What Knowledge Is Needed? Towards Explainable Memory for kNN-MT Domain Adaptation

Wenhao Zhu^{1,2}, **Shujian Huang**^{1,2}, **Yunzhe Lv**^{1,2}, **Xin Zheng**^{1,2}, **Jiajun Chen**^{1,2}
^{1} National Key Laboratory for Novel Software Technology, Nanjing University, China
^{2} Collaborative Innovation Center of Novel Software Technology and Industrialization
{zhuwh,lvyz,zhengxin}@smail.nju.edu.cn, {huangsj,chenjj}@nju.edu.cn

Abstract

kNN-MT presents a new paradigm for domain adaptation by building an external datastore, which usually saves all target language token occurrences in the parallel corpus. As a result, the constructed datastore is usually large and possibly redundant. In this paper, we investigate the interpretability issue of this approach: what knowledge does the NMT model need? We propose the notion of local correctness (LAC) as a new angle, which describes the potential translation correctness for a single entry and for a given neighborhood. Empirical study shows that our investigation successfully finds the conditions where the NMT model could easily fail and need related knowledge. Experiments on six diverse target domains and two language-pairs show that pruning according to local correctness brings a light and more explainable memory for kNN-MT domain adaptation[1].

1 Introduction

Domain adaptation in neural machine translation (NMT) 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., 2021). Fine-tuning (Luong and Manning, 2015) has been the de facto standard for adaptation. However, fine-tuning suffers from the catastrophic forgetting problem (McCloskey and Cohen, 1989; French, 1999).

Recently, Khandelwal et al. (2021) propose kNN-MT, showing a new paradigm for domain adaptation. kNN-MT first explicitly extracts translation knowledge in the target domain training data into a *key-value* datastore with a pre-trained NMT model. For each datastore entry, the key is a continuous representation and the value is a symbolic token. The datastore is then used to assist the NMT model during translation. The kNN-MT framework circumvents the necessity to disturb the parameters of the pre-trained NMT model and enables quick adaptation by switching datastores.

kNN-MT incorporates the symbolic datastore to assist the neural model (Khandelwal et al., 2021; Zheng et al., 2021; Jiang et al., 2021). However, the datastore usually 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.

To understand the relationship between the datastore and the NMT model, this paper conducts investigations on the interpretability issue: *what knowledge does the NMT model need?* Intuitively, the pre-trained NMT model only needs knowledge that remedies its weaknesses. Thus, we propose to explore this issue from the point of *local correctness* (Section 3). Our local correctness includes two aspects, the correctness of translating a given entry (*entry correctness*) and, more importantly, the correctness of performing translation in a given neighborhood in the representation space (*neighborhood correctness*).

For the entry correctness, we check whether the NMT could make correct translation for the entry itself and accordingly split the datastore entries into two categories, namely *known* and *unknown*. Based on entry correctness, we examine neighborhood correctness to more comprehensively evaluate the NMT model's underlying capability. Specifically, we propose a *knowledge margin* metric to evaluate the maximum size of the neighborhood where the NMT could make correct translation. Intuitively, the NMT model may fail when the knowledge margin is small.

To verify our interpretation, we devise a datastore pruning algorithm PLAC (**P**runing with **L**oc**A**l **C**orrectness), which simply removes entries with a higher knowledge margin value (Section 4). These entries are less useful for adaptation, because the

[1] Code will be released at https://github.com/NJUNLP/knn-box

Findings of the Association for Computational Linguistics: ACL 2023, pages 2824–2836
July 9-14, 2023 ©2023 Association for Computational Linguistics

NMT model translates well in their neighborhood.

We conduct experiments on six diverse target domains in two language pairs (Section 6). Compared with existing pruning baselines (Martins et al., 2022; Wang et al., 2022), PLAC prunes more entries (up to 45%) in four OPUS domains' datastore without hurting translation performance. Through ablation study, we reveal that simply relying on entry correctness is not enough, showing that the novel metric knowledge margin for the neighborhood correctness could be the key to build a light and more explainable memory for kNN-MT domain adaptation.

