Simple and Scalable Nearest Neighbor Machine Translation

Yuhan Dai*[1], Zhirui Zhang*[2], Qiuzhi Liu[2], Qu Cui[2], Weihua Li[2], Yichao Du[1], Tong Xu[1]
(∗ equal contribution)

[1]University of Science and Technology of China (USTC)
[2]Tencent AI Lab

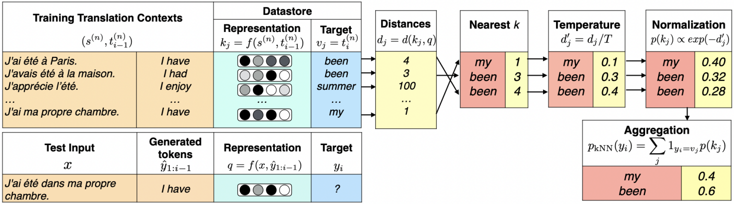
Table of contents

Background

Nearest Neighbor Machine Translation

What is kNN-MT [1]?

- is a simple non-parametric method for machine translation (MT) using nearest neighbor retrieval
- consists of two processes: datastore creation and generation with kNN retrieval
- improves translation accuracy without fine-tuning the entire model
- promising for fast domain adaptation



[1] Khandelwal, et al. "Nearest Neighbor Machine Translation." In ICLR 2021

Challenges

- Problem
  - its large-scale datastore brings massive storage overhead and high latency during inference
- Prior Work
  - Efficient *k*NN-MT[2] and Efficient Cluster-Based *k*NN-MT [3] proposed methods to reduce the datastore size, such as pruning the redundant records and reducing the dimension of keys
  - Fast *k*NN-MT [4] is designed to construct a smaller datastore for each source sentence instead of consulting the entire datastore, migrates the inefficient *k*NN retrieval from the target side to the source side

---

[2] Martins, et al. "Efficient machine translation domain adaptation."
[3] Wang, et al. "Efficient cluster-based k-nearest-neighbor machine translation." In ACL 2022.
[4] Meng, et al. "Fast nearest neighbor machine translation." In ACL Findings 2022.

Our Solution

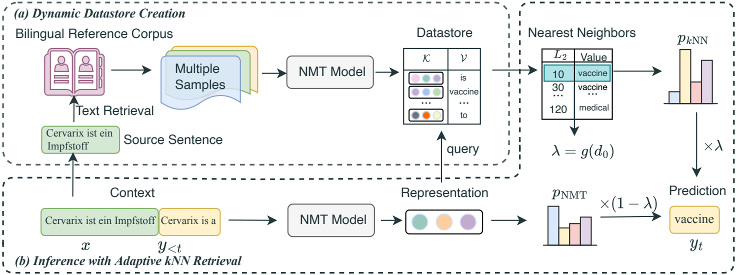
- Analysis
  - We investigate the involved sentence pairs during the decoding process of *k*NN-MT and found that a scarce number of training samples are important

Domain	Full				Involved Samples During Inference			
	Sents	Tokens	Datastore	*k*NN-MT	Sents	Tokens	Datastore	SK-MT
IT	223k	3.6M	6.9G	45.9	7.1	249	0.46M	46.3
Medical	248k	6.9M	14.0G	54.2	9.0	358	0.68M	57.8
Law	467k	19.0M	37.0G	61.3	14.2	730	1.5M	62.7

- Solution
  - Based on the phenomenon, we propose a simple and scalable nearest neighbor machine translation framework (SK-MT)
    - dynamically constructs an extremely small datastore for each input via sentence-level retrieval
    - further introduces a distance-aware adapter to adaptively incorporate the *k*NN retrieval results
  - We illustrate the effectiveness and efficiency in two translation settings
  - We illustrate the simplicity and scalability of our method

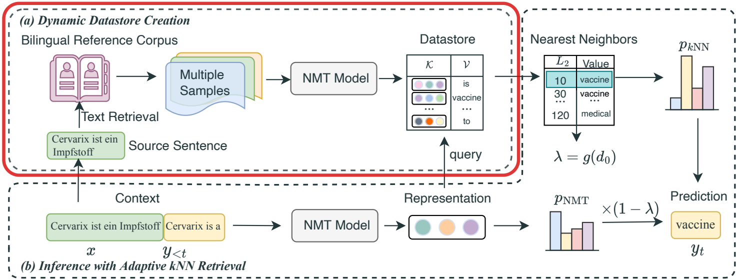
Methodology

Framework Overview



Inspired by our preliminary experiment which demonstrates only a few training samples in the reference corpus are involved during the decoding process, we design a simple and scalable nearest neighbor machine translation framework (SK-MT).

