Chunk-based Nearest Neighbor Machine Translation

Pedro Henrique Martins[ΨϺ] **Zita Marinho**[♭Ͽ] **André F. T. Martins**[Ψ♀Ϻ]

[Ψ]Instituto de Telecomunicações [Ϻ]Unbabel [♭]Institute of Systems and Robotics

[Ͽ]DeepMind [♀]LUMLIS (Lisbon ELLIS Unit), Instituto Superior Técnico

Lisbon, Portugal

pedrohenriqueamartins@tecnico.ulisboa.pt,

zmarinho@google.com, andre.t.martins@tecnico.ulisboa.pt.

Abstract

Semi-parametric models, which augment generation with retrieval, have led to impressive results in language modeling and machine translation, due to their ability to retrieve fine-grained information from a datastore of examples. One of the most prominent approaches, kNN-MT, exhibits strong domain adaptation capabilities by retrieving tokens from domain-specific datastores (Khandelwal et al., 2021). However, kNN-MT requires an expensive retrieval operation for every single generated token, leading to a very low decoding speed (around 8 times slower than a parametric model). In this paper, we introduce a *chunk-based* kNN-MT model which retrieves chunks of tokens from the datastore, instead of a single token. We propose several strategies for incorporating the retrieved chunks into the generation process, and for selecting the steps at which the model needs to search for neighbors in the datastore. Experiments on machine translation in two settings, static and "on-the-fly" domain adaptation, show that the chunk-based kNN-MT model leads to significant speed-ups (up to 4 times) with only a small drop in translation quality.[1]

1 Introduction

Machine translation has seen remarkable advances due to increasingly powerful neural models (Sutskever et al., 2014; Bahdanau et al., 2015; Vaswani et al., 2017). Most deployed systems are *fully-parametric* (the training data is fully compressed into the parameters of a neural model), but they often struggle when translating rare words or out-of-domain sentences (Koehn and Knowles, 2017), commonly requiring several stages of fine-tuning to adapt to data drift or to new domains. Recently, **semi-parametric methods** have shown great promise, by combining the strengths of parametric models with external databases of parallel sentences, such as translation memories (Gu et al., 2018; Zhang et al., 2018; Bapna and Firat, 2019a; Meng et al., 2021; Zheng et al., 2021a; Jiang et al., 2021; Martins et al., 2022).

One of the most prominent semi-parametric models for machine translation is the k-Nearest Neighbor Machine Translation model (kNN-MT) (Khandelwal et al., 2021), which has led to impressive results, particularly in domain adaptation settings, without requiring fine-tuning. The kNN-MT model constructs domain-specific datastores of parallel sentences and, at inference time, it retrieves similar examples from these datastores, which are used to improve the generation process, through the interpolation of probability distributions. However, kNN-MT only retrieves single tokens—this is inefficient, since the model needs to consult the datastore at every generation step, an expensive operation. Consequently, its decoding speed is around 8 times slower than that of a fully-parametric model.

Recent work has introduced several techniques to speed up kNN-MT. Meng et al. (2021) proposed Fast kNN-MT, which constructs a different datastore for each example, by first searching for the nearest neighbors of the source tokens. Wang et al. (2021) introduced Faster kNN-MT, similar to Fast kNN-MT but with reduced memory requirements. Martins et al. (2022) proposed pruning the datastore, reducing the keys' representation size, and using a cache of retrieval distributions. However, despite leading to some decoding speedups, these methods are limited as they still **retrieve a single token at each time step**.

In this paper, we propose a simple and efficient chunk-based kNN-MT model. Inspired by RETRO (Borgeaud et al., 2021), the chunk-based kNN-MT model retrieves *chunks* of tokens, instead of single tokens. But, similarly to kNN-MT and unlike RETRO, **it does not require any training or fine-tuning of the parametric component**: it simply uses a combination of caching and interpolation

---

[1]The code is available at https://github.com/deep-spin/chunk-based_knn-mt.

of probability distributions to incorporate the retrieved tokens. By doing this, the model leads to a similar translation quality while having to search for neighbors in the datastore less often. This leads to decoding speeds **up to 4 times faster** than the ones achieved using the vanilla kNN-MT model and only twice as slow as a fully-parametric model, but with considerably higher translation quality.

In sum, our main contributions are:

- We introduce a chunk-based kNN-MT model, which retrieves chunks of tokens from a datastore of examples.

- We propose and compare several approaches to deal with the retrieved chunks' tokens and to select the steps in which the model performs retrieval from the datastore.

- We compare the translation quality and decoding efficiency on domain adaptation, which shows the benefits of chunk-based kNN-MT.

- We propose using chunk-based kNN-MT for on-the-fly adaptation.

2 Background

In machine translation, a model is given a sentence or document in a source language, $\boldsymbol{x} = [x_1, \ldots, x_L]$, and the goal is to output a translation in a target language, $\boldsymbol{y} = [y_1, \ldots, y_N]$. This is commonly done using a parametric sequence-to-sequence model (Sutskever et al., 2014; Bahdanau et al., 2015; Vaswani et al., 2017), in which the encoder receives the source sentence as input and outputs a set of hidden states. Then, at each step t, the decoder attends to these hidden states and outputs a probability distribution over the vocabulary, $p_{\text{NMT}}(y_t|\boldsymbol{y}_{<t}, \boldsymbol{x})$. Finally, these probability distributions are used in a search procedure to generate the translation, typically using beam search (Reddy, 1977).

2.1 k-Nearest Neighbor Machine Translation

Khandelwal et al. (2021) introduced a semi-parametric model called k-**nearest neighbor machine translation** (kNN-MT). kNN-MT is composed of a parametric component that outputs a probability distribution over the vocabulary as above, $p_{\text{NMT}}(y_t|\boldsymbol{y}_{<t}, \boldsymbol{x})$, enhanced with an approximate nearest neighbor retrieval mechanism, which allows direct access to a datastore of examples.

The kNN-MT's datastore $\mathcal{D}$ consists of a key-value memory, where each key is the decoder's output representation, $\boldsymbol{f}(\boldsymbol{x}, \boldsymbol{y}_{<t}) \in \mathbb{R}^d$, and the value is the corresponding target token $y_t \in \mathcal{V}$:

$$\mathcal{D} = \{(\boldsymbol{f}(\boldsymbol{x}, \boldsymbol{y}_{<t}), y_t) \,\forall\, t \mid (\boldsymbol{x}, \boldsymbol{y}) \in \mathcal{S}\}, \quad (1)$$

where $\mathcal{S}$ denotes a set of parallel sentences. Then, at inference time, the model searches the datastore to (approximately) retrieve the set of k nearest neighbors $\mathcal{N}$. The retrieval distribution, $p_{k\text{NN}}(y_t|\boldsymbol{y}_{<t}, \boldsymbol{x})$, is computed, using the neighbors' distance to the current decoder's output representation, $d(\boldsymbol{f}(\boldsymbol{x}, \boldsymbol{y}_{<t}), \cdot)$:

$$p_{k\text{NN}}(y_t|\boldsymbol{y}_{<t}, \boldsymbol{x}) = \qquad\qquad (2)$$
$$\frac{\sum_{(\boldsymbol{k}_j, v_j) \in \mathcal{N}} \mathbb{1}_{y_t=v_j} \exp\left(-d\left(\boldsymbol{k}_j, \boldsymbol{f}(\boldsymbol{x}, \boldsymbol{y}_{<t})\right)/T\right)}{\sum_{(\boldsymbol{k}_j, v_j) \in \mathcal{N}} \exp\left(-d\left(\boldsymbol{k}_j, \boldsymbol{f}(\boldsymbol{x}, \boldsymbol{y}_{<t})\right)/T\right)},$$

where T is the softmax temperature, k_j denotes the key of the jth neighbor and v_j its value. Finally, the two distributions, $p_{\text{NMT}}(y_t|\boldsymbol{y}_{<t}, \boldsymbol{x})$ and $p_{k\text{NN}}(y_t|\boldsymbol{y}_{<t}, \boldsymbol{x})$, are combined, by performing interpolation, to obtain the final distribution, which is used to generate the translation through beam search:

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

where $\lambda \in [0, 1]$ is a hyperparameter that controls the weight given to the two distributions.

3 Chunk-based kNN-MT

We now describe our **chunk-based kNN-MT model**, illustrated in Figure 1. We first describe the datastore creation (§3.1) and how we can retrieve chunks of tokens from it (§3.2), describing how they are used by the model (§3.2.2). Finally, we describe how to select the steps in which the model performs retrieval (§3.3).

