On Retrieval Augmentation and the Limitations of Language Model Training

Ting-Rui Chiang **Xinyan Velocity Yu** **Joshua Robinson**
Ollie Liu **Isabelle Lee** **Dani Yogatama**
University of Southern California
{tingruic,xinyany,joshua.j.robinson,zliu2898,gunheele,yogatama}@usc.edu

Abstract

Augmenting a language model (LM) with k-nearest neighbors (kNN) retrieval on its training data alone can decrease its perplexity, though the underlying reasons for this remain elusive. In this work, we rule out one previously posited possibility — the "softmax bottleneck." We then create a new dataset to evaluate LM generalization ability in the setting where training data contains additional information that is not causally relevant. This task is challenging even for GPT-3.5 Turbo. We show that, for both GPT-2 and Mistral 7B, kNN retrieval augmentation consistently improves performance in this setting. Finally, to make kNN retrieval more accessible, we propose using a multi-layer perceptron model that maps datastore keys to values as a drop-in replacement for traditional retrieval. This reduces storage costs by over 25x.[1]

1 Introduction

Recent efforts to improve the performance of language models (LMs) have focused on scaling up model (Brown et al., 2020) and training data size (Hoffmann et al., 2022). The resulting models have reached near-human or even super-human performance on some tasks (Chowdhery et al., 2022), though with steep accompanying energy and compute resource costs (Schwartz et al., 2020; Brown et al., 2020; Touvron et al., 2023).

Another approach for improving LM performance has been retrieval augmentation. Khandelwal et al. (2020) proposed to build a datastore using LM training data. The datastore associates the next token of prefixes in the training data with the representations of the prefixes extracted from an intermediate layer of an LM. They found that when predicting the next token for a given prefix, using k-nearest neighbor (kNN) retrieval, which retrieves the next token based on the intermediate representation of a given prefix, reduced language models' perplexity. Because the datastore is drawn entirely from the LM's training data, the success of kNN augmentation suggests the standard LM training setup does not yield models that best utilize their parametric capacity or training data. Studying why LMs augmented with kNN retrieval (kNN-LMs) outperform vanilla LMs may shed light on ways to improve the standard LM training setup.

In this work, we base our study on the analyses of kNN-LMs by Xu et al. (2023). Among the aspects they explore are the limitations of model architecture and memorization. Xu et al. (2023) suggest the kNN component may be able to map intermediate representation of context to distributions in a more flexible way, while the last layer of LMs has a softmax bottleneck (Yang et al., 2018) that restricts LMs from generating certain distributions. This discrepancy of expressiveness may thus cause the performance gap. They also show that replacing the kNN component with an overfitted LM performs worse than kNN-LM, suggesting that kNN augmentation does not perform better solely because it memorizes the training data better.

In this work, we start with inspecting the bottlenecks in the model as suggested by Xu et al. (2023). We propose an experiment that shows that the softmax bottleneck is not the cause of the performance gap between kNN and vanilla LM. Our experimental results show that the last linear layers of LMs can generate distributions that approximate the distribution from a kNN-LM well. Therefore, we conclude that the bottleneck issues in the last layers, including the softmax bottleneck issue, are not the cause of the performance gap.

We then investigate the performance gap from the perspective of generalization. This explains why an overfitted LM is less effective than a kNN retrieval component (Xu et al., 2023). We identify a scenario which we refer to as *over-specification*.

[1]The source code is available at https://github.com/usc-tamagotchi/on-knnlm.

Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 2: Short Papers), pages 229–238

That is, when a statement about certain knowledge (e.g., relational knowledge (Petroni et al., 2019) or commonsense (Speer et al., 2017; Young et al., 2018; Sap et al., 2019)) contains redundant information. We create a synthetic dataset *Macondo* and use it to show that over-specification in training data prevents LMs from learning the knowledge in a robust way, i.e., LMs cannot generalize to test data which is not over-specified. Even GPT-3.5 Turbo, fails, indicating it is a fundamental limitation of LM training. It may be crucial when the size of training data is limited, because in this scenario, it is likely that there are only few statements about certain knowledge and all of them are over-specified. Decounfounding the effect of having redundant information also requires more training examples. This may explain why we need to scale up the training data size.