- dynamic datastore construction
- inference with adaptive *k*NN retrieval.

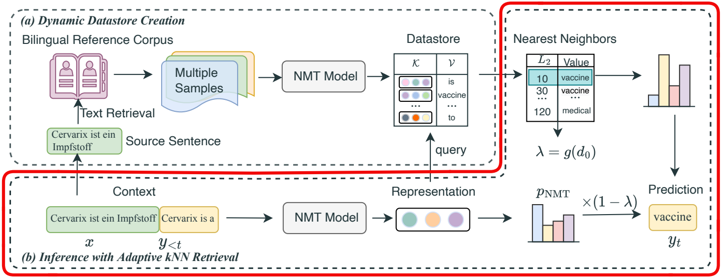


Step 1: Dynamic Datastore Construction

- obtains samples with highest relevance score (BM25).
- re-rank the retrieved bilingual sentences and maintain top-*m*.
- build the datastore by passing top-*m* samples forward to the pre-trained NMT model.

Benefits: filter out the noise and improve inference efficiency

Inference with Adaptive *k*NN Retrieval



Step 2: Inference with Adaptive *k*NN Retrieval

- interpolates the model prob and *k*NN prob as *k*NN-MT does
- explicitly calculate $\lambda = g(d_0) = \mathrm{ReLU}(1 - \frac{d_0}{\tau})$, where d_0 denotes the distance to the nearest neighbor.
- ignores the irrelevant records and magnifies the relevant ones

Benefits: Simple but effective, no need for further training.

More about SK-MT

- Scalability
    - corpus scale: easily adopted in a larger corpus. *k*NN-MT needs large space for datastore creation, while text retrieval is much more space-efficient.
    - translation samples: easily use different samples, but it is difficult to make changes to a pre-defined datastore
- Relation to Translation Memory
    - in a similar framework to [5] but introduces *k*NN retrieval to achieve shallow fusion
    - inherits the advantage of *k*NN-MT: does not need extra training
    - text retrieval improves *k*NN-MT's efficiency
    - achieves better or comparable performance than state-of-the-art translation-memory methods. The results are included in the Appendix of our paper.

[5] Gu, et al. "Search Engine Guided Neural Machine Translation." In AAAI 2018

Experiment

Task Settings

The experiments are conducted in two general settings: static domain adaptation and online learning from human feedback.

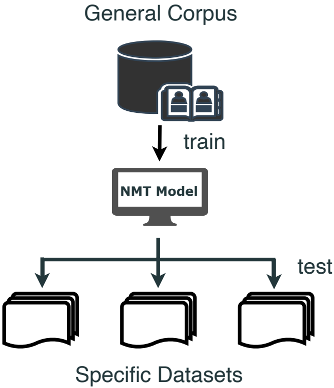


Figure 1: domain adaptation

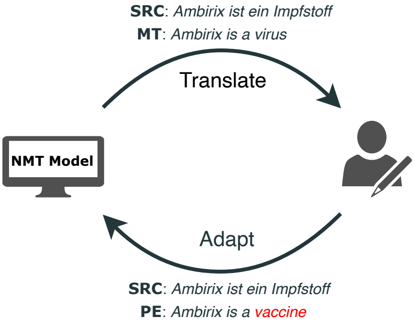


Figure 2: online learning

Dataset Description

Table 1: The statistics of EMEA and JRC-Acquis datasets for online learning.

Bucket	0-50	50-100	100-200	200-500	500-1000
			EMEA		
Documents	22	14	7	4	5
Ave sentences	38.4	73.0	157.9	392.8	759.2
Ave tokens	1174.7	1938.9	3466.1	9334.5	22725.6
			JRC-Acquis		
Documents	22	14	7	4	5
Ave sentences	38.1	73.1	158.5	373.8	734.8
Ave tokens	1347.1	2466.7	5345.4	12518.2	26409.2

- multi-domain datasets including IT, Medical, Koran, Law.
- online learning datasets including EMEA and JRC-Acquis.

Table 2: The statistics of multi-domain dataset.