3.1 Building the datastore

For the model to retrieve a chunk of tokens of size c instead of a single token, we first need to build a datastore $\mathcal{D}$ which also consists of a key-value memory, where each entry key is the decoder's output representation $\boldsymbol{f}(\boldsymbol{x}, \boldsymbol{y}_{<t})$, but the value is now a *chunk* of target tokens $y_{t:(t+c-1)}$:[2]

$$\mathcal{D} = \{(\boldsymbol{f}(\boldsymbol{x}, \boldsymbol{y}_{<t}), y_{t:(t+c-1)}) \,\forall\, t \mid (\boldsymbol{x}, \boldsymbol{y}) \in \mathcal{S}\}.$$
$$(4)$$

---

[2] When the chunk size c is larger than the number of the remaining tokens in the sentence, $N - (t-1)$, we add padding tokens to complete the chunk.
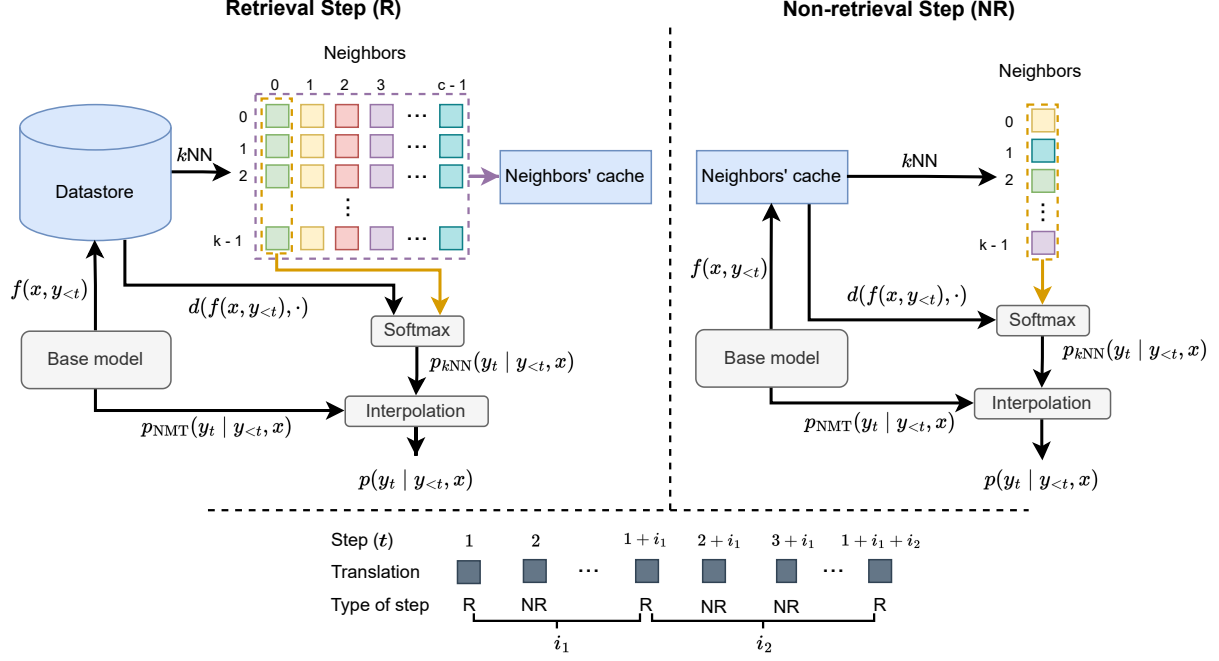
Figure 1: Chunk-based kNN-MT scheme. Top left: model procedure when retrieving neighbors from the datastore. Top right: procedure when not performing retrieval. Bottom: retrieval schedule scheme.

Note that the chunks are sliding windows, *i.e.*, they overlap.

3.2 Retrieving chunks of tokens

At inference time, when performing retrieval, the model searches the datastore and retrieves the set of k nearest neighbors $\mathcal{N}$, for each beam hypothesis and example in the current batch. We now describe several strategies to manipulate the retrieved chunks of tokens during generation.

3.2.1 Maintaining the chunk order

Since in the original sentences used to build the datastore each chunk is composed of an ordered sequence of tokens, the simplest way for the model to use the retrieved tokens is to consider them in the same order. For this, we simply need to compute a retrieval distribution for each token in the chunk, always using the same retrieval distances but aligning the chunk tokens with the corresponding time step, *i.e.* at the retrieval step we consider the first token of each neighbor chunk, at the following step we consider the second token, and so on. We can compute this retrieval distribution as it is done for the kNN-MT, in Eq. 2, just modifying the token indices according to the step. However, by doing this, we are ignoring the remaining tokens in the chunk, which can also contain relevant information for the current prediction. Also, the tokens that

are generated in the previous steps $t, \ldots, t + j - 1$ might not be well "aligned" with the next tokens in the jth position of the chunk for all neighbors.

3.2.2 Neighbors' Cache

To avoid the limitation stated above, we propose using a neighbors' cache instead. We keep the tokens of the retrieved chunks in this cache, so that the model has a higher flexibility about which tokens to select for the current step: it has access to all the tokens present in the retrieved chunks. The neighbors' cache, $\mathcal{M}$, consists of a key-value memory, where a key is the decoder's output representation, $\boldsymbol{f}(\boldsymbol{x}, \boldsymbol{y}_{<t})$, and a value is the corresponding target token y_t, as in the datastore:

$$\mathcal{M} = \{(\boldsymbol{f}(\boldsymbol{x}, \boldsymbol{y}_{<t+i}), y_{t+i}) \ \forall \ 0 \leq i < c \ \forall \ y_t \mid$$
$$y_{t:(t+c-1)} \in \mathcal{N}\}. \tag{5}$$

Note, however, that this cache requires having the decoder state for every token in the chunk, not just for the first one. Therefore, we need to have this information available in the datastore:[3]

$$\mathcal{D} = \{(\boldsymbol{f}(\boldsymbol{x}, \boldsymbol{y}_{<t}), y_{t:(t+c-1)},$$
$$[\boldsymbol{f}(\boldsymbol{x}, \boldsymbol{y}_{<t}), \ldots, \boldsymbol{f}(\boldsymbol{x}, \boldsymbol{y}_{<t+c-1})])$$
$$\forall \ t \mid (\boldsymbol{x}, \boldsymbol{y}) \in \mathcal{S}\}. \tag{6}$$

---

[3]To avoid having the same decoder states in several entries, we just store pointers to an array of decoder states.

Then, as shown in the diagram of Figure 1, at the retrieval steps, the model first searches for the nearest neighbors in the datastore, then builds the neighbors' cache with the tokens of the retrieved chunks, then finally uses the first token of each chunk to compute the current retrieval distribution. In contrast, at the non-retrieval steps, the model searches for the nearest neighbors in the neighbors' cache instead of retrieving from the datastore. Then, to compute the retrieval distribution it also uses the softmax, as in Eq. 2, but with a different softmax temperature, T'. To incorporate the retrieved tokens, it performs interpolation as before, Eq. 3, replacing the hyperparameter λ by λ'.[4]

Considering batch-beam-level neighbors. By building this neighbors' cache, the model can use all the tokens in the retrieved chunks that correspond to each beam hypothesis of the sentence being translated. However, it still ignores the chunks of tokens corresponding to the other beam hypotheses, which are often quite similar, and the other sentences being translated in the same batch, which can contain relevant contextual information if they belong to the same document.

To also leverage these, we propose increasing the number of tokens that the model has access to, by combining the chunks retrieved for the different beam hypotheses and the different examples of the same batch. To do so, we simply need to build a single neighbors' cache for the current batch, to which we feed all the retrieved chunks' tokens.

Considering sentence-level neighbors. To also consider the chunks of tokens retrieved in previous steps of the generation of the current sentences, we propose to keep these in the neighbors' cache, instead of resetting the cache at each retrieval step.

We empirically compare these different proposed approaches in §4.1.3.

3.3 Retrieval Steps Schedule

As the need to perform retrieval slows down decoding considerably, having an efficient retrieval schedule is key to achieve decoding speedups. The simplest scheduling option corresponds to performing retrieval every i steps. However, we noticed empirically that it is beneficial to perform retrieval

---

[4]We use a different temperature and interpolation coefficient at the non-retrieval steps because the number of entries in the neighbors' cache is much smaller than in the datastore, hence the neighbors might be less similar.

steps more frequently at the beginning of the sentence, as we will see in Table 3 of §4.1.4. To leverage this, we introduce the following schedule.

Having $k \in \{1, 2, \ldots\}$ as the retrieval step's index and t_k as the corresponding time step, (*i.e.* t_k is the position of the token generated after the kth retrieval step), we propose using a geometric progression to compute the interval (in tokens) between the kth and $(k+1)^{\text{th}}$ retrieval steps, $i_k = t_{k+1} - t_k$:

$$i_k = \lfloor \min\left(i_{\max}, \; i_{\min} \times 2^{\,r\,t_k}\right) \rfloor, \qquad (7)$$

where $i_{\max}$ and $i_{\min}$ are hyperparameters that define the maximum and minimum interval between retrieval steps, the rate at which the interval increases is defined as $r = \frac{1}{2} i_{\max}/|\boldsymbol{x}|$ where $|\boldsymbol{x}|$ is the source sentence size, and $\lfloor \cdot \rfloor$ denotes the floor function.[5] By using this progression, the frequency with which the model performs retrieval decays along the generation, until the interval between retrieval steps reaches $i_{\max}$. For example, with $i_{\min} = 2$, $i_{\max} = 16$, and $|\boldsymbol{x}| = 20$ the model performs retrieval at steps: $\{1, 3, 7, 20, 36, 52, \ldots\}$. Note that the chunk size, c, is independent of the interval between retrieval steps.

4 Experiments

To understand if the chunk-based kNN-MT is able to maintain the translation quality while speeding-up decoding, we performed experiments on domain adaptation (§4.1) and on-the-fly adaptation (§4.2).

4.1 Domain Adaptation

Dataset and metrics. For domain adaptation, we perform experiments on the Medical, Law, IT, and Koran domain data of the multi-domains dataset introduced by Koehn and Knowles (2017) using the splits redefined by Aharoni and Goldberg (2020). To build the datastores, we use the training sets that have 6,903,141, 19,061,382, 3,602,862, and 524,374 tokens, respectively. The validation and test sets have 2,000 examples for each domain. For evaluation, we use BLEU (Papineni et al., 2002; Post, 2018) and COMET (Rei et al., 2020).

Models. As a baseline, we consider the fully parametric base MT model: the winning system from the WMT'19 German-English news translation task (Ng et al., 2019) (with 269M parameters), available in Fairseq (Ott et al., 2019). We also compare our chunk-based kNN-MT model with other

---

[5]We always consider that the first retrieval step occurs at the first step of the generation (t_0=1).