Because the better generalization ability may be what makes the kNN component helpful, we explore alternatives to a kNN component by looking for components that also generalize well. It turns out that we can close the generalization gap on Moncodo by training another neural model that maps the intermediate representation to the target token. We also show that on the WikiText dataset, this approach reduces the perplexity by 1.45 while requiring less than 4% storage space of kNN augmentation. We suggest it is a promising future direction for improving LMs.

2 Background and Notations

LM We focus on Transformer LMs such as GPT-2. Given context $c = \{x_i\}_{i=1}^{t-1}$, we formulate next token prediction as

$$p_{\text{lm}}(x_t|c) = f \circ \text{g} \circ \text{enc}(c), \tag{1}$$

where f is the last linear layer with softmax activation, g is the two-layer MLP network with a residual connection in the last Transformer layer, and enc includes the earlier layers of the model.

kNN-LM Khandelwal et al. (2020) use the enc function from a trained LM (Eq 1) to build a datastore, where a key is the representation of a token sequence $\{x_i\}_{i=1}^{t-1}$ in the training data encoded by enc, and the value of the key is the next token x_t. When predicting the next token x'_t of given context $c = \{x'_i\}_{i=1}^{t-1}$, kNN-LM has a kNN retrieval module that maps $\text{enc}(c)$ to a distribution $p_{\text{knn}}(\cdot|c)$ by querying the datastore with $\text{enc}(c)$. Then a kNN-LM generates the next token distribution with

$$p_{\text{knnlm}}(x_t|c) = \lambda p_{\text{lm}}(x_t|c) + (1-\lambda)p_{\text{knn}}(x_t|c),$$

where λ is a hyperparameter for interpolation.

Original LMs		p_{proj} projected with Eq. 2	
LM	kNN-LM	$f \rightarrow p_{\text{knnlm}}$	$f \circ g \rightarrow p_{\text{knnlm}}$
20.13	16.92	16.76	16.78

Table 1: The perplexity of the LMs discussed in §3.

Softmax bottleneck Yang et al. (2018) theoretically show that the dimensionality of the last linear layer confines the possible vocabulary distribution the last softmax layer can generate. It implies that no matter what $g \circ \text{enc}$ generates, f can not generate certain distributions.

3 Capacity of LMs' Last Layers

Xu et al. (2023) hypothesize that the performance gap between kNN-LM and vanilla LM is because the softmax bottleneck prevents it from generating some distributions that kNN-LM can generate. In this section, we reinspect this hypothesis.

3.1 Projecting to the Probability Space

We study whether softmax bottleneck causes the performance gap by inspecting whether the last layers can generate a distribution that approximates the distribution generated by kNN-LM p_{knnlm}. We do the projection by solving

$$z^* \in \arg\min_{z \in \mathbb{R}^d} \text{KL}[f(z) \| p_{\text{knnlm}}], \tag{2}$$

where f is the last layer of the model with its trained parameters fixed (definition in Eq 1). By definition, if softmax bottleneck really prevents the model from generating p_{knnlm}, then $f(z^*)$ can not approximate p_{knnlm} well and thus its perplexity should be close to the vanilla LM's. Therefore, by comparing the perplexity of $p_{\text{proj}} = f(z^*)$ with vanilla LM's and kNN-LM's perplexity, we can infer the effect of softmax bottleneck in this problem.

Similarly, we can inspect whether the MLP layer has a bottleneck effect by replacing f in Eq 2 with $f \circ g$. We use $\text{enc}(\{x_i\}_{i=1}^{t-1})$ as the initialization of z and solve Eq 2 with gradient descent.

