Retrieval-based Language Models Using a Multi-domain Datastore

Rulin Shao
University of Washington
rulins@cs.washington.edu

Sewon Min
University of Washington
sewon@cs.washington.edu

Luke Zettlemoyer
University of Washington
lsz@cs.washington.edu

Pang Wei Koh
University of Washington
pangwei@cs.washington.edu

Abstract

Retrieval-based language models (LMs) can generalize well to unseen test domains, but typically assume access to a datastore of examples from the target domain. It remains an open question if these models are robust with more general datastores, which may include other out of domain data or cover multiple different test domains. In this paper, we study this question by constructing a multi-domain datastore, using a kNN-LM approach. We first show that, on domains that are part of the multi-domain datastore, the model is comparable to or even better than the model with an oracle test domain datastore. We also find that, on domains that are unseen during training and not part of the datastore, using a multi-domain datastore consistently outperforms an oracle single-domain datastore. Together, our results show that kNN-LM is highly robust at out-of-distribution generalization and can effectively target many domains at once, without the oracle domain knowledge assumptions included in all previous work.

1 Introduction

Generalizing to new domains that are unseen during training is a long-standing challenge in machine learning. Retrieval-based language models (LMs) have recently been shown to be effective at domain generalization (Min et al., 2023a), assuming that they have test-time access to a datastore of text samples that are perfectly in-distribution to the evaluation data. However, it is unexplored how the performance of retrieval-based LMs is affected by distribution shifts between the datastore and evaluation data, e.g., because the datastore includes other potentially distracting domains or the evaluation domain is not even included in the datastore. Such shifts would be common when building a general-purpose retrieval-based LM that can target multiple domains simultaneously.

In this paper, we study this question using kNN-LM retrieval language models (Khandelwal et al., 2020b). We first construct a multi-domain datastore by concatenating 8 single-domain datastores with a post-hoc merging method, making it easy to study the effect of different datastore compositions at scale. We then evaluate on 20 different domains, categorized along two dimensions: whether or not the domain was seen during training, and whether or not the domain data is included in the datastore.

Our results show that, as long as the test domain is included in the datastore, kNN-LM with a multi-domain datastore outperforms or matches the model with an oracle, single-domain datastore. This indicates that kNN-LM is highly robust to the out-of-distribution data in a datastore and can effectively target many domains simultaneously, unlike parametric models that suffer from competition between different target domains (Oren et al., 2019; Chowdhery et al., 2022; Xie et al.,

Workshop on Distribution Shifts, 37th Conference on Neural Information Processing Systems (NeurIPS 2023).

2023). When test domains are not included in the datastore, we find that a multi-domain datastore consistently outperforms an oracle single-domain datastore, although the gains from retrieval are relatively minor (9% vs. 66% for domains in the datastore). Together, our results highlight the promise of scaling the domains in a datastore for a general-purpose retrieval-based LM with no prior assumptions about the evaluation domain.

2 Related Work

Out-of-distribution (OOD) generalization. Prior work on distribution shifts have shown that models can suffer significant performance drops when evaluated on test data that is OOD to the training data (Koh et al., 2021; Sagawa et al., 2022). Generalizing across multiple domains simultaneously presents additional challenges in standard parametric models, because targeting one domain (e.g., by upweighting that domain in the training data) can lower performance on other domains (Oren et al., 2019; Sagawa et al., 2019; Chowdhery et al., 2022; Xie et al., 2023). In this work, we study domain generalization relative to the datastore provided at inference time. Furthermore, we demonstrate that using nonparametric retrieval from a multi-domain datastore effectively removes the competition between different domains, allowing the model to perform well on all domains in the datastore.

Retrieval-based LMs. Retrieval-based LMs reason with external data provided during inference (called a *datastore*). We distinguish work that retrieves a small set of text blocks from the data and feeds into the LM as an additional input (Guu et al., 2020; Borgeaud et al., 2022; Shi et al., 2023; Ram et al., 2023) from work that uses a nonparametric softmax over the data (Khandelwal et al., 2020b; Zhong et al., 2022; Min et al., 2023b). We use kNN-LM (Khandelwal et al., 2020b), a model that falls into the latter type, and leave extensions to other models for future work. Most work in retrieval-based LMs assume an in-distribution datastore to the evaluation data, with a few exceptions: Borgeaud et al. (2022); Piktus et al. (2021), which use only Web data, and Khandelwal et al. (2020a); Huang et al. (2023), which constructed a multi-domain datastore for an ablation study. To the best of our knowledge, our work is the first that studies the effect of different choices and compositions of the domains in the datastore, including on test domains that are not included in the datastore.

