Empirical Assessment of kNN-MT for Real-World Translation Scenarios

Pedro Henrique Martins[*1], **João Alves**[*1],
Tânia Vaz[1], **Madalena Gonçalves**[1], **Beatriz Silva**[1], **Marianna Buchicchio**[1],
José G. C. de Souza[1], **André F. T. Martins**[1,2,3]
[1]Unbabel, Lisbon, Portugal,
[2]Instituto de Telecomunicações, Lisbon, Portugal
[3]Instituto Superior Técnico, University of Lisbon, Portugal

Abstract

This paper aims to investigate the effectiveness of the k-Nearest Neighbor Machine Translation model (kNN-MT) in real-world scenarios. kNN-MT is a retrieval-augmented framework that combines the advantages of parametric models with non-parametric datastores built using a set of parallel sentences. Previous studies have primarily focused on evaluating the model using only the BLEU metric and have not tested kNN-MT in real-world scenarios. Our study aims to fill this gap by conducting a comprehensive analysis on various datasets comprising different language pairs and different domains, using multiple automatic metrics and expert-evaluated Multidimensional Quality Metrics (MQM). We compare kNN-MT with two alternate strategies: fine-tuning all the model parameters and adapter-based fine-tuning. Finally, we analyze the effect of the datastore size on translation quality, and we examine the number of entries necessary to bootstrap and configure the index.

1 Introduction

The remarkable advances in neural models have brought significant progress in the field of machine translation (Sutskever et al., 2014; Bahdanau et al., 2015; Vaswani et al., 2017). However, current systems rely heavily on a fully-parametric approach, where the entire training data is compressed into the model parameters. This can lead to inadequate translations when encountering rare words or sentences outside of the initial training domain (Koehn and Knowles, 2017), requiring several stages of fine-tuning to adapt to data drift or to new domains.

By combining the advantages of parametric models with non-parametric databases built from parallel sentences, retrieval-augmented models showed to be a promising solution, particularly in domain adaptation scenarios (Gu et al., 2018; Zhang et al., 2018; Bapna and Firat, 2019; Meng et al., 2021; Zheng et al., 2021; Jiang et al., 2021; Martins et al., 2022a; Martins et al., 2022b).

One notable example is the k-Nearest Neighbor Machine Translation model (kNN-MT) (Khandelwal et al., 2021), known for its simplicity and very promising results. The model first creates a token-level datastore using parallel sentences, and then it retrieves similar examples from the database during inference, enhancing the generation process via interpolation of probability distributions.

However, despite its potential, the kNN-MT model has yet to be tested in real-world scenarios. Previous studies have primarily focused on evaluating it using only the BLEU metric, which correlates poorly with human judgments. In order to gain a deeper understanding of when and how kNN-MT can be effective, we conduct a thorough analysis on various datasets which comprise 4 different language pairs and 3 different domains, using BLEU (Papineni et al., 2002; Post, 2018), COMET (Rei et al., 2020), and Multidimensional Quality Metrics (MQM) – quality assessments obtained from the identification of error spans in translation outputs by experts (Lommel et al., 2014; Freitag et al., 2021).

To sum up, our main contributions are:

[*]Equal contribution.
Contact: {pedro.martins, joao.alves}@unbabel.com.

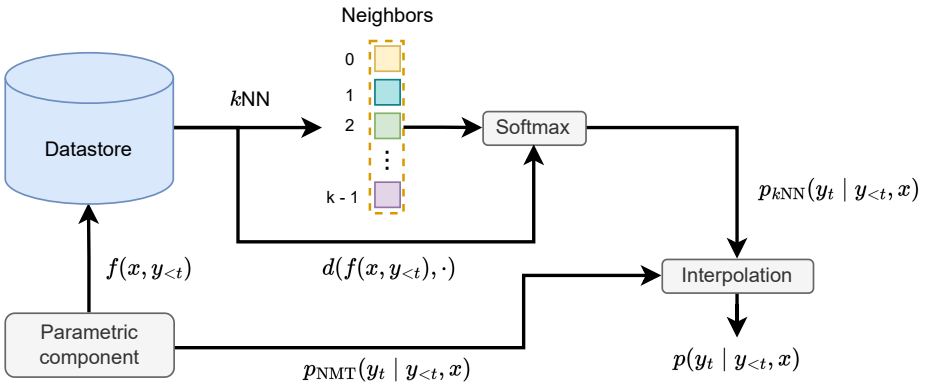

Figure 1: Diagram of the kNN-MT model.

- We compare using kNN-MT with directly using a pre-trained multilingual model, fine-tuning all the model parameters, and with adapter-based fine-tuning, reporting results in several automatic metrics.
- We analyze the effect of the datastore size on the quality of kNN-MT's translations and examine the number of entries necessary to bootstrap and configure the datastore's index.
- We perform MQM evaluation of the translations generated by a pre-trained model with and without retrieval, and by a fully fine-tuned model with and without retrieval.

