Non-parametric Online Learning from Human Feedback for Neural Machine Translation

Dongqi Wang[1], **Haoran Wei**[2], **Zhirui Zhang**[2], **Shujian Huang**[1*], **Jun Xie**[2], **Jiajun Chen**[1]
[1]National Key Laboratory for Novel Software Technology, Nanjing University, China
[2]Machine Intelligence Technology Lab, Alibaba DAMO Academy
wangdq@smail.nju.edu.cn, {huangsj, chenjj}@nju.edu.cn
{funan.whr, zhirui.zzr, qingjing.xj}@alibaba-inc.com

Abstract

We study the problem of online learning with human feedback in the human-in-the-loop machine translation, in which the human translators revise the machine-generated translations and then the corrected translations are used to improve the neural machine translation (NMT) system. However, previous methods require online model updating or additional translation memory networks to achieve high-quality performance, making them inflexible and inefficient in practice. In this paper, we propose a novel non-parametric online learning method without changing the model structure. This approach introduces two k-nearest-neighbor (KNN) modules: one module memorizes the human feedback, which is the correct sentences provided by human translators, while the other balances the usage of the history human feedback and original NMT models adaptively. Experiments conducted on EMEA and JRC-Acquis benchmarks demonstrate that our proposed method obtains substantial improvements on translation accuracy and achieves better adaptation performance with less repeating human correction operations.

Introduction

The quality of the neural machine translation (NMT) system has been significantly improved recently (Sennrich, Haddow, and Birch 2016a; Vaswani et al. 2017; Zhang et al. 2018b; Hassan et al. 2018). However, recent research has shown that machine translation still lags behind human parity on translation quality (Läubli, Sennrich, and Volk 2018; Freitag et al. 2021). In some scenarios where high-quality translation should be guaranteed, human-in-the-loop machine translation, i.e., machine translation with human post-editing, is still indispensable. As human corrections on machine-translated text are constantly produced, adapting the NMT model to these corrections could improve translation quality and effectively reduce human efforts, as shown in Figure 1. However, the training of NMT models requires a certain amount of data and consumes dozens or hundreds of GPU hours, which is unavailable in this scenario. Therefore, online learning from human feedback at a low training cost has become a promising research topic in recent years.

A series of studies (Turchi et al. 2017; Kothur, Knowles, and Koehn 2018) propose to on-the-fly update the NMT model right after humans correct the translation. Although these methods improve translation performance, they require extra costs for frequent gradient computations at the inference time. Tuning parameters only on the new-coming sentences also brings the risk of catastrophic forgetting problems. Another research line (Gu et al. 2018; Zhang et al. 2018a) augments the NMT model with the translation memory to cache the human-corrected sentences. For each sentence to translate, cached sentences with similar context are retrieved to help the decoding process of the NMT model. Memory-augmented NMT avoids updating parameters on-the-fly, but it requires careful network architecture design to model the retrieved translation memories. Also, extra training overhead is inevitable to optimize the memory network.

Recently, Khandelwal et al. (2020) proposed KNN-MT, a non-parametric and model-agnostic method for domain adaptation. It equips the pre-trained NMT model with a k-nearest-neighbor classifier over a datastore of cached context representations and corresponding target tokens. The final translation probability is the interpolation between the probability calculated by the NMT model and the KNN module. KNN-MT provides a simple but effective way to exploit human feedback by adding human-corrected translations into the datastore. However, the success of KNN-MT is owed to enough in-domain samples pre-cached in the datastore. In the human-in-the-loop translation, to learn the cor-

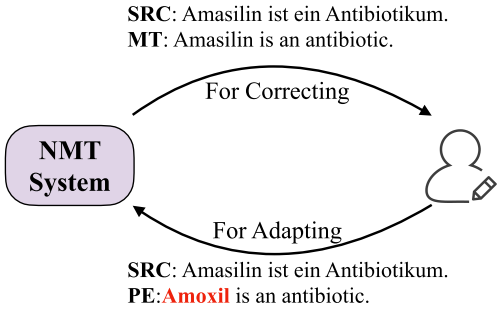

Figure 1: The workflow of online learning from human feedback in the human-in-the-loop machine translation. Given the source sentence *SRC*, the NMT System generates the translation *MT*. The user corrects it and produces *PE*. Then the corrected sample is used to adapt the NMT system.

*Corresponding author.
Copyright © 2022, Association for the Advancement of Artificial Intelligence (www.aaai.org). All rights reserved.

rected translation, human translators have to make repetitive corrections on the same mistake to provide enough samples for KNN-MT. Biasing the interpolation to the KNN module could accelerate this learning process, but it takes the risk of being interfered by the noise in retrieved neighbors when retrieved items are not relevant enough. Therefore, it is crucial to balance the usage of the probability from the NMT model and the KNN module when applying KNN-MT.

In this paper, we propose KNN-over-KNN (KoK), a plug-and-play non-parametric approach for online learning from human feedback. KoK equips the NMT model with two KNN modules, Token-KNN and Policy-KNN. Token-KNN is used to model the translation probability as in KNN-MT. Human-corrected sentences are incrementally added to the datastore of Token-KNN to improve the translation quality of proceeding sentences. Policy-KNN is introduced to model the balance mechanism between the Token-KNN module and the NMT model. It interpolates between the two modules with a Bernoulli distribution. The distribution is estimated by a k-nearest-neighbor classifier over a datastore. For building the datastore, we extract features from the retrieval results of Token-KNN as the key, and heuristically induce a binary flag for each result as the value (1 means using this result and vice versa).

We conduct experiments on translating domain-specific documents with various lengths. Oracle references are used to mimic human feedback. KoK obtains significant improvements (up to 12.9 BLEU improvement and 9.7 TER reduction) compared to the Pre-Trained NMT model. KoK also achieves consistent improvements over existing online learning methods on documents at all lengths. We further show that KoK can adapt to human feedback with less repeated corrections, which reduces human effort. Our code is open-sourced at https://github.com/wangqi1996/KoK.