3 Method

A kNN-LM models a next token probability distribution given a prefix: $P(y|x_1...x_n; \mathcal{D})$ conditioned on a datastore with N tokens, $\mathcal{D} = \{d_1...d_N\}$. It computes a contextualized representation for each token $\mathbf{d}_i = \text{Enc}(d_1...d_{i-1}) \in \mathbb{R}^h$, where Enc is a function that maps the text into a vector, and h is the number of hidden dimensions. The distribution is then defined as $P(y|x_1...x_n; \mathcal{D}) = (1-\lambda)P_{\text{LM}}(y|x_1...x_n) + \lambda P_{k\text{NN}}(y|x_1...x_n; \mathcal{D})$, where $P_{\text{LM}}(y|x_1...x_n)$ is an output from a regular LM (called a parametric-only LM), and

$$P_{k\text{NN}}(y|x_1...x_n; \mathcal{D}) \propto \sum_{1 \leq i \leq N} \mathbb{I}[d_i = y] \exp\left(-\frac{\text{dist}(\mathbf{d}_i, \text{Enc}(x_1...x_n))}{\tau}\right),$$

where dist is a distance function (squared L2 distance in our experiments), and λ and τ are hyperparameters. Typically, $P_{k\text{NN}}$ is approximated by considering only the k vectors nearest to $\text{Enc}(x_1...x_n)$ out of $\mathbf{d}_1...\mathbf{d}_N$—see Appendix A for the details.

The kNN-LM method can be used either with a single-domain dataset $\mathcal{D}$ or with multiple datasets in M different domains, $\mathcal{D}_1...\mathcal{D}_M$. We use a post-hoc index merging scheme to construct our multi-domain datastore, which allows us to reuse the datastore from each domain without constructing a new datastore from scratch. Details are provided in Appendix A. Our method allows studying the effect of different compositions of a multi-domain datastore at scale, as demonstrated in Section 4.

4 Experiments

4.1 Experimental Setup

Parametric LM. We use SILO (Min et al., 2023a) for both parametric inference and encoding for the datastore construction, as it is not exposed to most of the domains during training.

Table 1: Perplexity change brought by a **single-domain** datastore, compared to the parametric LM. Each column represents the domain used in a single-domain datastore, and each row represents the evaluation domain. For this table, we set a minimum value of $\lambda \geq 0.1$ to force the model to use the datastore. *Green* and *red* indicate the datastore helps and does not help, respectively. **Bold** indicates the datastore is in-domain to the eval domain.

Eval Domain	Datastore Domain						
	Wikipedia	Books3	Github	NIH ExPorter	Amazon	CC News	MIMIC III
Wikipedia	**-4.92**	-0.72	0.47	0.60	-0.16	-0.63	1.28
Books3	-0.19	**-1.31**	0.59	0.87	0.01	0.01	1.03
Github	0.15	0.12	**-0.19**	0.24	0.16	0.17	0.23
NIH ExPorter	-0.20	0.02	0.77	**-4.17**	0.56	0.10	0.96
Amazon	-0.07	-2.94	0.81	1.48	**-7.65**	-0.35	1.42
CC News	-1.01	-0.49	0.47	0.82	-0.25	**-15.07**	1.22
MIMIC III	0.20	-0.08	0.80	0.24	0.48	0.35	**-13.12**

Datastore. We build a multi-domain datastore containing 5 billion tokens from eight domains: Github, NIH ExPorter, Wikipedia, Books3, Enron Emails, CC-News, Amazon and MIMIC-III (see Appendix B for the details). These domains follow the choice made in Min et al. (2023a).

Evaluation domains. We consider 20 evaluation domains in total, including eight domains presented in the datastore and 12 more from the Pile (Gao et al., 2020). All domains fall in one of three categories, based on whether it was seen during training and whether it is in the datastore:[1]

- Seen during training & included in the datastore: These are domains that are available at any time.
- Unseen during training & included in the datastore: These are domains that cannot be used for model training but can be used at inference time, either because the data requires attribution or opt-out guarantees (Min et al., 2023a), or the data includes domains that were added after pretraining. Prior work in retrieval-based LMs often assumes this setting.
- Unseen during training & not included in the datastore: There are domains that were unseen at any stage of the model development but were given to the model run-time. These include the most strict out-of-distribution cases.

