Angular Dispersion Accelerates k-Nearest Neighbors Machine Translation

Evgeniia Tokarchuk
University of Amsterdam
evgeniia@tokarch.uk

Sergey Troshin
University of Amsterdam
s.troshin@uva.nl

Vlad Niculae
University of Amsterdam
v.niculae@uva.nl

Abstract

Augmenting neural machine translation with external memory at decoding time, in the form of k-nearest neighbors machine translation (k-NN MT), is a well-established strategy for increasing translation performance. k-NN MT retrieves a set of tokens that occurred in the most similar contexts recorded in a prepared data store, using hidden state representations of translation contexts as vector lookup keys. One of the main disadvantages of this method is the high computational cost and memory requirements. Since an exhaustive search is not feasible in large data stores, practitioners commonly use approximate k-NN lookup, yet even such algorithms are a bottleneck. In contrast to research directions seeking to accelerate k-NN MT by reducing data store size or the number of lookup calls, we pursue an orthogonal direction based on the performance properties of approximate k-NN lookup data structures. In particular, we propose to encourage angular dispersion of the neural hidden representations of contexts. We show that improving dispersion leads to better balance in the retrieval data structures, accelerating retrieval and slightly improving translations.

1 Introduction

k-Nearest Neighbors Machine Translation (k-NN MT) is a promising training-free approach demonstrated to improve the translation quality of neural MT in both in- and out-of-domain setups (Khandelwal et al., 2021; Meng et al., 2022; Dai et al., 2023). However, performance improvement comes with a high decoding cost because k-NN queries require nearest-neighbor lookups in a usually large *data store*: a key-value store where tokens in the training dataset are indexed by continuous key vectors (given by the hidden states of the neural MT system when processing the context).

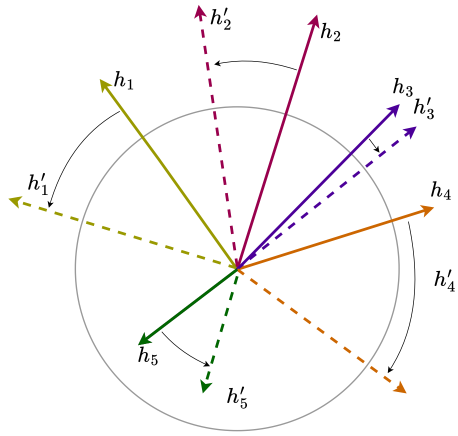

Figure 1: Angular dispersion applied to a set of vectors: dispersion spreads out the angles of the vectors, without changing their length (norm). By increasing angular dispersion of key-value data stores, we observe substantial speed-ups at the retrieval time. Here, solid lines represent vectors before dispersion, and dashed lines after dispersion.

With modern neural networks, keys tend to be high-dimensional (≥ 512), and, given the number of tokens in a data store, k-NN search requires substantial time and space. Thus, in order to search this large data collection, approximate k-NN (Johnson et al., 2019) is dominant in practice (Khandelwal et al., 2021; Martins et al., 2023; Gao et al., 2024). However, even approximate k-NN methods can not achieve a decoding speed of the baseline models out-of-the-box.

Many works focus on the efficiency of retrieval either by data store compression (He et al., 2021; Dai et al., 2023; Wang et al., 2022a; Martins et al., 2022a; Zhu et al., 2023a) or by adaptive retrieval at the decoding time (Martins et al., 2022a,b; Gao et al., 2024). Key representations have, perhaps surprisingly, received less attention, even though their geometric properties play an important role in retrieval. Wang et al. (2022b) argue that key vectors are not specific enough for fine-grained retrieval and suggest a contrastive learning approach

Findings of the Association for Computational Linguistics: EMNLP 2025, pages 14120–14132
November 4-9, 2025 ©2025 Association for Computational Linguistics

to decouple representations. Similarly, Wang et al. (2022a) show that there is a large overlap between data store keys, and that the overlap is not related to word frequency; they suggest a method based on contrastive learning that can mitigate the overlap.

Since modern data structures that allow billion-scale distance-based k-NN search (Johnson et al., 2019) rely heavily on space partitioning, the distribution of keys affects not only the quality of the retrieved samples but also plays a significant role in retrieval speed due to the cluster imbalance (Tavenard et al., 2011). Gao et al. (2019) and Ethayarajh (2019) draw attention to the clumping of the continuous representations of Transformer-based (Vaswani et al., 2017) models, *i.e.*, they show that model representations tend to occupy only a small subspace and semantically different words cluster together. Motivated by the work of Tokarchuk et al. (2025), we study distributional properties of the keys in k-NN MT, focusing on *angular dispersion* (Figure 1) . In particular, we show that higher angular dispersion of the key vectors leads to up to 5 times more efficient k-NN lookups in large (>10M) data stores. We also emphasize that angular dispersion results in an overall slight increase of the translation performance for both k-NN MT with in-domain and out-of-domain data stores.

2 Background

2.1 k-NN Machine Translation

We follow the k-NN MT model proposed by Khandelwal et al. (2021), in which a neural MT system is augmented with a non-parametric k-nearest neighbors classifier at the token level. Given a set of pairs of source and target sentences $(\mathcal{X}, \mathcal{Y})$, and a well-trained encoder-decoder model with parameters θ, let $h_\theta(x, y_{<i}) \in \mathbb{R}^d$ denote a hidden state representation of the context needed for translating the i-th output token.[1] Given a key-value data store, we associate the key $h_\theta(x, y_{<i})$ with the target y_i, for all prefixes of all pairs in the parallel dataset. This results in a data store of key-value pairs $D = \{(h_j, y_j)\}_{j=1}^N$, where N is the total number of contexts used (on the order of millions). Given some new query state h and a distance function d, the k-nearest neighbor set $N_k(h)$ is a subset of D containing exactly k pairs such that if $(h', y') \in N_k(h)$ and $(h'', y'') \in D \setminus N_k(h)$ then $d(h, h') \le d(h, h'')$. The k-nearest neighbors set can be used to induce a probabilistic classifier from h onto the vocabulary, essentially by counting each neighboring state as a "vote" toward that word, weighted by the distance:

$$p_{k\text{-NN}}(y|h) \propto \sum_{(h_j, y_j) \in N_k(h)} [\![y = y_j]\!] \exp\left(\frac{-\|h - h_j\|^2}{T} \right), \quad (1)$$

where $[\![\cdot]\!]$ is the Iverson bracket, equal to 1 or 0 depending on the truth value of its argument, $T > 0$ is a temperature parameter adjusting the flatness of the induced distribution, and $\propto$ denotes equality up to a normalization constant. When translating from the neural MT system, we interpolate its own predictive distribution with the one of the kNN classifier with weight $0 \le \lambda \le 1$:

$$\begin{aligned} p(y_i|x, y_{<i}) &:= (1 - \lambda) p_{\text{model}}(y_i|x, y_{<i}) \\ &\quad + \lambda p_{k\text{-NN}}(y_i|h_\theta(x, y_{<i}))\,. \quad (2) \end{aligned}$$

2.2 Approximate k-NN search

The efficiency bottleneck of k-NN MT is the nearest-neighbor search. For this reason, the search is typically not performed exactly; rather, it is common to employ *approximate* nearest-neighbor search, which enables larger data stores up to billions of tokens (Johnson et al., 2019). We follow the line of k-NN MT research (Khandelwal et al., 2021; Zheng et al., 2021; Meng et al., 2022; Wang et al., 2022a; Martins et al., 2022b; Deguchi et al., 2023b,a) and use the inverted file index with product quantization (IVFPQ), through the implementation in the `faiss` library (Johnson et al., 2019), which, with GPU acceleration, is state-of-the-art in terms of accuracy and speed for this application.

