Long-Tail Crisis in Nearest Neighbor Language Models

Yuto Nishida[1]   **Makoto Morishita**[2]   **Hiroyuki Deguchi**[1]
Hidetaka Kamigaito[1]   **Taro Watanabe**[1]
[1]Nara Institute of Science and Technology   [2]Future Corporation
{nishida.yuto.nu8, deguchi.hiroyuki.db0, kamigaito.h, taro}@is.naist.jp
m.morishita.pi@future.co.jp

Abstract

The k-nearest-neighbor language model (kNN-LM), one of the retrieval-augmented language models, improves the perplexity for given text by directly accessing a large datastore built from any text data during inference. A widely held hypothesis for the success of kNN-LM is that its explicit memory, i.e., the datastore, enhances predictions for long-tail phenomena. However, prior works have primarily shown its ability to retrieve long-tail contexts, leaving the model's performance remain underexplored in estimating the probabilities of long-tail target tokens during inference. In this paper, we investigate the behavior of kNN-LM on low-frequency tokens, examining prediction probability, retrieval accuracy, token distribution in the datastore, and approximation error of the product quantization. Our experimental results reveal that kNN-LM does not improve prediction performance for low-frequency tokens but mainly benefits high-frequency tokens regardless of long-tail contexts in the datastore.[1]

1 Introduction

Retrieval-augmented language model (Grave et al., 2017; Guu et al., 2018; Khandelwal et al., 2020; He et al., 2020; Borgeaud et al., 2022) is a novel paradigm for language modeling that incorporates a base language model (LM) with retrieved neighbor examples from an external datastore. These models have been proven effective by impressive language modeling performance and achieved strong results on various tasks. One of the standard models in this paradigm is the k-nearest-neighbor language model (kNN-LM; Khandelwal et al., 2020), which interpolates the output probabilities of the trained base LM at each time step with kNN probabilities computed from retrieved neighbors in the datastore. These neighbors are retrieved based on the distance

---

[1]Our code and dataset are publicly available at https://github.com/naist-nlp/knnlm-longtail-analysis.

between the contextualized embedding for each token in the test input and all key vectors in the datastore, which is constructed by caching intermediate representations for all tokens of external text data with the base LM. Therefore, kNN-LM can accurately predict the probability of each token on a given test input by directly accessing text data through the datastore during inference.

As to why retrieval-augmented language models have been successful, several studies put forward an intriguing hypothesis that the explicit use of external memory in these models enhances prediction performance, particularly for low-frequency tokens (Grave et al., 2017; Merity et al., 2017; Li et al., 2020). In line with this, Khandelwal et al. (2020) argued that direct access to the datastore is beneficial for long-tail phenomena, providing several qualitative examples. However, despite kNN-LM retrieving and incorporating the *target token*, a token to predict, during inference, the prior work has focused primarily on long-tail *context*, the sequence preceding the target token. While it is expected that kNN-LM would improve predictions for long-tail target tokens as it does for long-tail contexts, our preliminary experiments revealed no correlation between the frequency of context n-grams and the frequency of target tokens (§3.5). Thus, the behavior of kNN-LM with respect to long-tail target tokens remains unclear.

To further investigate this hypothesis, we conducted a detailed quantitative analysis, focusing specifically on long-tail target tokens. We examined prediction probability, retrieval accuracy, and distribution of contextualized token embeddings in the datastore.

Our experiments on the resplit WikiText-103 dataset, using GPT2-XL as the base LM, revealed four key findings: (1) The kNN probabilities for low-frequency tokens were lower than those of the base LM, whereas kNN probabilities for high-frequency tokens were higher (§4.1). Thus, con-
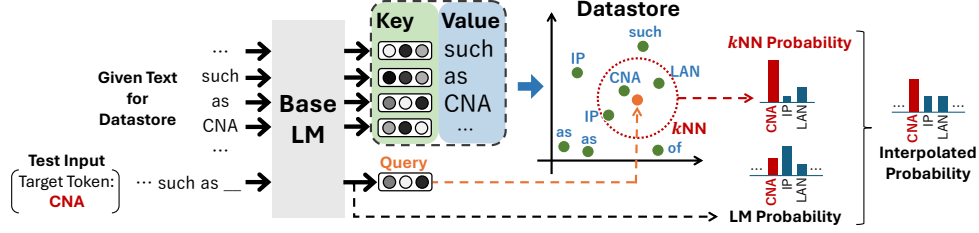
5980

Figure 1: Overview of kNN-LM: The target token *CNA* is retrieved as the nearest neighbor, and its prediction probability is enhanced by interpolating the kNN probability. Khandelwal et al. (2020) hypothesized that kNN-LM accurately predict long-tail phenomena, such as the low-frequency token *CNA*, as shown in this example through the use of explicit memory (e.g., the datastore).

trary to prior expectations, kNN-LM does not improve prediction performance for low-frequency tokens but instead enhances performance for high-frequency tokens. (2) Nearest neighbor search is more challenging for low-frequency tokens, as most target low-frequency tokens were not included in the retrieved nearest neighbors (§4.2). This likely explains why kNN-LM struggles to predict low-frequency tokens. (3) Low-frequency tokens have sparser distributions than high-frequency tokens in the datastore, and other tokens often appear mixed with the neighbors of low-frequency tokens (§4.3). It complicates the nearest neighbor search. (4) Long-tail tokens have larger reconstruction errors of the quantization in the datastore, making *approximate* nearest neighbor search more challenging (§4.4).

2 Problem Statement

2.1 Preliminary: kNN-LM

The k-nearest-neighbor language model (kNN-LM; Khandelwal et al., 2020) exploits knowledge of nearest neighbors retrieved from an external datastore with the base LM as shown in Figure 1.

As a fundamental process in kNN-LM, we preconstruct a datastore using arbitrary text. During inference, the kNN probabilities are calculated using the k-nearest-neighbor tokens retrieved from the datastore. These probabilities are then interpolated with the base LM probabilities. The kNN-LM improves the perplexity by utilizing explicit memory, i.e., the datastore.

Datastore Construction   Let a sequence of tokens $\boldsymbol{x} = (x_1, \cdots, x_{|\boldsymbol{x}|}) \in \mathcal{V}^*$ to be text stored in a datastore, where $\mathcal{V}$ denotes the vocabulary and $*$ represents the Kleene star. The datastore $\mathcal{D} \subseteq \mathbb{R}^D \times \mathcal{V}$ is constructed as a set of key-value pairs for all tokens in the text:

$$\mathcal{D} = \{(\boldsymbol{f}_\theta(\boldsymbol{x}_{<t}), x_t)\}_{t=1}^{|\boldsymbol{x}|}. \tag{1}$$

Each key is the intermediate representation $\boldsymbol{f}_\theta(\boldsymbol{x}_{<t})$ at each time step, where $\boldsymbol{f}_\theta \colon \mathcal{V}^* \to \mathbb{R}^D$ denotes the function that computes a D-dimensional hidden vector for the context $\boldsymbol{x}_{<t}$ using the base LM θ. Each value is the target token x_t, i.e., the next token in each context. The key-value pairs are not stored by token type, i.e., vocabulary element, but encompass all tokens. Consequently, the number of pairs in the datastore with a specific token type equals the frequency of that token type in the text from which the datastore is constructed.