2 k-Nearest Neighbor Machine Translation

In machine translation, the goal is to take a sentence or document in a source language, represented as $\boldsymbol{x} = [x_1, \ldots, x_L]$, and generate a corresponding translation in a target language, represented as $\boldsymbol{y} = [y_1, \ldots, y_N]$. This is typically achieved using a fully-parametric sequence-to-sequence model (Sutskever et al., 2014; Bahdanau et al., 2015; Vaswani et al., 2017). In these models, the encoder takes in the source sentence and outputs a set of hidden states. The decoder then generates the target translation one token at a time by attending to these hidden states and outputting a probability distribution over the vocabulary for each step, $p_{\text{NMT}}(y_t|\boldsymbol{y}_{<t}, \boldsymbol{x})$. Finally, a search procedure, such as beam search (Reddy, 1977), is applied using these probability distributions to generate the final translation.

The k-nearest neighbor machine translation model (kNN-MT) (Khandelwal et al., 2021), illustrated in Figure 1, is a retrieval-augmented model. It combines a standard sequence-to-sequence model as the one described above, with an approximate nearest neighbor retrieval mechanism, that allows the model to access a datastore of examples at inference time.

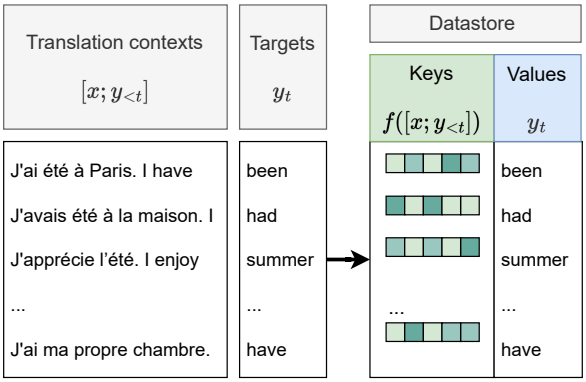

Figure 2: Diagram of the kNN-MT datastore.

2.1 Building the Datastore

Building kNN-MT's datastore, $\mathcal{D}$, requires a parallel corpus, $\mathcal{S}$, with the desired source and target languages, process illustrated in Figure 2. The datastore is a key-value memory, where each key is the decoder's output representation of the context (source and ground-truth translation until current step), $\boldsymbol{f}(\boldsymbol{x}, \boldsymbol{y}_{<t}) \in \mathbb{R}^d$. The value is the corresponding target token $y_t \in \mathcal{V}$:

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

Therefore, to construct the datastore, we simply need to perform force-decoding on the parallel corpus $\mathcal{S}$ and store the context vector representations and their corresponding ground-truth target tokens.

	Source	Reference
En-Tr	The Company has a 65+ year track record in supplying high quality pharmaceutical products across oral solid and liquid forms.	Şirket, oral katı ve sıvı formlarda yüksek kaliteli ilaç ürünleri tedarikinde 65 yılı aşkın geçmişe sahiptir.
En-Ko	A South Korean detective looks into the reason for his counterparts visit.	남한의 형사는 그가 남한에 파견된 이유를 알아내고자 한다.
En-De (1)	When I track your order it seems like it is lost in transit, I am so sorry about this.	Wenn ich Ihre Bestellung schicke, scheint es, als ob sie beim Versandverfahren verloren gegangen ist. Es tut mir sehr leid.
En-De (2)	I have put the request in to cancel the order.	Ich habe um eine Stornierung der Bestellung gebeten.
En-Fr	Sorry to hear about your domains, you can move them, so we can look at that together.	Désolé d'apprendre ce qui s'est passé pour vos domaines, vous pouvez les déplacer, afin que nous puissions examiner cela ensemble.

Table 1: Datasets translation examples.

2.2 Searching for k-NN

To find the closest examples in the datastore, the standard approach is to use a library for efficient similarity search such as FAISS (Johnson et al., 2019) to perform k-nearest neighbor search. To do this, a searchable index that encapsulates the datastore vectors must first be created. Since exact kNN search is computationally expensive, an approximate kNN search is performed by segmenting the datastore. This can be done by defining Voronoi cells in the d-dimensional space, which are defined by a centroid, and assigning each datastore key to one of these cells using k-means clustering (MacQueen, 1967). Then, during inference, the model searches the index hierarchically to approximately retrieve the set of k nearest neighbors $\mathcal{N}$.

2.3 Combining kNN with the NMT model

After retrieving the k nearest neighbors, we need a way to leverage this information. In kNN-MT this is done by computing a probability distribution based on the neighbors' values, which is then combined with the parametric component's distribution, at each step of the generation.

The retrieval distribution, $p_{k\text{NN}}(y_t|\boldsymbol{y}_{<}\, t, \boldsymbol{x})$, is calculated using the neighbors' distance to the current decoder's output representation, $d(\boldsymbol{f}(\boldsymbol{x}, \boldsymbol{y}_{<}\, t), \cdot)$:

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

where T is the softmax temperature, k_j denotes the key of the j^{th} neighbor and v_j its value.

Finally, the retrieval distribution, $p_{\text{NMT}}(y_t|\boldsymbol{y}_{<t}, \boldsymbol{x})$ and the parametric component distribution, $p_{k\text{NN}}(y_t|\boldsymbol{y}_{<t}, \boldsymbol{x})$, are combined, by performing interpolation, to obtain the final distribution, which is used to generate the translation through beam search:

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

where $\lambda \in [0, 1]$ is a hyperparameter that controls the weight given to the two distributions. This interpolation allows the model to benefit from the strengths of both the parametric component and the retrieval component.