Background

Neural Machine Translation

Neural machine translation systems formulate the translation task as a conditional probability model $p(\boldsymbol{y}|\boldsymbol{x})$, which defines the translation process from source sentence $\boldsymbol{x} = \{x_1, x_2, ..., x_m\}$ into target sentence $\boldsymbol{y} = \{y_1, y_2, ..., y_n\}$. The representative works decompose it in an auto-regressive manner from left to right :

$$p_{\text{NMT}}(\boldsymbol{y}) = \prod_{i=1}^{n} p_{\text{NMT}}(y_t|\boldsymbol{y}_{<t}, \boldsymbol{x}),$$

where $\boldsymbol{y}_{<t} = \{y_1, y_2, ..., y_{t-1}\}$ denotes the prefix tokens. Then the probability of each target token is defined as:

$$\begin{aligned} p_{\text{NMT}}(y_t|\boldsymbol{y}_{<t}, \boldsymbol{x}) &= \text{softmax}(\boldsymbol{W}\boldsymbol{h}_t + \boldsymbol{b}) \\ \boldsymbol{h}_t &= \text{Transformer}(\boldsymbol{x}, \boldsymbol{y}_{<t}), \end{aligned} \tag{1}$$

where $\boldsymbol{h}_t$ is the representation for context $(\boldsymbol{x}, \boldsymbol{y}_{<t})$, and $\boldsymbol{W}$, $\boldsymbol{b}$ are the trainable parameters to mapping the dimension of $\boldsymbol{h}_t$ to the vocab size.

Online Learning from Human Feedback

In the human-in-the-loop translation, a complete translation step is: the NMT model generates machine-translated text, and then the human translator makes revisions on it (human feedback). Online Learning from human feedback follows such a paradigm: after the human completes the revision of the current sentence, the machine translation system is adapted incrementally by taking this sample into account. Concretely, given a document $\mathcal{D} = \{\boldsymbol{x}^1, \boldsymbol{x}^2, .., \boldsymbol{x}^{|\mathcal{D}|}\}$, where $\boldsymbol{x}^i$ represents i^{th} sentence, the translation for $\boldsymbol{x}^i$ is generated as

$$\hat{\boldsymbol{y}}^i = f^{i-1}(\boldsymbol{x}^i)$$

where f^{i-1} is the NMT model adapted by $\boldsymbol{x}^{<i}$ sentences and f^0 is the pre-trained model. The human translator corrects $\hat{\boldsymbol{y}}^i$, and then produces the reference $\boldsymbol{y}^i$. The model f^i is acquired by adapting f^{i-1} over the new sample $(\boldsymbol{x}^i, \boldsymbol{y}^i)$:

$$f^i \leftarrow \text{Adaptation}(f^{i-1}, \boldsymbol{x}^i, \boldsymbol{y}^i).$$

The adaptation process could be tuning the model parameters, or updating the external translation memory. In the rest of the paper, we omit the superscript i for simplicity.

KNN-MT

Recently, Khandelwal et al. (2020) proposed KNN-MT, a non-parametric method. It augments the NMT model with a token-level k-nearest-neighbor retrieval mechanism, allowing it to directly access the cached examples stored in the datastore during inference. It consists of two steps: datastore construction and inference.

Datastore Construction The datastore consists of a set of key-value pairs. More specifically, given the parallel sentence in the dataset $(\boldsymbol{x}, \boldsymbol{y}) \in (\mathcal{X}, \mathcal{Y})$, the pre-trained NMT model generates the context representation $\boldsymbol{h}_t$ as Equation 1 at every timestep t, then the datastore takes the $\boldsymbol{h}_t$ as the key and the corresponding target tokens y_t as the value:

$$(\mathcal{K}, \mathcal{V}) = \{(\boldsymbol{h}_t, y_t), \forall y_t \in \boldsymbol{y} | (\boldsymbol{x}, \boldsymbol{y}) \in (\mathcal{X}, \mathcal{Y})\}.$$

Inference The inference process performs in a token-by-token manner. Given the source sentence $\boldsymbol{x}$ and the generated target prefix $\hat{\boldsymbol{y}}_{<t}$, the pre-trained NMT model generates the representation $\hat{\boldsymbol{h}}_t$ and then predicts a probability $p_{\text{NMT}}(\hat{y}_t|\boldsymbol{x}, \hat{\boldsymbol{y}}_{<t})$ for the target token $\hat{y}_t$.

Then, the k-nearest-neighbor model retrieves the similar K neighbours $\mathcal{R} = \{(\boldsymbol{k}_i, v_i), i \in \{1, 2, ..., K\}\}$ in the datastore according to Euclidean distance to $\hat{\boldsymbol{h}}_t$. The retrieved result is converted into a distribution over the vocabulary by

$$\begin{aligned} p(\boldsymbol{k}_i|\hat{\boldsymbol{h}}_t) &= \text{softmax}(\frac{-d(\boldsymbol{k}_i, \hat{\boldsymbol{h}}_t)}{T}) \\ p_{\text{KNN}}(\hat{y}_t|\boldsymbol{x}, \hat{\boldsymbol{y}}_{<t}) &= \sum_{(\boldsymbol{k}_i, v_i) \in \mathcal{R}} \mathbb{I}_{v_i = \hat{y}_t} p(\boldsymbol{k}_i|\hat{\boldsymbol{h}}_t)), \end{aligned} \tag{2}$$

where $d(\boldsymbol{k}_i, \hat{\boldsymbol{h}}_t)$ represents the Euclidean distance between $\boldsymbol{k}_i$ and $\hat{\boldsymbol{h}}_t$, $p(\boldsymbol{k}_i|\hat{\boldsymbol{h}}_t)$ is the probability of i^{th} retrieval neighbor $(\boldsymbol{k}_i, v_i)$. T is a hyper-parameter to control the sharpness of the softmax function.