Inverted file index (IVF). IVF first splits the keys h into clusters by using k-means clustering algorithm (Hartigan and Wong, 1979) by learning K centroids $\mu = [\mu_1, \ldots, \mu_K]$. Given the set of learned centroids, any vector $h \in \mathbb{R}^d$ is associated with its nearest centroid as $\mu(h) = \arg\min_{\mu'} \|h - \mu'\|$. The search vector space is then divided into Voronoi cells (see Figure 2) and the data store is split into clusters $D_{\mu_i} = \{(h_j, y_j) \,|\, \mu(h_j) = \mu_i, (h_j, y_j) \in D\}$. In practice, searching within a single IVF cluster is suboptimal, and several nearest IVF clusters (probes) are searched (Johnson et al., 2019). Given a query vector, we retrieve a shortlist of nearest neighbor candidates from the top-n (nprobes) nearest Voronoi cells: $\text{IVF}(h) = \{(h_j, y_j) \,|\, \mu(h_j) \in K_{IVF}(h), (h_j, y_j) \in$

[1] Often the last hidden layer in a transformer decoder.

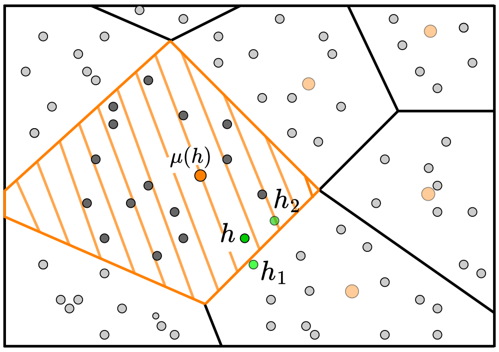

Figure 2: An illustration of the IVF index with a single probe. A query vector h is mapped to the nearest centroid $\mu(h)$. The search is then performed within the Voronoi cell of $\mu(h)$ (orange area), and then h_2 will be returned as the approximate nearest neighbor of h. With larger nprobes, it is possible to retrieve h_1, the closest neighbor of h.

$D\}$, where $K_{IVF}(h)$ denotes the set of nearest centroids given a query vector, namely probes.

Product quantization (PQ). Product quantization (Jégou et al., 2011) compresses the d-dimensional key vectors for fine-grained search. While IVF is the part of the data structure that we most target in our work, we describe the PQ stage for completeness. PQ splits the vector space into a Cartesian product of M subspaces, each d/m–dimensional: $h = [\tilde{h}_1, \ldots, \tilde{h}_M]$, and quantizes each subspace separately. For each subspace, we use k-means to learn L d/M-dimensional codewords per subspace: $C = [c_i]_{i=1}^{L}$. To quantize a key, one finds the closest codeword ids within each subspace, namely for a given subspace we search for the closest codeword using the Euclidean distance between a codeword and $\tilde{h}$ (the sub-vector of h): $\text{PQ}(\tilde{h}) = \arg\min_i \|c_i - \tilde{h}\|$. A standard practice is to represent each key vector using 8 bits.

IVFPQ. This strategy combines the inverted file index and product quantization. To store the keys, PQ compresses the residual key representations within an IVF cluster, where the residual key representation is a key vector minus the associated IVF centroid. Given a query vector, IVF first maps the query to a Voronoi cell of the nearest IVF centroid. Then, PQ performs a fine-grained search within the Voronoi cell using the stored quantized key representations.

The efficiency and accuracy of lookups in the IVFPQ data structure depends on the quality of the clustering stages, which depends in turn on the geometric distribution of the key vectors (Tavenard et al., 2011). Quantifying the properties of a point cloud that make it well-suited for IVFPQ lookup remains an open question. We propose to analyze the geometry of the *directions* of the key vectors, and, in particular, their *angular dispersion*.

2.3 Angular Dispersion

Directions in d-dimensional space can be represented as points on the sphere $\mathbb{S}_d \subset \mathbb{R}^d$. Unlike the entirety of $\mathbb{R}^d$, the sphere is compact and has many computationally attractive properties that allow us to quantify and optimize angular dispersion. Given two directions $s, s' \in \mathbb{S}_d$, their Euclidean dot product $\langle s, s' \rangle \in [-1, 1]$ corresponds to the cosine of the angle between them, and thus the angle is $\arccos \langle s, s' \rangle \in [0, \pi]$. A well-dispersed configuration $S \in (\mathbb{S}_d)^n$ is spread out uniformly over the entire sphere surface. One way to quantify dispersion is by the angle between the closest points:

$$d_{\text{min}}(S) := \min_{s \neq s' \in S} \arccos \langle s, s' \rangle. \tag{3}$$

An alternative and complementary measure of angular dispersion is spherical variance (Jammalamadaka and Sengupta, 2001; Mardia, 1975):

$$\text{svar}(S) = 1 - \left\| \frac{1}{|S|} \sum_{s \in S} s \right\|. \tag{4}$$

Spherical variance close to 1 indicates high concentration of directions, as the average will be close to the sphere surface; a value close to 0 does not necessarily imply dispersion but at least a more symmetrical configuration of the directions: for any $x \in \mathbb{S}_d$, the four-point configuration $(x, x, -x, -x)$ has both minimum angle and spherical variance zero.

Finding an optimally-dispersed configuration is known as the Tammes problem (Tammes, 1930), and it is generally not exactly tractable. Instead, it is common to define functions to optimize in order to encourage dispersion (Wang and Isola, 2020; Wang et al., 2020; Liu et al., 2018; Tokarchuk et al., 2025). A wide class of such functions are based on pairwise similarities or *kernels*. For example, the *minimum hyperspherical energy* objective is

$$R_{\text{MHE}}(S) = \frac{1}{|S|(|S|-1)} \sum_{s \neq s' \in S} k(s, s'). \tag{5}$$

where $k(s, s')$ is a kernel such as perhaps the Gaussian kernel $k(s, s') = \exp(\langle s, s' \rangle / \sigma)$.