Evaluation details. For each evaluation domain, we randomly sample 10,000 test samples and compute the perplexity. There are three key hyper-parameters in kNN-LM: k, τ, and λ. We report the perplexity using two hyper-parameter tuning methods: (1) we assume a validation set in distribution to the test set to optimize hyper-parameters, and (2) we choose a single configuration for all evaluation domains. We refer to Appendix B for more details.

Baselines. We consider two baselines. The first is the parametric-only LM without any datastore. The second is an oracle single-domain kNN-LM where the model computes the perplexity by iterating over all single-domain datastores and report the best performance as the oracle result.

4.2 Experimental Results

We first demonstrate the importance of datastore composition using pairwise combinations of evaluation domains and single-domain datastores. We then compare our multi-domain datastore with the baselines. In Appendix C.1 and Appendix C.2, we also study the effect of OOD data in the datastore and the effect of the datastore size respectively.

Composition of the datastore does matter. Table 1 shows the perplexity changes when using different single-domain datastores. First, as expected, using the in-domain datastore always gives the largest gains. Domains that contain general information (e.g., Wikipedia and Books3) are helpful for multiple domains, while specific domains like Github and NIH ExPorter can harm other domains. It is important to figure out the optimal target domain when using the single-domain datastore, however we next show our multi-domain setting does not need such prior knowledge.

[1]It is possible the domain is seen during training but not included in the datastore, but this is somewhat an unconventional setting. Therefore, we exclude it from our current scope.

Table 2: Perplexity on 20 evaluation domains. *Prm-only LM* refers to *Parametric-only LM*. We show the percentage of decrease of perplexity by retrieval in parenthesis. '*global*' indicates a single set of hyperparameters is used across all test domains, indicating that the model is **unaware of** any information about the test domain. Multi-domain datastore is on par with single-domain oracle on domains included in a datastore, and outperforms single-domain oracle on domains not in a datastore.

Eval domain	In-distribution		Prm-only LM	kNN-LM		
	Training	Datastore		Single-domain (oracle)	Multi-domain	Multi-domain (global)
Github	✓	✓	2.69	2.50 (-7%)	**2.44** (-9%)	2.61 (-3%)
Wikipedia	✗	✓	19.91	14.99 (-25%)	**14.30** (-28%)	14.32 (-28%)
CC News	✗	✓	23.13	**8.06** (-65%)	8.31 (-64%)	8.56 (-63%)
Books3	✗	✓	18.60	17.29 (-7%)	**16.79** (-10%)	17.63 (-5%)
NIH ExPorter	✗	✓	18.90	**14.72** (-22%)	14.77 (-21%)	14.93 (-21%)
Amazon	✗	✓	35.91	28.27 (-21%)	**27.20** (-24%)	27.20 (-24%)
Enron Emails	✗	✓	13.84	**6.45** (-53%)	6.52 (-53%)	6.65 (-52%)
MIMIC III	✗	✓	19.82	**6.70** (-66%)	6.74 (-66%)	7.35 (-63%)
ArXiv	✗	✗	8.53	8.53 (-0%)	8.53 (-0%)	8.87 (4%)
StackExchange	✗	✗	7.47	7.46 (-0%)	**7.42** (-0%)	7.58 (1%)
OpenSubtitles	✗	✗	18.68	**18.20** (-3%)	18.21 (-3%)	18.37 (2%)
PhilPapers	✗	✗	16.83	16.60 (-1%)	**16.35** (-3%)	16.54 (-2%)
OpenWebText2	✗	✗	23.13	22.32 (-4%)	**21.56** (-7%)	21.61 (-7%)
Pile-CC	✗	✗	21.38	20.91 (-2%)	**20.50** (-4%)	20.60 (-4%)
YoutubeSubtitles	✗	✗	16.58	16.09 (-3%)	**15.55** (-6%)	**15.55** (-6%)
EuroParl	✗	✗	21.20	19.51 (-8%)	**18.54** (-13%)	18.63 (-12%)
BookCorpus2	✗	✗	18.75	17.93 (-4%)	**17.77** (-5%)	**17.77** (-5%)
PubMed Abstracts	✗	✗	17.89	17.30 (-3%)	**17.22** (-4%)	17.51 (-2%)
PubMed Central	✗	✗	12.82	12.82 (-0%)	**12.77** (-0%)	13.15 (3%)
USPTO Backgrounds	✗	✗	10.83	10.83 (-0%)	10.83 (-0%)	11.19 (3%)

