Efficient Domain Adaptation for Non-Autoregressive Machine Translation

Wangjie You[♠*], **Pei Guo**[♠*], **Juntao Li**[♠†], **Kehai Chen**[♢], **Min Zhang**[♠]
[♠]Institute of Computer Science and Technology, Soochow University, China
[♢]Harbin Institute of Technology, Shenzhen
{wjyouuu, pguolst}@stu.suda.edu.cn;
{ljt, minzhang}@suda.edu.cn;
chenkehai@hit.edu.cn

Abstract

Domain adaptation remains a challenge in the realm of Neural Machine Translation (NMT), even in the era of Large language models (LLMs). Existing non-parametric approaches like nearest neighbor machine translation have made small Autoregressive Translation (AT) models achieve efficient domain generalization and adaptation without updating parameters, but leaving the Non-Autoregressive Translation (NAT) counterparts under-explored. To fill this blank, we introduce Bi-kNN, an innovative and efficient domain adaptation approach for NAT models that tailors a k-Nearest-Neighbor algorithm for NAT. Specifically, we introduce an effective datastore construction and correlated updating strategies to conform the parallel nature of NAT. Additionally, we train a meta-network that seamlessly integrates the kNN distribution with the NMT distribution robustly during the iterative decoding process of NAT. Our experimental results across four benchmark datasets demonstrate that our Bi-kNN not only achieves significant improvements over the Base-NAT model (7.8 Bleu on average) but also exhibits enhanced efficiency.[1]

1 Introduction

LLMs have dramatically shifted the paradigm of various language processing tasks, and there have also been extensive discussions recently to weigh up the pros and cons of LLMs to NMT (Lyu et al., 2023; Jiao et al., 2023; Hendy et al., 2023; Zhang et al., 2023). LLMs can achieve appealing generalization and task performance by pretraining on vast and varied corpora across languages and domains, surpassing the task expert model trained on supervised translation pairs under resource-rich scenarios, but might suffer from the prohibitive inference cost for highly concurrent service. Nevertheless, recent studies (Moslem et al., 2023; Yang et al., 2023; Jiao et al., 2023) have highlighted that while LLMs demonstrate impressive translation capabilities for mainstream languages, their performance significantly declines when confronted with specific domains. How to deal with the NMT task of specific domains in the era of LLMs, i.e., the domain adaptation setting, is still not well-known.

Models	**WMT**	**Domain (Avg.)**	**Speed** (tokens/s)
ChatGPT	39.59	32.63	-
LLaMA-2 (7B)	34.65	22.66	27.08
AT (65M)	37.15	32.41	148.99
NAT (73M)	37.03	30.48	240.86

Table 1: Comparison between LLMs, AT, and NAT models on general and domain-specific datasets (Aharoni and Goldberg, 2020).

To explore this problem further, we first briefly compare different lines of possible solutions for NMT tasks, including *closed-sourced LLM (ChatGPT)*, *open-sourced LLM (LLaMA-2)*, and *two types of transformer-based* expert models (i.e., the auto-regressive (AT) and non-auto-regressive (NAT) fashion).[2] LLMs are introduced to calibrate the domain-specific translation performance without any further task training due to possible prohibitive costs. Expert models are presented to learn the domain adaptation capabilities of small capacity models with supervised task training, e.g., initially converged on the WMT training dataset but inference on specific domains like IT, Medical, Law, and Koran. Table 1 averages the BLEU scores on four specific domains. We can see that though supervised task expert models no longer have performance superiority to LLMs on high-resource languages, they still show promising domain adaptation potential. Meanwhile, considering that ex-

[*] Equal Contribution
[†] Corresponding Author
[1]Our code is availabel at https://github.com/Moriarty0923/BIKNN.
[2]More details of the comparison experiments are given in Appendix A.1.

Findings of the Association for Computational Linguistics ACL 2024, pages 13657–13670

pert models are much more affordable in real-world usage at scale (e.g., over 10 $\times$ speedup), it is worth figuring out how to solve NMT for specific domains with small experts.

There are already some effective strategies to enhance the domain adaptation performance of small models, particularly the non-parametric paradigm. For instance, Khandelwal et al. (2020) present a k-Nearest-Neighbor Machine Translation method, which utilizes a trained NMT model to construct a datastore, consisting of *(query: context representations; value: the correlated target tokens)* pairs in the training set, and then retrieves relevant tokens during inference to enhance the translation accuracy. This non-parametric approach equips the model with rapid domain adaptation and generalization abilities without the need for parameter adjustments. However, most of the existing methods are tailored for AT models, leaving the domain adaptation problem of NAT models under-explored.

To fill this blank, we introduce an innovative domain adaptation approach for NAT models, namely Bidirectional-Iterative-$k$nn (Bi-kNN), which is an efficient method tailored for NAT models. Unlike k-Nearest-Neighbor NMT for AT models, NAT models struggle with producing accurate representations due to insufficient context with parallel decoding. To overcome this, we present a novel and effective framework for kNN-MT with NAT models, including *(1) building a bidirectional datastore*, *(2) renewing the indecipherable datastore*, *(3) training a robust Meta-network*, and *(4) iterative-kNN decoding*. We conducted experiments on multiple domain-specific NMT tasks. Across four domains, our approach achieved an average of 7.8 BLEU score improvement for the Base-NAT models and outperformed the specialized models which are trained on the corresponding datasets on most datasets. Furthermore, without tuning the parameters of pre-trained models, our method proved more efficient and avoided catastrophic forgetting compared to the straightforward fine-tuning method.

2 Related Work

Machine Translation with LLMs Large Language Models (LLMs), notably ChatGPT (Ouyang et al., 2022) and GPT-4 (Achiam et al., 2023), have demonstrated their substantial potential in the sphere of Neural Machine Translation (NMT). These models have delivered remarkable improvements in terms of translation accuracy and fluency compared to traditional machine translation systems, especially in the context of high-resource bilingual translation tasks (Agrawal et al., 2022; Hendy et al., 2023; Zhang et al., 2023). Moreover, their strong generality is believed to address traditional challenges in NMT, such as multilingual and domain-specific translation (Yang et al., 2023; Reinauer et al., 2023). However, challenges still persist. Numerous studies (Moslem et al., 2022; Jiao et al., 2023) have shown that though LLMs can effectively compete with commercial translation products like Google Translate for resource-rich European languages across various domains, their performance significantly deteriorates when dealing with resource-scarce or specific domains. This underscores the importance of domain adaptation, which remains a central challenge of NMT.