Sliced dispersion (Tokarchuk et al., 2025; Bonet et al., 2023) is an efficient alternative which avoids the quadratic complexity by making use of the fact that optimal dispersion is trivial on a circle (on $\mathbb{S}_2 \subset \mathbb{R}^2$). In this special case, any perfectly-dispersed configuration is made up of equidistant angles and given any input set of angles, the nearest dispersed configuration can be efficiently found. Let the sum of angles between a configuration S and the nearest optimally-dispersed configuration on a circle be $\delta(S)$. If $d = 2$ we can exactly minimize $\delta(S)$. To extend to higher dimensions, we will invoke projections onto great circles. On a sphere $\mathbb{S}_d$, the great circles correspond to pairs of orthogonal directions $C(\mathbb{S}_d) := \{(p, q) : p \in \mathbb{S}_d, q \in \mathbb{S}_d, \langle p, q\rangle = 0\}$. If we denote by S_{pq} the projection of a configuration of directions S onto the great circle (p, q), sliced dispersion optimizes

$$R_{\text{sliced}}(S) := \mathbb{E}_{p,q}\left[\delta(S_{pq})\right], \qquad (6)$$

where the expectation is over the uniform distribution on $C(\mathbb{S}_d)$. In words, this objective minimizes the expected distance to an optimally-sliced configuration along any great circle. Preliminary experiments in Appendix A show that sliced dispersion regularizer converges faster in terms of spherical variance, so we use it for all further experiments.

3 Key Representation with Dispersion

Dispersion on the sphere is well-established in the literature as discussed in §3, and it is common to apply dispersion of the angles for learning hyperspherical representations, *i.e.*, magnitudes of the vectors are discarded. However, previous works show that the norm of the keys, in fact, might contain important information (Gao et al., 2024). To avoid discarding norms, contrary to other works, we apply dispersion only to the angles of the model's outputs while keeping their magnitudes intact and do gradient updates in Euclidean space. Figure 1 shows how angular dispersion operates in $2D$.

Optimization for dispersion. In the context of k-NN MT, we also have to avoid a mismatch between the data store representation and the model's output to keep the decoding step consistent. Therefore, we cannot apply dispersion on the data store alone. We fine-tune a small portion of a pretrained model to optimize for the dispersion of the model's outputs before they are saved in a data store. We estimate dispersion over all possible keys using the mini-batches.

Given a dataset of paired (parallel) sentences $(\mathcal{X}, \mathcal{Y})$ as in §2.1, we employ the standard log-likelihood machine translation loss[2] *w.r.t.* the trainable model parameters θ:

$$\mathcal{L}_{\text{MT}}(\theta) := -\sum_{(x,y)} \sum_{i=1}^{|y|} \log p(y_i | x, y_{<i}). \qquad (7)$$

In addition, given a dispersion regularizer R that acts on the sphere $\mathbb{S}_d$, we define a dispersion loss on the directions of the hidden states. Concretely, writing $H := \{h(x, y_{<i}) : (x, y) \in \mathcal{X} \times \mathcal{Y}, i \in |y|\} \subset \mathbb{R}^d$ and $\bar{H} = \{h/\|h\| : h \in H\} \subset \mathbb{S}_d$, we have

$$\mathcal{L}_{\text{Disp}}(\theta) := R(\bar{H}),$$

and we optimize their weighted sum

$$\mathcal{L}(\theta) := \mathcal{L}_{\text{MT}}(\theta) + \gamma \mathcal{L}_{\text{Disp}}(\theta), \qquad (8)$$

estimating its stochastic gradients on minibatches.

Any directional dispersion regularizer can play the part of R; we use sliced dispersion as motivated in §3.

During the entire dispersion process, most of the network parameters are kept frozen. We only fine-tune the weights of the last two feed-forward layers, layer normalization, and output projection of the final decoder block. Hence, we do not introduce any new model parameters and utilize the existing architecture. Schematically, Figure 3 shows how we update the model representations during training.

Key representation. In kNN-MT, the input to the final Transformer decoder fully-connected (FC) block is usually used as the key representation for the datastore (Khandelwal et al., 2021). In our setup (Figure 3), following Martins et al. (2023), we instead use the *decoder output* as the key, and keep this choice consistent across experiments. Preliminary tests showed a small advantage for the conventional choice (input to the final FC decoder block) in vanilla kNN-MT. However, when fine-tuning, we found that adjusting the output projection is essential for good performance. This requires fine-tuning all weights up to the output, making the conventional approach less practical as it involves more parameters. How to fine-tune for dispersion more efficiently remains an open question.

[2] In practice, we additionally use label smoothing regularization (Szegedy et al., 2016) with $\epsilon = 0.1$.

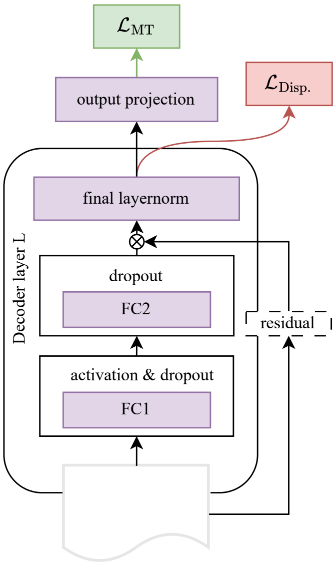

Figure 3: Fine-tuning of the NMT model for dispersion of outputs. Blocks in purple are trained, and the rest of the parameters are frozen.

4 Experimental Setup

4.1 Synthetic Data

To more directly control the effect of dispersion, we generate a family of synthetic data stores. Specifically, we sample 10M random vectors of dimensionality 128 from the mixture of 5 power spherical distributions (De Cao and Aziz, 2020). The scale parameter κ for the power spherical distribution defines the *concentration* of the data points, where higher κ means higher concentration. We build a data store using the keys generated with $\kappa = \{1, 10, 50, 100, 1000\}$ with 2048 centroids while picking a random length for each key in the $[1, 100]$ range.

4.2 MT Datasets and Evaluation

We provide empirical results on two different languages with in-domain and out-of-domain data stores. Below is the description of the datasets and evaluation criteria.

- WMT16 `ro-en` news translation dataset[3] with 612k training samples. We apply `sentencepiece`[4] tokenization (Kudo, 2018).
- Multi-domain `en-de` dataset (Aharoni and Goldberg, 2020). We prepare data similar to Khandelwal et al. (2021), applying BPE (Sennrich et al., 2016) and data filtering based on sentence length.[5] The number of training samples varies by domain: `Medical` (206K), `Law` (450K), `IT` (180K), and `Koran` (15K).
- WMT19 `en-de` dataset provided by HuggingFace.[6] We use this dataset only for fine-tuning `de-en` translation model. It contains 34M filtered `de-en` news data.

We evaluate the translation accuracy on the best checkpoint according to the validation BLEU score using SacreBLEU (Papineni et al., 2002; Post, 2018) and COMET (Rei et al., 2020) on `newstest2016` for `ro-en` and test sets of four domains (`Medical`, `Law`, `IT` and `Koran`) for `de-en`. We do all experiments using `fairseq` framework (Ott et al., 2019).