Multi-domain datastore performs comparably to single-domain oracle when the evaluation domain is included in the datastore. Both the single-domain design and our multi-domain design helps decrease the perplexity when the evaluation set is in-distribution to the training data. The gains are much larger when the domain is unseen during training than when the domain is seen during training, e.g., 66% vs. 9% relative improvements. Our multi-domain design has comparable performance to the single-domain oracle when the domain is included in the datastore. We hypothesize that using a multi-domain datastore is more beneficial when the evaluation domain is more heterogeneous and thus potentially benefits from a variety of domains, e.g., Wikipedia and Books3, whereas the single-domain datastore is better when the evaluation domain is highly specific and does not benefit from other domains, e.g., Enron Emails and MIMIC III.

A multi-domain datastore matching the in-distribution datastore indicates that kNN-LM is highly robust to out-of-distribution data in the datastore and maintains its performance even when more domains are added. This is different from the parametric models where adding more domains typically hurts performance on individual domains (Oren et al., 2019; Chowdhery et al., 2022; Xie et al., 2023).

Finally, we report results in a global hyper-parameter setting where our multi-domain datastore does not assume the availability of any validation samples. The model still achieves comparable performance to the single-domain oracle datastore, indicating its generalization capability even when there is no prior information about the test data, e.g., when the test data is provided on-the-fly.

Multi-domain datastore outperforms single-domain oracle when the evaluation domain is not in the datastore. When the evaluation domain is not included in the datastore (the bottom block of Table 2), using the multi-domain datastore generally achieves lower perplexity compared with both the LM and the single-domain oracle. The results indicate that the unseen domain is likely to benefit from multiple domains in the datastore, and constraining the datastore to a single domain hurts performance. Nonetheless, the gains brought by are relatively small, e.g., up to 13%, compared to up to 66% on domains that are included in the datastore. Generalizing to domains that are not explicitly included in the datastore is an open problem for future work.

5 Conclusion

In this work, we demonstrate the promise of constructing a generic datastore that performs comparably to a single in-distribution datastore and generalizes to unseen domains. Furthermore, we illustrate that a multi-domain datastore does not require specific information about the test domain to identify which data to use, as long as it includes in-domain data. Together, our results show that kNN-LM is highly robust at out-of-distribution generalization and can effectively target many domains at once, without the oracle domain knowledge assumptions included in previous work.

Acknowledgments and Disclosure of Funding

We thank Jacqueline He and Weijia Shi for the insightful discussions.

References

Baevski, A. and Auli, M. Adaptive input representations for neural language modeling. In *Proceedings of the International Conference on Learning Representations*, 2019.

Borgeaud, S., Mensch, A., Hoffmann, J., Cai, T., Rutherford, E., Millican, K., Van Den Driessche, G. B., Lespiau, J.-B., Damoc, B., Clark, A., et al. Improving language models by retrieving from trillions of tokens. In *Proceedings of the International Conference of Machine Learning*, 2022.

Chowdhery, A., Narang, S., Devlin, J., Bosma, M., Mishra, G., Roberts, A., Barham, P., Chung, H. W., Sutton, C., Gehrmann, S., et al. Palm: Scaling language modeling with pathways. *arXiv preprint arXiv:2204.02311*, 2022.

Gao, L., Biderman, S., Black, S., Golding, L., Hoppe, T., Foster, C., Phang, J., He, H., Thite, A., Nabeshima, N., et al. The Pile: An 800GB dataset of diverse text for language modeling. *arXiv preprint arXiv:2101.00027*, 2020.

Guu, K., Lee, K., Tung, Z., Pasupat, P., and Chang, M. Retrieval augmented language model pre-training. In *Proceedings of the International Conference of Machine Learning*, 2020.

He, R. and McAuley, J. Ups and downs: Modeling the visual evolution of fashion trends with one-class collaborative filtering. In *Proceedings of the World Wide Web Conference*, 2016.

Huang, Y., Gupta, S., Zhong, Z., Li, K., and Chen, D. Privacy implications of retrieval-based language models. *arXiv preprint arXiv:2305.14888*, 2023.