	BLEU					COMET				
	Medical	Law	IT	Koran	Average	Medical	Law	IT	Koran	Average
Base MT	40.01	45.64	37.91	16.35	34.98	.4702	.5770	.3942	-.0097	.3579
Fine-tuned	50.47	56.56	43.82	21.54	43.10	.5871	.6906	.5856	.0484	.4779
*k*NN-MT	54.47	61.23	45.96	21.02	45.67	.5760	.6781	.5163	.0480	.4546
Efficient *k*NN-MT	51.90	57.82	44.44	20.11	43.57	.5513	.6260	.4909	-.0052	.4158
Chunk-based *k*NN-MT (ours)	53.16	59.65	44.18	19.33	44.08	.5551	.6257	.4854	.0039	.4175

Table 1: BLEU and COMET scores on the multi-domains test set, for a batch size of 8.
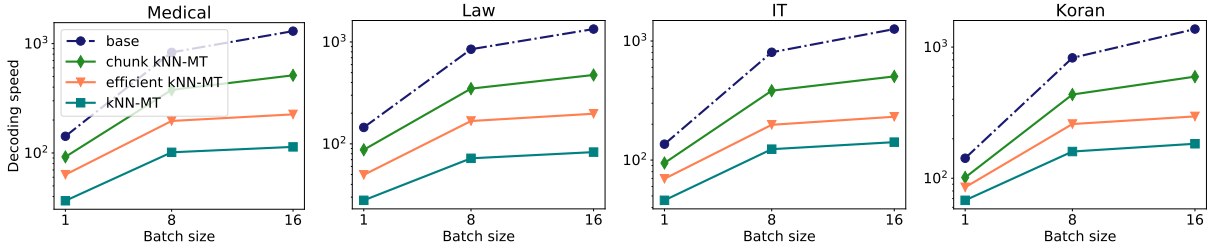

Figure 2: Plots of the decoding speed (tokens per second) for the different models on the medical, law, IT, and Koran domains, for different batch sizes (1,8,16). The generation speed (y-axis) is in log scale.

models that have access to the domain-specific training data: the base model above fine-tuned on the domain-specific datasets, the vanilla *k*NN-MT model (Khandelwal et al., 2021), and the Efficient *k*NN-MT model from Martins et al. (2022).

Settings. For all models that perform retrieval, we retrieve $k = 8$ neighbors and select the hyper-parameter λ for each method and each domain by performing grid search on $\lambda \in \{0.5, 0.6, 0.7, 0.8\}$. For the chunk-based *k*NN-MT, we also perform grid search on $\lambda' \in \{0.4, 0.5, 0.6\}$ and $T' \in \{1, 2, 3\}$. The selected values for the hyperparameters for each model for each domain, validated on the validation set, are stated in Table 10 of App. D. We use the softmax temperatures proposed by Khandelwal et al. (2021) and for the efficient *k*NN-MT, we use the efficiency methods' parameters proposed by Martins et al. (2022). Where not stated otherwise, we use the chunk-based *k*NN-MT with chunks of size $c = 16$, with a sentence-level neighbors' cache, and use the geometric progression heuristic, in Eq. 7, to select the retrieval steps, with $i_{\min} = 2$ and $i_{\max} = 16$, since this is the setting that leads to the best trade-off between translation quality and decoding speed on the validation set. We also follow Martins et al. (2022) and use PCA to reduce the datastore keys' dimension to 256 and the neighbors' cache keys' size to 64. To perform search in the datastore and in the neighbors' cache, we use FAISS (Johnson et al., 2019).

For the fine-tuned model, we perform fine-tuning

for a maximum of 20 epochs on each domain. We perform grid search on the validation set, using different learning rates, $\eta \in \{5 \times 10^{-6}, 1 \times 10^{-5}, 5 \times 10^{-5}, 1 \times 10^{-4}\}$ and two different learning rate schedules (reducing learning rate on plateau and by the inverse square root) with and without warmup during 1 epoch. The selected hyperparameters are stated in Table 11 of App. D.

Computational infrastructure. All experiments were performed on a server with 3 RTX 2080 Ti (11 GB), 12 AMD Ryzen 2920X CPUs (24 cores), and 128 Gb of RAM. For the decoding speed measurements, we ran each model on a single GPU while no other process was running on the server, to have a controlled environment. The nearest neighbor search in the datastore is performed on the CPUs, since not all datastores fit into GPU memory.

4.1.1 Results

The translation scores are reported in Table 1. We can see that the decrease of translation quality when comparing the chunk-based *k*NN-MT model with the vanilla *k*NN-MT model is not substantial in terms of BLEU (-1.5 points on average) and COMET (-.04 points on average). It can also be seen that the chunk-based *k*NN-MT model leads to considerably better translation scores than the base MT model (+9.1 BLEU and +.06 COMET points on average) and to slightly better results than the efficient *k*NN-MT model in terms of BLEU. When comparing fine-tuning the base model with the use

	Medical	Law	IT	Koran	Average
Maintain order	45.17	51.48	40.11	17.87	38.66
Neighbors' Cache					
Single chunk	48.01	53.81	42.09	18.49	40.60
Beam-batch-level	51.77	**58.80**	42.91	19.46	43.24
Sentence-level	**51.86**	58.68	**43.44**	**19.79**	**43.44**

Table 2: BLEU scores on the multi-domains test set, for a batch size of 8, with $c = 6$ and $i = 6$.

	Medical	Law	IT	Koran	Average
$i = 6$	51.86	58.68	43.44	**19.79**	43.44
$i = 8$	51.36	58.28	43.04	19.25	42.98
GP ($i_{\max} = 8$)	**53.38**	**59.87**	**44.41**	19.74	**44.35**
GP ($i_{\max} = 16$)	53.16	59.65	44.18	19.33	44.08
GP ($i_{\max} = 32$)	52.81	58.96	43.51	19.22	43.63

Table 3: BLEU scores on the multi-domains test set, for a batch size of 8. When using the geometric progression heuristic (GP) the average interval with $i_{\max} = 16$ is 5.97 and with $i_{\max} = 32$ is 6.85.

	Medical	Law	IT	Koran	Average
$i = 6$	374	328	398	441	385
$i = 8$	421	374	**429**	470	424
GP ($i_{\max} = 8$)	354	307	358	405	356
GP ($i_{\max} = 16$)	397	368	393	445	401
GP ($i_{\max} = 32$)	**436**	**423**	418	**481**	**440**

Table 4: Decoding speed (tokens per second) on the multi-domains test set, for a batch size of 8, with $c = i_{\max}$.

of semi-parametric models, the results are not conclusive: in terms of BLEU, the semi-parametric models lead to better translations, but according to COMET this is not the case. We present translation examples for the different domains in App. F.

4.1.2 Decoding speed

As can be seen in Figure 2, the chunk-based kNN-MT model leads to a decoding speed up to two times higher than the decoding speed of the efficient kNN-MT model of Martins et al. (2022) and up to four times higher than that of the vanilla kNN-MT model of Khandelwal et al. (2021). The chunk-based kNN-MT model is also able to reduce the decoding speed gap to the base MT model to a factor of two, compared to a factor of four from previous work. Moreover, according to the results on Table 1 this speed-up comes without substantially harming the model's translation quality.

4.1.3 What is the best way to incorporate the retrieved tokens?

To understand which chunk incorporation strategy works best, we perform a comparison using chunks of size $c = 6$ and performed retrieval every 6 steps ($i = 6$). The results reported in Table 2 show that using a neighbors' cache leads to substantially better BLEU scores. We can also see that having a beam-batch-level cache improves the BLEU score and that keeping the tokens from the previous retrieved chunks in the neighbors' cache further improves the translation quality.

4.1.4 When to perform retrieval?

To understand how we should select the retrieval steps, we compare performing retrieval every $i = 6$ or $i = 8$ steps against using the proposed geometric progression (GP), Eq. 7, with $i_{\min} = 2$ and $i_{\max} = 8$, $i_{\max} = 16$, or $i_{\max} = 32$, to compute the interval between retrieval steps. For this comparison, we used the model with a sentence-level neighbors' cache and considered $c = i$ or $c = i_{\max}$ if using the geometric progression. We

report the BLEU scores in Table 3 and the corresponding decoding speeds in Table 4. This comparison shows that performing retrieval steps more frequently at the beginning of the translation, by using the the proposed geometric progression heuristic (GP), leads to better BLEU scores while having a higher decoding speed.

4.2 On-the-fly Adaptation

To understand how the chunk-based kNN-MT model behaves in a realistic scenario where the data arrives in streams, we performed experiments where the model is continuously adapted on the fly.
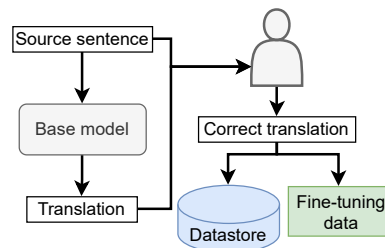

Figure 3: On-the-fly adaptation scheme. After the model translates an example, a human translator (e.g., a post-editor) corrects the generated translation which is added to the fine-tuning data or the domain-specific datastore. We use the references to simulate human translations.

