A Data-Efficient Nearest-Neighbor Language Model via Lightweight Nets

Anonymous ACL submission

Abstract

The kNN-MT method incorporates non-parametric techniques into Neural Machine Translation (NMT) to improve translation performance without fine-tuning the model. However, this approach utilizes the k-nearest neighbors algorithm at the token level, which requires significant storage space and slows down the inference speed. In this paper, we propose training a lightweight neural network as a substitute for kNN search, resulting in a substantial reduction in storage overhead while sacrificing a marginal amount of translation performance. Moreover, the lightweight network also ensures fast inference speed. According to our calculations, the storage overhead of our method is negligible compared to kNN-MT, and we can achieve its 92% to 99% translation performance on different datasets.

1 Introduction

The integration of retrieval-based approaches and language models has garnered substantial interest for enhancing specific NLP tasks (Robertson et al., 2009; Gu et al., 2018; Chen et al., 2017). This combination enables language models to leverage external knowledge, leading to improved performance in diverse NLP applications. kNN-MT (Khandelwal et al.) is a retrieval-based method that can improve the performance of the model on a specific translation dataset without training. Due to its property of not disrupting the knowledge of pre-trained models, it is natural to consider applying kNN-MT to cross-domain machine translation tasks. Unlike other methods based on fine-tuning (Wilkinghoff, 2023) or data augmentation (Chu et al., 2017; Tars and Fishel, 2018; Stergiadis et al.; Chan et al., 2023), kNN-MT does not suffer from catastrophic forgetting, which is a limitation present in most existing methods.

However, kNN-MT requires storing the entire training data and comparing the contextual information from the NMT output with each entry in the training data for every translation step. This results in significant storage requirements and slows down the inference speed of the entire system. Despite the utilization of the *faiss* library to accelerate the kNN search process, which partially alleviates the issue of slow inference speed, the storage overhead remains a hurdle for the practical application of this method. To address the storage overhead problem, various works have proposed filtering algorithms to eliminate redundant training data entries. These methods reduce the storage burden through compression and filtering techniques. In contrast, we propose replacing the search process in kNN-MT with a lightweight network. Specifically, for each domain, we utilize the output of the decoder hidden state as input and use the nearest neighbors obtained from kNN search, along with the target token, as supervision to train a lightweight neural network. At test time, this network will serve as a replacement for the kNN search module. This approach significantly improves inference speed and greatly reduces storage burden in practical applications. Besides, our experimental results demonstrate that our method recovers between 92% and 99% of the translation performance of kNN-MT across different domains.

2 Background: kNN-MT

In this section, we first elaborate on the background and principles of kNN-MT, describe its inference process, and then introduce recent works based on it.

kNN-MT is a recently proposed method that leverages external retrieval information to improve the performance of neural machine translation models. It works as an auxiliary component to a base NMT model, using the training dataset to construct a datastore for translation without requiring additional training. The datastore consists of a large

number of key-value pairs. For a given bilingual sentence pair (X, Y) in the training set, the NMT model translates every token y_t in t step with text information $(X, y_{<t})$. Then, the key is $f(X, y_{<t})$ and the value is y_t and the datastore is constructed by

$$(\mathcal{K}, \mathcal{V}) = \bigcup_{(x,y)\in(\mathcal{X},\mathcal{Y})} \{(f(x, y_{<t}), y_t), y_t \in y\}. \quad (1)$$

kNN-MT requires traversing and storing the entire training data for every domain. During inference, for each decoding step t, kNN-MT predicts token $\hat{y}_t$ with the combination of the output distribution of NMT and the kNN search results in datastore based on $f(x, y_{<t})$. Assuming $N_k = \{(h_i, v_i), i \in \{1, 2, ..., k\}\}$ represents the nearest neighbors obtained from kNN search, the associated predictive distribution can be represented as

$$p(y_t|x, \hat{y}_{<t}) = \sum_{(h_i,v_i)} 1_{y_t=v_i} exp(\frac{-d(h_i, f(x, \hat{y}_{<t}))}{T}). \quad (2)$$

The expression involves the temperature coefficient denoted by T and the Euclidean distance or l_2 distance denoted by $d(,)$. The calculation of the distribution is essentially based on the distances between the nearest neighbors obtained from the search and the query hidden state. Based on the above calculation process, the final predicted probability distribution $\hat{y}$ can be expressed as

$$\begin{aligned} p(y_t|x, \hat{y}_{<t}) = \lambda p_{kNN}(y_t|x, \hat{y}_{<t}) + \\ (1-\lambda) p_{NMT}(y_t|x, \hat{y}_{<t}), \end{aligned} \quad (3)$$

where λ denotes the interpolated superparameter of the two distributions.

kNN-MT significantly enhances the translation performance of NMT models on specific datasets. However, the substantial storage overhead and slower inference speed limit its practicality. In recent years, extensive research has been conducted to optimize kNN-MT by reducing storage overhead (Zhu et al., 2022; Zheng et al., 2021; Wang et al., 2022) and improve the search speed (Meng et al., 2022; Wang et al., 2021; Xu et al., 2023). These efforts eliminate redundant key-value pairs in the datastore and to some extent enhance the translation performance of kNN-MT.