Johnson, A. E., Pollard, T. J., Shen, L., Lehman, L.-w. H., Feng, M., Ghassemi, M., Moody, B., Szolovits, P., Anthony Celi, L., and Mark, R. G. MIMIC-III, a freely accessible critical care database. *Scientific data*, 2016.

Khandelwal, U., Fan, A., Jurafsky, D., Zettlemoyer, L., and Lewis, M. Nearest neighbor machine translation. *arXiv preprint arXiv:2010.00710*, 2020a.

Khandelwal, U., Levy, O., Jurafsky, D., Zettlemoyer, L., and Lewis, M. Generalization through memorization: Nearest neighbor language models. In *Proceedings of the International Conference on Learning Representations*, 2020b.

Khandelwal, U., Fan, A., Jurafsky, D., Zettlemoyer, L., and Lewis, M. Nearest neighbor machine translation. In *International Conference on Learning Representations (ICLR)*, 2021.

Klimt, B. and Yang, Y. The Enron corpus: A new dataset for email classification research. In *Proceedings of European Conference of Machine Learning*, 2004.

Koh, P. W., Sagawa, S., Marklund, H., Xie, S. M., Zhang, M., Balsubramani, A., Hu, W., Yasunaga, M., Phillips, R. L., Gao, I., Lee, T., David, E., Stavness, I., Guo, W., Earnshaw, B. A., Haque, I. S., Beery, S., Leskovec, J., Kundaje, A., Pierson, E., Levine, S., Finn, C., and Liang, P. WILDS: A benchmark of in-the-wild distribution shifts. In *International Conference on Machine Learning (ICML)*, 2021.

Mackenzie, J., Benham, R., Petri, M., Trippas, J. R., Culpepper, J. S., and Moffat, A. CC-News-En: A large english news corpus. In *Proceedings of the ACM International Conference on Information and Knowledge Management*, 2020.

Min, S., Gururangan, S., Wallace, E., Hajishirzi, H., Smith, N. A., and Zettlemoyer, L. SILO language models: Isolating legal risk in a nonparametric datastore. *arXiv preprint arXiv:2308.04430*, 2023a.

Min, S., Shi, W., Lewis, M., Chen, X., Yih, W.-t., Hajishirzi, H., and Zettlemoyer, L. Nonparametric masked language modeling. In *Findings of ACL*, 2023b.

Oren, Y., Sagawa, S., Hashimoto, T. B., and Liang, P. Distributionally robust language modeling. *arXiv preprint arXiv:1909.02060*, 2019.

Piktus, A., Petroni, F., Karpukhin, V., Okhonko, D., Broscheit, S., Izacard, G., Lewis, P., Oğuz, B., Grave, E., Yih, W.-t., et al. The web is your oyster-knowledge-intensive nlp against a very large web corpus. *arXiv preprint arXiv:2112.09924*, 2021.

Presser, S. Books3 corpus, 2020. URL `https://twitter.com/theshawwn/status/1320282149329784833`.

Ram, O., Levine, Y., Dalmedigos, I., Muhlgay, D., Shashua, A., Leyton-Brown, K., and Shoham, Y. In-context retrieval-augmented language models. *Transactions of the Association for Computational Linguistics*, 2023.

Sagawa, S., Koh, P. W., Hashimoto, T. B., and Liang, P. Distributionally robust neural networks for group shifts: On the importance of regularization for worst-case generalization. *arXiv preprint arXiv:1911.08731*, 2019.

Sagawa, S., Koh, P. W., Lee, T., Gao, I., Xie, S. M., Shen, K., Kumar, A., Hu, W., Yasunaga, M., Marklund, H., Beery, S., David, E., Stavness, I., Guo, W., Leskovec, J., Saenko, K., Hashimoto, T., Levine, S., Finn, C., and Liang, P. Extending the wilds benchmark for unsupervised adaptation. In *International Conference on Learning Representations (ICLR)*, 2022.

Shi, W., Min, S., Yasunaga, M., Seo, M., James, R., Lewis, M., Zettlemoyer, L., and Yih, W.-t. REPLUG: Retrieval-augmented black-box language models. *arXiv preprint arXiv:2301.12652*, 2023.

Xie, S. M., Pham, H., Dong, X., Du, N., Liu, H., Lu, Y., Liang, P., Le, Q. V., Ma, T., and Yu, A. W. Doremi: Optimizing data mixtures speeds up language model pretraining. *arXiv preprint arXiv:2305.10429*, 2023.