The final output distribution is an interpolation between distributions from the NMT model and the KNN retrieved neighbors with a tuned parameter $\lambda \in [0, 1]$

$$\begin{aligned} p(\hat{y}_t|\boldsymbol{x}, \hat{\boldsymbol{y}}_{<t}) &= \lambda * p_{\text{KNN}}(\hat{y}_t|\boldsymbol{x}, \hat{\boldsymbol{y}}_{<t}) \\ &\quad + (1 - \lambda) * p_{\text{NMT}}(\hat{y}_t|\boldsymbol{x}, \hat{\boldsymbol{y}}_{<t}). \end{aligned} \tag{3}$$

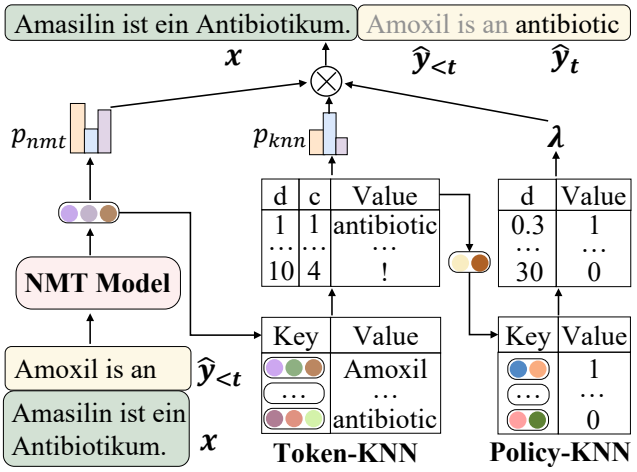

Figure 2: An illustration of proposed KoK.

Proposed Method

In this section, we describe the proposed method, KNN-over-KNN (KoK). Based on KNN-MT, we model a balance mechanism between the usage of the pre-trained NMT model and the KNN module cached history human corrections, i.e., we replace the fixed λ in Equation 3 into a dynamic λ_t varies at each predicting $\hat{y}_t$:

$$p(\hat{y}_t|\boldsymbol{x}, \hat{\boldsymbol{y}}_{<t}) = \lambda_t * p_{\text{KNN}}(\hat{y}_t|\boldsymbol{x}, \hat{\boldsymbol{y}}_{<t}) + (1-\lambda_t) * p_{\text{NMT}}(\hat{y}_t|\boldsymbol{x}, \hat{\boldsymbol{y}}_{<t}). \quad (4)$$

KoK learns to predict p_{KNN} and λ_t from history human feedback in a non-parametric way, and can be applied to NMT models with various architectures.

Architecture

Figure 2 shows the overall architecture of KoK. It parameterizes the final translation probability (Equation 4) with two KNN modules besides the pre-trained NMT model, in which the Token-KNN module predicts the probability p_{KNN} over target words and the Policy-KNN module predicts an interpolation coefficient λ_t.

During the translation process of one sentence, for each decoding step t, Token-KNN takes the hidden representation $\hat{\boldsymbol{h}}_t$ as query and retrieves from its datastore to generate the translation probability p_{KNN}. Then features extracted from the retrieval result of Token-KNN are used as the query of Policy-KNN to calculate the λ_t value. The final translation probability (Equation 4) is computed as an interpolation between p_{KNN} and p_{NMT} with λ_t.

Token-KNN

Token-KNN module predicts the translation probability p_{KNN} with k-nearest-neighbor classifier over the datastore. It is similar to the KNN module in the KNN-MT, but the datastore is built incrementally. During the translation process, with the $\hat{\boldsymbol{h}}_t$ as the query, the Token-KNN queries from the token datastore. The translation probability is computed as Equation 2. During the adaptation process, given source sentence $\boldsymbol{x}$ and its human-corrected translation $\boldsymbol{y}$, we build the key-values pairs and then store them into the datastore.

Policy-KNN

Policy-KNN module calculates the λ_t, which expresses the balance mechanism between the usage of the pre-trained NMT model and the Token-KNN module. We introduce a random variable $\boldsymbol{\lambda}$ which obeys the Bernoulli distribution. $\boldsymbol{\lambda} = 1$ represents to use the translation probability of Token-KNN and vice versa for the NMT model. Thus, predicting coefficient λ_t in Equation 4 is demonstrated as estimating the conditional probability $p(\boldsymbol{\lambda} = 1|\boldsymbol{x}, \boldsymbol{y}_{<t})$

$$\lambda_t = p(\boldsymbol{\lambda} = 1|\boldsymbol{x}, \boldsymbol{y}_{<t}).$$

Policy-KNN estimates this probability with k-nearest-neighbor classifier over the datastore as well. In the following, we will describe in detail the datastore construction and inference process of Policy-KNN. At the t^{th} decoding step, we denote the retrieval results from Token-KNN as $\mathcal{R}_t^{\text{tok}} = \{(\boldsymbol{k}_i^{\text{tok}}, v_i^{\text{tok}}), i \in \{1, 2, ..., K\}\}$.

Key Refer to Zheng et al. (2021a), we construct the key vector $\boldsymbol{s}_t$ using two kinds of features extracted from the retrieval results of Token-KNN. One feature is the distance, we denote the Euclidean distance between the context representation $\boldsymbol{h}_t$ and i^{th} retrieval result $\boldsymbol{k}_i^{\text{tok}}$ as d_i.

$$d_i = \left\|\boldsymbol{h}_t - \boldsymbol{k}_i^{\text{tok}}\right\|_2.$$

The other feature is the counts of distinct values for all retrieval results, marked as c_i. Specially, c_i is computed as:

$$c_i = |\text{UNIQUE}(v_1^{\text{tok}}, v_2^{\text{tok}}, ..., v_i^{\text{tok}})|$$

where UNIQUE represents the unique elements in the list. $\boldsymbol{s}_t$ is constructed by concatenating all the features:

$$\boldsymbol{s}_t = [d_1, d_2, ..., d_K; c_1, c_2, ..., c_K]. \quad (5)$$

Intuitively speaking, the distance features d_i stand for similarity, which directly evaluates the importance of each retrieval result. The count features c_i represents the consistency of retrieval results, as more distinct retrieved values mean less credible KNN retrieval. Both of these two features are crucial for predicting an optimal value of $\boldsymbol{\lambda}$.