	Koran	IT	Medical	Law
Train Sents	18k	223k	248k	467k
Dev Sents	2000	2000	2000	2000
Test Sents	2000	2000	2000	2000

Hyperparameters Selection

Since the parameters of the pre-trained models are frozen, the parameters we need to take care of include:

- m: the number of samples to build a datastore
- k: the number of neighbors retrieved
- τ: the temperature to control the sharpness of the softmax function

Table 4: Grid search on m and k on IT development set with the temperature τ fixed to 100. The $*$ marks the two selected models (SK-MT$_1$ and SK-MT$_2$) in our experiments.

m \ k	BLEU 1	BLEU 2	BLEU 3	BLEU 4	ChrF 1	ChrF 2	ChrF 3	ChrF 4
1	42.7	42.2	41.9	40.8	61.7	61.3	61.0	60.5
2	43.4*	43.0	42.5	41.6	62.1*	61.7	61.5	60.9
4	43.5	43.8	43.0	42.3	62.2	62.1	61.7	61.3
8	43.3	**43.9**	43.4	42.7	62.1	**62.3**	62.0	61.4
16	43.3	**43.9***	43.6	43.0	62.0	**62.3***	62.1	61.8

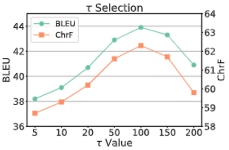


Figure 2: Temperature selection on IT development set.

We use two model architectures in our experiments:
$m = 2, k = 1, \tau = 100$ as SK-MT$_1$ (for efficiency) and
$m = 16, k = 2, \tau = 100$ as SK-MT$_2$ (for performance).

Main Results: Domain Adaptation

Model	BLEU IT	BLEU Medical	BLEU Koran	BLEU Law	BLEU Avg.	ChrF IT	ChrF Medical	ChrF Koran	ChrF Law	ChrF Avg.
NMT	39.1	41.8	16.9	45.9	35.9	58.9	61.4	39.8	66.0	56.5
*k*NN-MT	45.9	54.2	20.4	61.3	45.5	63.3	69.5	41.3	76.0	62.5
AK-MT	**46.9**	**56.4**	20.3	**62.6**	**46.6**	**64.4**	**71.0**	**41.7**	**76.9**	**63.5**
FK-MT	45.5	53.6	**21.2**	56.0	44.1	-	-	-	-	-
EK-MT	44.4	51.9	20.1	57.8	43.6	-	-	-	-	-
CK-MT	44.2	53.1	19.3	59.7	44.1	-	-	-	-	-
SK-MT$_1$	46.1	56.8	17.6	60.7	45.4	63.6	70.7	40.6	75.5	62.5
- w/o adapter	40.6	47.0	18.4	52.3	39.6	59.7	63.4	40.9	69.1	58.3
SK-MT$_2$	**46.2**	**57.6**	19.5	**62.3**	**46.4**	**64.0**	**71.3**	41.5	**76.4**	**63.3**
- w/o adapter	41.3	51.2	**20.5**	56.3	42.3	60.4	66.0	**42.3**	71.0	59.9

Figure 3: the performance of domain adaptation.

- SK-MT$_1$ do not notice significant performance degradation
- SK-MT$_2$ outperforms vanilla *k*NN-MT and achieves comparable performance to AK-MT
- SK-MT surpasses all the efficient methods
- It verifies the benefit of adjusting λ in an adaptive manner.

Main Results: Online Learning

Table 5: BLEU(↑) and ChrF(↑) on EMEA and JRC-Acquis datasets.

Model	BLEU [0, 50)	BLEU [50, 100)	BLEU [100, 200)	BLEU [200, 500)	BLEU [500, 1000)	BLEU Full	ChrF [0, 50)	ChrF [50, 100)	ChrF [100, 200)	ChrF [200, 500)	ChrF [500, 1000)	ChrF Full
EMEA												
NMT	44.2	43.2	38.3	42.4	40.6	41.7	64.8	63.6	61.5	63.4	64.8	64.1
kNN-MT	43.6	43.4	39.9	43.8	43.8	43.4	63.5	63.7	61.8	64	65.9	64.6
KoK	44.4	44.6	44.1	45.7	53.7	49.2	65.0	65.4	64.4	65.5	71.5	68.2
SK-MT$_1$	45.5	44.8	43.4	45.6	53.2	49.2	65.3	64.7	64.3	65.5	71.6	68.3
SK-MT$_2$	46.1	45.6	43.8	46.3	53.6	**49.7**	65.8	65.1	64.6	65.8	71.8	**68.6**
JRC-Acquis												
NMT	54.1	50.0	42.2	39.9	43.4	44.5	72.2	70.2	65.9	62.9	65.6	66.4
kNN-MT	55.5	52.2	45.7	43.6	47.7	48.1	72.0	70.7	67.8	65.1	68.3	68.3
KoK	56.3	52.4	47.7	44.7	50.1	**49.8**	73.9	72.0	69.2	66.1	70.2	**69.9**
SK-MT$_1$	56.6	52.8	47.2	43.5	47.8	48.5	74.0	72.0	69.0	65.3	68.8	69.1
SK-MT$_2$	57.4	53.7	48.2	44.7	49.5	**49.8**	74.5	72.5	69.4	66.0	69.7	69.8

Figure 4: the performance of online learning.