2 Background

For NMT domain adaptation, kNN-MT constructs a datastore $\mathcal{D}$ based on the given target domain bilingual corpus $\mathcal{C}$ and use it to provide helpful target domain translation knowledge for the pre-trained NMT model $\mathcal{M}$. In this section, we briefly introduce kNN-MT and its advanced variant, adaptive kNN-MT (Zheng et al., 2021).

2.1 Building a Domain Specific Datastore

Given target domain bilingual corpus $\mathcal{C}$, all translation pairs in $\mathcal{C}$ 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 in a key-value pair. For the entire corpus, the datastore $\mathcal{D}$ is consisted of key-value pairs:

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

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 $\mathcal{M}$ and $\mathcal{D}$ to generate target language translation $\mathbf{y}$=$\{y_1, y_2, \cdots, y_{|\mathbf{y}|}\}$. More specifically, at decoding time step t, kNN-MT queries the datastore with the decoder hidden state $h(\mathbf{x}, \mathbf{y}_{<t})$ generated by $\mathcal{M}$. The k 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 according to squared-L^2 distance, d. These retrieved knowledge are converted into a distribution over the vocabulary:

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

where T is the temperature. Then, kNN-MT interpolates $p_{k\mathrm{NN}}$ with the pre-trained NMT model's output distribution as the final translation distribution:

$$\begin{aligned} p(y_t|\mathbf{x}, \mathbf{y}_{<t}) &= \lambda\, p_{k\mathrm{NN}}(y_t|\mathbf{x}, \mathbf{y}_{<t}) \\ &\quad + (1-\lambda)\, p_{\mathrm{NMT}}(y_t|\mathbf{x}, \mathbf{y}_{<t}) \end{aligned} \tag{3}$$

The complete translation $\mathbf{y}$ 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 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., 2021).

3 What Knowledge Does the NMT Model Need?

Although less accurate, the pre-trained NMT model could perform translation without the datastore. This fact suggests that the NMT model knows some bilingual knowledge of the target domain. However, the construction of datastore dismisses this point and results in a huge amount of entries being stored.

Intuitively, the pre-trained NMT model only needs knowledge that remedies its weaknesses. To find out these weaknesses and build more explainable memory, we start from investigating entry correctness. Based on this basic concept, we further study neighborhood correctness and find that it precisely reflects the NMT model's strengths and weaknesses.

3.1 Known v.s. Unknown for Entry Correctness

The capability of the NMT model in target domain is difficult to describe directly. However, as the datastore consists of entries constructed on training set, it is easier to check whether the NMT model could make correct translation for them.

This can be efficiently accomplished by an extra evaluation during the teacher-forcing decoding. 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 y'_t, which is the target token with the highest probability $p_{\text{NMT}}(y'_t|\mathbf{x}, \mathbf{y}_{<t})$. Then we call an entry as a *known* entry if the NMT model could predict it correctly; and *unknown*, otherwise (Equation 4).

$$(h(\mathbf{x}, \mathbf{y}_{<t}), y_t) \text{ is } \begin{cases} \textit{known}, & \text{if } y'_t = y_t \\ \textit{unknown}, & \text{o.w.} \end{cases} \tag{4}$$

Obviously, the *unknown* entries in the datastore are important, because these are the points where the NMT model tends to make a mistake.

3.2 The Knowledge Margin Metric for Neighborhood Correctness

However, entry correctness alone could not fully reveal the NMT model's weaknesses. Because for *known* entries, the NMT model may still fail during inference where the context could be similar but different. Considering that the contextualized representations of similar context stay close in the representation space (Peters et al., 2018), we propose to investigate the NMT model's translation performance in a neighborhood.