The datastore often becomes too large because it stores high-dimensional contextualized representations for all tokens in the text. Therefore, product quantization (PQ; Jégou et al., 2011) is commonly used in kNN-LM to compress the key vectors in the datastore.

k**NN Probability Calculation**   When given a test input $\boldsymbol{x}$, the base LM θ computes a hidden vector $\boldsymbol{q}_t = \boldsymbol{f}_\theta(\boldsymbol{x}_{<t})$ for each target token x_t. This vector is used as a search query to retrieve the k-nearest-neighbors $\mathcal{N}(\boldsymbol{q}_t; \mathcal{D}) \subset \mathcal{D}$ from the datastore $\mathcal{D}$. The k-nearest-neighbors probability $p_{k\text{NN}}$ for the target token x_t is then calculated based on the distance between the query $\boldsymbol{q}_t$ and the retrieved keys as follows:

$$p_{k\text{NN}}(x_t|\boldsymbol{x}_{<t})$$
$$\propto \sum_{(\boldsymbol{k},v) \in \mathcal{N}(\boldsymbol{q}_t; \mathcal{D})} \mathbb{1}_{x_t=v} \exp\left(\frac{-d(\boldsymbol{k}, \boldsymbol{q}_t)}{\tau}\right), \tag{2}$$

where $d(\cdot, \cdot)$ is the distance function, and τ is the temperature parameter of the softmax function.[2] The kNN probability is assigned higher for tokens closer to the query than for those farther away.[3] If a token is not retrieved in the neighbors, its kNN probability is assigned to 0.

Interpolation Given a context $\boldsymbol{x}_{<t}$, the prediction probability p of the target token x_t is computed by linear interpolation of the kNN probability $p_{k\text{NN}}$ and the prediction probability of the base LM p_{LM} as follows:

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

where λ is a hyperparameter that controls the weights of the kNN probabilities. Note that the prediction performance of kNN-LM on a target token improves when $p_{k\text{NN}}$ of the token is greater than p_{LM} since the prediction probability p is a weighted average of $p_{k\text{NN}}$ and p_{LM}, regardless of the value of the interpolation factor λ when $\lambda > 0$.[4]

During inference, kNN-LM directly accesses text data via the datastore. Khandelwal et al. (2020) hypothesize that this explicit memory allows for better prediction of low-frequency phenomena, which is one factor in the success of kNN-LM.

2.2 Research Questions

Retrieval-augmented LMs, which explicitly use external memory, are hypothesized to be particularly effective for low-frequency tokens (Grave et al., 2017; Merity et al., 2017; Li et al., 2020). One such model, kNN-LM, is expected to improve predictions for low-frequency target tokens by interpolating kNN probabilities obtained from the datastore. While prior work has demonstrated that kNN-LM improves predictions for low-frequency contexts, our preliminary experiments (§3.5) revealed

no correlation between the frequency of context n-grams and the frequency of target tokens. Therefore, the behavior of kNN-LM for low-frequency target tokens remains unclear. To address this gap, we conducted a detailed analysis of the relationship between token frequency and the behavior of the kNN-LM. In particular, we investigate the following four research questions.

Research Question 1 Does kNN-LM improve the prediction performance for low-frequency target tokens? We investigate the relationship between token frequency and both the prediction probability of the base LM and the kNN probability. We show that kNN-LM actually worsens the prediction of low-frequency target tokens but improves the prediction of high-frequency target tokens (§4.1).

We then deeply probe the causes of kNN-LM's poor prediction performance for long-tail target tokens by conducting a detailed analysis of the kNN probability calculation process.

Research Question 2 Does the retrieval module effectively search for low-frequency tokens? Our analysis concludes that the failure to retrieve the majority of low-frequency tokens worsens their prediction with kNN-LM (§4.2).

We investigate the reason why retrieving long-tail tokens is challenging.

Research Question 3 How does the distribution of tokens in the key vector space of the datastore depend on their frequency? We show the distribution of the datastore makes it challenging to retrieve low-frequency tokens (§4.3).

Research Question 4 Are low-frequency tokens quantized accurately? We show the *approximate* nearest neighbor search with quantization tends to have large quantization errors on low-frequency tokens (§4.4).

3 Experimental Settings

We describe the experimental setup for the analysis of kNN-LM, which is shared across all experiments and analyses throughout this work.[5]

3.1 Base LM

We used GPT2-XL (Radford et al., 2019) as the base LM comprising 1.5B parameters.[6] This model was pre-trained on English web text data, which was crawled from outbound links on Reddit. Note

---

[2]Note that the temperature parameter τ does not appear in the original kNN-LM (Khandelwal et al., 2020). In this paper, we introduced the temperature to control the smoothness of the kNN probability distributions inspired by the k-nearest-neighbor machine translation (kNN-MT; Khandelwal et al., 2021), a variant of kNN-LM for encoder-decoder architectures.

[3]The temperature parameter τ adjusts the influence of distance on probability assignment. A higher value of τ reduces the impact of distance on the assigned probabilities. In the limit, $\tau \to \infty$, the kNN probability for a token becomes independent of its distance and is equal to the relative count of that token in the neighbors.

[4]Since $p = \lambda p_{k\text{NN}} + (1-\lambda)p_{\text{LM}} > \lambda p_{\text{LM}} + (1-\lambda)p_{\text{LM}} = p_{\text{LM}}$, i.e., the prediction probability p is greater than the base LM probability p_{LM} when $p_{k\text{NN}} > p_{\text{LM}}$ and $\lambda > 0$.

[5]The detailed settings are described in Appendix A.

[6]We also conducted experiments with models of different sizes, namely GPT2-Medium (355M) and GPT2-Large (774M), and confirmed that the results remained consistent.

that GPT-2 intentionally excludes all Wikipedia pages from the pre-training data because Wikipedia is often used as test data in language modeling tasks. It is popular to use GPT-2 as the base model for kNN-LM and is already proven to be effective in the previous research (Shi et al., 2022; Xu et al., 2023; Wang et al., 2023).

3.2 Token Frequency

We have investigated two types of frequencies when analyzing the low-frequency tokens in kNN-LM. First, we analyzed the frequency of tokens in the texts used to construct the datastore for the kNN-LM (*datastore frequency*). Second, we have also analyzed the frequency of tokens in the texts used to pre-train the base LM (*pre-training frequency*). Since the training data for the base LM, GPT-2, is not publicly available, we used the token frequency from its replica, the OpenWebTextCorpus (Gokaslan and Cohen, 2019), as a proxy for the base LM's frequency. By examining these frequencies, we aim to elucidate the relationship between token frequency and kNN-LM performance through comprehensive experiments.