Domain adaptation Many efforts have been made to address the domain adaptation challenge of NMT, particularly in the non-parametric paradigm. Notably, kNN-MT (Khandelwal et al., 2020), has been shown to be both simpler and more expressive, breaking the capacity limitation in a plug-and-play manner. Typically, it utilizes the decoder representations as keys and the corresponding target words as values to construct a datastore. During inference, the predicted distribution of the NMT model is interpolated with the kNN distribution using a hyper-parameter λ, based on the retrieved results. Subsequently, some studies (Zheng et al., 2021a; Jiang et al., 2021, 2022a; Wang et al., 2022b) have achieved improved results by dynamically estimating λ. Meanwhile, other researchers have made efforts to accelerate inference by compressing data (He et al., 2021; Wang et al., 2022a; Martins et al., 2022a) or limiting the search space (Meng et al., 2022; Martins et al., 2022b; Deguchi et al., 2023). However, the effectiveness of the kNN approach has only been tested on autoregressive models, and the non-autoregressive models, co-existing as a critical branch in the tree of machine translation, remain unexplored.

Non-autoregressive Machine Translation NATs (Gu et al., 2018) have been introduced to reduce decoding latency but might suffer from poor generation quality. Numerous studies (Gu and Kong, 2020; Qian et al., 2021; Zeng et al., 2022; Lv et al., 2023; Guo et al., 2023) have been dedicated to addressing this issue. Notably, iterative refinement in NATs (Lee et al., 2018; Ghazvininejad et al., 2019; Huang et al., 2021;

Xiao et al., 2023) has shown potential, achieving performance on par with autoregressive (AT) models by incorporating target-side dependencies. This is accomplished by conditioning each prediction on the output from the preceding iteration. Nevertheless, in the landscape dominated by LLMs, NATs lag behind AT models, primarily due to their limitations in leveraging pretraining effectively with LLMs. Furthermore, while the general translation capabilities of NATs have been the focus of much research, their domain adaptation proficiency has not been adequately addressed. An exception is Lv et al. (2023), they conducted preliminary exploration on the domain adaptation problem of the NAT models and proposed an N-gram-based method. However, their method failed to achieve significant improvement. In this work, we further explore the domain adaptation challenges for NAT models. We propose Bi-kNN, an efficient domain adaptation approach, adapting kNN for NATs, to enhance their adaptability across various domains. Our method has achieved significant performance improvements for NAT models, presenting a cost-effective strategy to enhance the versatility and applicability of NATs in domain-specific machine translation tasks.

3 Preliminary Study

3.1 Nearest-Neighbor Machine Translation

Khandelwal et al. (2020) pioneered the integration of k-nearest-neighbor (kNN) retrieval into machine translation, demonstrating notable advancements in NMT and domain adaptation challenge by introducing pre-stored external target-side information during the decoding stage. Specifically, kNN-MT contains two steps: datastore creation and kNN decoding. Given a bilingual sentence pair in the training set $(x, y) \in (\mathcal{X}, \mathcal{Y})$ and a pre-trained NMT model $f(\cdot)$. kNN-MT utilizes the hidden state $h_t = f(x, y_{<t})$ [3] when predicting the t-th target token y_t as key and the corresponding ground-truth tokens as value to construct key-value pairs. Then, the datastore is constructed by a single forward pass over each target token in the training set.

During inference, at time-step t, kNN-MT utilizes the hidden state $\hat{h}_t = f(x, \hat{y}_{<t})$ to query the datastore for k nearest neighbors according to l_2 distance. The kNN prediction probability is calculated by the distance between the query and retrieved keys,

$$p_t^{k\text{nn}}(y_t|x, \hat{y}_{<t}) \propto \sum_{(h_i,v_i)\in N} \mathbb{1}_{y_t=v_i} \exp(\frac{-d_k}{\tau}), \tag{1}$$

where d_k is the l_2 distance and τ denotes the temperature. The final prediction probability of y_t is calculated by interpolating the kNN prediction and model prediction with a hyper-parameter λ:

$$\begin{aligned} p(y_t|x, \hat{y}_{<t}) = \lambda\, & p_t^{k\text{nn}}(y_t|x, \hat{y}_{<t}) \\ & + (1-\lambda)\, p_t^{\text{NMT}}(y_t|x, \hat{y}_{<t}). \end{aligned} \tag{2}$$

3.2 Limitation of kNN for NAT Models

However, the vanilla kNN approach, originally tailored for AT models, exhibits obvious limitations when repurposed for NAT models, primarily due to the absence of dependencies on the target side. In detail, NAT models disrupt the conventional conditional dependencies, simultaneously generating all tokens. This process is mathematically represented by $p(y|x) = p(T_y|x) \cdot \prod_{t=1}^{T_y} p(y_t|x)$, where $p(T_y|x)$ signifies the target length prediction of the model. The typical decoder input for NAT models is an identical copy of source representations or an empty sequence, constructed using `[UNK]` or `[MASK]` tokens. Therefore, in the conventional kNN approach, the datastore keys for NAT models are synthesized using $h_i = f(x, y_{unk})$, where x represents the source sentences and y_{unk} denotes the substituted decoder input. Owing to the absence of pertinent target-side information, the constructed keys may lack clarity and precision and fail to aid the following decoding stage. Conversely, for AT models, the construction process benefits from the autoregressive pattern, which incorporates the previous steps' unidirectional target-side information, resulting in more precise and informative keys.

4 Methodology

In this section, we present the overall process of our proposed Bi-kNN, as illustrated in Figure 1. Concretely, our method contains four specific steps, i.e., build bidirectional datastore, renew indecipherable datastore, train robust Meta-network and iterative decoding with kNN.

4.1 Build Bidirectional Datastore

The primary obstacle for NATs during the datastore creation stage is the deficiency of target-side

[3] Following previous works (Khandelwal et al., 2020; Zheng et al., 2021b), we use the hidden state before the final softmax as h.