We propose a metric called knowledge margin, denoted as km, to measure the neighborhood correctness. Given an entry (h, y), its neighborhood is defined by its k nearest neighbors[2] in the datastore $\mathcal{N}_k(h) = \{(h^j, y^j)\}_1^k$. The knowledge margin of the entry, i.e. $km(h)$, is defined as:

$$\arg\max_k \, (h^j, y^j) \text{ is } \textit{known}, \forall (h^j, y^j) \in \mathcal{N}_k(h). \tag{5}$$

Intuitively, km is the maximum size of the neighborhood of the entry h where the NMT could make correct translation. If considering at most $\bar{k}$ nearest neighbors of h, its knowledge margin will be a number between 0 and $\bar{k}$.

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 $\mathcal{N}_k(q)$ could be defined by querying the datastore. This extension allows the investigation of the NMT model at any given point in the representation space.

3.3 Empirical Analysis

We now present an empirical analysis of the relationship between the NMT model and the datastore, and reveal the NMT model's weaknesses.

Settings We follow Zheng et al. (2021) and consider four domains in German-English OPUS dataset (Tiedemann, 2012) as target domains[3]. Table 1 lists statistics of four domains[4]. For pre-trained NMT model, we use the winner model of WMT'19 German-English news translation task [5] (Ng et al., 2019). The datastore for each domain is constructed on the corresponding training set with the pre-trained NMT model.

	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.

Entry Correctness 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 are *known* by the pre-trained NMT model. This high ratio strongly indicates that a large amount of datastore entries may be redundant.

Neighborhood Correctness We measure neighborhood correctness of each datastore entries and plot the distribution of knowledge margin for known and unknown entries in Figure 1 ($\bar{k}$ =

[2] In our implementation, we do not consider the given entry itself as its neighbor.

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

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

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

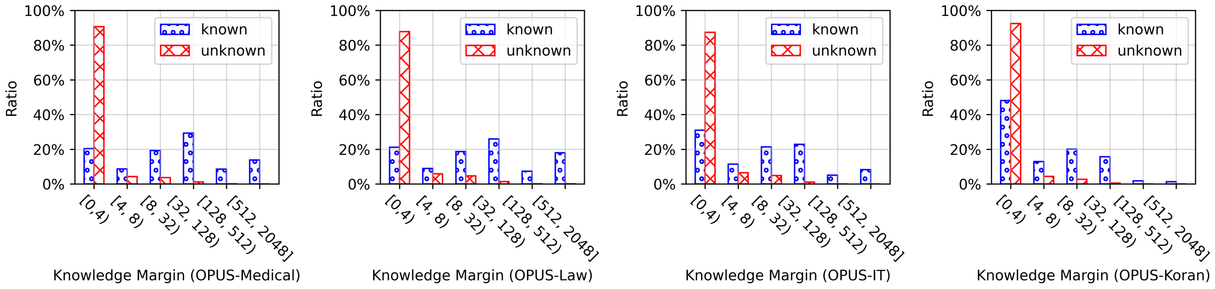

Figure 1: The ratio distribution on different knowledge margin for *known* and *unknown* entries on four OPUS domains.

	OPUS-Medical	OPUS-Law	OPUS-IT	OPUS-Koran
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.

2048). The distributions on four OPUS domains show the same trends. Most *unknown* entries has a very low knowledge margin, e.g., around 90% of *unknown* entries have a margin value between 0 and 4. In contrast, the distribution for *known* entries is more diverse. The results indicate that the neighborhood correctness is consistent with the entry correctness, but may provide more information for known entries.

To verify the relation between knowledge margin and NMT model's translation ability, we conduct experiments on the development set for each domain, where translation context are unseen. 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. We evaluate the knowledge margin of the query and the prediction accuracy of the NMT model.

Figure 2 shows the results. For tokens with higher margins, e.g. $km \geq 32$, the prediction accuracy of the NMT model is higher than 95%. In contrast, for tokens with lower margins, e.g. $km < 4$, the accuracy is lower than 50%. This is a strong evidence that the NMT model could easily fail when knowledge margin is small.