3.3 Dataset

We used WikiText-103 (Merity et al., 2016) for our evaluation. The original split of WikiText-103 is unsuitable for analyzing the behavior of low-frequency tokens because of the large lexical overlap between test/valid data and train data. Therefore, we resplit the original WikiText-103 training split into the new train, test, and valid data. These test and valid data were split to contain many low-frequency tokens in the datastore constructed from the new training data. The process involved:

1. Tokenize the original training data using the subword tokenizer of the base LM.

2. Extract documents based on the ratio of n-grams with a frequency of 1 to all n-grams in the original training data over $n = 1, 2, 3, 4$ until the number of tokens almost matches the original valid and test data combined.

3. Divide the extracted documents randomly to create new valid and test datasets, then treat the remaining documents as training data.

Table 1 shows the number of tokens of the original and resplit dataset. We compared the token frequency distributions of the original and resplit datasets in Appendix A.1. We confirmed that the resplit test data contains more low-frequency to-

Subset	Original	Resplit
Train	117,317,490	116,754,298
Validation	245,757	276,476
Test	280,573	289,787

Table 1: Dataset size: the number of tokens.

kens than the original data, making it more suitable for our analysis conducted in this paper.

3.4 kNN-LM

We used the knn-transformers (Alon et al., 2022) for the implementation of kNN-LM. For datastore creation and nearest neighbor search, we employed FAISS (Johnson et al., 2019; Douze et al., 2024). The squared-L2 was used as a distance function. The keys in the datastore were 1,600-dimensional representations corresponding to the inputs of the feed-forward network in the final layer. For efficiency, we quantized the datastore using IVFPQ (Jégou et al., 2011). Following Khandelwal et al. (2020), the interpolation factor λ was set to 0.25. The datastores were constructed using both the original and resplit training data, and these were evaluated on the test data for the corresponding split.

3.5 Preliminary Experiments

Overall perplexity Before conducting a detailed performance analysis, we first measured the perplexity (PPL) of kNN-LM on both the original WikiText-103 and the resplit datasets. The datastores for kNN-LM were constructed from the training data of each respective dataset, and the PPL was evaluated with the corresponding evaluation data. The number of neighbors k and the softmax temperature τ for kNN-LM were chosen to minimize the PPL on the validation data, resulting in $k = 1024$ and $\tau = 10$ for both the original and resplit data.

Table 2 compares PPL between the base LM and kNN-LM on the test data for each dataset. The results show that kNN-LM outperforms the base LM on both the original and resplit datasets, indicating its effectiveness even with data containing many low-frequency tokens. However, the improvement was less pronounced on the resplit data than the original data, suggesting that the increase of low-frequency tokens limits the effectiveness of kNN-LM.

In the following sections, we conduct a detailed analysis of the prediction of kNN-LM for each

| Model | Perplexity ↓ | |
	Original	Resplit
Base LM	14.37	17.30
kNN-LM	**10.74**	**15.68**

Table 2: PPL on the test data of WikiText-103.

$n = 1$	$n = 2$	$n = 3$	$n = 4$	$n = 5$
-0.12	-0.067	-0.050	-0.012	-0.019

Table 3: Pearson correlation coefficient between frequency of target tokens and frequency of context n-grams ($n = 1, \cdots, 5$) on the test data of WikiText-103.
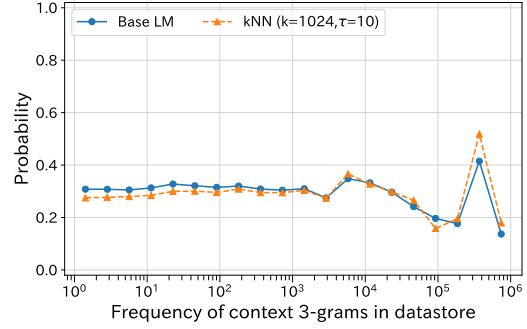

Figure 2: The relationship between the frequency of context 3-grams in the datastore and the expected values of kNN/LM probabilities on the resplit test.

token in the resplit test data, where the overall PPL shows improvement.

Predictions for long-tail contexts Retrieval-augmented LMs, such as kNN-LM, are hypothesized to be particularly effective for long-tail phenomena (Grave et al., 2017; Merity et al., 2017; Li et al., 2020). Khandelwal et al. (2020) provided qualitative examples showing that kNN-LM improves predictions for low-frequency context, i.e., the sequence preceding the target token. However, when we examined the relationship between the frequency of context n-grams and prediction probabilities using the resplit test, we found that, as shown in Figure 2 for $n = 3$, both kNN and LM probabilities exhibit similar trends, regardless of the frequency of the context.[7] This indicates that kNN-LM does not particularly improve predictions for long-tail contexts. Moreover, since the datastore of kNN-LM is constructed as pairs of keys, i.e., the intermediate representations of the context, and values, i.e., the target tokens, long-tail phenomena are primarily divided into long-tail contexts and long-tail target tokens.

Therefore, to fully understand kNN-LM's behavior on long-tail phenomena, we should analyze kNN-LM focusing on the target tokens. To investigate whether the relationship observed for contexts holds for target tokens as well, we show the correlation between the frequency of context n-grams ($n = 1...5$) and the frequency of target tokens in Table 3. Since there is no correlation between contexts and targets, we independently analyze the behavior of kNN-LM for long-tail target tokens, investigating the factors contributing to the success of kNN-LM.

---

[7] We showed the results for context n-grams with other n values ($n = 1, 2, 4, 5$) in Appendix B.1.

4 Analyses

4.1 kNN-LM Does NOT Improve Prediction Probability for Long-Tail Tokens

Khandelwal et al. (2020) hypothesized that kNN-LM improves prediction performance for long-tail phenomena by utilizing explicit memory, namely, the datastore, and they demonstrated improvements in predicting long-tail contexts. To further investigate this hypothesis, we focus on the target tokens, which are actually retrieved by kNN-LM, rather than the context. We perform a quantitative analysis to examine the relationship between token frequency and prediction probability. According to Equation 3, kNN-LM's prediction improves when the kNN probability exceeds the output probability of the base LM. Therefore, we compared these probabilities to determine whether the kNN probability surpasses the base LM's output probability for low-frequency tokens.

Figure 3 illustrates the relationship between the token frequency of the datastore, and the expected values of the kNN probability and the base LM probability on the resplit test data. The figure shows that for tokens with a frequency of approximately 10^5 or less in the datastore, the kNN probability was lower than the base LM probability, with the difference being particularly pronounced for frequencies of 10^3 or less. This result indicates that interpolating kNN probabilities with LM probabilities worsens prediction performance for low-frequency tokens. Conversely, the expected value of the kNN probability for tokens with a frequency exceeding 10^5 was higher than or comparable to the LM probability.