Analysis: Decoding Speed and Storage Overhead

Table 3: Storage overhead and inference speed on Law test set.

Model	Storage Overhead			Inference Speed (ms/sentence)			
	Datastore	Faiss Index	GPU	batch=1	batch=4	batch=8	batch=16
NMT	-	-	-	300.8 ($\times$1.00)	158.2 ($\times$1.00)	85.1 ($\times$1.00)	79.1 ($\times$1.00)
*k*NN-MT	37.0G	1.3G	✗	1986.4 ($\times$0.15)	749.5 ($\times$0.21)	467.4 ($\times$0.18)	410.5 ($\times$0.19)
			✓	409.6 ($\times$0.73)	195.3 ($\times$0.81)	110.5 ($\times$0.77)	100.5 ($\times$0.79)
SK-MT$_1$	0.16M	-	-	344.4 ($\times$0.87)	184.8 ($\times$0.86)	94.6 ($\times$0.90)	85.0 ($\times$0.93)
SK-MT$_2$	1.34M	-	-	430.5 ($\times$0.70)	225.1 ($\times$0.70)	131.1 ($\times$0.65)	136.2 ($\times$0.58)

- SK-MT requires far less space to create the datastore
- SK-MT's decoding speed is faster than kNN-MT and even comparable to NMT

Analysis: Experiment on Larger-scale WMT'14 Dataset

- *k*NN-based methods show little power in boosting the translation quality
- low similarity greatly attributes to the unfavourable performance

Model	De⇒En					En⇒De				
	BLEU	ChrF	Datastore	FAISS Index	Speed (ms/sent)	BLEU	ChrF	Datastore	FAISS Index	Speed (ms/sent)
NMT	31.4	58.1	-	-	36.7	27.2	57.8	-	-	32.8
*k*NN-MT	31.3	58.0	145G	9.0G	252.4	27.3	57.8	128G	7.9G	343.2
SK-MT$_1$	31.4	58.0	0.06M	-	50.1	27.0	57.8	0.06M	-	43.3
SK-MT$_2$	31.3	58.0	0.46M	-	63.4	27.0	57.8	0.46M	-	50.4

Similarity	WMT'14 De⇒En Sent	WMT'14 De⇒En Percent	WMT'14 En⇒De Sent	WMT'14 En⇒De Percent	JRC-Acquis De⇒En Sent	JRC-Acquis De⇒En Percent	JRC-Acquis En⇒De Sent	JRC-Acquis En⇒De Percent	IT De⇒En Sent	IT De⇒En Percent
$[0, 0.1)$	1	0%	0	0%	2	0%	0	0%	164	8.2%
$[0.1, 0.2)$	70	2.3%	61	2%	116	4.7%	70	2.8%	111	5.6%
$[0.2, 0.3)$	1210	40.3%	1101	36.7%	344	13.6%	288	11.4%	174	8.7%
$[0.3, 0.4)$	1096	36.5%	1125	37.5%	346	13.7%	353	14.1%	229	11.5%
$[0.4, 0.5)$	411	13.7%	468	15.6%	226	0.9%	225	8.9%	141	7.1%
$[0.5, 0.6)$	173	5.8%	198	6.6%	233	9.1%	246	9.7%	543	27.2%
$[0.6, 0.7)$	28	0.9%	35	1.2%	223	8.7%	213	8.4%	296	14.8%
$[0.7, 0.8)$	9	0.3%	11	0.4%	216	0.5%	228	9%	151	7.6%
$[0.8, 0.9)$	4	0.1%	4	0.1%	369	14.8%	418	16.8%	148	7.4%
$[0.9, 1]$	1	0%	1	0%	441	16.6%	442	17.7%	43	2.2%

Recap and Conclusion

Conclusion

- We present SK-MT, a simple and scalable nearest neighbor machine translation approach for fast domain adaptation
- We demonstrate SK-MT produces equivalent or even superior performance than previous *k*NN-based approaches
- We illustrate SK-MT does not require any extra training and is efficient in both decoding time and storage overhead
- It is promising that our proposed SK-MT has a wide range of applications
- Our paper can be found on `https://arxiv.org/abs/2302.12188`
- Our code is released on `https://github.com/dirkiedai/sk-mt`.