In Table 3, we also show a translation example for such a condition, where knowledge margin of the current query is 0 and the NMT model fails to generate the last subword of "Cyanokit"[6].

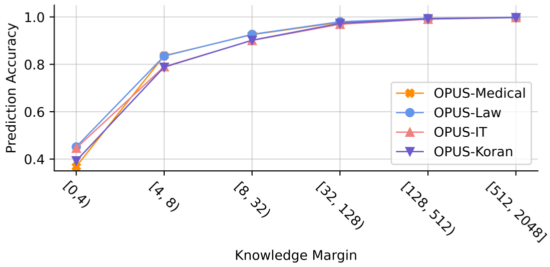

Figure 2: The NMT model's prediction accuracy at positions with different margin values in OPUS domains' unseen development set.

4 Building Explainable Memory Based on Local Correctness

Because local correctness are good indicators for translation failures. It could also be interpreted as the importance of datastore entries. To verify this interpretation, we propose a pruning algorithm, i.e. **P**runing with **L**oc**A**l **C**orrectness (**PLAC**), to cut off entries with a high knowledge margin (Algorithm 1).

There are two steps in the algorithm. In the first step, each entry (h, y) in the datastore $\mathcal{D}$ is checked for their local correctness. If knowledge margin of (h, y) is greater than or equal to the threshold k_p, the entry is collected as the pruning candidates [7].

In the second step, these pruning candidates are randomly selected and get removed from $\mathcal{D}$ until the required pruning ratio is reached. Since our method does not need to train any additional neural networks, it can be easily implemented. The pruned

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

[7] In practice, there exists a very small amount of *unknown* entries with high knowledge margin. In our implementation, we keep them in the datastore because the NMT model cannot make correct translation about these entries.

Source sentence ($\mathbf{x}$): Wie ist Cy@@ an@@ ok@@ it anzu@@ wenden ?
Previous translation ($\mathbf{y}_{<t}$): How to use Cy@@ an@@ ok@@

No.	Type	Retrieved Keys: source ($\mathbf{x}$)	Retrieved Keys: target ($\mathbf{y}_{<t}$)	Retrieved Values
1	*unknown*	Wie wird Cy@@ an@@ ok@@ it ange@@ wendet ?	How is Cy@@ an@@ ok@@	it
2	*unknown*	Sie erhalten Cy@@ an@@ ok@@ it als In@@ fusion in eine V@@ ene .	You will have Cy@@ an@@ ok@@	it
3	*unknown*	Wo@@ für wird Cy@@ an@@ ok@@ it ange@@ wendet ?	What is Cy@@ an@@ ok@@	it
4	*unknown*	Wel@@ ches Risiko ist mit Cy@@ an@@ ok@@ it verbundenn ?	What is the risk associated with Cy@@ an@@ ok@@	it
5	*unknown*	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	*unknown*	Warum wurde Cy@@ an@@ ok@@ it zugelassen ?	Why has Cy@@ an@@ ok@@	it
7	*unknown*	Beson@@ dere Vorsicht bei der Anwendung von Cy@@ an@@ ok@@ it ist erforderlich	Take special care with Cy@@ an@@ ok@@	it
8	*unknown*	Wie wirkt Cy@@ an@@ ok@@ it ?	How does Cy@@ an@@ ok@@	it

NMT's prediction (y'_t): ite
Correct target token (y_t): it

Table 3: An example where the NMT model fails (sentence are tokenized into subwords). At the current time step, all retrieved entries are *unknown* for the NMT model, so knowledge margin is 0. The prediction of NMT is highly likely to be wrong. With these retrieved entries, the kNN-MT could make a correct prediction.

datastore can be used in different kNN-MT models, such as adaptive kNN-MT.

Algorithm 1 Datastore Pruning by PLAC

```
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 $km(h) \geq k_p$ 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 $\mathcal{D}$
```

5 Experiment Setup

This section introduces general experiment setup for evaluating pruning effect. More implementation details can be found in Appendix C.