3 Experimental Settings

In order to analyze how kNN-MT performs in real-world scenarios, we performed experiments using datasets from several domains and different language pairs (as described in §3.1). We compared the results with that of a pre-trained multilingual model (referred to as the base model; see §3.2), fine-tuning all the parameters of the base model (as discussed in §3.3), and using adapter-based fine-tuning (as described in §3.4). The specific settings of kNN-MT are detailed in §3.5 and the automatic metrics employed are described in §3.6.

3.1 Datasets

In our experiments, we use 5 proprietary datasets across 4 different language pairs: English-Turkish (En-Tr), English-Korean (En-Ko), English-German (En-De (1) and En-De (2)), and English-French (En-Fr). The En-Tr and En-Ko datasets are composed of sentences related to press

	En-Tr			En-ko			En-De (1)			En-De (2)			En-Fr		
	k	λ	T	k	λ	T	k	λ	T	k	λ	T	k	λ	T
kNN-MT	16	0.4	10	16	0.5	10	4	0.5	100	4	0.5	10	4	0.6	10
Fine-tuned (Adapters) + kNN-MT	16	0.3	10	16	0.3	10	4	0.3	100	8	0.3	10	8	0.4	10
Fine-tuned (Full) + kNN-MT	8	0.5	100	4	0.3	10	4	0.3	10	16	0.2	100	16	0.3	1

Table 2: Hyperparameters values: number of neighbors k, interpolation coefficient λ, and retrieval softmax temperature T.

releases and media descriptions, respectively. The En-De (1), En-De (2) and En-Fr datasets belong to the customer service domain. We provide some translation examples in Table 1 as well as the data splits for each dataset in Table 3.

	Train set	Validation Set	Test set
En-Tr	10,281	944	492
En-Ko	197,945	973	496
En-De (1)	10,599	1000	2000
En-De (2)	556,972	1000	2000
En-Fr	1,353,257	1000	2000

Table 3: Number of sentences in each dataset split.

3.2 Base Model

The mBART50 model (Tang et al., 2020) serves as the base model for our study. Its "one-to-many" variation is pre-trained to translate English into 49 other languages, including the languages used in our study. The model architecture is a transformer-based encoder-decoder, with 12 layers in the encoder, 12 layers in the decoder, a hidden layer dimension of 1024 and 16 heads, encompassing a total of approximately 610 million parameters. It was first trained on a denoising task using monolingual data from 25 languages (mBART; (Liu et al., 2020)), and then further pre-trained on a larger set of monolingual data from 50 languages. It was then fine-tuned on parallel data for all 50 languages to adapt the model to the machine translation task.

3.3 Fine-tuning

We compare applying kNN-MT with fine-tuning all the base model parameters. To do so, we fine-tune the base model for each dataset, using its training set, the Adam optimizer with a learning rate of 3×10^{-5}, a batch size of 16, and gradient accumulation of 8 steps. We perform early stopping on the validation set, with a patience of 5 checkpoints, being the validation step computed every 100 steps for the En-Tr and En-De (1) datasets, every 500 steps for the En-Ko dataset, and every 1000 steps for the En-De (2) and En-Fr datasets.

3.4 Adapter-based Fine-tuning

We also explore the use of adapter-based fine-tuning as a method of light-weight adaptation. Adapters (Houlsby et al., 2019) are small residual layers inserted into the middle of a pre-trained model and are used to adapt the model to a new task, in this case, adapting the model to the dataset's domain. As it is possible to incorporate adapters corresponding to different datasets to the same model, this method is an efficient solution in terms of model parameters, since we only need to save one set of parameters for multiple datasets. For each domain we add adapters with 12.5M parameters, approximately 2% of the total number of parameters of the pretrained model (610M). To implement it, we employ the same hyper-parameters and training settings as previously described in the methodology section for fine-tuning the entire model. This allows a fair comparison of the effectiveness of adapter-based fine-tuning versus traditional fine-tuning methods.

3.5 kNN-MT

For the kNN-MT we build the token-based datastores using the training sets' parallel sentences. To set the parameters for kNN-MT, we conduct a grid search on the validation set for the interpolation coefficient λ, the temperature T, and the number of retrieved neighbors k. The grid search is performed on $\lambda \in \{0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8\}$, $T \in \{1, 10, 100\}$, and $k \in \{4, 8, 16\}$. The chosen values for each dataset are listed in Table 2. To perform the kNN search, we use the FAISS library (Johnson et al., 2019) with the IVFPQ index and set the number of centroids to 2000, the code size to 64, and perform the search over 32 partitions.

3.6 Automatic Metrics

To evaluate the model we use two automatic metrics: BLEU (Papineni et al., 2002; Post, 2018) – n-gram matching based metric – and COMET (Rei et al., 2020) – metric based on fine-tuned pre-trained language models.

	En-De (1)		En-De (2)		En-Fr		Average	
	BLEU	COMET	BLEU	COMET	BLEU	COMET	BLEU	COMET
Base Model	42.6	0.534	38.0	0.492	49.1	0.716	43.2	0.581
kNN-MT	48.0	0.668	49.2	0.673	71.2	0.945	56.1	0.762
Fine-tuned (Adapters)	53.2	0.737	53.9	0.720	**78.9**	1.009	62.0	0.822
Fine-tuned (Full)	53.5	0.742	52.4	0.720	76.8	1.004	61.5	0.822
Fine-tuned (Adapters) + kNN-MT	53.2	0.748	**54.7**	**0.724**	78.5	**1.014**	**62.1**	**0.829**
Fine-tuned (Full) + kNN-MT	**54.1**	**0.751**	53.2	**0.724**	77.5	1.011	61.6	**0.829**

Table 4: BLEU and COMET scores on the English-German and English-French customer-service test sets.