Task description. In this task, we attempt to simulate a real scenario, described in Figure 3, in which we have a model that translates sentences and a human translator (e.g., a post-editor) who
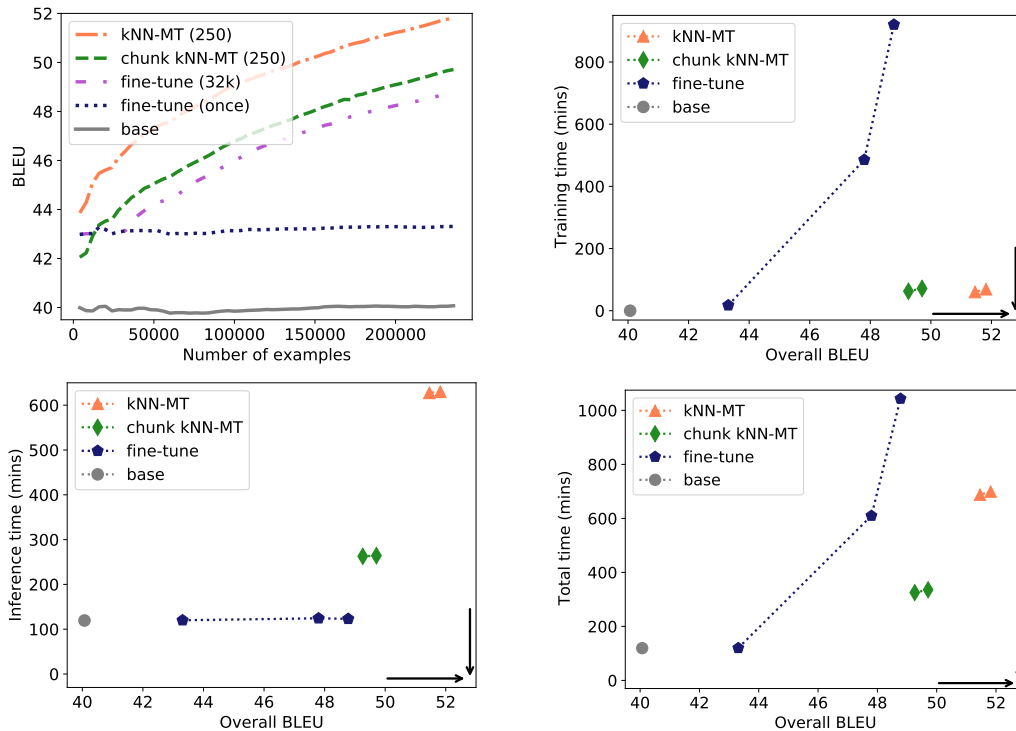
Figure 4: Analysis of the on-the-fly adaptation experiments on the medical domain. Top left: BLEU scores measured every 4,000 examples. Top right: Time (in minutes) spent by each model on training / creating and updating the datastore and the corresponding BLEU score. Bottom left: Inference time (in minutes) spent by each model to translate the whole set of examples and its BLEU score. Bottom right: Total time (training + inference) spent by each model to translate the set of examples and its BLEU score. For the time plots, the lower right corner is better.

corrects the generated translations. Our goal is to understand whether it is better to use the corrected translations to repeatedly fine-tune the base model or to add the corrected translations to the datastore without touching the base model. To do so, we consider that, before starting translation, we have access to 10% of the dataset (we use the training sets of the medical and law domains) and the goal is to translate the remaining 90% examples. To simulate the existence of human translators who correct the generated translations, we consider that, after the model translates a block of sentences, it has access to the corresponding reference sentences.

Models. As baselines, we use the base MT model and the base model fine-tuned on the initially available 10% of data (fine-tune (once)). We also compare fine-tuning the base model on the initially available data and then finetuning after every 32,000 and every 64,000 examples, with all the available data at the time. In all cases, we fine-tuned the model for a maximum of 5 epochs with a learning rate of 5×10^{-5} using the inverse square root scheduler. Concerning the semi-parametric models, we compare the kNN-MT model and the

chunk-based kNN-MT model, building the initial datastore with the initially available data and adding new examples to the datastore after every 250 or 1,000 sentences. We used the same configurations for the chunk-based kNN-MT model, as in §4.1: chunks of size $c = 16$, sentence-level neighbors' cache, and the geometric progression heuristic to select when to perform retrieval, with $i_{\min} = 2$ and $i_{\max} = 16$. We also use the same hyperparameters, stated on Table 10 of App. D.

4.2.1 Results

Figure 4 contains the results of the on-the-fly adaptation experiments on the medical domain. On the top left plot, we see that both the kNN-MT and the chunk-based kNN-MT lead to higher BLEU scores than the fine-tuned models. Also, on the top right, we see that the time needed to add examples to the datastore is much shorter than the time needed to fine-tune the model. This comes at the cost of a higher inference time (bottom left). However, we can see that the chunk-based kNN-MT model is able to substantially reduce the inference time gap to the fully-parametric models. Also, the bottom right plot shows that the chunk-based kNN-MT has

a shorter total time than kNN-MT and the models fine-tuned every 32,000 and 64,000 steps. Concerning the fine-tuned models, there is a BLEU increase when the model is fine-tuned more often. However, this leads to a substantially higher training time. On the other hand, increasing the datastore updates frequency leads to small BLEU improvements and small increases in training time. The results on the law domain show similar results (App. E).

5 Related Work

Semi-parametric models. Semi-parametric models, which augment a parametric model with a retrieval component, have been shown to be effective on several text generation tasks. For language modeling, Khandelwal et al. (2019) proposed the k-nearest neighbor language (kNN-LM), in which a language model is augmented with token-based retrieval and uses probability interpolation to incorporate these tokens. Yogatama et al. (2021) proposed to integrate the retrieved tokens with a gating mechanism. Borgeaud et al. (2021) proposed to retrieve chunks of tokens and incorporate them with cross-attention, using datastores with trillions of tokens. To increase the kNN-LM's decoding speed, He et al. (2021) proposed a range of techniques, such as datastore pruning, dimension reduction, and adaptive retrieval. Alon et al. (2022) proposed adding pointers to the next token on the original corpus to the datastore entries, so that the model can consider the pointed entries instead of performing retrieval. Similarly to our approach, this saves retrieval steps by leveraging the original corpus sequences, but in our case, we do not limit the candidate tokens to be the following ones and consider the succeeding tokens even if the model has not generated the same prefix token(s).

For machine translation, Gu et al. (2018) introduced a semi-parametric model which uses an out-of-the-box search engine to retrieve similar sentence pairs, and incorporate them with shallow and deep fusion. Zhang et al. (2018) proposed to retrieve n-grams and use them to up-weight token probabilities. Bapna and Firat (2019a) proposed to retrieve sentences similar to the source's n-grams, and incorporate them with attention. More recently, Khandelwal et al. (2021) proposed the kNN-MT model which Zheng et al. (2021a) extended with a network that determines the number of retrieved tokens to consider and Zheng et al. (2021b) pro-

posed building the datastore using monolingual sentences. As kNN-MT can be up to two orders of magnitude slower than a fully-parametric model, methods that improve its efficiency have been proposed. Meng et al. (2021) and Wang et al. (2021) proposed the Fast and Faster kNN-MT, in which the model has a higher decoding speed, by creating a different datastore, based on the source sentence, for each example. Martins et al. (2022) propose efficient kNN-MT, which we use as baseline (§4.1), by adapting the methods introduced by He et al. (2021) to machine translation and introducing a retrieval distributions cache to speed-up decoding. In this paper, we show that the chunk-based kNN-MT model can further speed-up decoding, by retrieving chunks of tokens instead of a single token.

Semi-parametric models have also been applied to other tasks as question answering (Lewis et al., 2020; Izacard and Grave, 2021a,b) and dialogue generation (Weston et al., 2014; Fan et al., 2021).

Domain adaptation for machine translation. Domain adaptation consists of adapting generic models to domain-specific data. The most common method for domain adaptation in machine translation is fine-tuning the model on each domain, but this can be expensive and often leads to catastrophic forgetting (Saunders, 2021). To simplify this, some work has proposed fine-tuning only part of the model (Wuebker et al., 2018; Bapna and Firat, 2019b; Lin et al., 2021; Liang et al., 2021). Farajian et al. (2017) performed on-the-fly adaptation, by fine-tuning the model on a set of retrieved examples for each source sentence. However, this still requires fine-tuning model parameters.

Several works have introduced domain adaptation methods without the need to fine-tune the model. Eidelman et al. (2012), Hasler et al. (2014), and Su et al. (2015) proposed using topic models while Bertoldi et al. (2014) proposed leveraging post-editing information. More recently, Khandelwal et al. (2021) proposed using semi-parametric models which retrieve from domain-specific datastores. The aim of our chunk-based kNN-MT method is to speed up kNN-MT's decoding, while maintaining its high translation quality.

6 Conclusions

In this paper, we propose a chunk-based kNN-MT model, which retrieves chunks of tokens from a datastore, instead of a single token. To do so, we proposed several alternatives to explore the re-

trieved chunks' tokens: keeping the original order or building a neighbors' cache. We also analyzed two approaches to select the retrieval steps: every i steps or using a geometric progression heuristic to define the interval between retrieval steps. Through experiments on domain adaptation, we showed that chunk-based kNN-MT leads to a considerable speed-up without substantially compromising the translation quality. Experiments on on-the-fly adaptation showed that chunk-based kNN-MT leads to high quality translations while being more efficient than previously proposed methods.

Limitations

The scope of this paper is limited to the usage of small to medium size datastores, due to the memory requirements needed for big size datastores, for which the proposed model could be even more beneficial. Additionally, we use the decoding speed (tokens per second), training time (in minutes), and inference time (in minutes) to compare the efficiency of the different models. However, these metrics depend on the computational infrastructure used, and, consequently, the speed-up gains can vary when using different hardware.

Acknowledgments

References

Roee Aharoni and Yoav Goldberg. 2020. Unsupervised domain clusters in pretrained language models. In *Proc. ACL*.

Uri Alon, Frank F Xu, Junxian He, Sudipta Sengupta, Dan Roth, and Graham Neubig. 2022. Neuro-Symbolic Language Modeling with Automaton-augmented Retrieval. In *Proc. ICML*.

Dzmitry Bahdanau, Kyung Hyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. In *Proc. ICLR*.

Ankur Bapna and Orhan Firat. 2019a. Non-Parametric Adaptation for Neural Machine Translation. In *Proc. NAACL*.