Zhong, Z., Lei, T., and Chen, D. Training language models with memory augmentation. In *Proceedings of Empirical Methods in Natural Language Processing*, 2022.

Supplementary Material

A Details in Methodology

Approximation in single-domain *k*NN-LM. Computing the full distribution $P_{k\text{NN}}$ with large $\mathcal{D}$ is very expensive. Therefore, most work uses an approximation using the nearest neighbor search index (henceforth called *index*). An index takes an input vector $\mathbf{x}$ and returns k number of tuples, $(i_1, \text{dist}(\mathbf{d}_{i_1}, \mathbf{x})), \cdots, (i_k, \text{dist}(\mathbf{d}_{i_k}, \mathbf{x}))$, denoted as $\mathcal{N}(\mathbf{x}|\mathcal{D})$, where $i_1...i_k = \text{argMin}_{1\leq i\leq N}\text{dist}(\mathbf{d}_i, \mathbf{x})$. Then, $P_{k\text{NN}}(y|x_1...x_n; \mathcal{D})$ is computed as:

$$\frac{\sum_{(i,d)\in\mathcal{N}(\text{Enc}(x_1...x_n)|\mathcal{D})} \mathbb{I}[d_i = y]\exp(-d/\tau)}{\sum_{(i,d)\in\mathcal{N}(\text{Enc}(x_1...x_n)|\mathcal{D})} \exp(-d/\tau)}$$

Constructing multi-domain *k*NN-LM. When there are M datasets under multiple domains: $\mathcal{D}_1...\mathcal{D}_M$, a naive method is to consider a union of datasets $\mathcal{D} = \cup_{1\leq m\leq M}\mathcal{D}_m$ and use the method that is the same as a single-domain kNN-LM. This was used in prior work that explores a multi-domain datastore (Khandelwal et al., 2021; Huang et al., 2023). However, adding or removing the domain in the datastore requires a reconstruction of an entirely new index, which can be expensive.

Instead, we construct the multi-domain datastore by re-using the index constructed for each domain data and aggregating the results during run-time. Formally, $P_{k\text{NN}}(y|x_1...x_n; \mathcal{D}_1...\mathcal{D}_M)$ is defined as:

$$\frac{\sum_{1\leq m\leq M}\sum_{(i,d)\in\mathcal{N}(\text{Enc}(x_1...x_n)|\mathcal{D}_m)} \mathbb{I}[d_i = y]\exp(-d/\tau)}{\sum_{1\leq m\leq M}\sum_{(i,d)\in\mathcal{N}(\text{Enc}(x_1...x_n)|\mathcal{D}_m)} \exp(-d/\tau).}$$

This offers more flexibility: adding or removing the domain data does not require reconstructing the index, and any update in the m-th domain data only requires updating the m-th index while the rest of the indices can remain the same. This allows us to efficiently study the effect of different compositions of the datastore at scale, at the cost of increasing the time required for the nearest neighbor query (since we query each domain separately).

B Evaluation Setup

Table 3: Composition of our 5B multi-domain datastore.

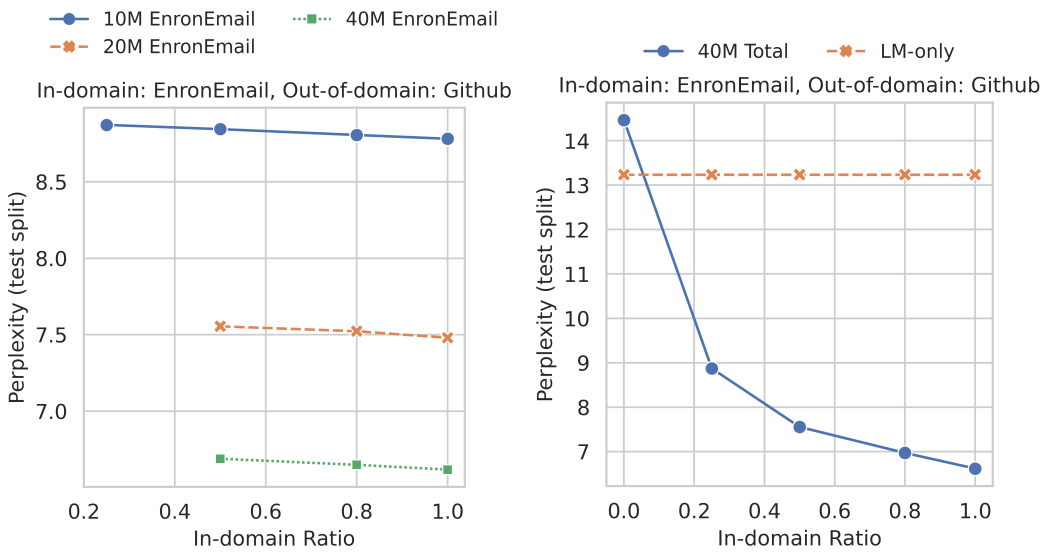
Domain	Github	Wikipedia	CC News	Books3	NIH ExPorter	Amazon	Enron Emails	MIMIC III
#Tokens	1000M	1000M	400M	1000M	100M	1000M	50M	500M