5.1 Data and Processing

We conduct datastore pruning for 6 different domains from 2 language pairs. Specifically, we take 4 OPUS domains for De-En experiments and 2 UM domains[8] for Zh-En experiments (Tian et al., 2014), which are all benchmark dataset for NMT domain adaptation research.

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

Table 4: Detailed statistics of UM dataset. We report the sentence number of each subset. "Train", "Dev", "Test" denote training, development, test set respectively.

For preprocessing, we use *moses*[9] toolkit to tokenize German and English corpus and *jieba*[10] to tokenize Chinese corpus. Byte pair encoding[11] (BPE) is applied for subword segmentation.

5.2 Pre-trained NMT Model

For De-En tasks, we use the winner model of WMT'19 De-En news translation task, which is based on the Transformer architecture (Vaswani et al., 2017). For Zh-En tasks, we train a base Transformer model from scratch on CWMT'17 Zh-En Dataset[12] (9 million sentence pairs), since we do not find any publicly available Zh-En pre-trained NMT model on the website.

The pre-trained NMT model is the unadapted general domain model for each language pair, which is the starting point for domain adaptation.

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

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

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

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

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

	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
Cluster	45%	53.31*	0.5689	45%	58.68*	0.6779*	40%	45.80*	0.5788	25%	20.04*	0.0410
Merge	45%	54.65*	0.5523*	45%	60.60*	0.6776*	40%	45.83*	0.5334*	25%	20.25*	0.0365
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*
PLAC (ours)	45%	57.66	0.5773	45%	63.22	0.6953*	40%	48.22	0.5560	25%	20.96	0.0442

Table 5: 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 ($p < 0.05$).

For kNN methods, it also serves as the base for building the datastore.

5.3 Systems for Comparison

We report the performance of the following systems for reference: the pre-trained NMT model (Base), the pre-trained model finetuned on each target domain (Finetune) (Luong and Manning, 2015), adaptive kNN-MT with full datastores built for each target domain on their training set (Adaptive kNN) (Zheng et al., 2021). Finetuning and Adaptive kNN are two popular alternatives for adaptation.

The following pruning methods are applied to the datastore of Adaptive kNN for comparison: randomly pruning (**Random**), cluster-based pruning (**Cluster**) (Wang et al., 2022), merging similar entries (**Merge**) (Martins et al., 2022), randomly pruning *known* entries (**Known**), pruning all *known* entries (**All Known**). Among them, Cluster and Merge are pruning methods based on the context similarity of different entries (Wang et al., 2022; Martins et al., 2022).

We report case-sensitive detokenized BLEU (Papineni et al., 2002) calculated by *sacrebleu*[13] and COMET (Rei et al., 2020) calculated by publicly available *wmt20-comet-da*[14] model. For the prunign methods, statistical significance test (Koehn, 2004) against the full datastore (Adaptive kNN) are conducted as well.

6 Experiment Results and Analysis

6.1 Safely Pruning with PLAC

Experiment results on OPUS domains are presented in Table 5. For the reference, the pre-trained NMT model usually does not translate well on target domains. Finetuning and Adaptive kNN have comparable performances.

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

Compared with using full datastore (Adaptive kNN), our method (**PLAC**) cutting off 25%-45%[15] entries of the datastore while achieving comparable performance. On the two largest domains, "OPUS-Medical" and "OPUS-Law", our method successfully prunes 45% datastore (millions of key-value pairs). Excellent pruning performance validates our analysis concerning with local correctness.

Cluster and Merge lead to a larger degradation of translation performance[16], showing that entries with identical target tokens indeed have different importance in assisting the NMT model. Simply pruning all *known* entries results in a significant drop of performance (All Known). Pruning *known* entries to the same ratio as PLAC also lead to degradation (Known), although it outperforms Cluster and Merge. These comparisons indicates that the entry correctness only partially reflects entry importance, demonstrating the necessity of the neighborhood correctness analysis with knowledge margin.