	En-Tr		En-Ko		Average	
	BLEU	COMET	BLEU	COMET	BLEU	COMET
Base Model	24.5	0.672	7.9	0.273	16.2	0.473
kNN-MT	31.1	0.857	19.2	0.545	25.2	0.701
Fine-tuned (Adapters)	33.8	0.912	20.9	0.574	27.4	0.743
Fine-tuned (Full)	35.7	0.931	23.0	0.612	29.4	0.772
Fine-tuned (Adapters) + kNN-MT	35.1	0.927	22.6	0.597	28.9	0.762
Fine-tuned (Full) + kNN-MT	**36.2**	**0.956**	**24.0**	**0.626**	**30.1**	**0.791**

Table 5: BLEU and COMET scores on the English-Turkish and English-Korean test sets.

4 Results with Automatic Metrics

We report the results of our experiments using automatic metrics in Tables 4 and 5, which we discuss in the following sections.

4.1 Does kNN-MT improve the base model's performance?

When comparing the performance of kNN-MT to the base model (mBART50) using automatic metrics, we see that kNN-MT leads to significant improvements in all datasets. Specifically, by retrieving examples from a datastore, kNN-MT results in an average increase of 12.9 BLEU points and 0.181 COMET points for the customer service datasets, and 9 BLEU points and 0.228 COMET points for the En-Tr and En-Ko datasets.

4.2 Is kNN-MT better than fine-tuning?

When comparing with fine-tuning all the model parameters or performing adapter-based fine-tuning (using each dataset's training data), kNN-MT falls short, according to the automatic metrics. However, MQM evaluation leads to different conclusions, as we will see in §5.

On average, for the customer-service datasets, kNN-MT results in a decrease of 5.9 BLEU points and 0.060 COMET points compared to adapter-based fine-tuning and of 5.4 BLEU points and 0.060 COMET points compared to fine-tuning the entire model. For the remaining datasets, kNN-MT shows an average decrease of 2.2 BLEU points and 0.042 COMET points compared to adapter-based fine-tuning and of 4.2 BLEU points and 0.071 COMET points compared to full fine-tuning. Despite these findings, applying kNN-MT can be computationally cheaper, since it reduces the need to fine-tune the model, and avoids having different models (or adapters) for each dataset.

4.3 Does kNN-MT improve fine-tuned model performance?

Applying kNN-MT to fine-tuned models results in small improvements. On customer-service datasets, it increases BLEU by 0.1 points and COMET by 0.007 points compared to adapter-based fine-tuning and fine-tuning the entire model. On other datasets, kNN-MT shows an average increase of 1.5 BLEU points and 0.019 COMET points compared to adapter-based fine-tuning, and 0.7 BLEU points and 0.019 COMET points compared to fine-tuning the entire model.

4.4 How does the datastore size influences the translation quality?

We analyzed the effect of the number of entries in the datastore on the translation quality of the model by using the base model (mBART50) extended with kNN-MT on the En-De (2) and En-Fr test sets. We calculated the COMET score for different datastore sizes and plotted the results in Figure 3. The results show that, for both datasets, as the number of entries in the datastore increases, the COMET score also improves. The rate of improvement is steepest for small datastore sizes but still present as the size increases. Additionally, we ob-

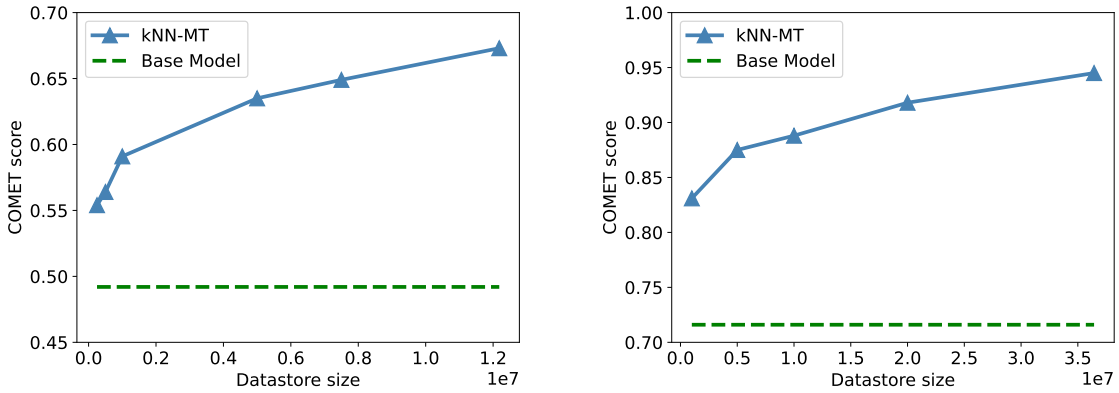

Figure 3: COMET scores when varying the number of entries on the datastore for the En-De (2) and En-Fr datasets, respectively.