Figure 4 presents the relationship between the token frequency measured on the pre-training data for the base LM, and the expected values of probabili-
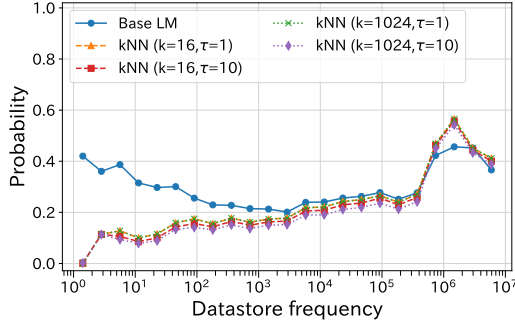
Figure 3: The relationship between datastore frequency and the expected values of kNN/LM probabilities on the resplit test: At low frequencies, the kNN probability was lower than the LM probability. At high frequencies, the opposite trend was observed.
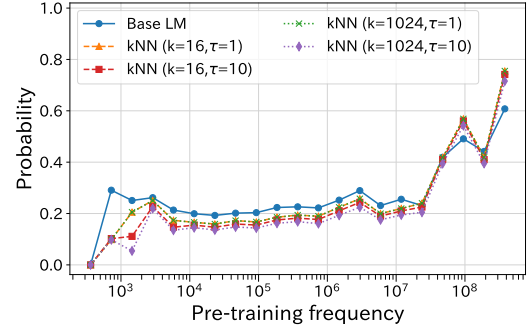

Figure 4: The relationship between pre-training frequency and the expected values of kNN/LM probabilities on the resplit test: At low frequencies, the kNN probability was lower than the LM probability. At high frequencies, the opposite trend was observed.

ties on the resplit test data. The figure reveals a similar pattern: at low frequencies, the kNN probability was lower than the LM probability, whereas at high frequencies, the opposite trend was observed.

For further analysis of low-frequency tokens, Appendix B.2 shows the relationship between token frequency and the proportion of tokens for which the kNN probability exceeds the LM probability, indicating an improvement in predictions by kNN-LM. The proportion decreases as the frequency decrease. Appendix B.2 compares the kNN probability and the base LM probability across different hyperparameters. The kNN probability for low-frequency tokens was consistently lower than the LM probability, regardless of the number of neighbors k, and this disparity became more pronounced as the temperature τ increased.

To better understand the relationship between token frequency and prediction probability through case studies, Table 4 shows qualitative examples in the resplit test data. For target tokens *intoxicated*, *FreeBSD*, and *recommendation* whose datastore frequencies were 10^3 or less, the kNN probabilities were lower than the base LM probabilities. These tokens, observed in sample contexts, would be related to the factual knowledge that Khandelwal et al. (2020) hypothesized kNN-LM would excel at; however, actually, kNN-LM worsens the predictions for them. In contrast, the target token *song*, whose frequency was approximately 10^5, had a higher kNN probability than the base LM probability.

In summary, our findings demonstrate that *kNN-LM does not improve the prediction performance of long-tail tokens* irrespective of the hyperparameters. Instead, it suggests that kNN-LM enhances the prediction performance for high-frequency tokens.

4.2 kNN Search for Long-Tail Tokens Is Challenging

We demonstrated that the kNN probabilities are lower for long-tail tokens in §4.1. As a reason for this, we hypothesize that low-frequency tokens are less likely to be retrieved in the kNN search. To validate this hypothesis, we analyzed the kNN hit rate, which is the proportion of the neighbors $\mathcal{N}$ that contain at least one instance of the target token.

Figure 5 shows the relationship between the token frequency of the datastore and the expected kNN hit rate.[9] The figure indicates that increasing the number of neighbors k tends to improve the kNN hit rate regardless of token frequency. For tokens with frequencies of 10^5 or more, the kNN hit rate approached nearly 100% with $k = 1024$. In contrast, for tokens with frequencies below 10^4 with $k = 16$, and below 10 with $k = 1024$, the kNN hit rate fell under 50%. This means that for most of the low-frequency tokens, *the target token does not appear among the neighbors at all*, resulting in the kNN probability of 0. This explains the low kNN probabilities for low-frequency tokens discussed in §4.1.

Additionally, increasing the number of neighbors k introduces more non-target tokens as noise, thereby lowering the relative frequency of instances with the value of the target token among the neighbors. This poses a trade-off between the relative frequency of the target token and the kNN hit rate. This trade-off would explain why the kNN probabilities do not improve with a larger number of

---

[8]In GPT-2, tokenization is applied to the entire string, including spaces, so tokens sometimes contain spaces.

[9]Figure 14 shows the plot for token frequency of the pre-training data in Appendix B.3.

Target token	Datastore freq.	Probability ↑		kNN hit rate ↑	A sample context for the target token of test input	Top-3 neighbors
		kNN	LM			
␣intoxicated	5	.137	.518	.600	...They also put honey bees in shuttle-boxes that used a stimulus to encourage the bees to move, and found that they were less mobile as they became more ___	in, with, intoxicated
␣FreeBSD	6	.029	.320	.520	...FreeBSD is a free Unix-like operating system descended from Research Unix via the Berkeley Software Distribution (BSD). Although for legal reasons ___	the, Perl, a
␣recommendation	966	.016	.309	.571	...Sleep hygiene is the recommended behavioral and environmental practice that is intended to promote better quality sleep. This ___	direct, direct, is
␣song	67,141	.525	.421	.989	...Yusuke Tanaka directed the accompanying music video for the single, which shows Perfume as robots and dancing with fairy lights around them. Perfume have performed the ___	song, song, song

Table 4: Quantitative examples ($k = 1024$, $\tau = 10$): For target tokens[8] *intoxicated*, *FreeBSD*, and *recommendation* whose datastore frequencies were 10^3 or less, the kNN probability was lower than the base LM probability, and the kNN hit rate was below 60%. In contrast, token *song*, whose frequency was approximately 10^5, had a higher kNN probability than the base LM probability, and in most cases, the nearest neighbors contained the target token. The right side of the table shows the sampled contexts for each token type in the resplit test data and displays the top-3 nearest neighbors for those contexts. For low-frequency tokens, the nearest neighbor for each context was an incorrect token, whereas for the high-frequency token, the correct target token was included in the nearest neighbors.
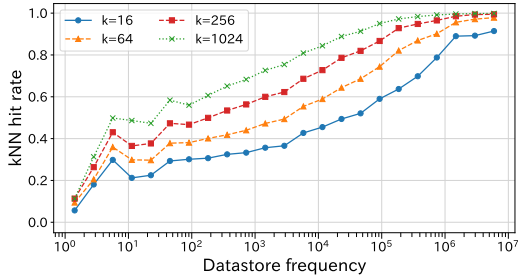

Figure 5: The relationship between datastore frequency and the expected kNN hit rate on the resplit test: For low-frequency tokens, the target token was not included in the neighbors at all.

neighbors, as observed in §4.1.

We present qualitative examples of the kNN hit rate in the resplit test data in Table 4. The table shows that target tokens *intoxicated*, *FreeBSD*, and *recommendation*, which have low frequency in the datastore, had low kNN hit rates. In contrast, the token *song* with high frequency had a high hit rate.

4.3 Long-Tail Tokens Are Distributed Sparsely with Contamination in Datastore

We showed that long-tail tokens were less likely to be retrieved during the kNN search in §4.2. We hypothesize that this is due to the distribution of tokens in the datastore. This section further analyzes the distribution in the datastore from two perspectives: the sparsity and contamination of neighbors

for low-frequency tokens.