Ankur Bapna and Orhan Firat. 2019b. Simple, Scalable Adaptation for Neural Machine Translation. In *Proc. EMNLP-IJCNLP*.

Nicola Bertoldi, Patrick Simianer, Mauro Cettolo, Katharina Wäschle, Marcello Federico, and Stefan Riezler. 2014. Online adaptation to post-edits for phrase-based statistical machine translation. *Machine Translation*.

Sebastian Borgeaud, Arthur Mensch, Jordan Hoffmann, Trevor Cai, Eliza Rutherford, Katie Millican, George van den Driessche, Jean-Baptiste Lespiau, Bogdan Damoc, Aidan Clark, et al. 2021. Improving language models by retrieving from trillions of tokens. *arXiv preprint*.

Vladimir Eidelman, Jordan Boyd-Graber, and Philip Resnik. 2012. Topic models for dynamic translation model adaptation. In *Proc. ACL*.

Angela Fan, Claire Gardent, Chloé Braud, and Antoine Bordes. 2021. Augmenting Transformers with KNN-Based Composite Memory for Dialog. *Transactions of the Association for Computational Linguistics*.

M Amin Farajian, Marco Turchi, Matteo Negri, and Marcello Federico. 2017. Multi-domain neural machine translation through unsupervised adaptation. In *Proc. Second Conference on Machine Translation*.

Jiatao Gu, Yong Wang, Kyunghyun Cho, and Victor OK Li. 2018. Search engine guided neural machine translation. In *Proc. AAAI*.

Eva Hasler, Phil Blunsom, Philipp Koehn, and Barry Haddow. 2014. Dynamic topic adaptation for phrase-based mt. In *Proc. EACL*.

Junxian He, Graham Neubig, and Taylor Berg-Kirkpatrick. 2021. Efficient Nearest Neighbor Language Models. In *Proc. EMNLP*.

Gautier Izacard and Edouard Grave. 2021a. Leveraging Passage Retrieval with Generative Models for Open Domain Question Answering. In *Proc. EACL*.

Gautier Izacard and Edouard Grave. 2021b. Distilling Knowledge from Reader to Retriever for Question Answering. In *Proc. ICLR*.

Qingnan Jiang, Mingxuan Wang, Jun Cao, Shanbo Cheng, Shujian Huang, and Lei Li. 2021. Learning Kernel-Smoothed Machine Translation with Retrieved Examples. In *Proc. 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 *Proc. ICLR*.

Urvashi Khandelwal, Omer Levy, Dan Jurafsky, Luke Zettlemoyer, and Mike Lewis. 2019. Generalization through Memorization: Nearest Neighbor Language Models. In *Proc. ICLR*.

Philipp Koehn and Rebecca Knowles. 2017. Six Challenges for Neural Machine Translation. In *Proceedings of the First Workshop on Neural Machine Translation*.

Patrick Lewis, Ethan Perez, Aleksandara Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, et al. 2020. Retrieval-augmented generation for knowledge-intensive nlp tasks. In *Proc. NeurIPS*.

Jianze Liang, Chengqi Zhao, Mingxuan Wang, Xipeng Qiu, and Lei Li. 2021. Finding Sparse Structures for Domain Specific Neural Machine Translation. In *Proc. AAAI*.

Zehui Lin, Liwei Wu, Mingxuan Wang, and Lei Li. 2021. Learning Language Specific Sub-network for Multilingual Machine Translation. In *Proc. ACL*.

Pedro Henrique Martins, Zita Marinho, and André F. T. Martins. 2022. Efficient Machine Translation Domain Adaptation. In *Proc. ACL 2022 Workshop on Semiparametric Methods in NLP: Decoupling Logic from Knowledge*.

Yuxian Meng, Xiaoya Li, Xiayu Zheng, Fei Wu, Xiaofei Sun, Tianwei Zhang, and Jiwei Li. 2021. Fast Nearest Neighbor Machine Translation.

Nathan Ng, Kyra Yee, Alexei Baevski, Myle Ott, Michael Auli, and Sergey Edunov. 2019. Facebook FAIR's WMT19 News Translation Task Submission. In *Proc. of the Fourth Conference on Machine Translation*.

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 *Proc. NAACL (Demonstrations)*.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proc. ACL*.

Matt Post. 2018. A Call for Clarity in Reporting BLEU Scores. In *Proc. Third Conference on Machine Translation*.

Raj Reddy. 1977. Speech understanding systems: summary of results of the five-year research effort at Carnegie-Mellon University.

Ricardo Rei, Craig Stewart, Ana C Farinha, and Alon Lavie. 2020. COMET: A Neural Framework for MT Evaluation. In *Proc. EMNLP*.

Danielle Saunders. 2021. Domain Adaptation and Multi-Domain Adaptation for Neural Machine Translation: A Survey.

Jinsong Su, Deyi Xiong, Yang Liu, Xianpei Han, Hongyu Lin, Junfeng Yao, and Min Zhang. 2015. A context-aware topic model for statistical machine translation. In *Proc. ACL*.

Ilya Sutskever, Oriol Vinyals, and Quoc V Le. 2014. Sequence to sequence learning with neural networks. In *Proc. NeurIPS*.

Yuqing Tang, Chau Tran, Xian Li, Peng-Jen Chen, Naman Goyal, Vishrav Chaudhary, Jiatao Gu, and Angela Fan. 2020. Multilingual Translation with Extensible Multilingual Pretraining and Finetuning.

Jörg Tiedemann. 2012. Parallel Data, Tools and Interfaces in OPUS. In *Proc. LREC*.

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 *Proc. NeurIPS*.

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*.

Jason Weston, Sumit Chopra, and Antoine Bordes. 2014. Memory networks.

Joern Wuebker, Patrick Simianer, and John DeNero. 2018. Compact Personalized Models for Neural Machine Translation. In *Proc. EMNLP*.

Dani Yogatama, Cyprien de Masson d'Autume, and Lingpeng Kong. 2021. Adaptive Semiparametric Language Models. *Transactions of the Association for Computational Linguistics*, 9:362–373.

Jingyi Zhang, Masao Utiyama, Eiichiro Sumita, Graham Neubig, and Satoshi Nakamura. 2018. Guiding Neural Machine Translation with Retrieved Translation Pieces. In *Proc. NAACL*.

Xin Zheng, Zhirui Zhang, Junliang Guo, Shujian Huang, Boxing Chen, Weihua Luo, and Jiajun Chen. 2021a. Adaptive Nearest Neighbor Machine Translation.

Xin Zheng, Zhirui Zhang, Shujian Huang, Boxing Chen, Jun Xie, Weihua Luo, and Jiajun Chen. 2021b. Non-Parametric Unsupervised Domain Adaptation for Neural Machine Translation. In *Proc. EMNLP Findings*.

A   Varying the Chunk Size along the Generation

When using the geometric progression to compute the interval between retrieval steps (§3.3), the model performs retrieval more frequently at the beginning of the generation of the translation. Because of this, we compare having a fixed chunk size equal to the maximum interval between retrieval steps ($c = i_{\max}$) with having the chunk size vary along the generation ($c_k = i_k$). For this comparison, we compute the retrieval steps using the geometric progression with $i_{\min} = 2$ and $i_{\max} = 16$, and use a sentence-level cache.

	Medical	Law	IT	Koran	Average
$c = i_{\max}$	**53.16**	**59.65**	**44.18**	**19.33**	**44.08**
$c_k = i_k$	52.70	59.40	43.96	19.10	43.79

Table 5: BLEU scores on the multi-domains test set, for a batch size of 8.

The results, in Table 5, indicate that keeping the chunk size fixed leads to slightly better translation quality.

B   Using Different Values for k.

In order to understand how the number of retrieved chunks (k) affects the translation quality and the decoding speed, we compare using different values of k. For this comparison, we compute the retrieval steps using the geometric progression with $i_{\min} = 2$ and $i_{\max} = 16$, and use a sentence-level cache.

	Medical	Law	IT	Koran	Average
$k = 2$	50.55	56.30	41.42	18.32	41.65
$k = 4$	51.46	58.28	43.55	19.07	43.09
$k = 8$	53.16	59.65	44.18	19.33	44.08
$k = 16$	**53.75**	**60.64**	**45.37**	**19.99**	**44.94**

Table 6: BLEU scores on the multi-domains test set, for a batch size of 8.

	Medical	Law	IT	Koran	Average
$k = 2$	**452**	**392**	**452**	**494**	**448**
$k = 4$	433	386	428	486	433
$k = 8$	397	368	393	445	401
$k = 16$	343	323	329	380	344

Table 7: Decoding speed (tokens per second) on the multi-domains test set, for a batch size of 8.

We report the BLEU score and decoding speed for different values of k in Tables 6 and 7, respec-

tively. These results show that there is a trade-off between the translation quality and the decoding speed, when varying the number of retrieved neighbors (k).

C   Experiments on ES-FR and ET-IT.

To understand if the proposed model, chunk-based kNN-MT, performs well on language pairs and datasets other than the ones used in the main experiments (§4.1.1), we perform experiments on Spanish-French (es-fr) and Estonian-Italian (et-it) on two datasets: EMEA and JRC-acquis (Tiedemann, 2012). For this experiment, we used the multilingual model mBART50 (Tang et al., 2020), compute the retrieval steps using the geometric progression with $i_{\min} = 2$ and $i_{\max} = 16$, and use a sentence-level cache.

	EMEA		JRC	
	es-fr	et-it	es-fr	et-it
Base MT	5.02	16.44	19.27	17.94
kNN-MT	36.65	38.79	45.87	39.88
Chunk-based kNN-MT	25.69	32.80	29.17	28.16

Table 8: BLEU scores on the EMEA and JRC test sets, for a batch size of 8.