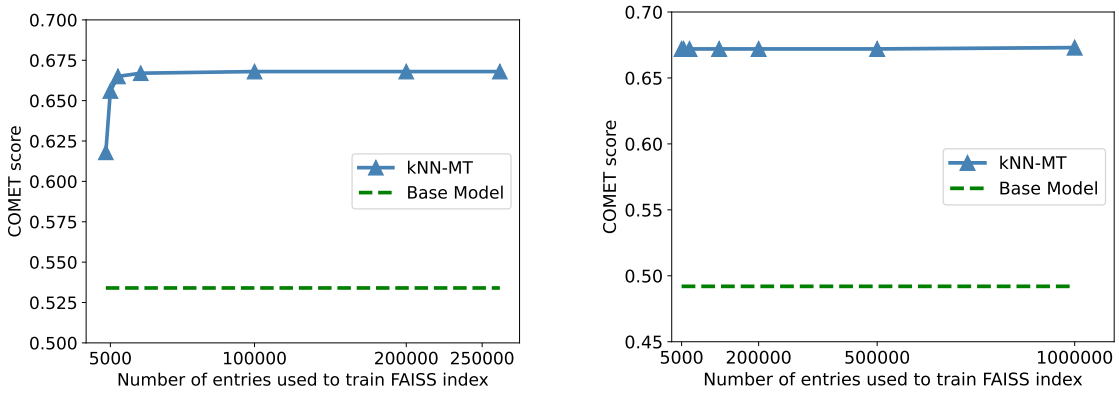

Figure 4: COMET scores when varying the number of entries used to train the FAISS index for the En-De (1) and En-De (2) datasets, respectively.

served that even using small datastores (250,000 and 1,000,000 entries for the En-De (2) and En-Fr datasets) already leads to a substantial improvement when compared to the base model.

4.5 How many entries are needed to train datastore index?

We also investigated the optimal number of entries to use for training the FAISS index for hierarchical approximate k-nearest neighbor search. We evaluated the performance of the kNN-MT model on the En-De (1) and En-De (2) datasets by measuring the COMET score using different numbers of entries for training the index. The results, as shown in Figure 4, indicate that a relatively small number of entries is sufficient for achieving the best COMET scores. For example, in the left plot, we can see that using only 2,000 or 5,000 entries leads to a reduction in COMET score, but increasing the number of entries to 10,000 results in a similar score as using the entire number of entries (261,669). Similarly, in the right plot, we see that even when using only 5,000 entries, the translation quality is already comparable to using the entire number of entries (1,000,000). This suggests that it is possible to create a datastore and train its index with a limited amount of data, and then add more entries as more data becomes available.

5 Results with MQM Assessments

To complement this analysis, we evaluated the performance of the pre-trained model with and without retrieval, as well as the fully fine-tuned model with and without retrieval using Multidimensional Quality Metrics (MQM) – quality assessments obtained from the identification of error spans in translation outputs (Lommel et al., 2014; Freitag et al., 2021). To conduct this assessment, we had professional linguists assessing the models' translations for the En-Ko, En-De (2), and En-Fr test sets. We asked the annotators to identify all errors and independently label them with an error category (accuracy, fluency, and style, each with a specific set of subcategories) and a severity level (neutral, minor, major, and critical).

Table 6 presents the MQM results while Fig-

	En-De (2)				En-Fr				En-Ko			
	MINOR	MAJOR	CRITICAL	MQM	MINOR	MAJOR	CRITICAL	MQM	MINOR	MAJOR	CRITICAL	MQM
Base Model	1301	896	439	61.24	499	237	266	88.42	713	185	28	75.23
kNN-MT	928	417	75	86.22	**335**	**116**	137	93.77	527	**95**	6	85.72
Fine-tuned	982	471	72	85.03	377	131	**3**	**97.14**	513	101	**3**	85.56
Fine-tuned + kNN-MT	**800**	**391**	**62**	**88.03**	363	118	5	96.87	**466**	99	5	**85.97**

Table 6: Error severity counts and MQM scores.