Sparsity Intuitively, if the keys of a target token type are sparsely distributed, the distances from these keys to the query corresponding to the target token type are expected to be larger, making the nearest neighbor search more challenging. We hypothesize that low-frequency tokens are sparsely distributed in the datastore and investigate the relationship between token frequency and the dispersion of their distribution.

To quantify the sparsity of tokens with a token type, we measured the Euclidean distance from the centroid of the keys whose values correspond to the token type to each key. We then calculated the coefficient of variation (CV) of these distances, i.e., the ratio of the standard deviation to the mean, which indicates the dispersion of the distribution.

Figure 6 shows the relationship between token frequency of the datastore and the CV of the distances. The figure reveals that low-frequency tokens have a higher CV compared to high-frequency tokens.[10] This result implies that *low-frequency tokens are more sparsely distributed in the datastore*, which suggests that their sparse distribution makes the search more challenging.

Contamination If the neighbors of the target token are contaminated with the non-target tokens

---
[10]Figure 15 shows the plot for token frequency of the pretraining data in Appendix B.4.
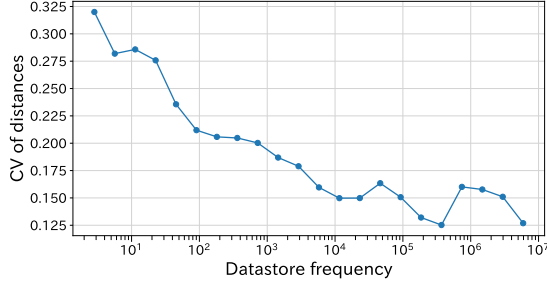
Figure 6: The relationship between token frequency of the datastore and the CV of the distances: Low-frequency tokens in the datastore have a higher CV compared to high-frequency tokens.
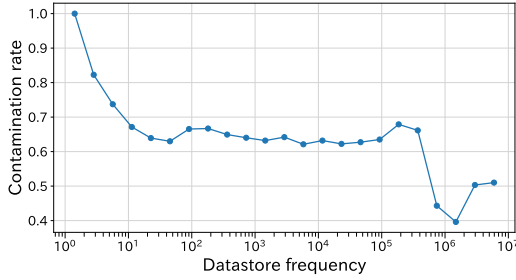

Figure 7: The relationship between the token frequency of the datastore and the contamination rate of the token type: Lower-frequency token types had higher contamination rates.

in the datastore, retrieving the target token as the nearest neighbor becomes challenging. We hypothesize that the neighbors of low-frequency tokens have different token types as the nearest neighbor, which might directly impact the kNN probabilities.

To validate this hypothesis, we investigated whether each instance in the datastore had the same token type as the nearest neighbor. Specifically, we quantified the contamination rate for each token type. We calculated the proportion of instances whose nearest neighbor instance had a different token type in the datastore.

Figure 7 shows the relationship between token frequency of the datastore and the contamination rate of the token type.[11] The figure reveals that lower-frequency token types have higher contamination rates. For token types with a frequency of less than 10, the nearest neighbor of the instance has a different token type in most cases. This result indicates that *low-frequency token types have contaminated neighbors in the datastore*, making the kNN search more difficult.

---

[11] Figure 16 shows the plot for token frequency of the pre-training data in Appendix B.4.
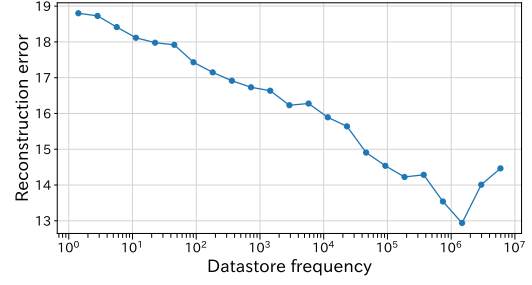

Figure 8: The relationship between the token frequency of the datastore and the reconstruction error of PQ: Low-frequency tokens have higher approximation errors in the datastore.

4.4 Long-Tail Tokens Have Large Quantization Error in Datastore

We hypothesize another reason why the nearest neighbor search for low-frequency tokens fails is the adverse impact of approximation errors in the search process. The product quantization (PQ; Jégou et al., 2011), commonly used for compressing key vectors in the datastore of kNN-LM, may accurately approximate the majority of tokens but might fail to approximate long-tail tokens.

We measured the approximation error introduced by datastore quantization as the average of the reconstruction error, defined as the Euclidean distance between the original key and the vector reconstructed from the PQ-encoded key.

Figure 8 shows the relationship between token frequency of the datastore and the reconstruction error of the PQ.[12] The figure indicates that the quantization error is larger for low-frequency tokens and smaller for high-frequency tokens. We then investigated the relationship between approximation error and kNN-LM's prediction performance. Figure 9 shows the relationship between kNN and base LM probabilities and PQ reconstruction error on the resplit test. When the error was small, the kNN probabilities exceeded the base LM probabilities, leading to improved predictions through kNN augmentation. In contrast, larger errors led to a decline in prediction performance. Additionally, we calculated the correlation between the average prediction gain of kNN-LM (i.e., $p_{\text{kNN}} - p_{\text{LM}}$) and the average reconstruction error in the datastore for each token type. The Pearson correlation coefficient was -0.20, and the Spearman's rank correlation coefficient was -0.16. These negative correlations

---

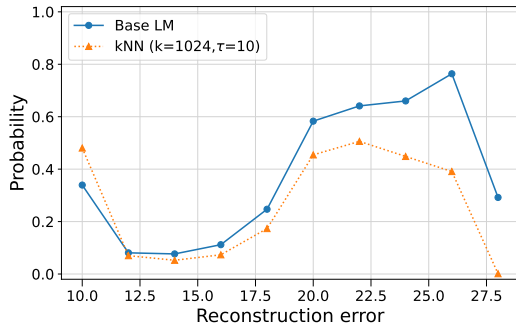[12] Figure 17 shows the plot for token frequency of the pre-training data in Appendix B.5.

5987

Figure 9: The relationship between kNN and base LM probability on the resplit test and the reconstruction error of the product quantization.

suggest that kNN-LM worsens the prediction performance for tokens with larger approximation errors in the datastore. Thus, *low-frequency tokens have higher approximation errors in the datastore*, making the approximate nearest neighbor search more difficult.

5 Related Work

Retrieval-augmented language model (Grave et al., 2017; Kaiser et al., 2017; Guu et al., 2018; Khandelwal et al., 2020; He et al., 2020; Borgeaud et al., 2022) incorporates a base language model (LM) with retrieved neighbor examples from a datastore for language modeling and other tasks. Kaiser et al. (2017) introduced a memory module capable of remembering rare events into a neural network and used it for text generation. Khandelwal et al. (2020) proposed kNN-LM, one of the standards for retrieval-augmented LMs, which interpolates the output probabilities of the trained base LM, with kNN probabilities computed from the retrieved neighbors in the datastore. Similar to Kaiser et al. (2017), they hypothesized that using an explicit memory, i.e., datastore, is particularly helpful for predicting long-tail phenomena based on qualitative examples. While they demonstrated that kNN-LM improves the prediction of long-tail contexts, our detailed analysis reveals that kNN-LM often fails to enhance the prediction of long-tail target tokens, regardless of the frequency of the context n-grams.