3 Methodology

Due to the necessity of computing distances between each query point and all points in the datastore, the computational complexity of kNN-MT

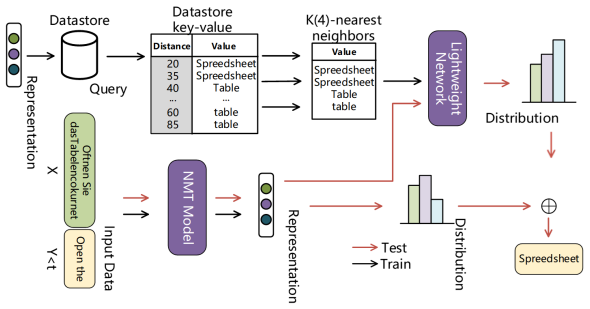

Figure 1: The main framework of our proposed method.

is quite high. Although acceleration libraries such as *faiss* can alleviate this issue, the process of constructing a datastore for every dataset and maintaining storage makes deployment challenging. As illustrated in Figure 1, we train a lightweight network as a replacement for the kNN datastore, while keeping the parameters of the NMT model frozen to ensure the retention of its existing knowledge. Specifically, we utilize the hidden state from the last layer of the NMT model as input. The nearest neighbors obtained through kNN search, along with the target token, are used as the ground truth for training. The corresponding loss function is formulated as

$$\begin{aligned} \mathcal{L}_{KL} &= \sum_{i}^{N} p_i^{KNN} \log \frac{p_i^{KNN}}{q_i^{nn}}, \\ \mathcal{L}_{CE} &= -\sum_{i}^{N} (y_i^{gt} \log q_i^{nn}), \end{aligned} \quad (4)$$

where p and q represent the probability distributions obtained from kNN search and the NMT output, respectively. We optimize the KL divergence loss of both. y_i^{gt} is the target token for the current context, which we use to optimize a CE loss. The variable N represents the current batch size. We utilize the aforementioned loss to train the lightweight network as a replacement for the kNN search module. Compared to original kNN-MT, our method eliminates the need for extensive nearest neighbor search during the testing phase. Furthermore, the storage cost of retaining a lightweight network with only a few layers is much lower than storing all the training data.

We further analyze the impact of the lightweight network for translation performance. Both kNN search and the lightweight network are fundamentally involved in classifying candidate words within a high-dimensional space. Our lightweight network can be seen as an additional transformation of the feature space generated by the NMT, with the expectation of aligning the resulting feature space

with the outcomes of the kNN search. However, even for the same token, it can have vastly different meanings in different contextual contexts, leading to highly complex decision boundaries in feature spaces. Therefore, it is challenging for the feature space transformed by lightweight networks to align perfectly with the results obtained from kNN search. So directly replacing the datastore with a lightweight network would result in some performance loss.

To address this issue, the feature space needs to be further optimized. Specifically, we propose further incorporating a sublinear layer between the lightweight network and the NMT. We train this layer to optimize a Mean Squre Error (MSE) loss, aiming to reduce the distance between word vectors and improve the decision boundaries. More specifically, we minimize the distance between the context representation, $f(x, y_{<t})$, generated by the NMT model, and the keys corresponding to y in the datastore. By bringing nearer neighbors closer to each other, this approach effectively alleviates the issue of lightweight networks easily misclassifying points located on different cluster boundaries in the feature space. The corresponding loss function can be formulated as follows:

$$\mathcal{L}_{dis} = \frac{1}{N}\sum_{i}^{N}(h_i - \frac{1}{M}\sum_{j}^{M} h_j)^2. \quad (5)$$

In the formula, h_i denotes the hidden state produced by the decoder of the NMT model, h_j represents the contextual information stored in the datastore, and M represents the number of nearest neighbors retrieved through kNN search for the target token. The final training loss for training lightweight network can be obtained by combining these three losses and the corresponding formula can be expressed as follows:

$$\mathcal{L}_{total} = \mathcal{L}_{CE} + \alpha_1\mathcal{L}_{dis} + \alpha_2\mathcal{L}_{KL}, \quad (6)$$

where α_1 and α_2 are hyperparameters for each extra loss.