3.2 Experiment, Result, and Discussion

We train an LM using WikiText following the setting in Khandelwal et al. (2020) and measure its

perplexity (details in §A). Table 1 shows that the approximation of p_{knnlm} by the last layer f has an average perplexity similar to the perplexity of kNN-LM. The average KL-divergence between p_{knnlm} and p_{proj} is also under 0.1 (Table 3). These results imply that the approximation is good enough for a good perplexity. It also implies the softmax bottleneck does not prevent the LM from generating a good distribution. Thus, the softmax bottleneck is not the cause of the gap between vanilla LM and kNN-LM. Projecting the p_{knnlm} to the output space of $f \circ g$ has a similar result. Therefore, LMs' last layers do not have a bottleneck that causes the performance gap. [2]

4 Generalization from Over-specification

As the last layers do not have a bottleneck issue that explains the performance gap, we turn to study the efficacy of kNN augmentation from the perspective of generalization. In this section, we identify a limitation of LM training that may cause the performance gap: The failure to generalize from *over-specified* descriptions.

4.1 Over-specification

We refer to the phenomenon that the prefix of a partial sentence contains information that is not causally relevant to its completion as *over-specification*. In other words, over-specification is the scenario where removing some information in the prefix (e.g. a phrase) does not change the likelihood of the continuation. This phenomenon often occurs in in the training data. The descriptions about factual knowledge or commonsense are usually over-specified with non-causally relevant information, but the causally irrelevant information may be absent during inference. Generalization from over-specified training data is thus important for an LM to utilize knowledge in the training data.

For example, in the training data, the text about the knowledge "being drunk" implies "dangerous to drive" may be over-specified as "I was drunk when I left the party, so it was dangerous to drive". In this example, "I was drunk" is causally relevant to "it was dangerous to drive" but "when I left the party" is not. An ideal LM should generalize and predict the same continuation when the non-causal information "when I left the party" is absent.

[2]However, we find it more difficult to solve Eq. 2 with a smaller learning rate for $f \circ g$. More discussions in §E.

4.2 Dataset: Macondo

We create a synthetic dataset Macondo to demonstrate the challenge of generalizing from over-specified training data. This dataset contains the names of 500 villagers, where 100 villagers have 1 to 5 child(ren), and each villager has a unique full name consisting of a random combination of a first name and a last name. Each child has a single-token and distinct first name. We construct each sentence in the training set using the template "`[villager]`, who `[desc]`, is the parent of `[child]`", where "`[desc]`" is a verb phrase about an attribute of the villager that is irrelevant to the parent-child relationship. As for the sentences in the test set, they follow the template "`[villager]`, is the parent of `[child]`". A perfect LM should predict each child of the villager with probability $\log(1/\#\text{ of children})$. (More details in §B.1)

4.3 Experiment, Results, and Discussion

To inspect how LMs are (un)able to generalize from over-specified training data, we fine-tune GPT-2 XL models with Macondo and test it with the test set where irrelevant "`[desc]`" is absent (details in §C). Figure 1 shows that the fine-tuned GPT-2 model has a likelihood much lower than the theoretical perfect likelihood ($\log(1/\#\text{ of children})$). It indicates that it cannot generalize from over-specification. Additionally, Figure 1 shows that the kNN-augmented model performs better than the vanilla model. The better generalization capability of an augmented LM may partly explain the performance gap between augmented and vanilla LMs. We also experiment with GPT-2 small models (Figure 3a) and find that GPT-2 XL models do not generalize much better, suggesting that scaling up the model may not close this generalization gap. We observe similar performance trend when fine-tuning a Mistral-7B-v0.1 model in Section C.1.

4.4 Experimenting with GPT-3.5-turbo

To inspect whether scaling mitigates the challenge of generalization, we further experiment with GPT-3.5-turbo. We construct a conversational version of Macondo, Macondo-Conv to fit the conversational format of GPT-3.5-turbo. In the training set, sentences follow the template "User: Who is the child of `[villager]`, the one who `[desc]`? Assistant: `[child]`.". The test examples follow the template "User: Who is the child of `[villager]`? Assistant: `[child]`.". The dataset contains 125 villagers