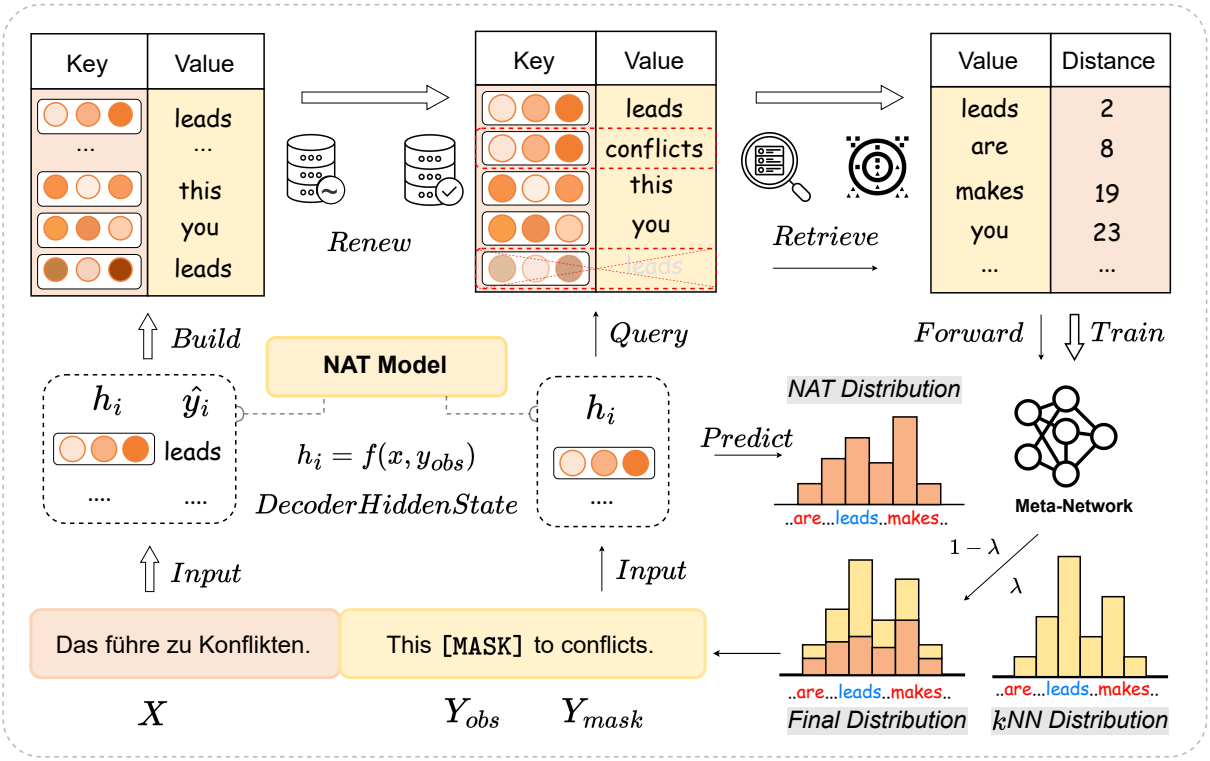

Figure 1: Overview of our proposed Bi-kNN. $\Uparrow$ represents the process of bidirectional datastore construction and Meta-network training, while $\uparrow$ represents the process of iterative decoding with kNN.

information. We draw inspiration from iterative-based NAT models, which iteratively enhance their outputs by utilizing predictions from previous iterations. This paradigm enables us to harness the abundant bidirectional information embedded in past iteration predictions, thereby aiding NAT models to obtain partial contextual information. Given a bilingual pair $(x, y) \in (\mathcal{X}, \mathcal{Y})$, we random mask part of positions in y, and split the original y into $[y_{mask}, y_{obs}]$, where y_{mask} is the positions replaced by [MASK] tokens and y_{obs} is the observed target tokens. We denote the hidden representation of i-th masked position y_i as $h_i = f(x, y_{obs})$, and then the datastore is constructed over each masked position in parallel:

$$(\mathcal{K}, \mathcal{V}) = \bigcup_{(x,y)\in(\mathcal{X},\mathcal{Y})} \{(h_i, y_i), \forall i \in y_{mask}\}, \quad (3)$$

In this context, we integrate explicit target-side contextual information during the construction process, thereby significantly enhancing the robustness and preciseness of key-value pairs. To capture more potential contextual information, we set up multiple iterations and set different mask ratios for each iteration. We initiate n iterations on a given sentence, which is determined by the target length y_{len}, n is computed as $n = \min(\sqrt{y_{\text{len}}}, \beta)$, where we β to 5 for most datasets in practice to achieve trade-off between performance and effectiveness. In t-th iteration, we define the mask ratio α based on t with linear decay mechanics as $\alpha = \frac{t}{n} + \lambda$, where λ is the pre-set parameter to limit the bound.

However, there remain several challenges:

- The constructed datastore lacks robustness due to the presence of [MASK] tokens in the decoder input, which causes the hidden state to point to an ambiguous representation.
- There is a discrepancy between the creation and inference stages. While the creation stage only considers the model's output, the inference stage also factors in the kNN probability distribution, leading to inconsistencies.

4.2 Renew Indecipherable Datastore

Upon completion of the initial stage, we get a basic bidirectional datastore. However, this datastore is of subpar quality and lacks distinguishability, as only a subset of the target tokens are evident during the construction phase. Moreover, the datastore might encompass ambiguous, irrelevant, and useless representations, conversely lacking essential information that the model cannot autonomously

generate.[4] To enhance the quality of the datastore, we propose a renewal strategy subsequent to the building stage. Concretely, we deploy a standard k nearest-neighbor decoding using the original model's outputs on the training dataset. For a given masked target position, we employ its corresponding hidden state as the query to conduct a search within the bidirectional datastore we have established. Adhering to a predefined value of k, we retrieve the top k closest key-value pairs, along with their distances to the query. We then proceed to polish the datastore from three distinct perspectives: **Inclusion of overlooked items:** If the top k retrieval candidates fail to include the correct corresponding tokens and the model's output diverges from the ground truth, we identify these instances as cases of overlooked key items. These are then incorporated into the datastore, with the current decoder representation serving as the new keys, and the corresponding ground truth target tokens as values. **Removal of indistinguishable items:** We meticulously track the retrieval frequency of each key-value pair within the original bidirectional datastore, noting both correct and incorrect times of retrieval. Subsequently, we cull key-value pairs that exhibit a higher frequency of incorrect retrievals compared to correct ones. We believe such pairs likely harbor ambiguous information, which fails to offer constructive external aid during the decoding phase but potentially impedes it. **Pruning of redundant items:** When a given query yields multiple correct target tokens in the retrieval results, while the model output itself is also accurate, we recognize this as an indication of redundancy within the key-value pairs. Consequently, we removed items with high similarity and more distant from the query, effectively pruning redundant information in the datastore.

4.3 Train Robust Meta-network

With previous steps, we have established a relatively accurate and comprehensive datastore. However, a challenge remains in the kNN decoding process, where the parameter λ is typically fixed to directly combine the kNN probability distribution and the model probability distribution for all situations, as depicted in formula 2. However, unlike decoding in AR, where the process is kept exactly the same at the datastore creation and inference stages, NATs do not probabilistically encounter the same distribution of giving y at construction and inference times. Following previous works (Zheng et al., 2021a; Jiang et al., 2022b), we train a lightweight Meta-network to combine model prediction with the $k$nn probability distribution organically and to enhance robustness in noisy situations.