	EMEA		JRC	
	es-fr	et-it	es-fr	et-it
Base MT	189	214	189	205
kNN-MT	37	37	26	27
Chunk-based kNN-MT	98	96	95	101

Table 9: Decoding speed (tokens per second) on the EMEA and JRC test sets, for a batch size of 8.

We report the BLEU scores and the decoding speeds (in tokens per second) on Tables 8 and 9, respectively. As can be seen, the chunk-based kNN-MT model is able to improve the translation quality considerably, when comparing with the base MT model, while leading to a decoding speed around 3 times faster than the vanilla kNN-MT model.

D   Hyperparameters

On Table 10 we report the values for the hyperparameters of the semi-parametric models: the interpolation coefficients $\lambda \in \{0.5, 0.6, 0.7, 0.8\}$ and $\lambda' \in \{0.4, 0.5, 0.6\}$, and the retrieval softmax temperatures T and $T' \in \{1, 2, 3\}$. For decoding we use beam search with a beam size of 5.

On Table 11 we report the values of the hyperparameters used to fine-tune the base model on each

domain: learning rate, learning rate scheduler, and whether warmup steps were used.

E   On-the-fly Adaptation on Law Domain

We report the results of the on-the-fly adaptation experiment, on the law domain, on Figure 5. In a similar way as in the medical domain, the top left plot shows that the kNN-MT and the chunk-based kNN-MT models lead to higher BLEU scores than the fine-tuned models. We can also see, on the top right plot, that the time the models take to add examples to the datastore along the generation is much shorter than the time needed to fine-tune the model. This comes at the cost of a higher inference time (as shown on the bottom left plot). However, we can see that the chunk-based kNN-MT model is able to substantially reduce the inference time gap between fully-parametric and semi-parametric models, having a shorter total time than the kNN-MT and the models fine-tuned every 32,000 and 64,000 steps (bottom right plot). Concerning the fine-tuned models, fine-tuning more often leads to a slightly better BLEU score. However, this also leads to a substantially higher training time.

F   Translation Examples

We report some translation examples on the medical domain in Figures 6 and 7, on the law domain in Figure 8, and on the IT domain in Figure 9. To simplify the examples, we use a batch size of 1 and a beam size of 1.

	Medical				Law				IT				Koran			
	λ	T	λ'	T'	λ	T	λ'	T'	λ	T	λ'	T'	λ	T	λ'	T'
kNN-MT	0.7	10	—	—	0.8	10	—	—	0.7	10	—	—	0.6	100	—	—
Efficient kNN-MT	0.7	10	—	—	0.8	10	—	—	0.7	10	—	—	0.7	100	—	—
Chunk-based kNN-MT	0.7	10	0.5	1	0.8	10	0.5	1	0.7	10	0.4	2	0.8	100	0.4	3
Ablations																
c=6 , i=6																
Maintain order	0.8	10	0.4	—	0.7	10	0.4	—	0.5	10	0.4	—	0.7	100	0.4	—
Single chunk	0.8	10	0.4	2	0.7	10	0.4	2	0.7	10	0.4	2	0.7	100	0.4	3
All chunks	0.7	10	0.5	1	0.8	10	0.5	1	0.7	10	0.4	2	0.6	100	0.4	1
Keep previous	0.7	10	0.5	1	0.8	10	0.5	1	0.7	10	0.4	1	0.7	100	0.4	3
Keep previous																
$i=6$	0.7	10	0.5	1	0.8	10	0.5	1	0.7	10	0.4	1	0.7	100	0.4	3
$i=8$	0.7	10	0.4	1	0.8	10	0.5	1	0.7	10	0.4	1	0.7	100	0.4	3
$\exp(i_{\max}=16)$	0.7	10	0.5	1	0.8	10	0.5	1	0.7	10	0.4	2	0.8	100	0.4	3
$\exp(i_{\max}=32)$	0.8	10	0.5	1	0.6	10	0.5	1	0.6	10	0.4	1	0.5	100	0.4	1

Table 10: Values of the hyperparameters: number of neighbors to be retrieved k, interpolation coefficient λ, and retrieval softmax temperature T.

	Medical				Law		
η	schedule	warmup		η	schedule	warmup	
1×10^{-5}	on plateau	no		5×10^{-5}	inverse sqrt	yes	
	IT				Koran		
η	schedule	warmup		η	schedule	warmup	
5×10^{-5}	inverse sqrt	yes		5×10^{-5}	inverse sqrt	no	

Table 11: Values of the hyperparameters for the fine-tuned model.
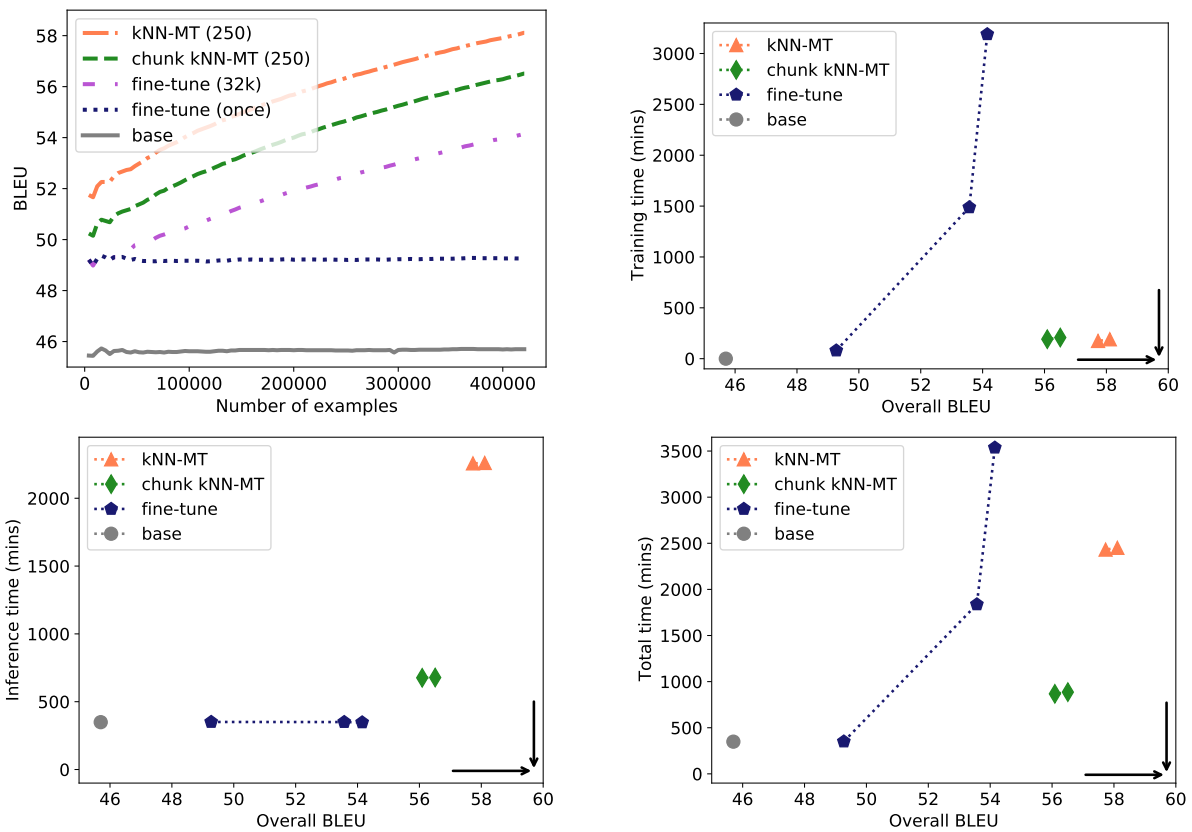
Figure 5: Analysis of the on-the-fly adaptation experiments on the law domain. Top left: BLEU scores measured every 4,000 examples. Top right: Time (in minutes) spent by each model on training / creating and updating the datastore and the corresponding BLEU score. Bottom left: Inference time (in minutes) spent by each model to translate the whole set of examples and its BLEU score. Bottom right: Total time (training + inference) spent by each model to translate the set of examples and its BLEU score. The lower right corner is better.

Source: - Informieren Sie Ihren Arzt, wenn sich bei Ihnen ein Hautausschlag entwickelt.

Target: - Tell your doctor if you develop a rash.