We further re-weight the key vector according to the significance of each element. We adopt exponential re-weighting on distance features and count features, respectively:

$$\boldsymbol{s}_t = \boldsymbol{s}_t \odot [\frac{1}{4}, \frac{1}{8}, ..., \frac{1}{2^K}, \frac{1}{2^K}; \frac{1}{2^K}, \frac{1}{2^K}, ..., \frac{1}{8}, \frac{1}{4}]. \quad (6)$$

Value The retrieval result of Token-KNN does not explicitly come with a golden value of $\boldsymbol{\lambda}$ indicating whether to use this result or not. Instead, we implicitly induce the value of $\boldsymbol{\lambda}$ through a simple heuristic:

$$\boldsymbol{\lambda} = \begin{cases} 1 & p_{\text{KNN}}(y_t|\boldsymbol{x}, \boldsymbol{y}_{<t}) > p_{\text{NMT}}(y_t|\boldsymbol{x}, \boldsymbol{y}_{<t}) \\ 0 & p_{\text{KNN}}(y_t|\boldsymbol{x}, \boldsymbol{y}_{<t}) \leq p_{\text{NMT}}(y_t|\boldsymbol{x}, \boldsymbol{y}_{<t}) \end{cases} \quad (7)$$

where $\boldsymbol{x}$ and $\boldsymbol{y}$ is the source sentence and its human-corrected translation, respectively.

This heuristic is actually mathematically meaningful. To find the value of λ_t to maximize the probability of KNN-MT (Equation 4), we need to solve the optimization problem below:

$$\begin{aligned}\lambda_t &= \text{argmax } p(y_t) \\ &= \text{argmax}_{\lambda_t}\ \lambda_t \cdot p_{\text{KNN}}(y_t) + (1-\lambda_t)\cdot p_{\text{NMT}}(y_t) \\ &= \text{argmax}_{\lambda_t}\ p_{\text{NMT}}(y_t) + (p_{\text{KNN}}(y_t) - p_{\text{NMT}}(y_t))\cdot\lambda_t.\end{aligned}$$

Inference During the translation process, at the t^{th} decoding step, given the retrieval result $\mathcal{R}_t^{\text{tok}}$ of Token-KNN, we build the query feature $\hat{\boldsymbol{s}}_t$ as in Equation 6, then Policy-KNN retrieves K neighborhoods, $\mathcal{R}_t^{\lambda} = \{(\boldsymbol{k}_i^{\lambda}, v_i^{\lambda}), i \in \{1,2,...,K\}\}$, from its datastore. The retrieval result λ_t is calculated by:

$$\begin{aligned}p(\boldsymbol{k}_i^{\lambda}|\hat{\boldsymbol{s}}_t) &= \text{softmax}(\frac{-d(\boldsymbol{k}_i^{\lambda}, \hat{\boldsymbol{s}}_t)}{T}) \\ \lambda_t &= p(\boldsymbol{\lambda}=1|\boldsymbol{x}, \hat{\boldsymbol{y}}_{<t}) \\ &= \sum_{(\boldsymbol{k}_i^{\lambda}, v_i^{\lambda})\in\mathcal{R}^{\lambda}} \mathbb{I}_{v_i^{\lambda}=1} * p(\boldsymbol{k}_i^{\lambda}|\hat{\boldsymbol{s}}_t).\end{aligned} \tag{8}$$

The final translation probability is calculated as Equation 4.

Implementation

Overall, KoK is performed as an online learning paradigm. It takes three steps when translating a source sentence $\boldsymbol{x}$:

a. **Translation**: KoK generates the translation $\hat{\boldsymbol{y}}$ by maximize Equation 4. At each decoding step, KoK computes p_{NMT} from the NMT model, p_{KNN} from the Token-KNN, and λ_t from the Policy-KNN, respectively.

b. **Correction**: The human translator corrects the translation result $\hat{\boldsymbol{y}}$ and produces the corrected version $\boldsymbol{y}$.

c. **Adaptation**: $\boldsymbol{x}$ and $\boldsymbol{y}$ are encoded by the NMT model with teacher-force decoding. All the tokens in $\boldsymbol{y}$ are used to retrieve from Token-KNN as well. Then we update Policy-KNN and Token-KNN by building key-value pairs, successively.

When the translation of a document is completed, one can choose to clear the datastores of two KNN modules for translating new documents, or keep them for proceeding translation on similar documents, which is dependent on user's demand. We clear the datastores for translating new documents in this paper. The details of the creating/updating process of datastore are shown in Appendix.

Experiments

Experimental Setup

We evaluate the proposed method by equipping the general-domain pre-trained NMT model with KoK and use it to translated in-domain documents with various document sizes. We use the oracle reference as the human-corrected translations to simulate the real human-in-the-loop scenario.

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

Table 1: Data statistics for the EMEA and JRC-Acquis dataset. *Documents* represents the number of documents in the bucket. *Ave sentences/tokens* stand for the average sentences/tokens of the documents in the bucket.

Dataset. We conduct the experiments on two specific domain datasets from OPUS (Tiedemann 2012), which are widely employed by previous works (Zheng et al. 2021a; Cai et al. 2021): (1) European Medicines Agency (EMEA) dataset[1] (Tiedemann 2009), which consists of sentence-aligned documents focusing on medical products. (2) JRC-Acquis corpus (Steinberger et al. 2006), which contains the European Union laws applicable to the EU member states. Following common practices, we use the Moses toolkit[2] to tokenize the documents, remove the same sentences in a document and then segment the words into subword units (Sennrich, Haddow, and Birch 2016b) with the bpe-codes provided by the pre-trained model.

We validate our method on documents with various lengths. Specifically, we divide the documents into five buckets based on their length (0-50, 50-100, 100-200, 200-500 and 500-1000). We randomly select some documents so that the total document length of each bucket is upper than 1000. Detailed for EMEA/JRC dataset statistics are shown in Table 1.

Implementation Details. We apply the FAIRSEQ[3] (Ott et al. 2019) toolkit for NMT implementation, and Faiss[4](Johnson, Douze, and Jégou 2017) with *Exact Search for L2* setting for efficient KNN retrieval. Following the previous experiences (Zheng et al. 2021a; Khandelwal et al. 2020), we employ the WMT19 German-English news translation task winner model (Ng et al. 2019) as the pre-trained model. The K for Token-KNN and Policy-KNN is 8.