4.3 Models

We compare translation quality and decoding speed across the following models:

- **Baseline (base).** We train a transformer-base model (Vaswani et al., 2017) for `ro-en`, with additional linear projection of the decoder outputs to dimensionality 128. We deliberately choose this dimensionality since the preliminary results show a performance increase compared to a model with output dimensionality 512 (see Appendix C). For `de-en`, we use the pre-trained model (Ng et al., 2019) with the dimensionality of the decoder output 1024.
- **Baseline fine-tuned with the dispersion objective (base-D).** For both `ro-en` and `de-en` we fine-tune the baseline model as shown in Figure 3. All layers, except the last two fully connected layers, layer normalization of the last decoder block, and an output projection, are frozen during fine-tuning. For `de-en` fine-tuning, we use WMT 19 training set, and for `ro-en` we use WMT 16 training set. We fine-tuned all models with 5k optimization steps. We choose γ equal to 1 for all experiments and use the sliced dispersion regularizer with one great circle per batch.
- **Vanilla k-NN MT with (k-NN-D) and without (k-NN) dispersion.** These models are obtained without additional training, by combining the aforementioned models (base and

[3] https://www.statmt.org/wmt16/

[4] `https://github.com/google/sentencepiece`

[5] `https://github.com/urvashik/knnmt/blob/master/examples/translation/prepare-domadapt.sh`

[6] `https://huggingface.co/datasets/wmt/wmt19`

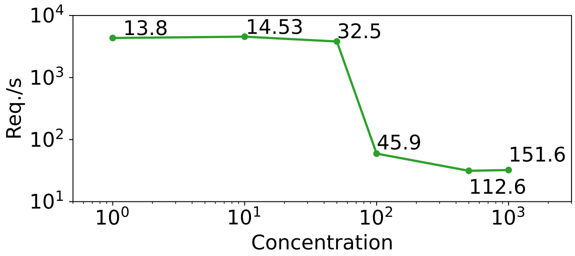

Figure 4: The number of requests per second (req./s) for the data stores generated with different concentration parameters κ. Higher concentration means lower dispersion. We also show the imbalance factor for each data store above the req./s value. Notably, increasing or decreasing dispersion past certain points does not influence speed much, but a certain critical range of concentration makes a substantial difference when crossed.

base-D) with k-NN interpolation at test time (Khandelwal et al., 2021). The data store is built from the model checkpoint with best development BLEU.

More details of the model training and inference can be found in Appendix B.

4.4 Data Store Construction and Search

To build a data store from a NMT model, we extract the hidden states from the last decoder feed-forward layer $h^{(L)}$. As described in §2.2, we use `faiss` with IVFPQ index (Johnson et al., 2019) and train the index on 1M random samples.[7] For all models, we use 2048 IVF centroids and retrieve according to the squared Euclidean distance.

5 Results

5.1 Synthetic Results

We first verify our hypothesis in a controlled experiment. Using the collection of synthetic data stores indexed by concentration (§4.1), we measure the time to retrieve $k = 8$ nearest neighbors for 10K random queries with batch size 10 and nprobe 32. Figure 4 shows that the data store lookup speed (measured as the number of queries per second) correlates negatively with the dispersion.

5.2 In-domain Results

First, we analyze fine-tuning dynamics of the model with sliced regularizer described in §3 and without dispersion regularizer. We only report spherical variance, as the number of keys is highly large, and minimum distance has quadratic complexity. Figure 5 shows that fine-tuning the model with the sliced regularizer increases the spherical variance value to a large extent, compared to the fine-tuning without the dispersion regularizer.

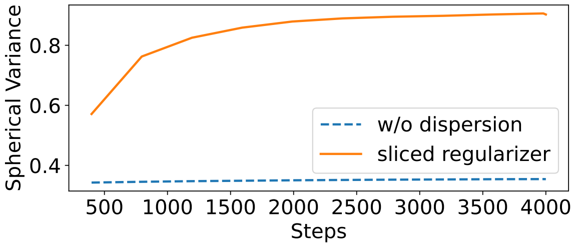

Figure 5: Fine-tuning dynamics of `ro-en` models in terms of spherical variance. A step is a gradient update on the effective batch size described in Appendix B.

Table 1 shows translation results for base and base-D models, both with and without k-NN. We also vary the number of centroids to probe between 32 and 8. The total amount of keys stored in the data store is equal to 21M. Alongside the translation scores, we report the number of tokens processed in one second (tok/s).

model	#pr.	BLEU$_{(\uparrow)}$	COMET$_{(\uparrow)}$	tok/s$_{(\uparrow)}$
base	-	31.5	78.95	75
base-D	-	31.7	78.96	79
k-NN	32	32.4	79.89	12
k-NN	8	32.2	79.69	28
k-NN-D	32	32.6	79.91	53
k-NN-D	8	**32.6**	**79.93**	**63**

Table 1: `ro-en` translation scores on `newstest16` test set. tok/s calculated as a median over three runs.

k-NN MT decoding with dispersion (base-D) performs as well as the base k-NN MT model while being about 5 times faster. The speedup agrees with the synthetic results, and so we attribute it to the properties of the IVFPQ index. To verify this hypothesis, we conduct an analysis of the data store properties in §6 and establish a connection to the NMT model performance.

We provide additional results with embeddings dimensionality equal to 512 in Appendix C.

5.3 Domain Adaptation

Following the line of k-NN MT works (Khandelwal et al., 2021; Gao et al., 2024; Meng et al., 2022), we provide results for five different domains for `de-en`, comparing the models with and without dispersion. As shown in Table 2, a model with dispersion can achieve slightly faster data store lookup and over-

[7] For data stores with less than $1M$ elements, we use all elements.

all better performance on all domains. The main difference between the two setups is that all de-en keys have larger dimension (1024 instead of 128), and the data store size, except for Law, is much smaller than for ro-en.

Our experiments also show that domain adaptation k-NN MT is robust against decreasing the number of clusters to probe, while without dispersion, we can see a performance drop both in BLEU and COMET. The most pronounced lookup speed improvements can be seen on the Law domain since it has the largest k-NN data store out of the four domains. Note that we fine-tuned for dispersion using only WMT 19 training data and did not use any domain-specific data. We measure average tok/s over 10 random subsets, each consisting of 200 sentences drawn uniformly with replacement from the test set. We report the median measurement.

Translation analysis. While studying the difference in the system outputs using compare-mt (Neubig et al., 2019), we do not see many systematic patterns that can plausibly explain the performance gain of the k-NN MT-D model. We find that sentence length is predicted more accurately in the dispersed model and sentence-level quality increases for short sentences. We leave further investigation to future work.

6 Analysis and Data Store Geometric Properties

Empirically, dispersion has a positive effect in terms of lookup speed for the synthetic and k-NN MT experiments. To further explore how dispersion affects the IVFPQ search, we measure how it affects the properties of the IVF clusters.