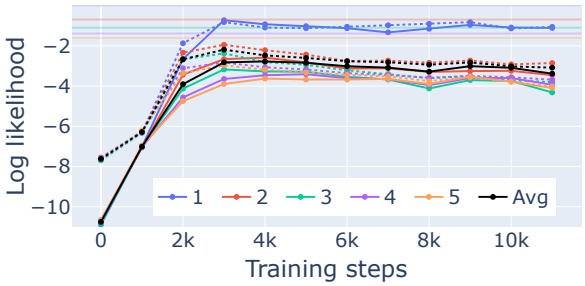


Figure 1: Test log-likelihood of children names in our synthetic dataset Macondo predicted by a fine-tuned GPT-2 XL model for parents with 1-5 children (average of **5** random seeds). The dotted lines represent the results of the kNN augmented LM. The horizontal lines represent the theoretically best log-likelihood a perfect model can achieve ($\log(1/\#\text{ of children})$). See Table 5 for the exact statistics shown in this figure.

Macondo		WikiText	
LM	kNN/MLP-LM	LM	kNN/MLP-LM
19.66	17.69 / 10.76	20.13	16.92 / 18.68

Table 2: The perplexity of LMs augmented with different a kNN model or a MLP model (§5).

having 2 children for lower fine-tuning costs.

The result in Figure 2 shows that GPT-3.5-turbo can not generalize to a test set without overspecification. This suggests that scaling up the model size alone cannot solve this generalization challenge. This failure to generalize may be a fundamental limitation of LM training.

5 An Alternative to kNN-augmentation

Motivated by the results in §4.3, we explore whether using a datastore is necessary to improve perplexity. The success of kNN-augmentation in §4.3 shows that it is possible to generalize better by utilizing the intermediate representation with a kNN module. We wonder whether we can use a classification model instead of a kNN module.

Because deep models have been known for their generalization ability (Neyshabur et al., 2015, 2019), we explore using an MLP model to replace kNN retrieval. We use the key-value pairs in the datastore for kNN retrieval to train an MLP model to map the keys to the values (details in §D). Results in Table 2 show that interpolating the original LM with this MLP model effectively reduces the perplexity while requiring less than 4% of storage. This indicates a promising future direction.

Figure 2: GPT-3.5-turbo fine-tuned with Macondo-Conv using OpenAI API. The results are the average of 5 runs with 5 datasets generated with 5 random seeds. Note that the presented loss involves special tokens, e.g., end-of-string tokens, so the theoretical perfect likelihood is greater than $\log 0.5$. The gray line is the test loss we achieve when we use the test data to train the model.

6 Related Work

LMs that solely rely on parametric knowledge learned during training time are known to hallucinate (Shuster et al., 2021; Dhuliawala et al., 2023; Zhang et al., 2023a; Ye et al., 2023; Zhang et al., 2023b), suffer to learn long-tail knowledge (Roberts et al., 2020), and fail to adapt to new knowledge over time (De Cao et al., 2021; Chen et al., 2021; Kasai et al., 2022). To overcome these limitations, recent works (Khandelwal et al., 2020; Lewis et al., 2020; Guu et al., 2020; Yogatama et al., 2021; Borgeaud et al., 2022; Izacard et al., 2023; Zhong et al., 2022; Min et al., 2023) include an external datastore with the parametric model, resulting in a retrieval-augmentated model paradigm. Meanwhile, Drozdov et al. (2022) and Wang et al. (2023) analyzes the effect of kNN-LM on generation tasks, while Shi et al. (2022) focuses on using kNN-LM on few- and zero-shot classification tasks.

The traditional LM training setup has been shown to yield models that fail to generalize to test data with reversed relations (Berglund et al., 2023), respective readings (Cui et al., 2023), and longer tasks (Anil et al., 2022). These models can also struggle with linguistic generalization between unseen but related contexts (Wilson et al., 2023) and learn shortcuts that harm generalization (McCoy et al., 2019). Bender and Koller (2020) have also argued that such models will necessarily be limited due to the ungrounded nature of their training data.