Neighbors' cache

['-', '**T@@**', '**ell**', '**your**', '**doctor**', '**if**', '**you**', 'get', 'any', 'of', 'these', 'symptoms', '**.**', '</s>']
['-', '**T@@**', '**ell**', '**your**', '**doctor**', '**if**', '**you**', 'have', 'been', 'treated', 'with', 'medic@@', 'ines', 'that', 'may', 'reduce']
['-', '**T@@**', '**ell**', '**your**', 'do@@', 'ct', 'or', '**if**', '**you**', 'have', 'ha@@', 'em@@', 'ophi@@', 'lia', '**.**', '</s>']
['Some', 'patients', 'taking', '</s>']
['-', '**T@@**', '**ell**', '**your**', '**doctor**', '**if**', '**you**', 'have', 'diabetes', '**.**', '</s>']
['**T@@**', '**ell**', '**your**', '**doctor**', 'or', 'pharmac@@', 'ist', '**.**', '</s>']
['A', 'gain', ',', 'loss', 'or', 're@@', 'distribution', 'of', '</s>']
['-', '**T@@**', '**ell**', '**your**', '**doctor**', '**if**', '**you**', 'have', 'a', 'heart', 'pac@@', 'em@@', 'aker', 'or', 'if', 'there']

['**ell**', 'the', '**doctor**', '**if**', 'your', 'child', 'has', 'recently', 'received', 'a', 'vaccine', 'or', '**if**', 'one', 'is', 'scheduled']
['**ell**', '**your**', '**doctor**', '**if**', '**you**', 'are', 'taking', 'anti', '@-@', 'HIV', 'therapy', '**.**', '</s>']
['**ell**', '**your**', '**doctor**', '**if**', '**you**', 'are', 'di@@', 'abe@@', 'tic', '**.**', '</s>']
['**ell**', '**your**', '**doctor**', '**if**', 'the', 'child', 'has', 'he@@', 'pati@@', 'tis', 'C', '**.**', '</s>']
['**ell**', '**your**', '**doctor**', '**if**', '**you**', 'are', 'breast', '@-@', 'feeding', 'or', 'about', 'to', 'start', 'breast', '@-@', 'feeding']
['**ell**', '**your**', '**doctor**', '**if**', '**you**', 'have', 'he@@', 'pati@@', 'tis', 'C', '**.**', '</s>']
['**ell**', '**your**', '**doctor**', 'or', 'pharmac@@', 'ist', '**if**', '**you**', 'are', 'pregnant', 'or', 'think', 'you', 'may', 'be', 'pregnant']
['**ell**', '**your**', '**doctor**', '**if**', '**you**', 'are', 'breast', '@-@', 'feeding', '**.**', '</s>']

['**develop**', '**a**', 'ra@@', '**sh**', '**.**', '</s>']
['experience', '**ra@@**', '**sh**', 'and', 'any', 'of', 'the', 'other', 'side', 'effects', 'of', 'a', 'hyper@@', 'sensi@@', 'tivity', '(']
['**develop**', 'any', 'of', 'these', 'symptoms', 'you', 'should', 'seek', 'medical', 'attention', 'immediately', '**.**', '</s>']
['get', 'any', 'of', 'these', '**.**', '</s>']
['get', 'a', 'skin', '**ra@@**', '**sh**', '**.**', '</s>']
['get', 'these', 'symptoms', '**.**', '</s>']
['get', 'any', 'of', 'these', 'symptoms', '**.**', '</s>']
['develop', 'lymp@@', 'homa', 'or', 'other', 'canc@@', 'ers', 'while', 'you', 'are', 'taking', 'Re@@', 'mic@@', 'ade', '**.**', '</s>']

Translation

-	T@@	ell	your	doctor	if	you	develop	a	ra@@	sh	.

Figure 6: Example of translation from the medical domain. The tokens generated at the retrieval steps are highlighted in the same color as the retrieved chunks of tokens. These chunks of tokens are added to the neighbors' cache, after being retrieved. The retrieved tokens which are present in the translation are bolded.

Source: Wenn Sie glauben, dass Sie eine Dosis vergessen haben, informieren Sie sofort Ihren Arzt.

Target: If you think that you have missed a dose, tell your doctor straight away.

Neighbors' cache

['**If**', '**you**', 'forget', '**a**', '**dose**', ',', '**talk**', '**to**', '**your**', '**doctor**', 'or', 'pharmac@@', 'ist', 'as', 'soon', 'as']
['In', 'case', '**you**', 'in@@', 'ject', 'an', 'incor@@', 'rec@@', 't', '**dose**', 'or', '**you**', 'miss', 'an', 'in@@', 'jection']
['24', '**If**', '**you**', 'take', 'more', 'N@@', 'ex@@', 'av@@', 'ar', 'than', '**you**', 'should', 'T@@', 'ell', '**your**', '**doctor**']
['**If**', '**you**', '**have**', 'acci@@', 'dentally', 'taken', 'more', 'than', 'the', 'prescri@@', 'bed', '**dose**', ',', '**you**', 'should', 'contact']
['**If**', '**you**', '**have**', 'forgotten', '**a**', '**dose**', 'of', 'Fil@@', 'gra@@', 'st@@', 'im', 'HE@@', 'X@@', 'AL', ',', '**you**']
['**If**', '**you**', 'are', 'concerned', 'that', '**you**', 'may', '**have**', '**missed**', '**a**', '**dose**', ',', 'contact', '**your**', '**doctor**', 'or']
['**If**', '**you**', '**have**', 'liver', 'or', 'ki@@', 'dney', 'problems', ',', '**talk**', '**to**', '**your**', '**doctor**', ',', 'since', '**your**']
['**If**', '**you**', '**have**', 'liver', 'or', 'ki@@', 'dney', 'problems', ',', '**talk**', '**to**', '**your**', '**doctor**', ',', 'since', '**your**']

['**think**', '**you**', 'might', '**have**', 'any', 'of', 'these', ',', '**talk**', '**to**', '**your**', '**doctor**', '**immediately**', '.', '</s>']
['**think**', '**you**', 'may', 'be', 'aller@@', 'gic', 'to', 'vil@@', 'd@@', 'ag@@', 'lip@@', 'tin', 'or', 'any', 'of', 'the']
['**think**', '**you**', 'may', 'be', 'aller@@', 'gic', 'to', 'any', 'of', 'these', ',', '**talk**', '**to**', '**your**', '**doctor**', 'before']
['**think**', 'that', '**a**', '**dose**', 'has', 'been', 'forgotten', '.', '</s>']
['**think**', '**you**', 'are', 'having', 'any', 'of', 'these', 'types', 'of', 'reaction', ',', 'stop', 'taking', 'this', 'medicine', 'and']
['**think**', '**you**', 'may', '**have**', 'been', 'given', 'too', 'much', 'medicine', ',', 'tell', '**your**', '**doctor**', 'straight', 'away', '**.**']
['**think**', '**you**', 'are', 'pregnant', '.', '</s>']
['**think**', '**you**', 'are', 'experiencing', 'any', 'of', 'these', 'side', 'effects', '.', '</s>']

['**dose**', ',', 'ad@@', 'minister', 'it', 'as', 'soon', 'as', 'possible', '.', '</s>']
['**dose**', ',', 'contact', '**your**', '**doctor**', 'or', 'pharmac@@', 'ist', '.', '</s>']
['**dose**', ',', '**talk**', '**to**', '**your**', '**doctor**', 'or', 'pharmac@@', 'ist', 'as', 'soon', 'as', 'possible', '.', '</s>']
['**dose**', ',', 'just', 'carry', 'on', 'with', 'the', 'next', '**dose**', 'as', 'normal', '.', '</s>']
['**dose**', ',', 'take', 'it', 'as', 'soon', 'as', '**you**', 'remember', '.', '</s>']
['**dose**', ',', 'just', 'resume', 'your', 'usual', 'schedule', 'the', 'following', 'day', '.', '</s>']
['**dose**', 'of', 'M@@', 'IR@@', 'C@@', 'ER@@', 'A', 'ad@@', 'minister', 'the', 'missed', '**dose**', 'as', 'soon', 'as', '**you**']
['**dose**', ',', 'take', 'it', 'as', 'soon', 'as', '**you**', 'remember', '.', '</s>']

Translation

| If | you | think | you | have | missed | a | dose | , | talk | to | your | doctor | immediately | . |

Figure 7: Example of translation from the medical domain. The tokens generated at the retrieval steps are highlighted in the same color as the retrieved chunks of tokens. These chunks of tokens are added to the neighbors' cache, after being retrieved. The retrieved tokens which are present in the translation are bolded.

Source: (8) Zahlt ein Mitglied seinen Beitrag zum Verwaltungshaushalt nicht binnen sechs Monaten nach Beginn des

Target: 8. If a member does not pay its contribution to the administrative budget in full within the six months

Neighbors' cache

['**8**', '**.**', 'Where', 'the', 'last', 'day', 'of', 'the', 'period', 'is', 'not', 'a', 'working', 'day', ',', '**the**']
['**8**', '**.**', 'In', 'the', 'case', '**of**', 'ir@@', 'regul@@', 'arities', 'affecting', 'at', 'least', '5', '%', '**of**', '**the**']
['**8**', '**.**', 'In', 'the', 'case', '**of**', 'ir@@', 'regul@@', 'arities', 'affecting', 'at', 'least', '5', '%', '**of**', '**the**']
['**8**', '**.**', 'A', 'new', 'ent@@', 'rant', 'which', 'has', 'been', 'offered', 'sl@@', 'ots', '**within**', 'two', 'hours', 'before']
['**8**', '**.**', '**If**', 'any', 'of', 'the', 'def@@', 'ici@@', 'en@@', 'cies', 'referred', 'to', 'in', 'par@@', 'ag@@', 'rap@@']
['(', '**8**', ')', 'Where', 'several', 'representatives', 'are', 'appointed', 'by', 'the', 'same', 'party', ',', 'they', 'may', ',']
['**8**', '**.**', 'The', 'office', 'referred', 'to', 'in', 'par@@', 'ag@@', 'rap@@', 'hs', '4', 'and', '5', 'shall', 'keep']
['**8**', '**.**', 'The', 'date', '**of**', 'issue', '**of**', '**the**', 'movement', 'certific@@', 'ate', 'must', 'be', 'indicated', 'in', '**the**']

['**If**', 'on', '1', 'June', '**a**', 'Member', 'State', 'has', 'not', 'submitted', 'its', 'annual', 'invent@@', 'ory', 'report', '**to**']
['**If**', '**a**', '**member**', '**of**', '**the**', 'F@@', 'UND', 'fails', '**to**', 'ful@@', 'fi@@', 'l', 'any', '**of**', 'its', 'oblig@@']
['**If**', '**a**', '**member**', '**has**', '**not**', '**paid**', '**its**', 'full', '**contribution**', '**to**', '**the**', '**administrative**', '**budget**', '**within**', 'four', '**months**']
['**If**', 'actual', 'recovery', '**has**', '**not**', 'taken', 'place', 'by', '**the**', 'due', 'date', 'sti@@', 'pul@@', 'ated', 'in', '**the**']
['**If**', 'no', 'products', 'have', 'been', 'put', 'into', 'free', 'circu@@', 'lation', 'during', '**the**', 'course', '**of**', 'one', '**of**']
['Where', '**the**', 'time', 'limit', 'set', 'by', '**the**', 'customs', 'office', 'pursu@@', 'ant', '**to**', 'par@@', 'ag@@', 'rap@@', 'h']
['Where', '**a**', 'con@@', 'sign@@', 'ment', '**has**', '**not**', 'been', 'presented', 'at', '**the**', 'office', '**of**', 'destination', 'and', '**the**']
['**If**', '**a**', '**member**', 'decl@@', 'ares', 'himself', 'un@@', 'available', 'for', '**a**', 'stated', 'period', ',', 'the', 'Chairman', 'may']