Given a query $\hat{h}_i$, we identify the set of retrieved neighbor pairs $N_i = \{(h_k, v_k)|1 \leq k \leq K\}$. To calibrate the probability distribution of the query, we consider two factors: the L_2 distance between $\hat{h}_i$ and each neighbor key h_k, denoted as d_k, and the count of distinct values among the top k neighbors, denoted as c_k. These metrics are concatenated to form the input feature for Meta-network, which modulates the temperature of the kNN distribution. The kNN distribution is formally defined as:

$$p_{k\text{NN}}(y_i|\hat{h}_i) \propto \sum_{(h_k,v_k)\in N_i} \mathbb{1}_{y_i=v_k} \exp\left(\frac{-d_k}{T}\right), \tag{4}$$

where the temperature T is calculated as:

$$T = \mathbf{W}_1 \left(\tanh\left(\mathbf{W}_2[d_1, \ldots, d_K; c_1, \ldots, c_K]\right)\right), \tag{5}$$

[5]

Upon establishing the kNN distribution, we then explore its adaptive integration with the model predictions, enhancing robustness in noisy scenarios. Following Jiang et al. (2022b), we consider the confidence of NMT distribution and kNN distribution to estimate the weight λ_i adaptively. Furthermore, we incorporate the mask ratio α of the decoder input as a factor to reflect the reliability of the NAT distribution:

$$\lambda_i = \frac{\exp(s_{k\text{NN}})}{\exp(s_{k\text{NN}}) + \exp(s_{\text{NAT}})}, \tag{6}$$

$$s_{k\text{NN}} = \mathbf{W}_3(\tanh(\mathbf{W}_2[d_1, ..., d_K; r_1, ..., r_K])), \tag{7}$$

$$s_{\text{NAT}} = \mathbf{W}_4[p_{\text{NAT}}(v_k|\hat{h}_i); p_{\text{NAT}}(v_k|h_k); \alpha], \tag{8}$$

where k ranges from 1 to K and α is the current mask ratio of decoder input.

4.4 Iterative Decoding with kNN

During inference, we further integrate iterative decoding of NAT models with kNN decoding. The conventional iterative decoding process is defined as follows:

$$P(y_i^{(t)}) = P(y_i^{(t)}|X, \hat{Y}^{(t-1)}; \Theta) \tag{9}$$

[4] We have provided more concrete examples of these deficiencies in A.2

[5] $\mathbf{W}_*$ refers to parameter matrices.

where $y_i^{(t)}$ denotes the predicted word at position i at iteration t, $\hat{Y}^{(t-1)}$ represents the prediction from the previous iteration, and Θ encapsulates the model parameters.

Subsequently, we incorporate kNN retrieval at each iteration for all positions of interest in parallel, employing the Meta-Network we have trained to combine the NAT distribution and the kNN distribution as follows:

$$p(y_i^{(t)}|X, \hat{Y}^{(t-1)}) = \lambda_i p_{k\text{NN}}(y_i^{(t)}|X, \hat{Y}^{(t-1)}) + (1 - \lambda_i) p_{\text{NAT}}(y_i^{(t)}|X, \hat{Y}^{(t-1)}) \quad (10)$$

In our experiments, we limit the maximum number of iterations to 10, which implies that we perform at most 10 kNN retrieval processes. We introduce an early stopping mechanism: if the predictions across all positions remain unchanged between two consecutive iterations, we terminate the iterative process ahead of schedule. This approach not only accelerates inference but also, as we observed, enhances performance to a certain degree.

5 Experiments

5.1 Experimental Settings

Datasets and Evaluation We follow the previous works (Khandelwal et al., 2020; Zheng et al., 2021b) and utilize the German-English multi-domain dataset released by Aharoni and Goldberg (2020). We consider four commonly used domains including IT, Medical, Law, and Koran. We use the Moses[6] toolkit to tokenize sentences and split words into subword units (Sennrich et al., 2015). For evaluation, we use SacreBLEU[7] (Post, 2018) to measure all results with case-sensitive detokenized BLEU (Papineni et al., 2002).

Models We set up three experimental groups as the LLMs, AT, and NAT models and test our proposed method on the NAT model. For LLMs, we utilize GPT-3.5-Turbo (OpenAI, 2022) and GPT-4-Turbo (OpenAI, 2023) APIs and following the prompt and settings given by Jiao et al. (2023). We utilize the vanilla Transformer-base model (Vaswani et al., 2017) as the AT backbone model and CMLMC (Huang et al., 2021) for the NAT model, as they demonstrate similar capabilities on the WMT19 de-en test set. Both models are trained from scratch on the training split of WMT19 de-en datasets, following the default settings in the respective papers to ensure a fair comparison. In the domain adaptation task for NAT models, we establish three baselines: the **Base-NAT** model, which is trained on the WMT19 de-ne dataset, **Domain-specific models**, which are trained on the training split of four domain datasets respectively, and the **vanilla kNN** method directly adapted on the Base-NAT model. Further details can be found in Appendix A.1.

Implementation We adopt fairseq toolkit[8] (Ott et al., 2019) for NMT models and faiss[9] (Johnson et al., 2019) for kNN to conduct datastore creation and retrieval. For NAT models, we set the max iteration number to 10, which is decided adaptively in our proposed method. For all datasets, we use faiss to learn 4k cluster centroids and set the code size to 64. During inference, we set the max search clusters as 8, except for the Koran dataset.

5.2 Main Results

The experimental results are listed in Table 2. Our proposed Bi-kNN strategy demonstrates substantial enhancements over the Base-NAT model across all domains, with an average improvement of 7.8 BLEU points. Additionally, our method surpasses the domain-specific NAT model in a majority of domains, achieving an average uplift of 2.4 BLEU points. In contrast, the vanilla kNN approach fails to yield superior outcomes, and in most cases, it even degrades the original predictions. This suggests that the vanilla kNN approach may not be directly applicable to NAT models as we mentioned in section 3.2. When compared with LLMs and AT models, our NAT model, aided by our proposed Bi-kNN method, exhibits performance on par with the domain-specific AT model. Compared to LLMs and AT models, our NAT model, aided by our proposed Bi-kNN method, demonstrates significant competitiveness with fewer model parameters and enhanced inference speed. We have expanded our evaluation to include COMET and ChrF metrics, the results can be found at A.1.

5.3 Cost Anlysis

Training Cost A significant merit of kNN methods is their non-parametric characteristic, eliminating the necessity for extra training to adapt the original model's parameters to domain-specific datasets.