7 Conclusion

We study the performance gap between vanilla and kNN-augmented LMs. We develop an experiment that allows us to directly inspect the bottleneck is-

sue and exclude the possibility that it causes the performance gap (§3). We further identify the over-specified scenario where vanilla LMs fail to generalize while kNN-LMs generalize better (§4). We also show with GPT-3.5-turbo that this failure of generalization can not be solved by scaling up the model size, suggesting that this is a fundamental limitation of LM training. Finally, we show the potential of augmenting LMs with an MLP model, indicating a promising future direction (§5).

Limitations

While we gain more insights by closely inspecting the phenomena observed by Xu et al. (2023), why kNN augmentation is beneficial remains not fully clear. In §3, we focus on the bottleneck issues of the last layers $f \circ g$ and show that there exists an intermediate representation z^* such that $f \circ g(z^*)$ approximates p_{knnlm} well. However, it is unclear why enc does not map the context to z^*. In §4, we identify the over-specification scenario where kNN-LMs generalize better than vanilla LMs. However, the mechanism behind this remains unclear. In §5, we show that augmenting LMs with another MLP can improve the perplexity of the model but does not fully close the gap between kNN-LM and vanilla LM on WikiText. Further analysis is required to understand the generalization behavior of the kNN and the MLP models.

References

Cem Anil, Yuhuai Wu, Anders Andreassen, Aitor Lewkowycz, Vedant Misra, Vinay Ramasesh, Ambrose Slone, Guy Gur-Ari, Ethan Dyer, and Behnam Neyshabur. 2022. Exploring length generalization in large language models.

Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. 2016. Layer normalization. *arXiv preprint arXiv:1607.06450*.

Emily M. Bender and Alexander Koller. 2020. Climbing towards NLU: On meaning, form, and understanding in the age of data. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 5185–5198, Online. Association for Computational Linguistics.

Lukas Berglund, Meg Tong, Max Kaufmann, Mikita Balesni, Asa Cooper Stickland, Tomasz Korbak, and Owain Evans. 2023. The reversal curse: Llms trained on "a is b" fail to learn "b is a".

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 *Proceedings of International conference on machine learning*.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel Ziegler, Jeffrey Wu, Clemens Winter, Chris Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. 2020. Language models are few-shot learners. In *Advances in Neural Information Processing Systems*.

Wenhu Chen, Xinyi Wang, and William Yang Wang. 2021. A dataset for answering time-sensitive questions. In *Proceedings of the Conference on Neural Information Processing Systems Datasets and Benchmarks Track*.

Aakanksha Chowdhery, Sharan Narang, Jacob Devlin, Maarten Bosma, Gaurav Mishra, Adam Roberts, Paul Barham, Hyung Won Chung, Charles Sutton, Sebastian Gehrmann, et al. 2022. Palm: Scaling language modeling with pathways. *arXiv*.

Ruixiang Cui, Seolhwa Lee, Daniel Hershcovich, and Anders Søgaard. 2023. What does the failure to reason with "respectively" in zero/few-shot settings tell us about language models? In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 8786–8800, Toronto, Canada. Association for Computational Linguistics.

Nicola De Cao, Wilker Aziz, and Ivan Titov. 2021. Editing factual knowledge in language models. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 6491–6506, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Shehzaad Dhuliawala, Mojtaba Komeili, Jing Xu, Roberta Raileanu, Xian Li, Asli Celikyilmaz, and Jason Weston. 2023. Chain-of-verification reduces hallucination in large language models. *arXiv*.

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.

Kelvin Guu, Kenton Lee, Zora Tung, Panupong Pasupat, and Mingwei Chang. 2020. Retrieval augmented language model pre-training. In *Proceedings of the International Conference on Machine Learning*.

Jordan Hoffmann, Sebastian Borgeaud, Arthur Mensch, Elena Buchatskaya, Trevor Cai, Eliza Rutherford, Diego de Las Casas, Lisa Anne Hendricks, Johannes Welbl, Aidan Clark, Tom Hennigan, Eric Noland, Katie Millican, George van den Driessche, Bogdan Damoc, Aurelia Guy, Simon Osindero, Karen Simonyan, Erich Elsen, Jack W. Rae, Oriol Vinyals, and L. Sifre. 2022. Training compute-optimal large language models. *arXiv*.

Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. 2021. Lora: Low-rank adaptation of large language models. *arXiv preprint arXiv:2106.09685*.

Gautier Izacard, Patrick Lewis, Maria Lomeli, Lucas Hosseini, Fabio Petroni, Timo Schick, Jane Dwivedi-Yu, Armand Joulin, Sebastian Riedel, and Edouard Grave. 2023. Atlas: Few-shot learning with retrieval augmented language models. *Journal of Machine Learning Research*.

Albert Q Jiang, Alexandre Sablayrolles, Arthur Mensch, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Florian Bressand, Gianna Lengyel, Guillaume Lample, Lucile Saulnier, et al. 2023. Mistral 7b. *arXiv preprint arXiv:2310.06825*.

Jungo Kasai, Keisuke Sakaguchi, Yoichi Takahashi, Ronan Le Bras, Akari Asai, Xinyan Yu, Dragomir Radev, Noah A Smith, Yejin Choi, and Kentaro Inui. 2022. Realtime qa: What's the answer right now? *arXiv*.

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

Diederik Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *International Conference on Learning Representations (ICLR)*, San Diega, CA, USA.

Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, et al. 2020. Retrieval-augmented generation for knowledge-intensive nlp tasks. In *Advances in Neural Information Processing Systems*.

Tom McCoy, Ellie Pavlick, and Tal Linzen. 2019. Right for the wrong reasons: Diagnosing syntactic heuristics in natural language inference. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 3428–3448, Florence, Italy. Association for Computational Linguistics.

Sewon Min, Weijia Shi, Mike Lewis, Xilun Chen, Wen-tau Yih, Hannaneh Hajishirzi, and Luke Zettlemoyer. 2023. Nonparametric masked language modeling. In *Findings of the Association for Computational Linguistics: ACL 2023*, pages 2097–2118, Toronto, Canada. Association for Computational Linguistics.

Behnam Neyshabur, Zhiyuan Li, Srinadh Bhojanapalli, Yann LeCun, and Nathan Srebro. 2019. The role of over-parametrization in generalization of neural networks. In *International Conference on Learning Representations*.

Behnam Neyshabur, Ryota Tomioka, and Nathan Srebro. 2015. In search of the real inductive bias: On the role of implicit regularization in deep learning. In *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Workshop Track Proceedings*.

Fabio Petroni, Tim Rocktäschel, Sebastian Riedel, Patrick Lewis, Anton Bakhtin, Yuxiang Wu, and Alexander Miller. 2019. Language models as knowledge bases? 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 2463–2473, Hong Kong, China. Association for Computational Linguistics.

Adam Roberts, Colin Raffel, and Noam Shazeer. 2020. How much knowledge can you pack into the parameters of a language model? In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 5418–5426, Online. Association for Computational Linguistics.

Maarten Sap, Ronan Le Bras, Emily Allaway, Chandra Bhagavatula, Nicholas Lourie, Hannah Rashkin, Brendan Roof, Noah A. Smith, and Yejin Choi. 2019. Atomic: An atlas of machine commonsense for if-then reasoning. *Proceedings of the AAAI Conference on Artificial Intelligence*, 33(01):3027–3035.

Roy Schwartz, Jesse Dodge, Noah A Smith, and Oren Etzioni. 2020. Green ai. *Communications of the ACM*.

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.

Kurt Shuster, Spencer Poff, Moya Chen, Douwe Kiela, and Jason Weston. 2021. Retrieval augmentation reduces hallucination in conversation. In *Findings of the Association for Computational Linguistics: EMNLP 2021*, pages 3784–3803, Punta Cana, Dominican Republic. Association for Computational Linguistics.

Robyn Speer, Joshua Chin, and Catherine Havasi. 2017. Conceptnet 5.5: An open multilingual graph of general knowledge. In *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence*, AAAI'17, page 4444–4451. AAAI Press.

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, et al. 2023. Llama 2: Open foundation and fine-tuned chat models. *arXiv*.

Shufan Wang, Yixiao Song, Andrew Drozdov, Aparna Garimella, Varun Manjunatha, and Mohit Iyyer. 2023. Knn-lm does not improve open-ended text generation. *arXiv*.