4 Experiments

4.1 Experimental Setup

Datasets. We use the WMT19 data for German-English (Ng et al., 2019) as the general domain corpus and we follow adaptive kNN-MT (Zheng et al., 2021) which consider domains including IT, Medical, Koran, and Law as specific domain corpus. The statistics of the above datasets are illustrated in 2. Before training, we use Moses toolkit to tokenize the sentences and we split words into subwords with bpe-codes. We utilize ScareBLEU to evaluate the translation performance of different methods.

Implementation Details. We use the *fairseq* framework and accelerate the kNN search process using the *faiss* acceleration library, which is also used in most kNN-MT based work. We use the same training hyperparameters as in Adaptive-kNN-MT (Zheng et al., 2021). Specifically, we set the distribution interpolation parameter λ to 0.7, the initial learning rate to 3e-4, the batch size to 8, and the update frequency to 16. We optimize the network parameters using the Adam optimizer.

4.2 Main Results

The main experimental results are presented in Table 1. We compare our method with other kNN-MT based approaches in various domains. For instance, Adaptive kNN-MT achieves 5.37 BLEU improvement in the IT domain. In comparison, our method achieves a significant optimization in storage overhead while sacrificing a small amount of translation performance. For example, in the IT domain, our method achieved 88.6% of the performance with just 0.56% of the storage overhead. Similar effects can be observed on datasets of other domains. Moreover, it can be seen that fine-tuning in specific domain dataset can improve the translation performance to some extent. For instance, in the Medical domain, fine-tuning achieves a BLEU score of 51.98, surpassing the 49.00 BLEU score of kNN-MT. However, these approaches can disrupt the exsiting knowledge of the base model, leading to a decline in its performance in general domain.

4.3 Ablation Study

We conduct ablation experiments on the IT and Koran datasets to evaluate the impact of the proposed KL loss and MSE loss. As shown in Table 3, in both domains, incorporating the two proposed losses results in improvements in translation performance compared to using only target token supervision.

The kNN algorithm is highly sensitive to the choice of parameter K, which often needs to be manually set. A larger value of K may introduce irrelevant samples into the search, thereby causing interference in the training process. Conversely, a smaller value of K may result in the retrieval of a

	Method	IT		Koran		Law		Medical	
		BLEU	Params	BLEU	Params	BLEU	Params	BLEU	Params
	Ours	42.02	**2.09M**	19.87	**2.09M**	50.88	**2.09M**	47.56	**2.09M**
	PLAC kNN-MT	47.00	221M	19.9	34M	62.8	10B	56.2	3B
	Adaptive kNN-MT	**47.39**	369M	**20.30**	53M	**62.90**	19B	**56.10**	7B
	kNN-MT	45.23	369M	19.65	53M	54.98	19B	49.00	7B
Baseline	Mix domain Train	31.68	-	15.95	-	46.50	-	40.66	-
	Finetune	43.19	-	22.85	-	56.47	-	51.98	-
	General Train	36.81	-	16.96	-	45.50	-	39.51	-

Table 1: The BLEU scores and additional storage overhead of different Methods. The baseline method does not involve additional storage, but fine-tuning disrupts the existing knowledge of NMT. "Finetune" means fine-tuning on a specific domain, "General Train" means testing directly without fine-tuning on any domain, and "Mix domain Train" means fine-tuning after mixing training data from all domains. "-" means no additional storage overhead.

Dataset	IT	Medical	Koran	Laws
Train	222, 927	248, 009	17, 982	467, 309
Dev	2000	2000	2000	2000
Test	2000	2000	2000	2000

Table 2: Statistics of dataset in different domains.

	IT	Koran
Ours	42.02	19.87
w/o MSE loss	40.82	18.55
w/o KL loss	40.02	19.67

Table 3: Ablation experiments for KL and MSE loss on IT and Koran domains.

limited number of relevant samples, affecting the generalization ability of the trained lightweight network. We conduct experiments to investigate the impact of different K values on translation performance, as shown in figure 2 left. Based on experimental results, we uniformly set the K value to 4 in our experiments.

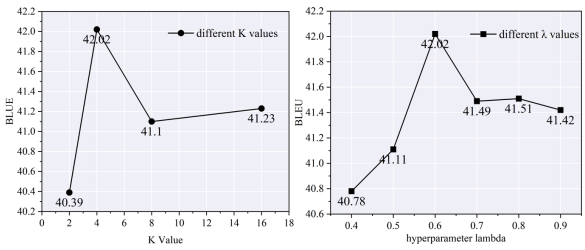

Figure 2: Performance of lightweight network with different hyperparameters

Figure 2 right illustrates the translation performance of the lightweight network on IT domain data for different values of the distribution interpolation parameter λ. It can be observed that the translation performance is highest when the value of λ is set to 0.6. Moreover, when λ is greater than 0.5, the performance generally surpasses that when λ is less than 0.5, which indicates that when the distribution of the output from the lightweight network has a higher proportion, the performance improvement is more significant.