[6] https://github.com/moses-smt/mosesdecoder
[7] https://github.com/mjpost/sacrebleu
[8] https://github.com/pytorch/fairseq
[9] https://github.com/facebookresearch/faiss

	WMT '19	IT	Medical	Law	Koran	Avg.
Test set sizes	2,000	2,000	2,000	2,000	2,000	-
LLM						
- `GPT-3.5-turbo`	39.59	33.66	41.31	38.52	17.03	32.63
- `GPT-4-turbo`	39.17	33.67	41.49	41.01	18.00	33.56
Base AT	37.15	36.01	37.38	41.85	14.41	32.41
- Domain-specific models	-	40.80	50.70	57.86	12.32	40.22
Base NAT	37.03	33.25	35.84	38.93	13.91	30.48
- Domain-specific models	-	36.59	46.30	**49.77**	10.75	35.85
+kNN:						
- vanilla kNN	37.03	32.30	33.80	36.28	10.49	28.21
- Bi-kNN(ours)	37.03	**39.30**	**46.47**	49.27	**18.10**	**38.28**

Table 2: Experimental results on German-English Multi-domain translation tasks. **Bold** denotes the best performance for NAT models. The performance improvements over Base-NAT are statistically significant with $p < 0.05$. All experiments are our own implementation; more details can be found in Appedix A.1.

Models	**Training Cost** (gpu·hours)	**Inference Speed** (tokens/s)
Domain-specific AT	2.13	149.63
Domain-specific NAT	3.39	263.66
Base-AT		
+ fine-tuning	1.68	154.51
+ vanilla-kNN	< 0.1	124.18
+ adaptive-kNN	0.26	118.41
Base-NAT		
+ fine-tuning	2.67	259.65
+ vanilla-kNN	< 0.1	218.74
+ Bi-kNN (ours)	0.38	203.60

Table 3: Comparsion of different strategies on training cost and inference speed on koran dataset. The batch size is set to 1, and the beam is set to 4 for all the strategies during inference.

The training cost of kNN methods mainly lies in the datastore creation stage, which utilizes forward passes of the NMT models but without updating the original parameters of the model. Our method incorporates additional meta-network; however, it is quite light-weight, and the number of parameters is negligible compared to the model itself.

We assess the training cost of different strategies, as illustrated in Table 3. Obviously, parameter-updating methods i.e., training a domain-specific or fine-tuning on a pre-trained NMT model, are more time-consuming compared to the non-parametric method. The vanilla kNN only includes a single forward pass over all examples in the training set during the datastore creation stage, so its training cost can be essentially disregarded. On the other hand, adaptive kNN and our method introduce an additional meta-network that requires training, which to some extent increases the training cost, but this cost is completely acceptable compared to parameter tuning methods.

Decoding Speed The primary drawback of the kNN approach lies in its decoding speed; retrieval keys from a dataset containing billions of items in the decoding stage significantly decrease its generation speed. For AT models, retrieval of each position must be performed sequentially, resulting in retrieval times equivalent to the length of the predicted target. In contrast, NAT models generate results on all positions in parallel, originally decreasing the translation latency. Our method incorporated kNN retrieval into the iterative decoding strategy for NAT models, enabling simultaneous retrieval across all positions, thereby reducing retrieval times to the number of iterations. Furthermore, we adaptively set the iteration numbers based on the current prediction, which effectively reduces the number of interactions.

We also evaluate the inference speed of different strategies, as shown in Tabel 3. As clearly demonstrated, the NAT model itself has an advantage in terms of inference speed compared to the AT model, achieving 1.5 to 2 times acceleration. While incorporating kNN leads to a certain loss in inference speed, Bi-kNN still maintains a significant advantage in inference speed compared to AT models especially those equipped with kNN retrieval.

6 Analysis

6.1 Catastrophic Forgetting of NAT

A straightforward approach for domain adaptation involves fine-tuning pre-trained models on specific target domain data. However, studies (Chu et al., 2017; Chu and Wang, 2018; Saunders) have suggested that a direct continuation of training on new

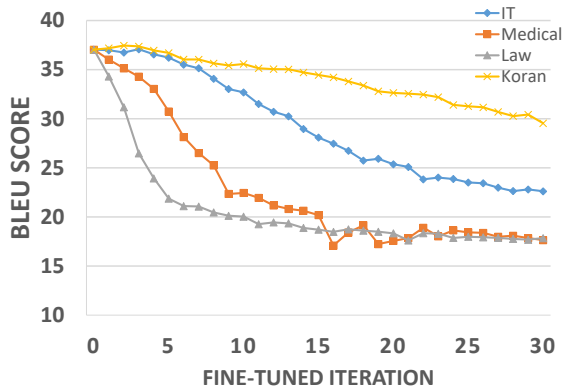

Figure 2: The results of Base-NAT on WMT19 de-en test set with fine-tuning on four domain-specific data.

Models	IT	Medical
(1): Base-NAT	33.25	35.84
(2): (1) + Build Datastore	37.40	44.45
(3): (2) + Renew Datastore	37.79	45.41
(4): (2) + Training Meta-network	38.13	45.76
(5): (3) + Training Meta-network	**39.30**	**46.47**

Table 4: The ablation study on our proposed Bi-kNN. Results on settings without Meta-network utilized the default hypermeters provided by (Khandelwal et al., 2021).

data often results in overfitting on the new data and catastrophic forgetting (French, 1999) of performance on previous domains for AT models. We conducted experiments to verify if NAT models exhibit similar issues, and the results are depicted in Figure 2. While fine-tuning the model with data from the new domain, we observe a consistent decrease in the model's proficiency in the initial training data across all four domains. This indicates that NAT models, akin to their AR counterparts, are susceptible to catastrophic forgetting, Which further amplifies the advantages of non-parametric methods in domain adaptation tasks.