Baselines. We compare our method with several representative researches, including:

- Pre-Trained: We only use the pre-trained NMT model during translation, which measures the domain diversity without any adaptation.
- Online Tuning: Online updating the pre-trained NMT model with human-corrected sentences. Following

[1]http://opus.lingfil.uu.se/EMEA.php
[2]https://github.com/moses-smt/mosesdecoder
[3]https://github.com/pytorch/fairseq
[4]https://github.com/facebookresearch/faiss

Bucket	0-50		50-100		100-200		200-500		500-1000		Average	
	BLEU	TER	BLEU	TER	BLEU	TER	BLEU	TER	BLEU	TER	BLEU	TER
Pre-Trained	43.8	**52.1**	43.1	52.8	38.3	54.0	41.9	53.8	40.8	53.4	41.6	53.2
Online Tuning	44.0	52.2	43.5	52.3	39.6	51.4	43.8	51.8	44.7	49.3	43.1	51.4
KNN-MT	43.8	52.6	43.6	52.5	40.0	53.1	43.8	52.3	44.2	50.8	43.1	52.3
Adaptive KNN-MT	29.7	70.2	28.9	70.3	35.9	58.4	37.2	61.2	48.2	50.3	36.0	62.1
KoK	**44.4**	**52.1**	**43.9**	**52.4**	**44.1**	**50.0**	**45.7**	**51.1**	**53.7**	**43.7**	**46.4**	**49.9**

Table 2: BLEU (↑) and TER (↓) on the EMEA dataset.

Bucket	0-50		50-100		100-200		200-500		500-1000		Average	
	BLEU	TER	BLEU	TER	BLEU	TER	BLEU	TER	BLEU	TER	BLEU	TER
Pre-Trained	54.0	37.2	49.9	41.2	41.9	47.1	39.9	48.8	43.4	45.3	45.8	43.9
Online Tuning	54.4	37.0	50.9	40.5	43.8	45.4	42.8	46.3	47.5	42.2	47.9	42.3
KNN-MT	55.5	38.0	52.2	40.9	45.7	44.7	43.6	46.4	47.7	42.4	48.9	42.5
Adaptive KNN-MT	42.5	53.0	41.7	52.8	40.2	52.3	39.2	52.4	45.4	46.3	41.8	51.3
KoK	**56.3**	**35.1**	**52.4**	**39.3**	**47.7**	**43.1**	**44.7**	**45.6**	**50.1**	**40.1**	**50.2**	**40.6**

Table 3: BLEU (↑) and TER (↓) on the JRC-Acquis dataset.

Kothur, Knowles, and Koehn (2018), we train the model 5 steps on every single sentence with the adam optimizer. The hyper-parameters setting follows the pre-trained models, except the learning rate is set to 1.0×10^{-7} and fixed during the inference process.

- KNN-MT: After every sentence is translated, we add the human-corrected sentences to the datastore of KNN-MT. Similar to our method, the K is 8. The λ value for the EMEA dataset and the JRC-Acquis dataset are 0.2 and 0.3, respectively.
- Adaptive KNN-MT (Zheng et al. 2021a): A variant of KNN-MT, which introduces a network to dynamically determine the number of K for each target token. We use the model pre-trained on the IT domain (provided by the paper) and incrementally update the datastore as in the KNN-MT.

Evaluation. We use SacreBLEU[5] (Post 2018) to measure the result with case-sensitive detokenized BLEU (Papineni et al. 2002). We concatenate the translations of all the documents in each bucket and calculate the corpus-level BLEU. For the completeness of the results, we also report the TER (Snover et al. 2006) metric to compute the edit distance between the reference and system translation.

Main Results

We can see that consistent improvements (up to 12.9 BLEU improvement and 9.7 TER reduction) are achieved by KoK on different document lengths. As a comparison, Online Tuning achieves relatively minor improvement on long documents, probably because that several thousand samples are still not enough for a gradient-based approach. KNN-MT fails to attain similar improvements as KoK on long documents because of the small but fixed λ value. This indicates the benefit of adjusting λ adaptively. Instead, the performance of adaptive KNN-MT deteriorates when the length of a document is relatively small. As the training of the adaptive KNN-MT is based on a static datastore built from all in-domain training data, it is not compatible with the human-in-the-loop machine translation scenario that the human feedback comes in a streaming way. Overall, the results demonstrate the effectiveness and generalization of our method on different document conditions.

We also conduct experiments on the JRC-Acquis dataset, which contains documents related to the law domain. All the results are listed in Table 3. KoK consistently outperforms all the baselines, which demonstrates that our approach can be generalized to other domains. We show the performance on the full test sets in Appendix.

Comparing With Translation Memory

We also compare our method with the NMT model augmented by the translation memory (TM-NMT) (Kuang et al. 2021; Bapna and Firat 2019). Due to the training data of the WMT19 NMT model is unavailable, we implement the TM-NMT model and the NMT model of KoK on the WMT14 DE-EN dataset for a fair comparison, and evaluate them on the EMEA dataset. For the NMT model, we use the transformer big architecture, and the training detail follows the introduction of fairseq[6]. For the TM-NMT, we follow the *source similarity* method in Cai et al. (2021), and the architecture is the same as the transformer big model.

The result is illustrated in Table 4. As seen, KoK still achieves improvements on all the document sizes. However, TM-NMT fails to exceed the pretrained NMT model except on the 200-500 bucket, which may be resulted from the inability to handle the low quality of retrieval memories (Cai et al. 2021).

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

[6]https://github.com/pytorch/fairseq/issues/202

Bucket	0-50		50-100		100-200		200-500		500-1000		Average	
	BLEU	TER	BLEU	TER	BLEU	TER	BLEU	TER	BLEU	TER	BLEU	TER
Pre-Trained	38.8	55.7	36.6	57.1	33.2	59.2	33.9	60.1	32.2	62.0	29.1	58.8
TM-NMT	38.1	**55.6**	35.6	57.7	32.7	59.0	35.9	59.8	32.0	61.8	29.1	58.8
KoK	**39.0**	**55.6**	**37.3**	**56.7**	**36.6**	**56.6**	**38.6**	**57.6**	**44.1**	**52.5**	**32.6**	**55.8**

Table 4: Performance comparison between KoK and NMT model augmented by the translation memory (TM-NMT).