Michael Wilson, Jackson Petty, and Robert Frank. 2023. How Abstract Is Linguistic Generalization in Large Language Models? Experiments with Argument Structure. *Transactions of the Association for Computational Linguistics*, 11:1377–1395.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Remi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander Rush. 2020. Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 38–45, Online. Association for Computational Linguistics.

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

Zhilin Yang, Zihang Dai, Ruslan Salakhutdinov, and William W. Cohen. 2018. Breaking the softmax bottleneck: A high-rank RNN language model. In *Proceedings of the International Conference on Learning Representations*.

Hongbin Ye, Tong Liu, Aijia Zhang, Wei Hua, and Weiqiang Jia. 2023. Cognitive mirage: A review of hallucinations in large language models. *arXiv*.

Dani Yogatama, Cyprien de Masson d'Autume, and Lingpeng Kong. 2021. Adaptive semiparametric language models. *Transactions of the Association for Computational Linguistics*, 9:362–373.

Tom Young, Erik Cambria, Iti Chaturvedi, Hao Zhou, Subham Biswas, and Minlie Huang. 2018. Augmenting end-to-end dialogue systems with commonsense knowledge. In *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence and Thirtieth Innovative Applications of Artificial Intelligence Conference and Eighth AAAI Symposium on Educational Advances in Artificial Intelligence*, AAAI'18/IAAI'18/EAAI'18. AAAI Press.

Muru Zhang, Ofir Press, William Merrill, Alisa Liu, and Noah A Smith. 2023a. How language model hallucinations can snowball. *arXiv*.

Yue Zhang, Yafu Li, Leyang Cui, Deng Cai, Lemao Liu, Tingchen Fu, Xinting Huang, Enbo Zhao, Yu Zhang, Yulong Chen, et al. 2023b. Siren's song in the ai ocean: A survey on hallucination in large language models. *arXiv*.

Zexuan Zhong, Tao Lei, and Danqi Chen. 2022. Training language models with memory augmentation. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 5657–5673, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

A Experiment Details of §3

A.1 Hyperparameters for the Baseline Models

We implement kNN-LM based on the package `transformers 4.34.0` (Wolf et al., 2020). We train a 16-layer transformer model following the hyperparameters used by Khandelwal et al. (2020) and Xu et al. (2023). We use $k = 1024$, $\lambda = 0.25$ and L2 distance for kNN retrieval. Please refer to the repository of Xu et al. (2023) (`https://github.com/frankxu2004/knnlm-why`) for more details about datastore building.

A.2 Hyperparameters for Solving Eq. 2

We use learning rate 0.1 and Adam optimizer (Kingma and Ba, 2015) for solving Eq. 2 using gradient descent. We do gradient descent until the update changes the KL-divergence is by less than 0.001.

B Details about the Macondo Dataset

B.1 Generation Process

We construct the Macondo dataset using the template "`[villager]`, who `[desc]`, is the parent of `[child]`". In each example, the "`[villager]`" placeholder is replaced with a villager's full name. We generate the full name of a villager by randomly sampling a given name from a corpora by Mark Kantrowitz and a surname from a list of the most common surnames under Creative Commons Attribution 4.0 International License. The "`[villager]`" placeholder is replaced with a single-token given name from the corpora by Mark Kantrowitz. We associate each villager with 6 attributes described below. When generating an example in the training set, we randomly sample one of the six attributes and replace the "`[desc]`" placeholder with a relative clause describing the attribute:

- Year of birth: "who was born in `[year]`". The year is randomly sampled between 1800 and 2005.
- City of birth: "who was born in `[city]`". The city is randomly sampled from a word city database by simplemaps under the license Creative Commons Attribution 4.0.

- Living city: "who used to live in `[city]`". The city is randomly sampled from the word city database by simplemaps.
- Friend: "who was a friend of `[villager]`".
- Graduate from: "who graduated from `[university]`". The list of university is from THE world university ranking.
- Marry year: "who married in `[year]`". The year is randomly sampled between 1800 and 2023 and is guaranteed to be at least 18 years after the year of birth.
- Work: "who used to work for `[company]`". The company is randomly sampled from a list of California Companies on Wikipedia.