6.2 Effect of Each Part

Our proposed method includes multiple strategies, i.e., building a bidirectional datastore, renewing the indecipherable datastore, and training a robust Meta-network. We conduct an ablation study to analyze each component's contributions to the whole process in this section. The results are listed in Table 4. The introduction of bidirectional information as the key value information of the kNN datastore has already significantly improved the effectiveness of obtaining externally stored knowledge during the

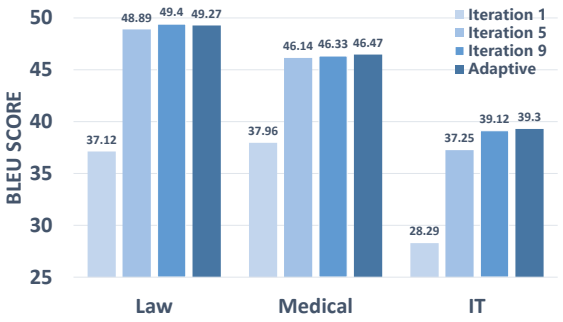

Figure 3: The performance of setting different iteration numbers (1, 5, 9 and adaptive numbers) in the decoding stage on three domain datasets.

decoding stage. The updating of the datastore and the training of a robust original network is aimed at improving the quality of the datastore and better integrating it with the model's prediction results. It is evident that although the improvement brought by the renewed datastore is relatively modest, it reduces a large amount of redundant information in the datastore and increases retrieval efficiency. At the same time, it is apparent that without datastore update operations, subsequent network training may be affected by dirty data in the datastore, leading to the final effect. The Meta-network not only brings about certain improvements but also increases the overall stability of the method, allowing the kNN probability distribution to be adaptively combined with the model's own output, rather than relying on fixed hyperparameters.

6.3 Effect of Iteration numbers

In this subsection, we explore the influence of varying iteration numbers during the decoding phase, as depicted in Figure 3. Evidently, with a small number of iterations (=1), the performance is notably inadequate across all the domains. This shortfall is attributable to the NAT model's initial inability to extract valuable latent contextual information from the preceding iteration, which further leads to an imprecise kNN search. As we increase the number of iterations ($\geq$5), there emerges a remarkable improvement in performance. Nevertheless, this pattern does not suggest that performance enhancements are inexorably tied to more iteration numbers as has proven in past works (Gu et al., 2019; Kasai et al., 2020). Our investigations further indicate that employing adaptive strategies to calibrate the number of iterations can frequently yield superior results while mitigating the decoding cost.

7 Conclusion

In this paper, we highlight the ongoing challenge of domain adaptation for NMT and point out the deficiency of NAT models for domain adaptation tasks. Subsequently, we introduce Bi-kNN, an innovative domain adaptation approach, tailoring kNN for NAT models, which creates a robust bidirectional datastore and integrates iterative decoding with kNN retrievals. Extensive evaluation results and in-depth analysis consistently demonstrate the overall effectiveness and efficiency of our method.

8 Limitations

Despite the promising results of our proposed Bi-kNN, several limitations remain that should be addressed in future work:

- While our method's reduction of kNN searches to the number of iterations, retrieving from the datastore continues to impose an overhead on the decoding process. Future work will aim to minimize retrieval costs and further accelerate the inference stage.
- The introduction of the Meta-network in our approach adds an extra training process over the standard kNN-MT, thus marginally increasing the overall training cost.
- Our experiments focused solely on domain adaptation tasks, showcasing the effectiveness of Bi-kNN. The potential effect of incorporating our method into other tasks, such as Language Modeling and Question Answering, has yet to be explored.

These limitations highlight further investigation and refinement to enhance our method's applicability and performance in a wider range of scenarios.

9 Acknowledgements

We want to thank all the anonymous reviewers for their valuable comments. This work was supported by the National Science Foundation of China (NSFC No. 62206194), the Natural Science Foundation of Jiangsu Province, China (Grant No. BK20220488), Young Elite Scientists Sponsorship Program by CAST (2023QNRC001). The work of Kehai Chen was supported by the National Natural Science Foundation of China under Grant 62276077 and Guangdong Basic and Applied Basic Research Foundation (2024A1515011205).

References

Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, et al. 2023. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*.

Sweta Agrawal, Chunting Zhou, Mike Lewis, Luke Zettlemoyer, and Marjan Ghazvininejad. 2022. In-context examples selection for machine translation. *arXiv preprint arXiv:2212.02437*.

Roee Aharoni and Yoav Goldberg. 2020. Unsupervised domain clusters in pretrained language models. *arXiv preprint arXiv:2004.02105*.

Chenhui Chu, Raj Dabre, and Sadao Kurohashi. 2017. An Empirical Comparison of Simple Domain Adaptation Methods for Neural Machine Translation. Comment: 6 pages.

Chenhui Chu and Rui Wang. 2018. A Survey of Domain Adaptation for Neural Machine Translation. Comment: COLING 2018, 16 pages, 9 figures.

Hiroyuki Deguchi, Taro Watanabe, Yusuke Matsui, Masao Utiyama, Hideki Tanaka, and Eiichiro Sumita. 2023. Subset retrieval nearest neighbor machine translation. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 174–189.

Robert M French. 1999. Catastrophic forgetting in connectionist networks. *Trends in cognitive sciences*, 3(4):128–135.

Marjan Ghazvininejad, Omer Levy, Yinhan Liu, and Luke Zettlemoyer. 2019. Mask-predict: Parallel decoding of conditional masked language models. *arXiv preprint arXiv:1904.09324*.

Jiatao Gu, James Bradbury, Caiming Xiong, Victor OK Li, and Richard Socher. 2018. Non-autoregressive neural machine translation. In *International Conference on Learning Representations*.

Jiatao Gu and Xiang Kong. 2020. Fully non-autoregressive neural machine translation: Tricks of the trade. *arXiv preprint arXiv:2012.15833*.

Jiatao Gu, Changhan Wang, and Junbo Zhao. 2019. Levenshtein transformer. *Advances in Neural Information Processing Systems*, 32.

Pei Guo, Yisheng Xiao, Juntao Li, and Min Zhang. 2023. Renewnat: Renewing potential translation for non-autoregressive transformer.

Junxian He, Graham Neubig, and Taylor Berg-Kirkpatrick. 2021. Efficient nearest neighbor language models. In *Conference on Empirical Methods in Natural Language Processing*.

Amr Hendy, Mohamed Abdelrehim, Amr Sharaf, Vikas Raunak, Mohamed Gabr, Hitokazu Matsushita, Young Jin Kim, Mohamed Afify, and Hany Hassan Awadalla. 2023. How good are gpt models at machine translation? a comprehensive evaluation.

Xiao Shi Huang, Felipe Perez, and Maksims Volkovs. 2021. Improving non-autoregressive translation models without distillation. In *International Conference on Learning Representations*.

Hui Jiang, Ziyao Lu, Fandong Meng, Chulun Zhou, Jie Zhou, Degen Huang, and Jinsong Su. 2022a. Towards Robust k-Nearest-Neighbor Machine Translation. Comment: Accepted to EMNLP 2022.

Hui Jiang, Ziyao Lu, Fandong Meng, Chulun Zhou, Jie Zhou, Degen Huang, and Jinsong Su. 2022b. Towards robust k-nearest-neighbor machine translation. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 5468–5477.