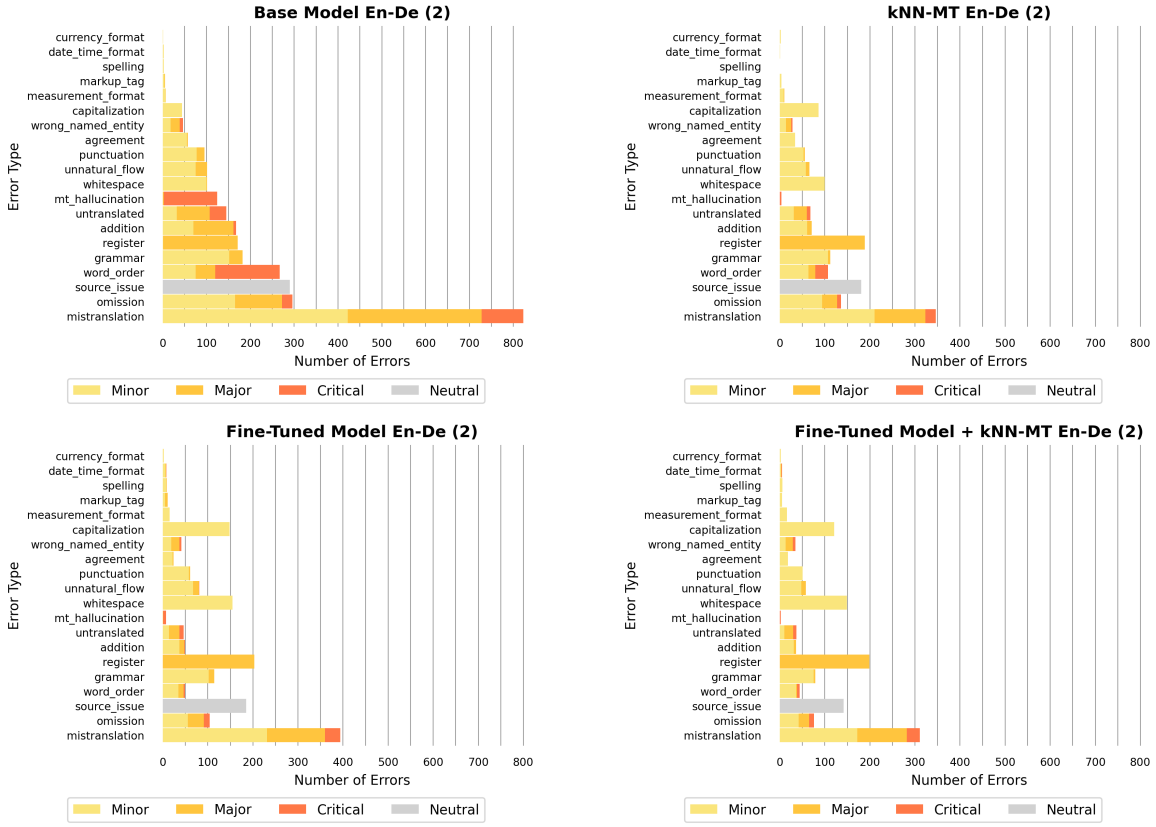

Figure 5: Error typology and severity level breakdown for the En-De (2) test set.

ures 5, 6, and 7 provide a breakdown of the error typology distribution. The MQM assessment indicates that both fine-tuning and kNN-MT significantly improve translation performance when compared to the base model, resulting in a substantial increase in MQM score and a notable reduction in critical, major, and minor errors. Interestingly, according to the MQM scores and in contrast to the automatic metric scores, kNN-MT slightly outperforms fine-tuning in two out of the three datasets. Moreover, in the customer service datasets (En-Fr and En-De (2)), kNN-MT proved to be useful in mitigating source sentence errors, which are prevalent in this domain and can adversely impact the translation quality (Gonçalves et al., 2022). Additionally, combining kNN-MT with fine-tuning results in marginal improvements for two datasets.

6 Related Work

In recent years, retrieval-augmented models have gained attention for their effectiveness in various text generation tasks. One such model is the k-nearest neighbor language model (kNN-LM; (Khandelwal et al., 2019)), which combines a parametric model with a retrieval component. Other works have proposed methods to integrate the retrieved tokens using gating mechanisms (Yogatama et al., 2021) or cross-attention (Borgeaud et al., 2021), and techniques to improve the efficiency of the kNN-LM by performing datastore pruning, adaptive retrieval (He et al., 2021) and adding pointers to the next token on the original corpus to the datastore entries (Alon et al., 2022). Retrieval-augmented models have also been explored in the field of machine translation. Ear-