The results on UM domains are presented in Table 6. 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.

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

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

[15] The best pruning ratios are different because for different target domains, the translation knowledge inside the pre-trained NMT model is different. For the target domain which is more distant away, the best pruning ratio is naturally smaller.

[16] Here we show the pruning effect of the two methods under a large pruning ratio. According to the original published results, these methods may suffer less performance degeneration when the pruning ratio is lower, e.g. 10% (Wang et al., 2022).

	UM-Law			UM-Thesis		
	Ratio	BLEU↑	COMET↑	Ratio	BLEU↑	COMET↑
Base	-	30.36	0.3857	-	13.13	-0.0442
Finetune	-	58.82	0.6375	-	16.86	-0.0295
Adaptive kNN	0%	58.64	0.6017	0%	17.49	-0.0146
Random	30%	53.78*	0.5661*	15%	16.14*	-0.0280*
Cluster	30%	49.65*	0.5274*	15%	15.73*	-0.0419*
Merge	30%	56.51*	0.5873*	15%	17.00*	-0.0296*
Known	30%	56.92*	0.5762*	15%	17.25	-0.0143
All Known	63%	46.45*	0.4720*	47%	15.33*	-0.0525*
PLAC (ours)	30%	58.65	0.6056	15%	17.52	-0.0122

Table 6: 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 ($p < 0.05$).

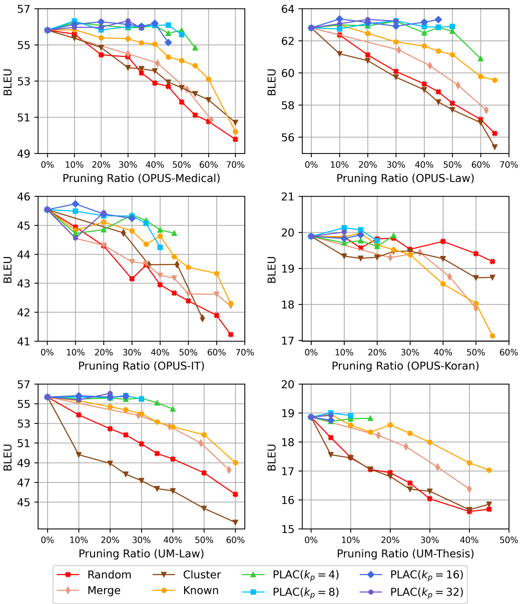

Figure 3: BLEU scores of adaptive kNN-MT models with pruned datastore on different domains' development set. Different symbols represent different ways of pruning datastore.

6.2 How Knowledge Margin Affects Pruning Performance?

In this section, we examine how knowledge margin affects pruning performance and provide more insight into our proposed method. Figure 3 plots BLEU scores of adaptive kNN-MT models with pruned datastore under different pruning ratios on development sets. We can observe that trends are mostly similar in different domains. Pruning by PLAC achieves the best performance over the other baselines and the performance is more stable even with a higher pruning ratio.

Note that Known is a case where neighborhood correctness is dismissed during entry pruning. Although it outperforms Random, Cluster and Merge in most scenarios, its performance is still unstable.

When tuning the hyperparameter k_p among {4, 8, 16, 32}, we can see a trade-off between BLEU score and the pruning ratio. Large k_p leads to a small sacrifice of BLEU score but a lower pruning ratio. Small k_p allows us to prune more entries but causes significant BLEU scores decline after a specific threshold ratio. For example, when $k_p = 4$, it is allowed to prune 55% "OPUS-Medical" datastore, but translation performance declines drastically after the pruning ratio reaches 50%. Finally, we choose the top-right point[17] in each subfigure as the best-performed setting for each domain, which are used in other experiments.

6.3 Datastore Entries With Lower Knowledge Margin Are Indeed Valuable

In this section, we want to verify that entries with low knowledge margin are truly important for NMT adaptation. For this purpose, we remove entries from datastore with a reversed strategy, i.e. the knowledge margin of (h, y) is less than k_p.