Dimensionality. Intuitively, clumping occurs more often when dimensionality is relatively small. Previous studies on dispersion (Tokarchuk et al., 2025) and our experiments discussed in §5.2 and §5.3 show that higher dimensionality alleviates clumping problem to some extent. However, increasing dimensionality does not necessarily leads to better performance (*e.g.*, ro-en model with output dimensionality 128 performs overall better than the same model with output dimensionality 512, as per Table 1 and Table 7), but it often incurs additional computation and memory costs. While having larger dimensionality showed to be useful in large-scale setups (), with k-NN MT such scaling is non-trivial given the size of resulting data store. However, there is a performance and speed gain even with higher dimensionality as per Table 2. Therefore, dispersion might be a good and relatively cheap tool that helps balancing clusters in the data store, and as a result improve retrieval speed with no additional handcrafted optimizations. We do believe that our method can be successfully applied in scenarios where large amount of points have to be distributed in high-dimensional space.

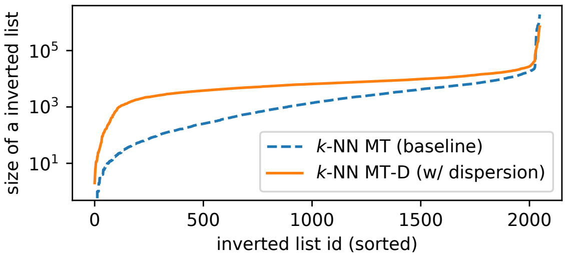

Figure 6: Dispersion tends to reduce the variance of IVF cluster sizes.

Distribution of IVF cluster sizes. We investigate whether dispersion improves the distribution of cluster sizes, which is known to have a positive effect on the retrieval speed and leads to a more predictable retrieval time (Tavenard et al., 2011). In the Figure 6, we compare the distributions of the IVF cluster sizes for the k-NN MT and the k-NN MT-D data stores. To numerically quantify the effect, we report the imbalance factor (Tavenard et al., 2011). Given a partitioning of a data store elements into K clusters with sizes N_i, and the total size $N = \sum_{i=1}^{K} N_i$, the imbalance factor is:

$$\text{IF} = K \sum_{i=1}^{K} \left(\frac{N_i}{N} \right)^2 . \qquad (9)$$

We observe that dispersion improves the imbalance factor (IF), as observed from Table 3 and Figure 4.

Number of clusters to probe. Next, we are asking whether dispersion improves the quantization accuracy such that we need fewer IVF probes to obtain the same quality.

To complement the analysis above, we also propose the expected number of probes (ENP) metric that aims to reflect the number of probes that is enough to obtain high-quality retrieval. First, given a vector h, we rank the IVF centroids $\mu = [\mu_i]_{i=1}^{K}$, by the distance from the h: $R_h(\mu_i) \in \{1, \ldots, K\}$. Next, given k nearest neighbors $[h_i]_{i=1}^{k} \in K(h)$, and their corresponding IVF centroids ranks $\mu_h =$

model	#pr.	Medical (5.7M)			Law (18.4M)			IT (3.1M)			Koran (0.5M)		
		BLEU	COMET	tok/s	BLEU	COMET	tok/s	BLEU	COMET	tok/s	BLEU	COMET	tok/s
base	-	40.4	83.05	91	45.8	85.24	90	37.6	82.07	70	16.9	72.48	89
base-D	-	40.5	83.30	86	46.0	85.37	92	38.3	82.45	82	17.1	72.58	76
k-NN MT	32	55.0	84.41	55	61.0	87.03	17	45.1	83.13	51	20.8	72.38	62
k-NN MT	8	54.6	84.37	63	60.7	86.87	36	44.6	83.12	55	20.1	71.44	63
k-NN MT-D	32	**55.2**	**84.54**	62	**61.5**	**87.07**	41	**45.8**	**83.54**	55	**21.4**	**72.58**	62
k-NN MT-D	8	55.1	84.47	66	61.3	87.01	47	45.8	83.49	58	21.3	72.27	63

Table 2: Translation quality (BLEU/COMET scores) on different domains for de-en alongside the tok/s decoding speed. The #npr column shows the number of probes used during approximate k-NN retrieval (§2.2).

$[\mu(h_i)]_{i=1}^{k}$, we estimate the ENP metric:

$$\text{ENP} = \mathbb{E}_{h \sim \mathcal{D}} \max_{h' \in K(h)} R_h(\mu(h')). \qquad (10)$$

ENP takes values from 1 to K, where 1 means 1 is the perfect score (meaning we need 1 probe to find k-NNs). We use the approximate search with 32 probes to estimate the reference $[h_i]_{i=1}^{k}$.

For the ro-en data stores, the ENP metric for k-NN MT is 6.296 ± 0.667, while for k-NN MT-D the ENP is 3.290 ± 0.215; for de-en (Law) data stores, we observe a marginal improvement in ENP (1.59 ± 0.056 vs 1.5 ± 0.049). The improvements in ENP are more pronounced for the ro-en data stores. We attribute this to the difference in key dimensionality and the amount of training data used.

Clustering algorithm evaluation. Further, we are asking whether dispersion improves other clustering metrics. Namely, we look at the commonly used clustering metrics, namely *homogeneity*, *completeness*, and *v-measure* (Rosenberg and Hirschberg, 2007) implemented via scikit-learn (Pedregosa et al., 2011). Motivated by the precision, recall, and f-measure (Van Rijsbergen, 1979), homogeneity, completeness, and v-measure estimate the quality of a clustering given keys and classes (labels), where we use the next tokens v from the vocabulary V. *Homogeneity* measures if all of its clusters contain only data points that are members of a single class; *completeness* measures if all the data points that are members of a given class are elements of the same cluster; and *v-measure* is the harmonic mean between homogeneity and completeness. For the definitions, we refer the reader to appendix D.

As we show in Table 3, the clustering performance improves for the dispersed model. In particular, homogeneity and v-measure improve for both ro-en and de-en data stores by a large margin, while completeness improves for the ro-en data store.

metric	ro-en		de-en	
	w/o disp.	w/ disp.	w/o disp.	w/ disp.
homogeneity (↑)	51.54	71.98	67.56	74.06
completeness (↑)	44.41	73.90	70.27	70.11
v-measure (↑)	61.39	70.16	68.89	72.03
IF (↓)	68.34	11.12	27.73	7.79

Table 3: Clustering metrics for in-domain ro-en k-NN MT and Law de-en k-NN MT data stores with and without dispersion. Dispersion improves homogeneity and v-measure for both ro-en and de-en data stores.

Symmetry of IVF keys. One of the ways to quantify the balance of the representation space is to measure how symmetrical the space is, *i.e.*, we are asking whether the dispersion helps to increase the symmetry of the key space. We measure the central symmetry as the Euclidean norm of the mean vector over the data store. Namely, if the norm of the mean vector is low, the vector representations must be well-balanced relative to the origin. Given a set of keys h, sampled from the data store $\mathcal{D}$, we estimate $\|\mathbb{E}[h]\|$. For the ro-en data stores, k-NN MT $\|\mathbb{E}[h]\| = 68.35$, and k-NN MT-D $\|\mathbb{E}[h]\| = 11.12$; for de-en (Law) data stores, k-NN MT $\|\mathbb{E}[h]\| = 6.03$, and k-NN MT-D $\|\mathbb{E}[h]\| = 2.91$, which indicates that dispersion improves the symmetry of the vector representations.[8]

6.1 Sensitivity to Dispersion Weight

In order to verify that speed-up and performance gains are the positive effect of dispersion rather than additional fine-tuning, we vary the value of γ in Equation (8) where $\gamma = 0$ is equivalent to fine-tuning with only cross-entropy loss. Results in Table 4 that positive values of gamma lead to improvements in both speed and translation quality, while being not sensitive to the specific value of γ.

