch-checkpoints \
  --fp16 \
  --num-workers 0
```

Figure 3: The finetuning script.