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 2021 Conference on Empirical Methods in Natural Language Processing*, pages 7280–7290, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Wenxiang Jiao, Wenxuan Wang, Jen-tse Huang, Xing Wang, Shuming Shi, and Zhaopeng Tu. 2023. Is ChatGPT A Good Translator? Yes With GPT-4 As The Engine. Comment: Analyzed/compared the outputs between ChatGPT and Google Translate; both automatic and human evaluation.

Jeff Johnson, Matthijs Douze, and Hervé Jégou. 2019. Billion-scale similarity search with gpus. *IEEE Transactions on Big Data*, 7(3):535–547.

Jungo Kasai, James Cross, Marjan Ghazvininejad, and Jiatao Gu. 2020. Non-autoregressive machine translation with disentangled context transformer. In *International conference on machine learning*, pages 5144–5155. PMLR.

Urvashi Khandelwal, Angela Fan, Dan Jurafsky, Luke Zettlemoyer, and Mike Lewis. 2020. Nearest neighbor machine translation. In *International Conference on Learning Representations*.

Urvashi Khandelwal, Angela Fan, Dan Jurafsky, Luke Zettlemoyer, and Mike Lewis. 2021. Nearest Neighbor Machine Translation. Comment: ICLR 2021.

Yoon Kim and Alexander M Rush. 2016. Sequence-level knowledge distillation. *arXiv preprint arXiv:1606.07947*.

Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Jason Lee, Elman Mansimov, and Kyunghyun Cho. 2018. Deterministic non-autoregressive neural sequence modeling by iterative refinement. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 1173–1182.

Rui Lv, Junliang Guo, Rui Wang, Xu Tan, Qi Liu, and Tao Qin. 2023. N-Gram Nearest Neighbor Machine Translation.

Chenyang Lyu, Jitao Xu, and Longyue Wang. 2023. New trends in machine translation using large language models: Case examples with chatgpt. *arXiv preprint arXiv:2305.01181*.

Pedro Martins, Zita Marinho, and André FT Martins. 2022a. Efficient machine translation domain adaptation. In *Proceedings of the 1st Workshop on Semi-parametric Methods in NLP: Decoupling Logic from Knowledge*, pages 23–29.

Pedro Henrique Martins, Zita Marinho, and André FT Martins. 2022b. Chunk-based nearest neighbor machine translation. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 4228–4245.

Yuxian Meng, Xiaoya Li, Xiayu Zheng, Fei Wu, Xiaofei Sun, Tianwei Zhang, and Jiwei Li. 2022. Fast nearest neighbor machine translation. In *Findings of the Association for Computational Linguistics: ACL 2022*, pages 555–565.

Yasmin Moslem, Rejwanul Haque, John D Kelleher, and Andy Way. 2022. Domain-specific text generation for machine translation. *arXiv preprint arXiv:2208.05909*.

Yasmin Moslem, Rejwanul Haque, John D Kelleher, and Andy Way. 2023. Adaptive machine translation with large language models. *arXiv preprint arXiv:2301.13294*.

Nathan Ng, Kyra Yee, Alexei Baevski, Myle Ott, Michael Auli, and Sergey Edunov. 2019. Facebook fair's wmt19 news translation task submission. *arXiv preprint arXiv:1907.06616*.

OpenAI. 2022. Gpt-3.5-turbo.

OpenAI. 2023. Gpt-4.

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. *NAACL HLT 2019*, page 48.

Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, et al. 2022. Training language models to follow instructions with human feedback. *Advances in Neural Information Processing Systems*, 35:27730–27744.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th annual meeting of the Association for Computational Linguistics*, pages 311–318.

Matt Post. 2018. A call for clarity in reporting bleu scores. *arXiv preprint arXiv:1804.08771*.

Lihua Qian, Hao Zhou, Yu Bao, Mingxuan Wang, Lin Qiu, Weinan Zhang, Yong Yu, and Lei Li. 2021. Glancing transformer for non-autoregressive neural machine translation. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 1993–2003.

Raphael Reinauer, Patrick Simianer, Kaden Uhlig, Johannes E. M. Mosig, and Joern Wuebker. 2023. Neural Machine Translation Models Can Learn to be Few-shot Learners.

Danielle Saunders. Domain adaptation for Neural Machine Translation.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2015. Neural machine translation of rare words with subword units. *arXiv preprint arXiv:1508.07909*.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008.

Dexin Wang, Kai Fan, Boxing Chen, and Deyi Xiong. 2022a. Efficient cluster-based k-nearest-neighbor machine translation-nearest-neighbor machine translation. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2175–2187.

Dongqi Wang, Haoran Wei, Zhirui Zhang, Shujian Huang, Jun Xie, and Jiajun Chen. 2022b. Non-parametric online learning from human feedback for neural machine translation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 36, pages 11431–11439.

Yisheng Xiao, Ruiyang Xu, Lijun Wu, Juntao Li, Tao Qin, Tie-Yan Liu, and Min Zhang. 2023. Amom: adaptive masking over masking for conditional masked language model. In *Proceedings of the Thirty-Seventh AAAI Conference on Artificial Intelligence and Thirty-Fifth Conference on Innovative Applications of Artificial Intelligence and Thirteenth Symposium on Educational Advances in Artificial Intelligence*, AAAI'23/IAAI'23/EAAI'23. AAAI Press.

Xinyi Yang, Runzhe Zhan, Derek F Wong, Junchao Wu, and Lidia S Chao. 2023. Human-in-the-loop machine translation with large language model. *arXiv preprint arXiv:2310.08908*.

Chun Zeng, Jiangjie Chen, Tianyi Zhuang, Rui Xu, Hao Yang, Qin Ying, Shimin Tao, and Yanghua Xiao. 2022. Neighbors are not strangers: Improving non-autoregressive translation under low-frequency lexical constraints. In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 5777–5790.

Biao Zhang, Barry Haddow, and Alexandra Birch. 2023. Prompting large language model for machine translation: A case study. *arXiv preprint arXiv:2301.07069*.

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 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, pages 368–374, Online. Association for Computational Linguistics.

Xin Zheng, Zhirui Zhang, Junliang Guo, Shujian Huang, Boxing Chen, Weihua Luo, and Jiajun Chen. 2021b. Adaptive nearest neighbor machine translation. *arXiv preprint arXiv:2105.13022*.

A Appendix

A.1 Experiment Setup

Datasets We utilize four domain datasets for domain adaptation tasks and the WMT19de-en dataset for general NMT tasks. Their statistics are listed in Table 5. Following Aharoni and Goldberg (2020), we use the bpecodes provided by Ng et al. (2019) to process datasets.