[8] Note that symmetry around zero implies zero mean, but zero mean does not necessarily imply symmetry.

γ	Medical (5.7M)		Law (18.4M)		IT (3.1M)	
	BLEU	tok/s	BLEU	tok/s	BLEU	tok/s
-	53.8	53.2	60.4	16.2	42.8	48.5
0	53.9	46.7	60.5	27.5	43.0	48.2
1	54.2	51.2	60.7	40.4	43.0	53.0
100	53.7	53.7	61.4	46.6	43.3	55.0

Table 4: BLEU scores on different domains for de-en development sets alongside the tok/s decoding speed with various values of γ.

7 Related Work

In the literature, there are two main lines of work addressing the decoding speed of k-NN MT. One of them is related to the creation of the data store and includes methods such as pruning and dimensionality reduction. Another direction of the k-NN MT improvements lies in the decoding-time improvement, including caching and adaptive retrieval. He et al. (2021) discuss various pruning strategies (including greedy merge pruning) and dimensionality reduction in the context of k-NN language models. Martins et al. (2022a) apply similar techniques in the context of NMT domain adaptation with k-NN. Zhu et al. (2023a) propose to prune data stores based on a local correctness metric, which is defined as the maximum number of correct predictions in the neighborhood of a specific key. Wang et al. (2022a) and Meng et al. (2022) build a source-side small data store to avoid searching across all possible target contexts. Dai et al. (2023) also use a small data store and perform sentence-based retrieval. They incorporate results into the NMT model using an adapter network.

Zhang et al. (2022) also show that there are redundant features in the keys and that a compression network can be learned for efficient search, while Zhu et al. (2023b) use adapter and alignment loss to learn low-dimensional keys representation.

To improve the decoding speed at the decoding time, Martins et al. (2022b) introduced chunk-based retrieval that retrieves multiple tokens at once instead of one token at a time. Martins et al. (2022a) use a caching mechanism in decoding time and a simple decision mechanism for skipping retrieval from the data store based on a predicted interpolation value λ rather than a static λ as in the original k-NN MT model (Khandelwal et al., 2021). Similar in nature, Gao et al. (2024) propose to use a classifier to decide when to skip retrieval from the k-NN data store and introduce a timestep aware k-NN MT threshold λ.

In contrast, the study of the key representation properties is limited. Wang et al. (2022a) analyze the distributions of the keys and show that even unrelated words can be seen in the same clusters, which can negatively affect the distance-based retrieval. They train adapter network with contrastive loss to promote more separable representations and introduce compression. Similarly Wang et al. (2022b) show that context representation from the NMT model is suboptimal for the k-NN MT retrieval and more fine-grained representation can be obtained via training small adapter network with a contrastive loss approach. In our work, we show that a similar goal can be achieved with a simple regularizer without the need for negative samples. Also, we show that we can achieve speed up even without compression.

Orthogonal to the speed-up of the k-NN MT is the design choice of the k-NN search index itself that affects retrieval speed greatly. In our work, we have focused on IVF-PQ since it is an index that that is well established and validated in the k-NN MT literature and methodology (Khandelwal et al., 2021; Martins et al., 2022a; Gao et al., 2024). Our method has direct impact on improving IVF lookup of the index rather then improving search over quantized vectors (PQ). Using such methods as Optimized Product Quantization (OPQ, Ge et al., 2014) might further improve search speed.

8 Conclusion

In this work, we show that angular dispersion of the data store benefits the retrieval speed and performance in k-NN MT for both in-domain and domain adaptation scenarios. Our analysis of cluster properties and synthetic data stores further indicates that dispersion of the keys balances the data store clustering, accelerating the approximate search, especially for larger data store sizes. Our findings indicate that fine-tuning with dispersion can be a first efficient k-NN MT component. Since there are no changes in architecture or k-NN data store construction, we hypothesize that other efficient methods that rely on distance metrics can be easily applied on top, which we leave for future work.

Limitations

Limited study of dimensionality effect. Dispersion and dimensionality have a tight connection with each other. Specifically, larger dimensionality naturally allows for large dispersion but induces greater costs for storage and retrieval. However, in this work, we focus on three different choices of dimensionality: 128, 512 and 1024. The dimensionality of modern LLMs is typically on the larger side, and it remains a question to what extent we can see the benefits of dispersion in application to language modeling, particularly at a larger scale.

Search Metric. Similar to previous works, we use squared Euclidean distance for the k-NN retrieval. Wang et al. (2022b) specified in their work that using the same metric as in fine-tuning leads to better performance. We use angular dispersion, and it is possible that resorting to angular-based distances, such as cosine similarity, during pretraining and fine-tuning will lead to better performance of the dispersed models.

Data store Index Design Our analysis relies on a specific index design with a fixed number of IVF centroids. Among all parameters, the number of centroids plays a key role in determining data store lookup speed. In addition, optimized product quantization can further accelerate index search. In this work, however, we do not explore modifications to the data store index, and leave this direction for future investigation.

Acknowledgments

We thank Caio Corro and the UvA Language Technology Lab, especially Yang Meng, Maya Nachesa, Di Wu and Melika Davoodzadeh for discussion and feedback on the manuscript. This work is supported by the Dutch Research Council (NWO) via VI.Veni.212.228. The authors also thank SURF (www.surf.nl) for the support in using the National Supercomputer Snellius.

References

Roee Aharoni and Yoav Goldberg. 2020. Unsupervised domain clusters in pretrained language models. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Association for Computational Linguistics.

Clément Bonet, Paul Berg, Nicolas Courty, François Septier, Lucas Drumetz, and Minh Tan Pham. 2023. Spherical Sliced-Wasserstein. In *The Eleventh International Conference on Learning Representations*.

Yuhan Dai, Zhirui Zhang, Qiuzhi Liu, Qu Cui, Weihua Li, Yichao Du, and Tong Xu. 2023. Simple and scalable nearest neighbor machine translation. In *The Eleventh International Conference on Learning Representations*.

Nicola De Cao and Wilker Aziz. 2020. The power spherical distrbution. *Proceedings of the 37th International Conference on Machine Learning, INNF+*.

Hiroyuki Deguchi, Hayate Hirano, Tomoki Hoshino, Yuto Nishida, Justin Vasselli, and Taro Watanabe. 2023a. knn-seq: Efficient, extensible knn-mt framework. *Preprint*, arXiv:2310.12352.