To understand the strengths and limitations of kNN-LM, several studies have investigated the key factors underlying its performance. Drozdov et al. (2022) find that large overlapping n-grams between the datastore and the evaluation data is crucial for the strong performance of kNN-LM. This find-

ing aligns with our results, showing that kNN-LM performs poorly on the resplit evaluation data of WikiText-103, which has less overlap with the training data. Wang et al. (2023) revealed that focusing on individual tokens, the majority of tokens' predictions deteriorate with kNN-LM, leading to lower performance in open-ended text generation. Our deep probing of individual tokens revealed that this majority primarily consists of low-frequency tokens. Xu et al. (2023) implies through various ablation studies that memorization is not a key factor in the success of kNN-LM. Our analysis of rare tokens further supports this hypothesis.

Several methods have been proposed to enhance the performance of kNN-LM and its derivative, k-nearest-neighbor machine translation (kNN-MT; Khandelwal et al., 2021). These methods include adaptively determining the number of neighbors k and the interpolation factor λ (Zheng et al., 2021; Jiang et al., 2021, 2022), as well as recalculating distances using two different datastores (Bhardwaj et al., 2023). However, these approaches do not explicitly address the negative impact on the prediction of low-frequency tokens, as demonstrated in this paper.

6 Conclusion

In this paper, we analyzed the behavior of kNN-LM, focusing on low-frequency target tokens. Our analysis revealed that kNN-LM does not improve the prediction for long-tail target tokens, regardless of the frequency of the context. This lack of improvement can be attributed to the challenges of kNN search for long-tail tokens, which is complicated by factors such as the distribution of the datastore and quantization errors.

Based on our findings, improving kNN-LM would require designing robust embedding and retrieval methods for low-frequency tokens and developing datastore compression techniques with fewer errors. One approach could involve applying inverse document frequency (IDF) weights to target tokens, which would help adjust the output probabilities for informative low-frequency tokens. Additionally, Zipfian whitening (Yokoi et al., 2024), which normalizes embeddings based on token frequencies, may help mitigate frequency bias of the datastore. We hope that our insights will contribute to exploring more effective methods with kNN-LM for language modeling and other natural language processing tasks in future research.

Limitations

Our work revealed that kNN-LM deteriorates the prediction of low-frequency tokens, which was contrary to the prior hypothesis, and we deeply probed the reasons behind this phenomenon. While our analyses focused on the prediction of subword tokens, extending these analyses to the level of words and phrases would be insightful.

We compared the expected values, i.e., the *arithmetic* means, of kNN probabilities and the base LM probabilities. Note that the evaluation metric commonly used in language modeling tasks, perplexity (PPL), corresponds to the *geometric* mean of the linear interpolation of kNN probabilities and the base LM probabilities. While the trends in the arithmetic means of each probability should resemble the trends in PPL, they are not completely identical.

We conducted analyses using GPT-2, which does not include Wikipedia text in its training data, to evaluate WikiText-103 in an out-of-domain scenario where kNN-LM is known to be most effective. It is also known that kNN-LM improves language modeling performance even in in-domain settings. Analyzing kNN-LM in in-domain settings remains our future work.

Acknowledgements

References

Uri Alon, Frank Xu, Junxian He, Sudipta Sengupta, Dan Roth, and Graham Neubig. 2022. Neuro-symbolic language modeling with automaton-augmented retrieval. In *International Conference on Machine Learning*, pages 468–485. Proceedings of Machine Learning Research.

Rishabh Bhardwaj, George Polovets, and Monica Sunkara. 2023. Adaptation approaches for nearest neighbor language models. In *Findings of the Association for Computational Linguistics: ACL 2023*, pages 1135–1146, Toronto, Canada. Association for Computational Linguistics.

Sebastian Borgeaud, Arthur Mensch, Jordan Hoffmann, Trevor Cai, Eliza Rutherford, Katie Millican, George Bm Van Den Driessche, Jean-Baptiste Lespiau, Bogdan Damoc, Aidan Clark, et al. 2022. Improving language models by retrieving from trillions of tokens. In *International conference on machine learning*, pages 2206–2240. Proceedings of Machine Learning Research.

Matthijs Douze, Alexandr Guzhva, Chengqi Deng, Jeff Johnson, Gergely Szilvasy, Pierre-Emmanuel Mazaré, Maria Lomeli, Lucas Hosseini, and Hervé Jégou. 2024. The faiss library. *arXiv preprint arXiv:2401.08281*.

Andrew Drozdov, Shufan Wang, Razieh Rahimi, Andrew McCallum, Hamed Zamani, and Mohit Iyyer. 2022. You can't pick your neighbors, or can you? when and how to rely on retrieval in the kNN-LM. In *Findings of the Association for Computational Linguistics: EMNLP 2022*, pages 2997–3007, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

Aaron Gokaslan and Vanya Cohen. 2019. Openwebtext corpus. http://Skylion007.github.io/OpenWebTextCorpus.

Edouard Grave, Moustapha Cisse, and Armand Joulin. 2017. Unbounded cache model for online language modeling with open vocabulary. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, NIPS'17, page 6044–6054, Red Hook, NY, USA. Curran Associates Inc.

Kelvin Guu, Tatsunori B. Hashimoto, Yonatan Oren, and Percy Liang. 2018. Generating sentences by editing prototypes. *Transactions of the Association for Computational Linguistics*, 6:437–450.

Junxian He, Taylor Berg-Kirkpatrick, and Graham Neubig. 2020. Learning sparse prototypes for text generation. In *Proceedings of the 34th International Conference on Neural Information Processing Systems*, NIPS '20, Red Hook, NY, USA. Curran Associates Inc.

Hervé Jégou, Matthijs Douze, and Cordelia Schmid. 2011. Product quantization for nearest neighbor search. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 33(1):117–128.

Hui Jiang, Ziyao Lu, Fandong Meng, Chulun Zhou, Jie Zhou, Degen Huang, and Jinsong Su. 2022. Towards robust k-nearest-neighbor machine translation. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 5468–5477, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

Qingnan Jiang, Mingxuan Wang, Jun Cao, Shanbo Cheng, Shujian Huang, and Lei Li. 2021. Learning kernel-smoothed machine translation with retrieved examples. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 7280–7290, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

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

Lukasz Kaiser, Ofir Nachum, Aurko Roy, and Samy Bengio. 2017. Learning to remember rare events. In

International Conference on Learning Representations.

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

Urvashi Khandelwal, Omer Levy, Dan Jurafsky, Luke Zettlemoyer, and Mike Lewis. 2020. Generalization through memorization: Nearest neighbor language models. In *International Conference on Learning Representations.*

Ke Li, Daniel Povey, and Sanjeev Khudanpur. 2020. Neural Language Modeling with Implicit Cache Pointers. In *Proceedings of the 21st Annual Conference of the International Speech Communication Association*, pages 3625–3629.