Details in eight domains included in the datastore. We first take five domains from the Pile: the **Github** codes, grant abstracts from the **NIH ExPorter**, English **Wikipedia** articles, the **Books3** (Presser, 2020), and emails from **Enron Emails** (Klimt & Yang, 2004). Additionally, we consider news articles from **CC-News** (Mackenzie et al., 2020), **Amazon** product reviews from He & McAuley (2016), and clinical notes from **MIMIC-III** (Johnson et al., 2016). This choice was made following Min et al. (2023a).

For each domain, we randomly sample up to 1B tokens from the training set to build the datastore. The composition of the 5B multi-domain datastore is provided in Table 3. For the single-domain baseline, we take each column as one single-domain datastore.

Evaluation metrics. We report language modeling perplexity as the evaluation metric. We merge all text into one stream of text and split them into batches with a maximum sequence length of 1,024 and a sliding window of 512, a setup that is standard in prior language modeling literature (Baevski & Auli, 2019; Khandelwal et al., 2020b).

Hyperparameter search. When we use the same set of hyperparameters for all evaluation domains, we choose the hyperparameters based on the average performance of the domains included in the datastore. The chosen hyperparameters are $k = 4096, \tau = 20, \lambda = 0.5$ for the eight domains



(a) The size of the in-domain data (i.e., Enron Emails) is fixed and the in-domain ratio varies.

(b) The size of the datastore is fixed and the in-domain ratio varies.

Figure 1: Perplexity on Enron Emails with a datastore built with in-domain Enron Email data and OOD Github data.

included in the datastore, and $k = 4096, \tau = 20, \lambda = 0.2$ for the other domains excluded in the datastore, where we reduced the λ to 0.2 as the unseen domains are supposed to benefit less from the datastore.

C Additional Results

C.1 Effect of the out-of-distribution data in the datastore

We can measure the effect of out-of-domain (OOD) data in the datastore using Enron Emails as the in-domain data and the Github as the OOD data. As shown in Figure 1a, adding OOD data to the datastore (i.e., decreasing the in-domain ratio) can negatively impact performance, albeit to a moderate extent. However, the absolute size of in-domain data is crucial to performance. Figure 1b shows that, with a fixed total number of tokens in the datastore, changing the ratio of the in-domain data can drastically improve performance.

C.2 Effect of the datastore size

We also present an ablation study on datastore size. We show the effect of multi-domain datastore size on the performance of kNN-LM in Table 4. We build another multi-domain datastore that has 3B tokens in total with up to 0.5B tokens per domain. Increasing the number of tokens of one domain in the multi-domain datastore consistently improve the corresponding performance. For example, it further decreases the perplexity of Github, Wikipedia, Books3, and Amazon when including more in-distribution tokens in the multi-domain datastore. In the 3B datastore, we keep the tokens for CC News, NIH ExPorter, Enron Emails, and MIMIC III to be the same as in the 5B datastore. We found that increasing OOD data might even help improve the kNN-LM.

Table 4: The effect of the datastore size on the performance. The 3B datastore is constructed by reducing the sizes of the domains marked "*" from 1B tokens to 0.5B tokens. Other domains that have less than 0.5B tokens maintain the same sizes.

Domain	Github*	Wikipedia*	CC News	Books3*	NIH ExPorter	Amazon*	Enron Emails	MIMIC III
LM	2.695	19.913	23.134	18.601	18.901	35.915	13.840	19.826
3B Datastore	2.507	14.948	8.353	17.149	14.785	28.331	6.530	**6.744**
5B Datastore	**2.444**	**14.309**	**8.317**	**16.790**	**14.777**	**27.209**	**6.524**	6.747