Hiroyuki Deguchi, Taro Watanabe, Yusuke Matsui, Masao Utiyama, Hideki Tanaka, and Eiichiro Sumita. 2023b. Subset retrieval nearest neighbor machine translation. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 174–189, Toronto, Canada. Association for Computational Linguistics.

Kawin Ethayarajh. 2019. How contextual are contextualized word representations? Comparing the geometry of BERT, ELMo, and GPT-2 embeddings. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 55–65, Hong Kong, China. Association for Computational Linguistics.

Jun Gao, Di He, Xu Tan, Tao Qin, Liwei Wang, and Tieyan Liu. 2019. Representation degeneration problem in training natural language generation models. In *International Conference on Learning Representations*.

Yan Gao, Zhiwei Cao, Zhongjian Miao, Baosong Yang, Shiyu Liu, Min Zhang, and Jinsong Su. 2024. Efficient k-nearest-neighbor machine translation with dynamic retrieval. In *Findings of the Association for Computational Linguistics: ACL 2024*, pages 7990–8001, Bangkok, Thailand. Association for Computational Linguistics.

Tiezheng Ge, Kaiming He, Qifa Ke, and Jian Sun. 2014. Optimized product quantization. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 36(4):744–755.

John A. Hartigan and Manchek A. Wong. 1979. Algorithm AS 136: A k-means clustering algorithm. *Journal of the Royal Statistical Society. Series C (Applied Statistics)*, 28(1):100–108.

Junxian He, Graham Neubig, and Taylor Berg-Kirkpatrick. 2021. Efficient nearest neighbor language models. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 5703–5714, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

S Rao Jammalamadaka and Ambar Sengupta. 2001. *Topics in circular statistics*. Series on multivariate analysis; vol. 5. World Scientific, Singapore.

Jeff Johnson, Matthijs Douze, and Hervé Jégou. 2019. Billion-scale similarity search with GPUs. *IEEE Transactions on Big Data*, 7(3):535–547.

Hervé Jégou, Matthijs Douze, and Cordelia Schmid. 2011. Product quantization for nearest neighbor search. *IEEE Trans. Pattern Anal. Mach. Intell.*, 33(1):117–128.

Urvashi Khandelwal, Angela Fan, Dan Jurafsky, Luke Zettlemoyer, and Mike Lewis. 2021. Nearest neighbor machine translation. In *International Conference on Learning Representations*.

Diederik P. Kingma and Max Welling. 2014. Auto-Encoding Variational Bayes. In *2nd International Conference on Learning Representations, ICLR 2014, Banff, AB, Canada, April 14-16, 2014, Conference Track Proceedings*.

Taku Kudo. 2018. Subword regularization: Improving neural network translation models with multiple subword candidates. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 66–75, Melbourne, Australia. Association for Computational Linguistics.

Weiyang Liu, Rongmei Lin, Z. Liu, Lixin Liu, Zhiding Yu, Bo Dai, and Le Song. 2018. Learning towards minimum hyperspherical energy. In *Neural Information Processing Systems*.

Kantilal Varichand Mardia. 1975. Statistics of directional data. *Journal of the Royal Statistical Society Series B: Statistical Methodology*, 37(3):349–371.

Pedro Martins, Zita Marinho, and Andre Martins. 2022a. Efficient machine translation domain adaptation. In *Proceedings of the 1st Workshop on Semiparametric Methods in NLP: Decoupling Logic from Knowledge*, pages 23–29, Dublin, Ireland and Online. Association for Computational Linguistics.

Pedro Henrique Martins, João Alves, Tânia Vaz, Madalena Gonçalves, Beatriz Silva, Marianna Buchicchio, José G. C. de Souza, and André F. T. Martins. 2023. Empirical assessment of kNN-MT for real-world translation scenarios. In *Proceedings of the 24th Annual Conference of the European Association for Machine Translation*, pages 115–124, Tampere, Finland. European Association for Machine Translation.

Pedro Henrique Martins, Zita Marinho, and André F. T. Martins. 2022b. Chunk-based nearest neighbor machine translation. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 4228–4245, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

Yuxian Meng, Xiaoya Li, Xiayu Zheng, Fei Wu, Xiaofei Sun, Tianwei Zhang, and Jiwei Li. 2022. Fast nearest neighbor machine translation. In *Findings of the Association for Computational Linguistics: ACL 2022*, pages 555–565, Dublin, Ireland. Association for Computational Linguistics.

Graham Neubig, Zi-Yi Dou, Junjie Hu, Paul Michel, Danish Pruthi, Xinyi Wang, and John Wieting. 2019. compare-mt: A tool for holistic comparison of language generation systems. *CoRR*, abs/1903.07926.

Nathan Ng, Kyra Yee, Alexei Baevski, Myle Ott, Michael Auli, and Sergey Edunov. 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)*, pages 314–319, Florence, Italy. Association for Computational Linguistics.

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 *Proceedings of NAACL-HLT 2019: Demonstrations*.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, pages 311–318, Philadelphia, Pennsylvania, USA. Association for Computational Linguistics.

F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. 2011. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830.

Matt Post. 2018. A call for clarity in reporting BLEU scores. In *Proceedings of the Third Conference on Machine Translation: Research Papers*, pages 186–191, Brussels, Belgium. Association for Computational Linguistics.

Ricardo Rei, Craig Stewart, Ana C Farinha, and Alon Lavie. 2020. COMET: A neural framework for MT evaluation. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 2685–2702, Online. Association for Computational Linguistics.

Andrew Rosenberg and Julia Hirschberg. 2007. V-measure: A conditional entropy-based external cluster evaluation measure. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*, pages 410–420, Prague, Czech Republic. Association for Computational Linguistics.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016. Neural machine translation of rare words with subword units. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1715–1725, Berlin, Germany. Association for Computational Linguistics.

Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jon Shlens, and Zbigniew Wojna. 2016. Rethinking the inception architecture for computer vision. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2818–2826.

Pieter Merkus Lambertus Tammes. 1930. *On the origin of number and arrangement of the places of exit on the surface of pollen-grains*. Ph.D. thesis, University of Groningen. Relation: http://www.rug.nl/ Rights: De Bussy.

Romain Tavenard, Hervé Jégou, and Laurent Amsaleg. 2011. Balancing clusters to reduce response time variability in large scale image search. In *2011 9th International Workshop on Content-Based Multimedia Indexing (CBMI)*, page 19–24, Madrid, Spain. IEEE.

Evgeniia Tokarchuk, Hua Chang Bakker, and Vlad Niculae. 2025. Keep your distance: Learning dispersed embeddings on $\mathbb{S}_m$. *Transactions on Machine Learning Research*.

C. J. Van Rijsbergen. 1979. Information retrieval. *Dept. of Computer Science, University of Glasgow*.

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 *Advances in Neural Information Processing Systems*, Red Hook, NY, USA. Curran Associates Inc.

Dexin Wang, Kai Fan, Boxing Chen, and Deyi Xiong. 2022a. Efficient cluster-based k-nearest-neighbor machine translation. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2175–2187, Dublin, Ireland. Association for Computational Linguistics.