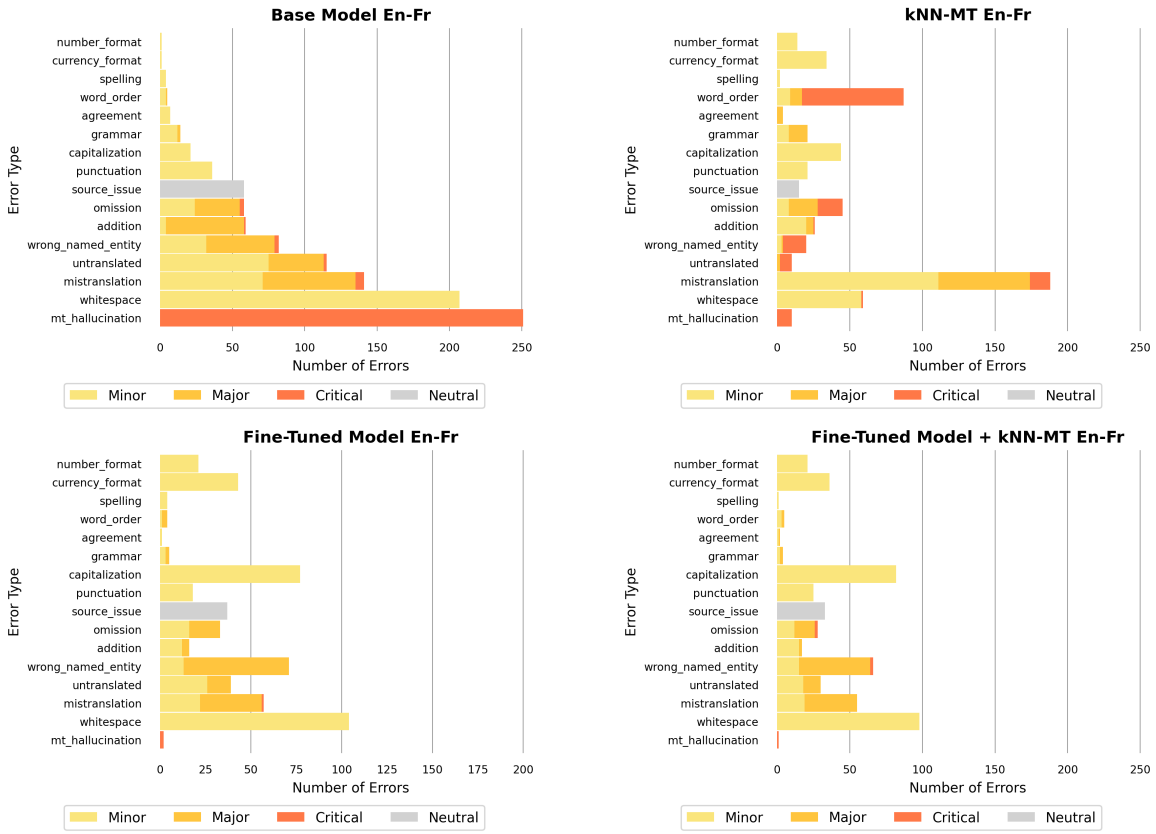

Figure 6: Error typology and severity level breakdown for the En-Fr test set.

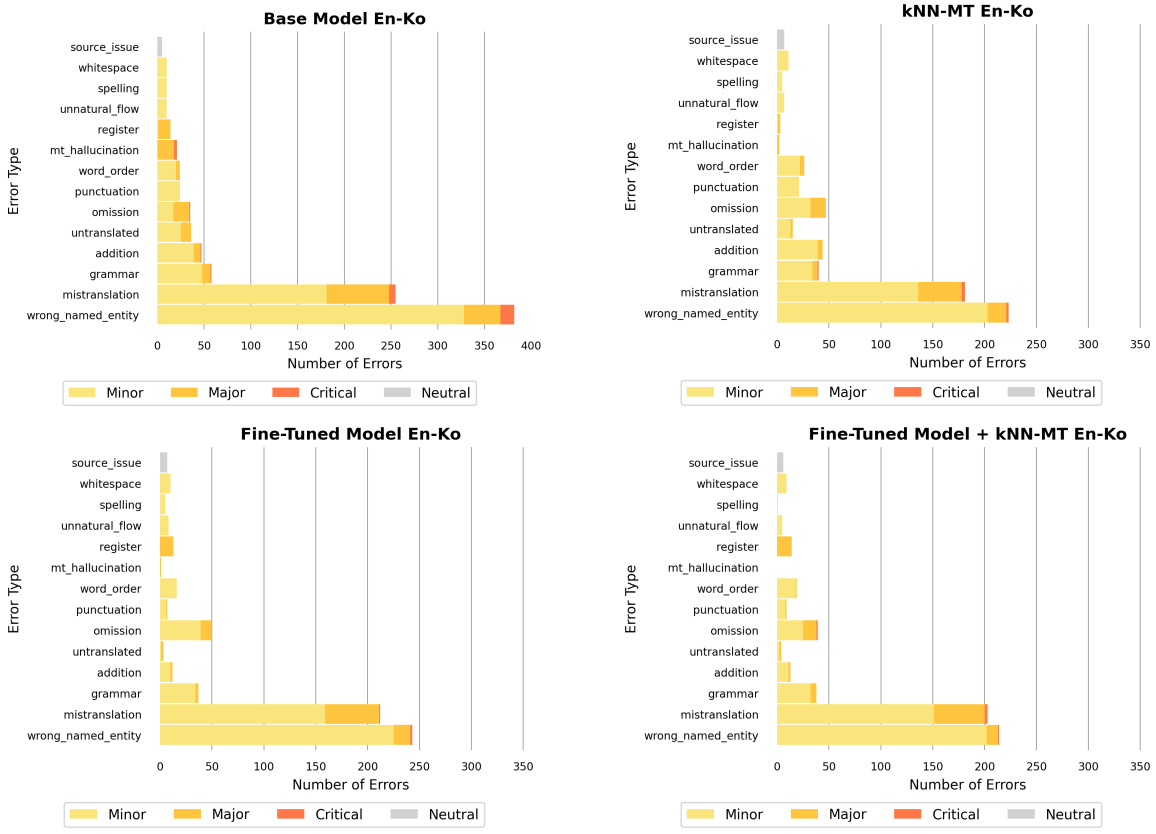

Figure 7: Error typology and severity level breakdown for the En-Ko test set.

lier works have proposed using a search engine to retrieve similar sentence pairs and incorporating them through shallow and deep fusion (Gu et al., 2018) or attention mechanisms (Bapna and Firat, 2019), or retrieving n-grams to up-weight token probabilities (Zhang et al., 2018). More recently, the kNN-MT model has been proposed as an adaptation of the kNN-LM for machine translation (Khandelwal et al., 2021), and was then extended with a network that determines the number of retrieved tokens to consider (Zheng et al., 2021). As kNN-MT can be up to two orders of magnitude slower than a fully-parametric model, (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., 2022a) proposed efficient kNN-MT by adapting the methods introduced by (He et al., 2021) to machine translation and introducing a retrieval distributions cache to speed-up decoding. (Martins et al., 2022b) proposed retrieving chunks of tokens instead of single tokens. However, most of these methods have been evaluated on a limited number of datasets and language pairs, and using only the BLEU metric. Our paper addresses this gap by evaluating kNN-MT across five "real-world" datasets and four language pairs using COMET and MQM evaluation.