Datasets	WMT19	IT	Medical	Law	Koran
Train	37M	223k	248k	467k	18k
Dev	2k	2k	2k	2k	2k
Test	2k	2k	2k	2k	2k

Table 5: Statistics of the datasets.

LLMs We utilize Llama2-chat 7b, ChatGPT, and GPT-4 in our experiment. The Llama2-chat 7b is open-sourced and available at https://huggingface.co/meta-llama. ChatGPT and GPT-4 are closed-sourced, and we utilize the API released by OpenAI. The version for ChatGPT is GPT-3.5-turbo, and for GPT-4, it is GPT-4-1106-preview. We utilize few-shot examples to exploit their abilities better. For ChatGPT and GPT-4, we provide clearer instruction because they have better instruction-following ability, and we set the example number to 1 in our experiments. The prompt we used is depicted in Figure 5. For Llama2, we set the example number to 3 since we find it hard to control the format of its output. The prompt we used is depicted in Figure 4. We post-processed their output results as they may contain irrelevant information such as "English:".

AT and NAT models We establish our baseline models by training them from scratch using the Fairseq library(Ott et al., 2019). The model architectures for both AT and NAT align with the configuration outlined in the respective papers (Vaswani et al., 2017; Huang et al., 2021): 6 layers per stack, 8 attention heads per layer, 512 model dimensions, and 2048 hidden dimensions. For our foundational models, both the Base-AT and Base-NAT are trained on the WMT19de-en dataset from scratch. The Base-AT model adheres to the implementation strategies described by Ng et al. (2019), while the hyperparameters for the Base-NAT model are tuned by us, as illustrated in Table 6. For domain-specific models, we emulate the implementations for both AT and NAT models as proposed by (Aharoni and Goldberg, 2020) as depicted in Table 6. For details, we use the ADAM optimizer (Kingma and Ba, 2014) with an initial learning rate of 0.0005 and a maximum of 4096 tokens per batch. We set the dropout rate to 0.2 and the maximum number of epochs to 200. Early stopping is employed if the BLEU score on the domain-specific development set does not improve in 10 consecutive checkpoints. Notably, we do not adopt sequence-level knowledge distillation (Kim and Rush, 2016), which is a common practice for NAT models, in our experiments to ensure a fair comparison.

Hyperparameters	GPUs	lr	warm-up	dropout	epoches	tokens/GPU
Base-NAT	4xA5000	0.0007	40,000	0.2	250	8,192
Domain-specific	1xA5000	0.0005	4,000	0.2	200	4,096

Table 6: The hyperparameters configuration we used for model training.

Decoding For AT models, we follow the traditional decoding configuration provided by (Ng et al., 2019). For NAT models, we set the beam size to 5 and fix the iteration number to 10 for baselines, while setting the iteration number adaptively in the decoding with kNN.

Other Evaluation Metrics We employed the BLEU metric to maintain consistency with our baseline work, namely $k$nnMT. We have expanded our evaluation to include COMET and ChrF metrics and conducted additional experiments. Specifically, we used the "wmt22 comet da" for COMET and implemented chrF as provided in ScaleBLEU. In figure7, we present the results of our NAT experiments under these expanded evaluation metrics.

AT Baselines Our methodology was primarily designed for NAT models, hence our focus was largely on comparing it with non-autoregressive generation methods. The inherent attributes of NMT models can significantly influence the effectiveness of the knn method. Consequently, a direct comparison between conventional AT models with knn might not present a fair comparison. Nevertheless, we have performed additional experiments on some AT baselines for comparison in 8. The performance gap is still It is evident that there remains a substantial performance gap between the AT and NAT methods. Future efforts should be dedicated to bridging this gap.

A.2 Example

To illustrate the deficiencies as mentioned in section 4, let's consider three scenarios:

1. **Missing Information**: This is the most intuitive case. During the construction of the

	IT	Medical	Law	Koran
Domain Models	0.7819 / 58.10	0.7846 / 64.04	**0.8089 / 68.92**	0.5231 / 33.17
BaseNAT	0.7902 / 55.37	0.7798 / 57.59	0.7945 / 60.49	**0.6546** / 36.85
+ vanilla	0.7663 / 53.94	0.7337 / 56.09	0.7368 / 59.12	0.5669 / 34.20
+ Bi-kN	**0.8153 / 58.35**	**0.7870 / 64.66**	0.7905 / 67.79	0.6335 / **40.51**

Table 7: Evaluation results using COMET / ChrF metrics.

	IT	Medical	Law	Koran
Transformer-base	36.01	37.38	41.85	14.41
+ Vanilla-kNN	43.80	53.25	60.06	16.94
+ Adaptive-kNN	46.73	55.95	61.69	18.52

Table 8: Performance of Auto-regressive translation methods

database, we perform random masking on the input, which means there is a possibility that crucial information may not be selected and included in the database.

2. **Ambiguity**: This issue arises because NATs do not provide as clear a context as ATs in the teacher forcing pattern during datastore construction. The NAT model generates hidden states in parallel under conditions where only a fraction of the words are visible, which can lead to ambiguity. For example, given the sentence "All work and no play makes Jack a dull boy," the input to the NAT model might be "All [MASK1] and no [MASK2] makes Jack a dull boy" during construction. The hidden state of the NAT at [MASK2] could correspond to either "work" or "play." However, for the AR model, when at the [MASK2] position, "all work and no" is known, allowing the model to generate a more accurate key.

3. **Redundancy**: Since we adopt random masking during the construction phase, the same position might be selected multiple times. Additionally, there may be multiple similar hidden states pointing to the same value, leading to redundancy.

To address these issues, we have an in-depth discussion in the "Renew Datastore" section of our methodology and propose corresponding solutions.

Prompt:

```
You are a helpful and precise assistant, following the examples and translate the following
German sentence into English. You only need to give the translation directly, no explanation or
other information.

German: In diesem Sinne untergraben diese Maßnahmen teilweise das demokratische System der USA.
English: In this sense , the measures will partially undermine the American democratic

German: {Input}
```

Figure 4: Prompt we used in our experiments for ChatGPT and GPT-4.

Prompt:

```
You are a helpful and precise assistant, following the examples and translate the following
German sentence into English.

German: In diesem Sinne untergraben diese Maßnahmen teilweise das demokratische System der USA.
English: In this sense , the measures will partially undermine the American democratic

German: Eine Irren-Anstalt, wo sich heute Jugendliche begegnen sollen.
English: A mental asylum, where today young people are said to meet\n.

German: Heute liegt Untersendling mitten in der Stadt.
English: Today, Untersendling lies in the middle of the city.

German: {Input}
```

Figure 5: Prompt we used in our experiments for Llama2.