Model	Latency	Speedup
Pre-Trained	269.5 ms	$\times 1.00$
Online Tuning	289.9 ms	$\times 0.93$
KNN-MT	324.7 ms	$\times 0.83$
Adaptive KNN-MT	409.8 ms	$\times 0.66$
KoK	384.6 ms	$\times 0.70$

Table 5: Decoding latency of different models on the EMEA dataset. We also list the speedup w.r.t the pre-trained NMT model.

Latency

In the human-in-the-loop translation scenario, human translators are sensitive to decoding latency. Therefore, we also evaluate the latency of the proposed KoK. We simulate the Computer Aided Translation(CAT) task setting and measure the decoding latency sentence-by-sentence[7]. The latency is calculated on the EMEA dataset, and we report the average latency in milliseconds. The speed-up ratio is also computed by comparing it with the Pre-Train Model following the previous practice (Zheng et al. 2021a). The result is summarized in Table 5. Our method slightly slows down the inference time, but significantly improves translation quality, which is tolerable for users.

Analysis

In this section, we perform several analyses and ablation studies for the proposed KoK.

Can KoK Learn Optimal λ

KoK equips KNN-MT with a Policy-KNN to adaptively decide between adopting knowledge from human feedback or keeping decoding results from the pre-trained NMT model on every token. In this section, we demonstrate that KoK indeed learns an optimal policy to make the decision.

We first compare KoK with KNN-MT which chooses different λ values. The results are shown in Figure 3. We try $\lambda \in [0.0, 1.0)$ with 0.1 as the step size, and report average BLEU scores on the various document length buckets on the EMEA dataset. KNN-MT is sensitive to the choice of λ value (BLEU varies from 18 to 43), which indicates the importance of selecting a proper λ value. KNN-MT achieves the best performance when λ is around 0.2, but still lags behind KoK. This experiment shows that it is crucial to decide the λ value token-by-token adaptively.

[7]The latency is measured on a single GeForce GTX 1080-Ti GPU.

We further evaluate whether KoK could learn an optimal λ. Given oracle target $\boldsymbol{y}$, for each $y_t \in \boldsymbol{y}$, we compute the Token-KNN probability of y_t as $p_{knn}(\boldsymbol{y}_t)$, and the corresponding λ_t predicted by Policy-KNN. To show the relationship between λ_t and $p_{knn}(\boldsymbol{y}_t)$, we divide all the Token-KNN probability values into five buckets and compute the average λ values in every bucket. The result is shown in Figure 4. As seen, the λ value becomes high when the Token-KNN probability gets bigger, and vice versa. This is in line with the motivation that KoK should bias to the usage of Token-KNN probability when retrieved human feedbacks indeed help the translation. We also evaluate the effectiveness of K in the Appendix.

Zero-Shot and Few-Shot Ability of KoK

In machine translation with the human-in-the-loop scenario, users not only care about the quality of the translation model, but also care about how fast the model can learn from their feedback to avoid repeatedly making the same mistakes they already corrected before. Therefore, in this section, we evaluate the adaptation speed of KoK to user feedback.

We follow the R-indicator used in Simianer, Wuebker, and DeNero (2019), which measures the translation recall of words with different occurrence times in users' feedback. We denote R_i as the recall of tokens that have appeared i times in the previous corrected sentences:

$$R_i = \frac{\sum_{j=1}^{|\mathcal{D}|} |\mathcal{H}_j \cap \mathcal{R}_{i,j}|}{\sum_{j=1}^{|\mathcal{D}|} |\mathcal{R}_{i,j}|} \tag{9}$$

where $\mathcal{H}_j$ represents unique words in the j^{th} machine-generated translation and $\mathcal{R}_{i,j}$ represents unique words in the j^{th} reference that are their $(i+1)^{th}$ occurrence in the whole document. Specifically, R_0 evaluates the tokens that first appear in the translating document and R_1 considers those that have appeared once.

We conduct experiments on documents with $[200, 500]$ bucket from EMEA. R_0, R_1, $R_{2\sim5}$, $R_{5\sim9}$ and R_{9+} are computed for KoK and other baselines as Equation 9 respectively ($R_{m\sim n}$ is defined as the micro average of $R_m, R_{m+1}, ..., R_n$). The results are shown in Figure 5. KoK achieves the comparable R_0, but the R_i values improve quickly and outperform all the other methods. It indicates KoK's ability to reduce interference from unrelated retrieval results of Token-KNN, and to adapt to helpful human feedback faster. Adaptive KNN-MT achieves a lower R_0, which may due to the same reason about worse performance in main experiments.

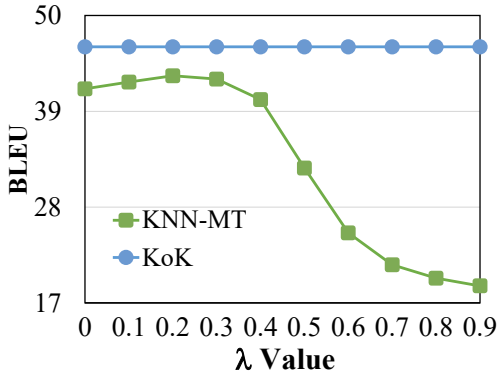

Figure 3: Results of KoK v.s. KNN-MT with different λ values on EMEA datasets.

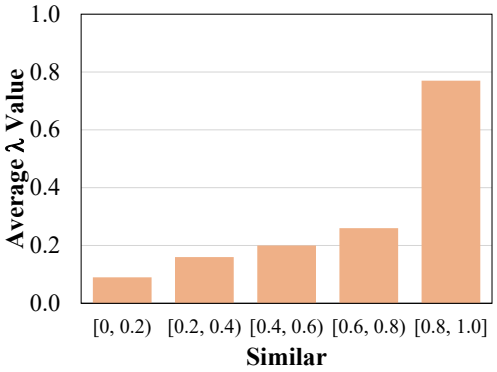

Figure 4: Average λ in different ranges of reference Token-KNN probability. We compute the Token-KNN probability and its corresponding λ value by Policy-KNN for each words in the reference.