Stephen Merity, Caiming Xiong, James Bradbury, and Richard Socher. 2016. Pointer sentinel mixture models. *arXiv preprint arXiv:1609.07843.*

Stephen Merity, Caiming Xiong, James Bradbury, and Richard Socher. 2017. Pointer sentinel mixture models. In *International Conference on Learning Representations.*

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, et al. 2019. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9.

Weijia Shi, Julian Michael, Suchin Gururangan, and Luke Zettlemoyer. 2022. Nearest neighbor zero-shot inference. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 3254–3265, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

Shufan Wang, Yixiao Song, Andrew Drozdov, Aparna Garimella, Varun Manjunatha, and Mohit Iyyer. 2023. kNN-LM does not improve open-ended text generation. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 15023–15037, Singapore. Association for Computational Linguistics.

Frank F. Xu, Uri Alon, and Graham Neubig. 2023. Why do nearest neighbor language models work? In *Proceedings of the 40th International Conference on Machine Learning*, ICML'23. JMLR.org.

Sho Yokoi, Han Bao, Hiroto Kurita, and Hidetoshi Shimodaira. 2024. Zipfian whitening. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems.*

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.
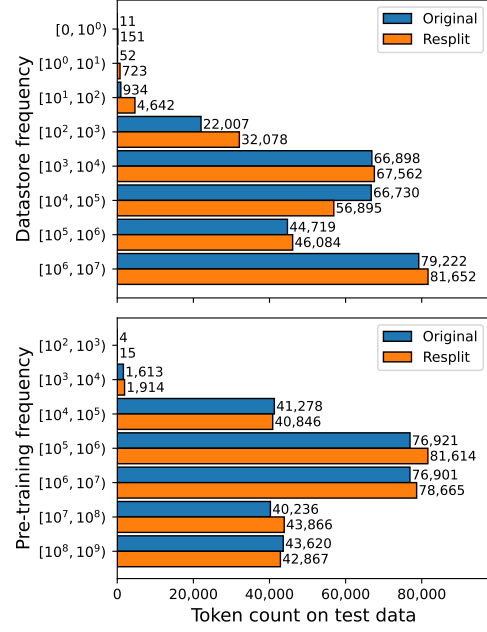
Figure 10: The number of token occurrences in the original and resplit test data of WikiText-103 for each bin of datastore frequency (upper) and pre-training frequency (lower): The resplit test data contains more low-frequency tokens than the original test data.

A Detailed Settings

A.1 Dataset

The upper plot of Figure 10 presents the number of tokens in the test data of the original and resplit datasets for datastore frequency, i.e., token frequency in the corresponding training data. The figure shows that the resplit test data contains more tokens with low datastore frequency (less than 10^3) compared to the original data. The lower plot of Figure 10 shows the count of tokens in original/resplit test data for pre-training frequency. The figure indicates that the resplit test data contains slightly more tokens with low pre-training frequency than the original split. Therefore, our resplit data is more suitable for analyzing low-frequency tokens than the original data.

A.2 kNN-LM

We quantized the datastore with IVFPQ, setting the code size to 64, the number of bits to 8, and the number of centroids to 4096. We used IndexFlatL2 as the coarse quantizer. The index looks up 32 cluster centroids while searching for the nearest neighbors.

A.3 Preliminary Experiments

In the preliminary experiments in §3.5, we tuned hyperparameters k and τ for kNN-LM, selecting from $k \in \{2, 4, 8, 16, 32, 64, 128, 256, 512, 1024\}$ and $\tau \in \{0.01, 0.1, 1, 10, 100, 1000\}$, minimizing PPL with each validation data of original/resplit WikiText-103. We used $k = 1024, \tau = 10$ for both the original and resplit evaluation data.

B Further Analyses

B.1 Predictions for long-tail contexts

In §3.5, we showed the relationship between the frequency of context n-grams in the datastore and the expected values of kNN/LM probabilities on the resplit test for $n = 3$. Figure 11 shows the relationship between the frequency of context n-grams and the probabilities for other n values ($n = 1, 2, 4, 5$). For these other n values, we observed a similar trend to $n = 3$, where no substantial improvement from kNN-LM was observed for low-frequency context n-grams.

B.2 kNN Probability

We analyzed the relationship between token frequency, kNN probability, and base LM probability, revealing that kNN-LM worsens the prediction probabilities for low-frequency tokens in §4.1. To deeply understand, we investigate the influence of the hyperparameters, the number of neighbors k and the temperature τ. To this end, we sorted the token types in the base LM's vocabulary by their datastore frequency and divided them into three equal categories: LOW, MED, and HIGH. For each of these categories, we plot the relationship between probabilities and the hyperparameters.

Figure 12 shows the relationship between kNN/LM probabilities and the number of neighbors k for each token frequency category at $\tau = 1$ on the resplit test data. The figure indicates that the kNN probability varies little with the number of neighbors and consistently remains lower than the LM probability for tokens in the low-frequency category, regardless of the number of neighbors k.

Figure 13 shows the relationship between kNN/LM probabilities and the temperature τ. The figure reveals that increasing the temperature τ accentuates the lower kNN probability for the LOW frequency category.

These results indicate that the tendency of kNN-LM to worsen prediction probabilities for low-frequency tokens, as we revealed, cannot be resolved by adjusting the hyperparameters.

B.3 kNN Hit Rate

We demonstrated that tokens with low datastore frequency have a low kNN hit rate in §4.2.

Figure 14 shows the relationship between pre-training frequency and kNN hit rate on the resplit test data. Similar to the case with datastore frequency, the figure reveals that tokens with low pre-training frequency have a low kNN hit rate. The majority of tokens with a frequency of approximately 10^3 or less were not being retrieved at all, even with $k = 1024$.

B.4 Token Distribution in Datastore

We demonstrated that tokens with low datastore frequency are sparsely distributed in the datastore and that their neighbors are contaminated with different token types in §4.3.

Figure 15 shows the relationship between pre-training frequency and the coefficient of variation (CV) of distances on the resplit test data. Similar to the case with datastore frequency, the figure reveals that tokens with lower pre-training frequency tend to be more sparsely distributed, except for tokens with a frequency greater than 10^8. Some high-frequency tokens exhibit a different trend due to their polysemy, which affects only the right tail.

Figure 16 shows the relationship between pre-training frequency and the contamination rate on the resplit test data. Similar to the case with datastore frequency, the figure indicates that the neighbors of tokens with lower pre-training frequency are often mixed with other token types.

B.5 Quantization Error

We demonstrated that tokens with low datastore frequency have larger approximation errors of PQ in §4.4.