['**paid**', '**its**', 'full', '**contribution**', '**to**', '**the**', '**administrative**', '**budget**', '**within**', 'four', '**months**', 'after', 'such', '**contribution**', 'becomes', 'due']
['**paid**', '**its**', '**contribution**', '**within**', 'two', '**months**', 'after', 'such', 'request', ',', 'that', '**member**', 'shall', 'be', 'requested', '**to**']
['**paid**', '**its**', '**contribution**', ',', 'its', 'voting', 'rights', 'shall', 'be', 'suspended', 'and', 'an', 'interest', 'charge', 'shall', 'be']
['**paid**', '**the**', 'purchase', 'price', '**to**', '**the**', 'producer', '**within**', '**the**', 'deadline', 'laid', 'down', 'in', 'Article', '65', '(']
['**paid**', 'for', '**the**', 'cer@@', 'e@@', 'als', '**within**', '**the**', 'period', 'laid', 'down', 'in', '**the**', 'first', 'par@@', 'ag@@']
['**paid**', '**the**', 'producer', '**the**', 'purchase', 'price', 'referred', '**to**', 'in', 'par@@', 'ag@@', 'rap@@', 'h', '6', '**within**', '**the**']
['made', '**the**', 'payment', 'referred', '**to**', 'in', 'par@@', 'ag@@', 'rap@@', 'h', '2', '**within**', '**the**', 'speci@@', 'fied', 'period']
['**paid**', '**the**', 'producer', '**the**', 'purchase', 'price', 'referred', '**to**', 'in', 'par@@', 'ag@@', 'rap@@', 'h', '7', 'of', 'this']

['**year**', '**.**', 'The', 'part', 'of', 'the', 'balance', 'exce@@', 'eding', 'the', 'amount', 'of', 'the', 'Community', 'subsi@@', 'dy']
['quanti@@', 'ties', 'with@@', 'drawn', ':', '</s>']
['storage', 'contract', 'or', 'the', 'final', 'day', 'for', 'sub@@', 'mission', 'of', 'ten@@', 'ders', 'in', 'response', 'to', 'an']
['Commission', 'shall', ':', '</s>']
['period', 'referred', 'to', 'in', 'Article', '10', '(', '1', ')', 'of', 'the', 'Act', 'of', '</s>']
['entry', 'into', 'force', '</s>']
['compet@@', 'ent', 'official', 'body', 'concerned', ',', 'or', '</s>']
['final', 'day', 'for', 'sub@@', 'mission', 'of', 'ten@@', 'ders', 'in', 'response', 'to', 'an', '</s>']

Translation

| 8 | . | If | a | member | has | not | paid | its | contribution | to | the | administrative | budget | within | six | months | of | the |

| commen@@ | cement | of | the | year | . |

Figure 8: Example of translation from the law domain. The tokens generated at the retrieval steps are highlighted in the same color as the retrieved chunks of tokens. These chunks of tokens are added to the neighbors' cache, after being retrieved. The retrieved tokens which are present in the translation are bolded.

Source: Mit dem Drehregler können Sie die Anzahl Tage in die Zukunft festlegen, für die anstehende Aufgaben angezeigt werden sollen. Der maximale Einstellung ist ein Jahr.

Target: Use this spinbox to set the number of days to show pending To-dos up to 1 year in the future.

Neighbors' cache

['**Use**', 'this', '**sp@@**', '**in@@**', '**box**', '**to**', '**set**', '**the**', '**number**', '**of**', '**days**', '**to**', '**show**', 'upcoming', 'special', 'occasions']
['**Use**', 'this', '**sp@@**', '**in@@**', '**box**', '**to**', '**set**', '**the**', '**number**', '**of**', '**days**', '**to**', '**show**', 'upcoming', 'events', '**up**']
['This', '**sp@@**', '**in@@**', '**box**', 'allows', 'you', '**to**', '**set**', '**the**', 'reminder', 'trigger', 'time', '**.**', 'The', 'time', 'unit']
['With', '**the**', 'sli@@', 'der', ',', 'you', 'can', '**set**', '**the**', 'speed', 'of', '**the**', 'anim@@', 'ation', '**.**', '</s>']
['**Use**', '**the**', 'sli@@', 'der', '**to**', '**set**', '**the**', 'mo@@', 'dem', 'volume', '**.**', 'Left', 'is', 'low', 'volume', ',']
['The', 'sli@@', 'der', 'bar', 'can', 'be', 'used', '**to**', 'ad@@', 'just', '**the**', 'dur@@', 'ation', '**of**', '**the**', 'visible']
['**Use**', '**the**', 'sli@@', 'der', '**to**', 'select', 'a', 'value', 'between', '5@@', 'x@@', '5', 's@@', 'quar@@', 'es', 'and']
['With', 'this', 'sli@@', 'der', 'you', 'can', '**set**', '**the**', 'bri@@', 'ght@@', 'ness', 'when', '**the**', 'system', 'is', 'plu@@']

['**sp@@**', '**in@@**', '**box**', '**to**', '**set**', '**the**', '**number**', '**of**', '**days**', '**to**', '**show**', 'upcoming', 'special', 'occasions', '**up**', '**to**']
['**sp@@**', '**in@@**', '**box**', '**to**', '**set**', '**the**', '**number**', '**of**', '**days**', '**to**', '**show**', 'upcoming', 'events', '**up**', '**to**', '**1**']
['sli@@', 'der', '**to**', '**set**', '**the**', 'mo@@', 'dem', 'volume', '**.**', 'Left', 'is', 'low', 'volume', ',', 'center', 'is']
['sli@@', 'der', '**to**', 'select', 'a', 'value', 'between', '5@@', 'x@@', '5', 's@@', 'quar@@', 'es', 'and', '10@@', 'x@@']
['sli@@', 'der', ',', 'you', 'can', '**set**', '**the**', 'speed', '**of**', '**the**', 'anim@@', 'ation', '**.**', '</s>']
['spin', '**box**', 'combination', '**.**', '</s>']
['rot@@', 'ate', 'butt@@', 'ons', 'visible', '</s>']
['sli@@', 'der', ')', 'before', '**the**', 'key@@', 'stroke', 'will', 'be', 'accepted', '**.**', 'This', 'helps', 'prevent', 'acci@@', 'dental']

['**set**', '**the**', '**number**', '**of**', '**days**', '**to**', '**show**', 'upcoming', 'events', '**up**', '**to**', '**1**', '**year**', '**in**', '**the**', '**future**']
['**set**', '**the**', '**number**', '**of**', '**days**', '**to**', '**show**', 'upcoming', 'special', 'occasions', '**up**', '**to**', 'one', '**year**', '**in**', '**the**']
['speci@@', 'fy', '**the**', 'dur@@', 'ation', 'for', 'displa@@', 'ying', 'a', 'frame', 'and', '**the**', '**number**', '**of**', 'times', 'an']
['define', 'how', 'broad', '**the**', 'trail', '**of**', 'a', 'vehicle', 'should', 'be', '**.**', 'Mo@@', 'ving', 'the', 'sli@@', 'der']
['determine', 'how', 'many', 'lines', '**of**', 'text', 'one', 'step', '**of**', '**the**', 'm@@', 'ouse', 'wheel', 'will', 'sc@@', 'roll']
['speci@@', 'fy', '**the**', '**number**', '**of**', 'rows', 'you', 'want', 'in', '**the**', 'voc@@', 'ab@@', 'ul@@', 'ary', '**.**', 'You']
['ad@@', 'just', 'the', 'time', ',', 'in', 'milli@@', 'seconds', ',', 'that', '**the**', 'OS@@', 'D', 'is', 'displayed', 'on']
['**set**', '**the**', '**number**', '**of**', 'pages', '**to**', 'be', 'displayed', '**.**', '</s>']

['**pending**', 'To', '@-@', 'dos', '</s>']
['upcoming', 'events', '**up**', '**to**', '**1**', '**year**', '**in**', '**the**', '**future**', '**.**', '</s>']
['**pending**', 'To', '@-@', 'dos', '</s>']
['upcoming', 'special', 'occasions', '**up**', '**to**', 'one', '**year**', '**in**', '**the**', '**future**', '**.**', '</s>']
['upcoming', 'events', '</s>']
['upcoming', 'events', '</s>']
['**pending**', '**tasks**', ',', 'record', 'your', 'occ@@', 'ur@@', 'ren@@', 'ces', ',', 'experiences', ',', 'and', 'refle@@', 'ctions', 'and']
['upcoming', 'events', 'and', 'to@@', 'dos', '**.**', '</s>']

Translation

| Use | the | sp@@ | in@@ | box | to | set | the | number | of | days | to | show | pending | tasks | up | to | 1 |

| year | in | the | future | . |

Figure 9: Example of translation from the IT domain. The tokens generated at the retrieval steps are highlighted in the same color as the retrieved chunks of tokens. These chunks of tokens are added to the neighbors' cache, after being retrieved. The retrieved tokens which are present in the translation are bolded.