Related Work

Online Learning from Human Feedback. Online updating (Álvaro Peris and Casacuberta 2019; Turchi et al. 2017; Weng et al. 2019) the NMT model has been proved to be an effective and efficient method to adapt the model to the target domain. Kothur, Knowles, and Koehn (2018) proposed the single-sentence adaptation, which performed online adaptation one sentence at a time. Simultaneously, they introduced the dictionary adaptation to address the problem of translating novel words. Karimova, Simianer, and Riezler (2018); Domingo et al. (2019b, 2020) presented a user study to prove the advantages of online adaptation. Domingo et al. (2019a) applied online adaptation technology in a production environment. Mueller and Lal (2019) proposed to fine-tune the model over a subset of similar sentences extracted from the training set.

However, these methods require updating the model frequently, which brings a non-negligible computation cost in the human-in-the-loop translation scenario, and is prone to catastrophic forgetting problems.

Translation Memory. The NMT model augmented with the translation memory (Kuang et al. 2021; Bapna and Firat 2019) followed the generate-then-retrieve manner. Tu et al. (2018) proposed to retrieve samples similar to the source sentence and then encoded retrieved source-target pairs using key-value memory networks. Cai et al. (2021) proposed to store the target-language sentence in the memory and used a cross-lingual retrieval model. Cao and Xiong (2018); Cao, Kuang, and Xiong (2019) designed gating mechanisms to balance the impact of the translation memory. Zhang et al. (2018a) up-weighted the NMT output with the retrieved n-gram. Bulte and Tezcan (2019); Xu, Crego, and Senellart (2020) augmented NMT training data with fuzzy TM matches. Xia et al. (2019) packed the TM into a graph and using the attention mechanisms to integrating the TM representation into the NMT model. However, these methods need to modify the architecture carefully to leverage the retrieved similar sentences, for example, adding extra components into the decoder of the NMT model to encoding and incorporate the retrieved sentences.

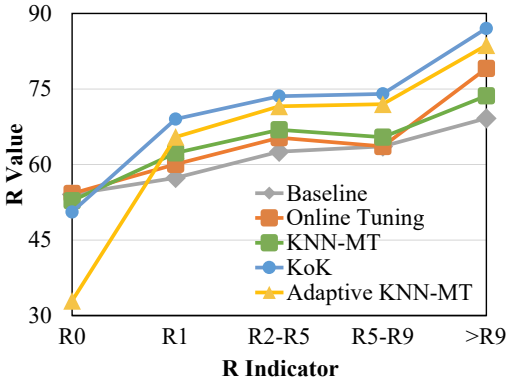

Figure 5: Results on documents with $[200, 500]$ bucket from EMEA dataset for the proposed R-indicator.

A series of research incorporated the knowledge into NMT systems through a non-parametric method. Khandelwal et al. (2019) proposed KNN-LM, augmented language model with the retrieved similar sentences using a KNN model. Khandelwal et al. (2020) generated the translation with the nearest neighbor classifier over a large datastore of cached examples. Zheng et al. (2021a) proposed to dynamically determine the number of K for each target token. Zheng et al. (2021b) further extended the capability of KNN-MT on unsupervised domain adaptation setting.

Conclusion

In this paper, we propose KNN-over-KNN (KoK), a plug-and-play non-parametric method for online learning from human feedback. KoK introduces two KNN modules, one to memorize the corrected sentences by the human and the other to balance the importance of the in-domain corrected sentences and the general-domain pretrain NMT model. In the experiments, our method significantly improves the performance and achieves better performance than state-of-the-art baselines on one-shot or few-shot learning for domain-specific lexical items. As KoK achieves competitive performance compared with existing online learning methods, applying our approach to other generation tasks is a promising direction for future work.

Acknowledgements

We would like to thank the anonymous reviewers for their insightful comments. This work is supported by National Science Foundation of China (No. U1836221, 6217020152) and National Key R&D Program of China (2018YFB1403202). The work was done when the first author was an intern at Alibaba Group.

References

Álvaro Peris; and Casacuberta, F. 2019. Online Learning for Effort Reduction in Interactive Neural Machine Translation. arXiv:1802.03594.

Bapna, A.; and Firat, O. 2019. Non-Parametric Adaptation for Neural Machine Translation. In *NAACL-HLT*, 1921–1931.

Bulte, B.; and Tezcan, A. 2019. Neural Fuzzy Repair: Integrating Fuzzy Matches into Neural Machine Translation. In *ACL*, 1800–1809.

Cai, D.; Wang, Y.; Li, H.; Lam, W.; and Liu, L. 2021. Neural Machine Translation with Monolingual Translation Memory. arXiv:2105.11269.

Cao, Q.; Kuang, S.; and Xiong, D. 2019. Learning to reuse translations: Guiding neural machine translation with examples. *arXiv preprint arXiv:1911.10732*.

Cao, Q.; and Xiong, D. 2018. Encoding Gated Translation Memory into Neural Machine Translation. In *EMNLP*, 3042–3047.

Domingo, M.; García-Martínez, M.; Estela, A.; Bié, L.; Helle, A.; Peris, Á.; Casacuberta, F.; and Herranz, M. 2019a. Demonstration of a neural machine translation system with online learning for translators. *arXiv preprint arXiv:1906.09000*.

Domingo, M.; García-Martínez, M.; Peris, A.; Helle, A.; Estela, A.; Bié, L.; Casacuberta, F.; and Herranz, M. 2019b. Incremental adaptation of NMT for professional post-editors: A user study. *arXiv preprint arXiv:1906.08996*.

Domingo, M.; García-Martínez, M.; Peris, Á.; Helle, A.; Estela, A.; Bié, L.; Casacuberta, F.; and Herranz, M. 2020. A User Study of the Incremental Learning in NMT. In *EAMT*, 319–328.

Freitag, M.; Foster, G. F.; Grangier, D.; Ratnakar, V.; Tan, Q.; and Macherey, W. 2021. Experts, Errors, and Context: A Large-Scale Study of Human Evaluation for Machine Translation. *ArXiv*, abs/2104.14478.

Gu, J.; Wang, Y.; Cho, K.; and Li, V. O. 2018. Search engine guided neural machine translation. In *AAAI*, volume 32.