KL	BLEU	MSE	BLEU
0.1	40.05	0.1	40.93
0.3	41.15	0.3	41.03
0.5	41.35	0.5	41.45
1	42.02	1	42.02

Table 4: The experiments of different α values in loss function on IT domain.

We conduct experiments of different α in final loss function as shown in figure 4. We observe that as the α increases, the translation performance gradually improves and it can be seen that when the α_1 and α_2 are 1.0, the translation performance is best. We also explore the structure of lightweight networks and find that a single layer of nonlinear units yield best performance. Increasing the depth of the network make it difficult to train and fit kNN results.

5 Conclusion

In this paper, we propose a approach which replaces the search and matching process of kNN-MT with a lightweight network. Compared to using a datastore, the lightweight network significantly reduces storage overhead and ensure fast inference speed while sacrificing only a small amount of performance. We conduct cross-domain machine translation experiments on four datasets from different domains. Compared with the original kNN-MT, the performance loss of our method is acceptable while greatly reducing storage overhead.

6 Limitations

This study presents several limitations. First, the lightweight network cannot completely replace datastore in terms of translation quality. It significantly reduce storage costs, but it also has a certain performance loss. Besides, although BLEU score is widely used as an evaluation metric, it has certain limitations. The use of large language models (LLMs) as evaluation models needs to be explored in the future. In addition, separate lightweight networks need to be trained for different domains, which is the same as kNN-MT requiring separate datastore for different domains.

References

David M Chan, Shalini Ghosh, Ariya Rastrow, and Björn Hoffmeister. 2023. Domain adaptation with external off-policy acoustic catalogs for scalable contextual end-to-end automated speech recognition. In *ICASSP 2023-2023 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 1–5. IEEE.

Danqi Chen, Adam Fisch, Jason Weston, and Antoine Bordes. 2017. Reading wikipedia to answer open-domain questions. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1870–1879.

Chenhui Chu, Raj Dabre, and Sadao Kurohashi. 2017. An empirical comparison of simple domain adaptation methods for neural machine translation. *arXiv preprint arXiv:1701.03214*.

Jiatao Gu, Yong Wang, Kyunghyun Cho, and Victor OK Li. 2018. Search engine guided neural machine translation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 32.

Urvashi Khandelwal, Angela Fan, Dan Jurafsky, Luke Zettlemoyer, and Mike Lewis. Nearest neighbor machine translation. *Distances*, 4(3):100.

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.

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*.

Stephen Robertson, Hugo Zaragoza, et al. 2009. The probabilistic relevance framework: Bm25 and beyond. *Foundations and Trends® in Information Retrieval*, 3(4):333–389.

Emmanouil Stergiadis, Satendra Kumar, Fedor Kovalev, and Pavel Levin. Multi-domain adaptation in neural machine translation through multidimensional tagging.

Sander Tars and Mark Fishel. 2018. Multi-domain neural machine translation. *arXiv preprint arXiv:1805.02282*.

Dexin Wang, Kai Fan, Boxing Chen, and Deyi Xiong. 2022. 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.

Shuhe Wang, Jiwei Li, Yuxian Meng, Rongbin Ouyang, Guoyin Wang, Xiaoya Li, Tianwei Zhang, and Shi Zong. 2021. Faster nearest neighbor machine translation. *arXiv preprint arXiv:2112.08152*.

Kevin Wilkinghoff. 2023. Design choices for learning embeddings from auxiliary tasks for domain generalization in anomalous sound detection. In *ICASSP 2023-2023 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 1–5. IEEE.

Frank F Xu, Uri Alon, and Graham Neubig. 2023. Why do nearest neighbor language models work? *arXiv preprint arXiv:2301.02828*.

Xin Zheng, Zhirui Zhang, Junliang Guo, Shujian Huang, Boxing Chen, Weihua Luo, and Jiajun Chen. 2021. Adaptive nearest neighbor machine translation. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, pages 368–374.

Wenhao Zhu, Shujian Huang, Yunzhe Lv, Xin Zheng, and Jiajun Chen. 2022. What knowledge is needed? towards explainable memory for knn-mt domain adaptation. *arXiv preprint arXiv:2211.04052*.