7 Conclusions

In this paper, we conducted a study to assess the performance k-Nearest Neighbor Machine Translation (kNN-MT) in real-world scenarios. To do so, we augmented a pre-trained multilingual model with kNN-MT's retrieval component and compared it against using the pre-trained model, performing fine-tuning, and doing adapter-based fine-tuning on five datasets comprising four language pairs and three different domains. The results on automatic metrics, COMET and BLEU, revealed that while kNN-MT significantly improves the translation quality over the pre-trained language model, it falls short when compared to fine-tuning and adapter-based fine-tuning. Furthermore, we observed that incorporating kNN-MT's retrieval component into a fine-tuned model resulted in small improvements. We also assessed the kNN-MT model using Multidimensional Quality Metrics (MQM) by having professional linguists evaluate the translations for the En-Ko, En-De (2), and En-Fr test sets. The MQM scores revealed a significant improvement in the kNN-MT model over the base model, with kNN-MT slightly outperforming fine-tuning in two out of three language pairs. Combining kNN-MT with a fine-tuned model resulted in minor improvements. Additionally, we analyzed the effect of the number of entries in the datastore on translation quality and the number of entries required to train the FAISS index. Our findings suggest that having larger datastores improves translation quality, with the improvement steepness being higher when increasing the size of a small datastore. The number of entries used to train the FAISS index has a small impact on the final translation quality, which is relevant when creating a dynamic datastore that can be updated when more data becomes available.

Acknowledgements

This work was supported by EU's Horizon Europe Research and Innovation Actions (UTTER, contract 101070631), by the Portuguese Recovery and Resilience Plan through project C645008882-00000055 (NextGenAI, Center for Responsible AI), and by the Fundação para a Ciência e Tecnologia through contract UIDB/50008/2020.

References

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

[Bahdanau et al.2015] Bahdanau, Dzmitry, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. In *Proc. ICLR*.

[Bapna and Firat2019] Bapna, Ankur and Orhan Firat. 2019. Non-Parametric Adaptation for Neural Machine Translation. In *Proc. NAACL*.

[Borgeaud et al.2021] Borgeaud, Sebastian, 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.

[Freitag et al.2021] Freitag, Markus, George Foster, David Grangier, Viresh Ratnakar, Qijun Tan, and Wolfgang Macherey. 2021. Experts, Errors, and Context: A Large-Scale Study of Human Evaluation for Machine Translation. *Transactions of the Association for Computational Linguistics*.

[Gonçalves et al.2022] Gonçalves, Madalena, Marianna Buchicchio, Craig Stewart, Helena Moniz, and Alon Lavie. 2022. Agent and User-Generated Content and its Impact on Customer Support MT. In *Proc. EAMT*.

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

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

[Houlsby et al.2019] Houlsby, Neil, Andrei Giurgiu, Stanislaw Jastrzebski, Bruna Morrone, Quentin De Laroussilhe, Andrea Gesmundo, Mona Attariyan, and Sylvain Gelly. 2019. Parameter-Efficient Transfer Learning for NLP. In *Proc. ICML*.

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

[Johnson et al.2019] Johnson, Jeff, Matthijs Douze, and Hervé Jégou. 2019. Billion-scale similarity search with gpus. *IEEE Transactions on Big Data*.

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

[Khandelwal et al.2021] Khandelwal, Urvashi, Angela Fan, Dan Jurafsky, Luke Zettlemoyer, and Mike Lewis. 2021. Nearest neighbor machine translation. In *Proc. ICLR*.

[Koehn and Knowles2017] Koehn, Philipp and Rebecca Knowles. 2017. Six Challenges for Neural Machine Translation. In *Proceedings of the First Workshop on Neural Machine Translation*.

[Liu et al.2020] Liu, Yinhan, Jiatao Gu, Naman Goyal, Xian Li, Sergey Edunov, Marjan Ghazvininejad, Mike Lewis, and Luke Zettlemoyer. 2020. Multilingual Denoising Pre-training for Neural Machine Translation. *Transactions of the Association for Computational Linguistics*.

[Lommel et al.2014] Lommel, Arle, Hans Uszkoreit, and Aljoscha Burchardt. 2014. Multidimensional quality metrics (MQM): A framework for declaring and describing translation quality metrics. *Revista Tradumàtica: tecnologies de la traducció*.

[MacQueen1967] MacQueen, J. 1967. Classification and analysis of multivariate observations. In *5th Berkeley Symp. Math. Statist. Probability*.

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

[Martins et al.2022b] Martins, Pedro Henrique, Zita Marinho, and André FT Martins. 2022b. Chunk-based Nearest Neighbor Machine Translation. In *Proc. EMNLP*.

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

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

[Post2018] Post, Matt. 2018. A Call for Clarity in Reporting BLEU Scores. In *Proc. Third Conference on Machine Translation*.

[Reddy1977] Reddy, Raj. 1977. Speech understanding systems: summary of results of the five-year research effort at Carnegie-Mellon University.

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

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

[Tang et al.2020] Tang, Yuqing, 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.

[Vaswani et al.2017] Vaswani, Ashish, 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*.

[Wang et al.2021] Wang, Shuhe, Jiwei Li, Yuxian Meng, Rongbin Ouyang, Guoyin Wang, Xiaoya Li, Tianwei Zhang, and Shi Zong. 2021. Faster Nearest Neighbor Machine Translation.

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

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

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