Table 7 shows pruning effect. We can see that pruning entries with reverse strategy suffers significant performance decline at even a small pruning ratio, demonstrating the importance of these entries for domain adaptation. We also show some cases for each domain in Table 8. We can see that target tokens of these valuable entries are more domain-specific, e.g. "dose" and "Executive".

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

Table 7: Translation performance difference (BLEU) compared with Adaptive kNN using full datastore under different pruning ratios.

6.4 PLAC Is Applicable to Different kNN-MT Variants

For more comprehensive evaluation, we plug our pruned datastore into different kNN-MT variants, i.e. vanilla kNN (Khandelwal et al., 2021), KSTER (Jiang et al., 2021) and adaptive kNN. Experiment results on OPUS-Law domain show that our pruned datastore does almost no harm to the translation performance of different variants, demonstrating the effectiveness of PLAC.

[17] Hyper-parameter values of these points are reported in Appendix C.

Domain	Source Sentence (x)	Target Sentence (y)
OPUS-Medical	Die Höchst@@ do@@ sis sollte 30 mg - Tag nich überschrei@@ ten .	The maximum dose should not exce@@ ed 30 mg / day .
OPUS-Law	Das Direkt@@ ori@@ um entscheidet über die Organisation seiner Sitz@@ ungen .	The Executive Board shall decide on the organisation of its meetings .
OPUS-IT	Sie haben eventuell einen Programm@@ fehler entdeckt .	You may have encounter@@ ed a bu@@ g in the program .
OPUS-Koran	Das ist eine schmerz@@ hafte P@@ ein .	Target sentence: That would be a grie@@ v@@ ous aff@@ li@@ ction .
UM-Law	保险公司 依法 接受 监督 检查 。	Any insurance company shall accept supervision and inspection according to law .
UM-Thesis	中国 能源需求 及其 风险管理 研究	The Research on Energy Demand and Its Risk Management in China

Table 8: Case study for remaining knowledge in different domain's pruned datastore. The underlined parts are target tokens of entries with small margin values.

	OPUS-Medical		OPUS-Law		OPUS-IT		OPUS-Koran		UM-Law		UM-Thesis	
	Space	Δ	Space	Δ	Space	Δ	Space	Δ	Space	Δ	Space	Δ
Full Datastore	492	-	1,328	-	265	-	54	-	680	-	810	-
PLAC (ours)	279	43%	739	44%	166	37%	45	17%	479	30%	690	15%

Table 9: Memory Space (MB) comparsion between pruned datastore and full datastore. "Space" denotes the memory space taken by the index file and "Δ" denotes the percentage of space saved by our method.

kNN-MT Variants	Full	Pruned
Vanilla kNN (Khandelwal et al., 2021)	61.34	61.24
KSTER (Jiang et al., 2021)	62.45	62.30
Adaptive kNN (Zheng et al., 2021)	63.53	63.22

Table 10: Translation performance (BLEU) of different kNN-MT variants with full and pruned datastore on OPUS-Law domain's test set.

6.5 Pruned Datastore Occupies Less Memory Space

In practice, the datastore must be loaded to CPU and GPU memory during inference. So its size affects the efficiency. Since Faiss index is used to index and represent the datastore, we compare the size of index file before and after pruning (Table 9). For all the domains, our pruning method PLAC significantly reduces the memory occupation. The ratio of saved memory space is roughly identical with the PLAC pruning ratio. For the largest datastore, "OPUS-Law", the memory space can be reduced by 44%.

7 Related Work

Less attention have been paid to the research of interpretability of kNN-MT. To the best of our knowledge, we are the first to systematically study the relationship between the NMT model and the datastore. As for datastore pruning, Wang et al. (2022) and Martins et al. (2022) prune the datastore based on the hypothesise that entries with similar translation are redundant. Actually, entries with similar translations may have different importance to the translation. Our analysis suggests one way to understand these differences.