Figure 17 shows the relationship between pre-training frequency and PQ reconstruction error in the resplit test data. Similar to the case with datastore frequency, the figure reveals that tokens with lower pre-training frequency tend to have larger approximation errors.
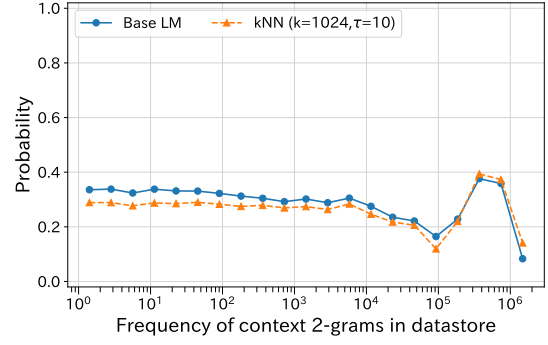
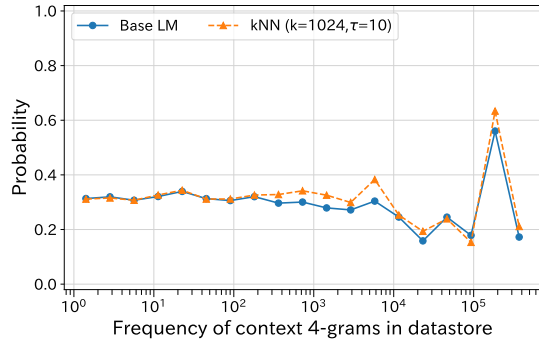## C Used Data, Model, and Software

C.1 Data

WikiText-103 created by Merity et al. (2016). License: CC BY-SA 3.0.
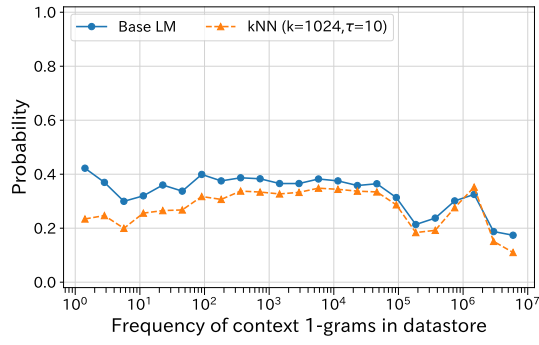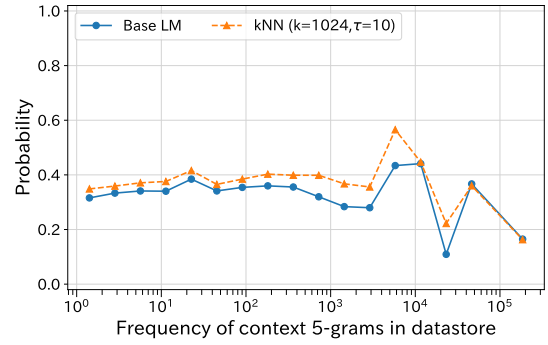
(a) $n = 1$

(b) $n = 2$

(c) $n = 4$

(d) $n = 5$

Figure 11: The relationship between the frequency of context n-grams in the datastore and the expected values of kNN/LM probabilities on the resplit test.
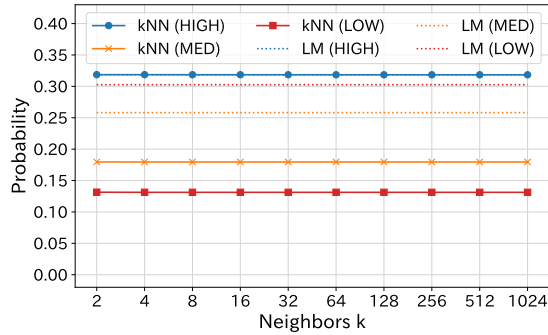

Figure 12: The relationship between kNN/LM probabilities and the number of neighbors k across frequency categories ($\tau = 1$).
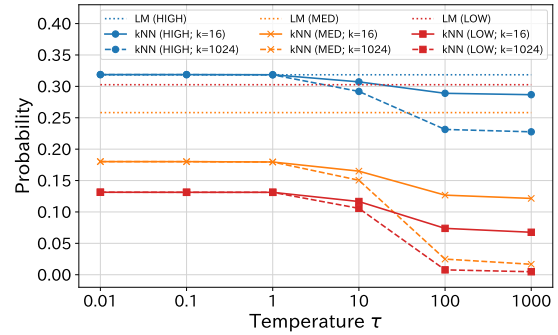

Figure 13: The relationship between kNN/LM probabilities and the number of temperature τ across frequency categories.

OpenWebTextCorpus created by Gokaslan and Cohen (2019). License: CC0 1.0.

C.2 Model

GPT2-XL created by Radford et al. (2019). Download: `https://huggingface.co/openai-community/gpt2-xl`. License: MIT.

C.3 Software

FAISS created by Johnson et al. (2019); Douze et al. (2024). Download: `https://github.com/facebookresearch/faiss`, License: MIT.

knn-transformers created by Alon et al. (2022). Download: `https://github.com/neulab/knn-transformers`, License: MIT.
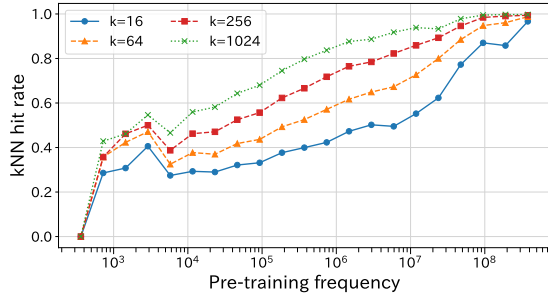
Figure 14: The relationship between pre-training frequency and the expected kNN hit rate on the resplit test.
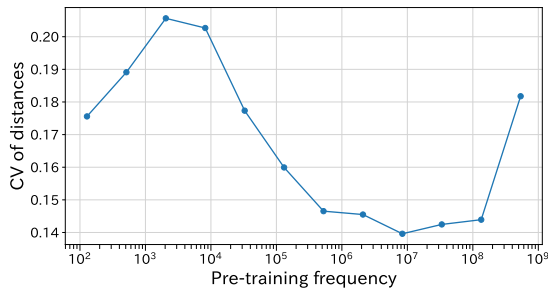

Figure 15: The relationship between pre-training frequency and the CV of the distances.
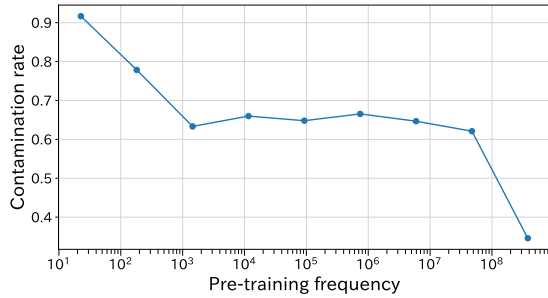

Figure 16: The relationship between token frequency of the pre-training data and the contamination rate of the token type.
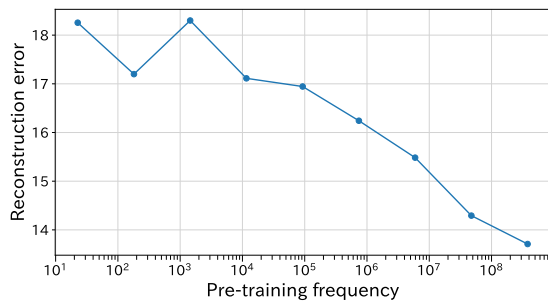

Figure 17: The relationship between token frequency of the pre-training data and the reconstruction error of the product quantization.