Qiang Wang, Rongxiang Weng, and Ming Chen. 2022b. Learning decoupled retrieval representation for nearest neighbour neural machine translation. In *Proceedings of the 29th International Conference on Computational Linguistics*, pages 5142–5147, Gyeongju, Republic of Korea. International Committee on Computational Linguistics.

Tongzhou Wang and Phillip Isola. 2020. Understanding contrastive representation learning through alignment and uniformity on the hypersphere. In *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pages 9929–9939. PMLR.

Zhennan Wang, Canqun Xiang, Wenbin Zou, and Chen Xu. 2020. Mma regularization: Decorrelating weights of neural networks by maximizing the minimal angles. In *Advances in Neural Information Processing Systems*, volume 33, pages 19099–19110. Curran Associates, Inc.

Haokui Zhang, Buzhou Tang, Wenze Hu, and Xiaoyu Wang. 2022. Connecting compression spaces with transformer for approximate nearest neighbor search. In *Computer Vision – ECCV 2022*, pages 515–530, Cham. Springer Nature Switzerland.

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

Wenhao Zhu, Shujian Huang, Yunzhe Lv, Xin Zheng, and Jiajun Chen. 2023a. What knowledge is needed? towards explainable memory for kNN-MT domain adaptation. In *Findings of the Association for Computational Linguistics: ACL 2023*, pages 2824–2836, Toronto, Canada. Association for Computational Linguistics.

Wenhao Zhu, Jingjing Xu, Shujian Huang, Lingpeng Kong, and Jiajun Chen. 2023b. INK: Injecting kNN knowledge in nearest neighbor machine translation. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 15948–15959, Toronto, Canada. Association for Computational Linguistics.

A Regularizers

We compare the behavior of sliced and MHE regularizers (see §3) during the `ro-en` model fine-tuning. Figure 7 shows batched spherical variance for both regularizers on the training data. We can see that sliced requires fewer updates for convergence, and the overall spherical variance is higher. Therefore, we use the sliced regularizer for all experiments.

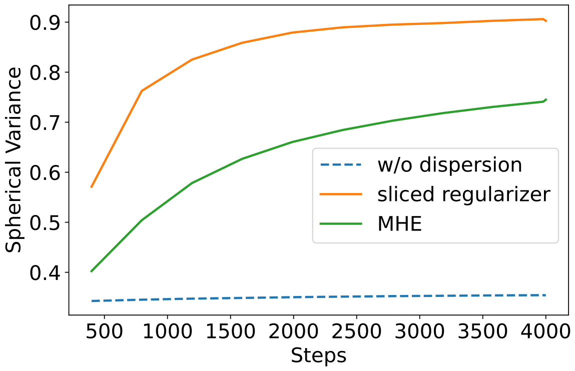

Figure 7: Comparison of the MHE and sliced regularizer.

B Training and Inference Details

Fine-tuning. We fine-tune base models with cross-entropy loss, label smoothing equal to 0.1, and sliced regularizer with γ equal to 1. We

	ro-en	en-de
# total	46M	356M
# trained	4M	51M
time	2h	20h
GPU	NVIDIA GeForce GTX TITAN X	
# GPUs	1	

Table 5: Total number of model parameters and number of trained parameters for ro-en and de-en alongside GPU model, number of GPUs and total training time.

test set	k	T	λ
ro-en newstest	8	100	0.3
de-en Law	8	10	0.8
de-en Medical	4	10	0.8
de-en IT	8	10	0.7
de-en Koran	8	100	0.7

Table 6: k-NN MT parameters.

model	**BLEU**	**COMET**	tok./sec.
base	31.2	0.7855	94.6
base-D.	31.1	0.7862	78.9
k-NN MT	31.8	**0.7937**	26.1
k-NN MT-D	**31.9**	0.7905	54.4

Table 7: BLEU and COMET on newstest16 for ro-en 512 model. k-NN MT is with the in-domain data store.

optimize for 5k steps using Adam (Kingma and Welling, 2014) optimizer with 500 warm-up steps and effective batch size 65.5K tokens. ro-en is fine-tune with learning rate $5 \cdot 10^{-5}$, de-en model is fine-tuned with lr $7 \cdot 10^{-5}$. Details of the model size and hardware set up is shown in Table 5

Decoding Speed. To make a fair comparison of the data store lookup, we use the same GPU cluster node with NVIDIA GeForce GTX TITAN X.

***k*-NN MT parameters.** k-NN MT relies on the three hyperparameters: number of retrieved neighbors (k), temperature (T), and interpolation parameter λ. Table 6 shows the exact parameters we used for each k-NN MT model.

Evaluation. We use SacreBLEU with the nrefs:1, case:mixed, eff:no, tok:13a, smooth:exp, version:2.3.1 signature; and for COMET, we use default Unbabel/wmt22-comet-da[9] model.

C Additional Results

Table 7 shows the results for ro-en models with the output dimension of 512.

D Clustering Analysis

Here define homogeneity, completeness and v-measure clustering metrics (Rosenberg and Hirschberg, 2007). To define these metrics, let us first denote the entropy and conditional entropy of clusters and next tokens:

$$H(V|K) = -\sum_{i=1}^{K}\sum_{v=1}^{|V|} \frac{n_{v,i}}{N} \log \frac{n_{v,i}}{\sum_{v=1}^{|V|} n_{v,i}}$$

$$H(V) = -\sum_{v=1}^{|V|} \frac{\sum_{i=1}^{K} n_{v,i}}{N} \log \frac{\sum_{i=1}^{K} n_{v,i}}{N}$$

$$H(K|V) = -\sum_{v=1}^{|V|}\sum_{i=1}^{K} \frac{n_{v,i}}{N} \log \frac{n_{v,i}}{\sum_{i=1}^{K} n_{v,i}}$$

$$H(K) = -\sum_{i=1}^{K} \frac{\sum_{v=1}^{|V|} n_{v,i}}{N} \log \frac{\sum_{v=1}^{|V|} n_{v,i}}{N},$$

where by $n_{v,i}$, we denote the number of data store elements with a next token v and assigned with the cluster i. Homogeneity measures if all of its clusters contain only data points which are members of a single class

$$\text{hom} = \begin{cases} 1 & \text{if } H(V) = 1 \\ 1 - \frac{H(V|K)}{H(V)} & \text{else;} \end{cases} \tag{11}$$

completeness measures if all the data points that are members of a given class are elements of the same cluster

$$\text{compl} = \begin{cases} 1 & \text{if } H(K) = 1 \\ 1 - \frac{H(K|V)}{H(K)} & \text{else;} \end{cases} \tag{12}$$

and v-measure is the harmonic mean between homogeneity and completeness, where β weights the two components (we use $\beta = 1$):

$$v\text{-measure} = \frac{(1+\beta)\,\text{hom}\,\text{compl}}{(\beta\,\text{hom}) + \text{compl}}. \tag{13}$$

[9] https://huggingface.co/Unbabel/wmt22-comet-da