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 PLAC 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 different kNN-MT variants and easy to implement. Future directions maybe using local correctness to explore more interpretability issue of NMT domain adaptation, e.g. catastrophic forgetting.

9 Limitation

During inference, kNN-MT have to query the datastore at each decoding step, which is time-consuming. Although up to 45% datastore entries can be safely pruned by our method, deploying a high-quality kNN-MT system with fast inference speed is still an open challenge.

10 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. In the future, more efforts may be put into addressing this issue.

Acknowledgement

We would like to thank the anonymous reviewers for their insightful comments. Shujian Huang is the corresponding author. This work is supported by National Science Foundation of China (No. 62176120), the Liaoning Provincial Research Foundation for Basic Research (No. 2022-KF-26-02).

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)*.

Pedro Martins, Zita Marinho, and Andre Martins. 2022. Efficient machine translation domain adaptation. In *Proceedings of the Workshop on Semiparametric Methods in NLP: Decoupling Logic from Knowledge*.

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: Human Language Technologies (NAACL-HLT)*.

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)*.

Matthew E. Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. In *Proceedings of the Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT)*.

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.

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: Human Language Technologies (NAACL-HLT)*.

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. 2022. Efficient cluster-based k-nearest-neighbor machine translation. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL)*.

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. 2021. Adaptive nearest neighbor machine translation. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL)*.

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 Implementation Details

We implement adaptive kNN-MT with Zheng et al. (2021)'s released code and script[18] based on *fairseq*[19] (Ott et al., 2019). Due to the large space of hyper-parameters, we follow Zheng et al. (2021) to set the number of retrieved entries (k_a) as 8 when training adaptive kNN-MT models for most experiments, and report pruning performance under different k_a in Appendix D. During inference, we set beam size as 5 and length penalty as 1.0.

For implementing PLAC, 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., 2021) and use Faiss[20] index (Johnson et al., 2019) to represent the datastore and accelerate nearest neighbors search.

In Table 11, we report hyperparameters to reproduce our main results in Table 5 and 6. 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 11: Hyperparameters for pruning datastore and training adaptive kNN-MT models.

D Pruning Effect is Insensitive to Hyperparameter k_a

To demonstrate the reliability 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 12). Results show that our pruning method enjoys consistent performance under different k_a.

OPUS-Law	$k_a = 4$	$k_a = 8$	$k_a = 16$	$k_a = 32$
Adaptive kNN	63.31	63.53	63.56	63.33
PLAC (ours)	62.93	63.22	63.18	63.22

Table 12: Pruning performance under different k_a.

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

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

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

ACL 2023 Responsible NLP Checklist

A For every submission:

☑ A1. Did you describe the limitations of your work?
section 9

☑ A2. Did you discuss any potential risks of your work?
section 10

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

☒ A4. Have you used AI writing assistants when working on this paper?
Left blank.

B ☑ Did you use or create scientific artifacts?

appendix B

☑ B1. Did you cite the creators of artifacts you used?
section 5

☑ B2. Did you discuss the license or terms for use and / or distribution of any artifacts?
appendix B

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

□ B4. Did you discuss the steps taken to check whether the data that was collected / used contains any information that names or uniquely identifies individual people or offensive content, and the steps taken to protect / anonymize it?
Not applicable. Left blank.

□ B5. Did you provide documentation of the artifacts, e.g., coverage of domains, languages, and linguistic phenomena, demographic groups represented, etc.?
Not applicable. Left blank.

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

C ☑ Did you run computational experiments?

Left blank.

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

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

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

☑ C3. Did you report descriptive statistics about your results (e.g., error bars around results, summary statistics from sets of experiments), and is it transparent whether you are reporting the max, mean, etc. or just a single run?
section 5

☑ C4. If you used existing packages (e.g., for preprocessing, for normalization, or for evaluation), did you report the implementation, model, and parameter settings used (e.g., NLTK, Spacy, ROUGE, etc.)?
section 5

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

Left blank.

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

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

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

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

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