Hassan, H.; Aue, A.; Chen, C.; Chowdhary, V.; Clark, J.; Federmann, C.; Huang, X.; Junczys-Dowmunt, M.; Lewis, W.; Li, M.; Liu, S.; Liu, T.-Y.; Luo, R.; Menezes, A.; Qin, T.; Seide, F.; Tan, X.; Tian, F.; Wu, L.; Wu, S.; Xia, Y.; Zhang, D.; Zhang, Z.; and Zhou, M. 2018. Achieving Human Parity on Automatic Chinese to English News Translation. *ArXiv*, abs/1803.05567.

Johnson, J.; Douze, M.; and Jégou, H. 2017. Billion-scale similarity search with GPUs. *arXiv preprint arXiv:1702.08734*.

Karimova, S.; Simianer, P.; and Riezler, S. 2018. A User-Study on Online Adaptation of Neural Machine Translation to Human Post-Edits. arXiv:1712.04853.

Khandelwal, U.; Fan, A.; Jurafsky, D.; Zettlemoyer, L.; and Lewis, M. 2020. Nearest Neighbor Machine Translation. In *ICLR*.

Khandelwal, U.; Levy, O.; Jurafsky, D.; Zettlemoyer, L.; and Lewis, M. 2019. Generalization through Memorization: Nearest Neighbor Language Models. In *ICLR*.

Kothur, S. S. R.; Knowles, R.; and Koehn, P. 2018. Document-Level Adaptation for Neural Machine Translation. *ACL 2018*, 64.

Kuang, S.; Yu, H.; Luo, W.; and Wang, Q. 2021. Translation Memory Guided Neural Machine Translation. *OpenReview*.

Läubli, S.; Sennrich, R.; and Volk, M. 2018. Has Machine Translation Achieved Human Parity? A Case for Document-level Evaluation. In *EMNLP*.

Mueller, A.; and Lal, Y. K. 2019. Sentence-Level Adaptation for Low-Resource Neural Machine Translation. In *Proceedings of the 2nd Workshop on Technologies for MT of Low Resource Languages*, 39–47.

Ng, N.; Yee, K.; Baevski, A.; Ott, M.; Auli, M.; and Edunov, S. 2019. Facebook FAIR's WMT19 News Translation Task Submission. In *Proceedings of the Fourth Conference on Machine Translation (Volume 2: Shared Task Papers, Day 1)*, 314–319.

Ott, M.; Edunov, S.; Baevski, A.; Fan, A.; Gross, S.; Ng, N.; Grangier, D.; and Auli, M. 2019. fairseq: A Fast, Extensible Toolkit for Sequence Modeling. In *NAACL-HLT: Demonstrations*.

Papineni, K.; Roukos, S.; Ward, T.; and Zhu, W.-J. 2002. Bleu: a Method for Automatic Evaluation of Machine Translation. In *ACL*, 311–318.

Post, M. 2018. A Call for Clarity in Reporting BLEU Scores. In *Proceedings of the Third Conference on Machine Translation: Research Papers*, 186–191.

Sennrich, R.; Haddow, B.; and Birch, A. 2016a. Improving neural machine translation models with monolingual data. *ACL*.

Sennrich, R.; Haddow, B.; and Birch, A. 2016b. Neural Machine Translation of Rare Words with Subword Units. In *ACL*, 1715–1725.

Simianer, P.; Wuebker, J.; and DeNero, J. 2019. Measuring Immediate Adaptation Performance for Neural Machine Translation. In *NAACL*, 2038–2046.

Snover, M.; Dorr, B.; Schwartz, R.; Micciulla, L.; and Makhoul, J. 2006. A Study of Translation Edit Rate with Targeted Human Annotation. In *AMTA*, 223–231.

Steinberger, R.; Pouliquen, B.; Widiger, A.; Ignat, C.; Erjavec, T.; Tufiş, D.; and Varga, D. 2006. The JRC-Acquis: A Multilingual Aligned Parallel Corpus with 20+ Languages. In *LREC*.

Tiedemann, J. 2009. News from OPUS-A Collection of Multilingual Parallel Corpora with Tools and Interfaces. In *Recent Advances in Natural Language Processing*, 237–248.

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

Tu, Z.; Liu, Y.; Shi, S.; and Zhang, T. 2018. Learning to Remember Translation History with a Continuous Cache. *TACL*, 6: 407–420.

Turchi, M.; Negri, M.; Farajian, M. A.; and Federicoa, M. 2017. Continuous Learning from Human Post-Edits for Neural Machine Translation. *The Prague Bulletin of Mathematical Linguistics*, (108): 233–244.

Vaswani, A.; Shazeer, N.; Parmar, N.; Uszkoreit, J.; Jones, L.; Gomez, A. N.; Kaiser, Ł.; and Polosukhin, I. 2017. Attention is all you need. In *NIPS*, 5998–6008.

Weng, R.; Zhou, H.; Huang, S.; Li, L.; Xia, Y.; and Chen, J. 2019. Correct-and-Memorize: Learning to Translate from Interactive Revisions. In *IJCAI*, 5255–5263.

Xia, M.; Huang, G.; Liu, L.; and Shi, S. 2019. Graph Based Translation Memory for Neural Machine Translation. *AAAI*, 33(01): 7297–7304.

Xu, J.; Crego, J.; and Senellart, J. 2020. Boosting Neural Machine Translation with Similar Translations. In *ACL*, 1580–1590.

Zhang, J.; Utiyama, M.; Sumita, E.; Neubig, G.; and Nakamura, S. 2018a. Guiding Neural Machine Translation with Retrieved Translation Pieces. In *NAACL*, 1325–1335.

Zhang, Z.; Liu, S.; Li, M.; Zhou, M.; and Chen, E. 2018b. Joint Training for Neural Machine Translation Models with Monolingual Data. In *AAAI*.

Zheng, X.; Zhang, Z.; Guo, J.; Huang, S.; Chen, B.; Luo, W.; and Chen, J. 2021a. Adaptive Nearest Neighbor Machine Translation. In *ACL-IJCNLP*.

Zheng, X.; Zhang, Z.; Huang, S.; Chen, B.; Xie, J.; Luo, W.; and Chen, J. 2021b. Non-Parametric Unsupervised Domain Adaptation for Neural Machine Translation. In *EMNLP*.