Table 6 contains some examples in this dataset. We have 1500 examples in total.

B.2 The Conversational Version

We use the `tiktoken` tokenizer to ensure that the names of the villagers' children are single-token. Table 7 contains some examples in this dataset.

C Experiment Details of §4

We fine-tuned GPT-2 small and GPT-2 XL with a warm-up ratio equal to 0.05, batch size 4, and Adam optimizer (Kingma and Ba, 2015) for 50 epochs. We use the default hyperparameters of the Trainer API of the `transformers` package (Wolf et al., 2020), i.e., learning rate 1e-5, max gradient norm 1.0, etc. We use version 0613 for our experiments that use GPT-3.5 Turbo. We execute this experiment with NVIDIA RTX A6000 GPUs.

C.1 Additional Experiments with Mistral

We report additional Macondo experiments conducted on a more capable model, namely Mistral-7B-v0.1 (Jiang et al., 2023). We follow the same dataset setup as in 4.3, and fine-tune the Mistral model with LoRa (Hu et al., 2021). We report performance curves in Figure 3b, and attain qualitatively similar observations to those in Section 4.3. Our experiments add favorable evidence that neither concurrent methods in pre-training language models nor model scaling is an effective solution for circumventing over-specification. But kNN-augmented language models can partially reduce the optimality gap between the backbone language model and a perfect model.

f	$f \circ g$
0.07	0.10

Table 3: The minimum KL-Divergence achievable by solving Eq 2 with gradient descent.

Original LMs		p_{proj} projected with Eq. 2		
LM	kNN-LM	f	$f \circ g$	$f \circ g \to y$
20.13	16.92	16.70	19.97	19.41

Table 4: The perplexity of projecting to LMs' output space as discussed in §3 when using learning rate 0.001.

Mistral Fine-Tuning Details. Following standard practices, we add LoRa adaptors to the embedding matrix, to the query, key, value, and output projections of each attention layer, as well as to all projections of each MLP layer. We set the rank of all update matrices to be 8, the LoRa scaling factor to be 16, and a LoRa dropout probability of 0.05. We use a warm-up ratio of 0.05, and train with a global batch size of 128 using the Adam optimizer (Kingma and Ba, 2015). We use default hyperparameters of the Hugging Face Trainer API to fine-tune the model for 30 epochs.

D Experiment Details of §5

We use a learning rate of 1e-5 to train an MLP model that maps the keys in the datastore to the values. The batch size is the same as the number of tokens in each batch when training the vanilla language model, i.e., 3×3072. For Macondo, we train the model for 10 epochs. For WikiText, we train the model for 2 epochs. The model architecture is the same as the last MLP layer of the vanilla language model, i.e.

$$\text{logits} = W(z + \text{LN} \circ \text{MLP}(z)),$$

where the MLP model has 1 hidden layer with the hidden size 4096 and LN is the layer normalization module (Ba et al., 2016). We execute this experiment with RTX 2080Ti GPUs.

E A Potential *MLP Hurdle*

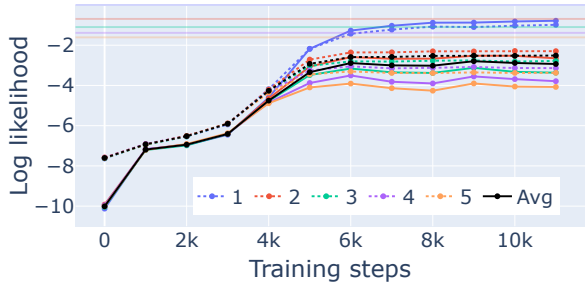
Even though we can solve the optimization problem in Eq. 2 with a learning rate of 0.1, we find it more difficult to solve it for $f \circ g$ with a learning rate below 0.001. Table 4 shows the perplexity of solving Eq. 2 using a learning rate below 0.001 for 100 steps. The perplexity of projecting to the

output space of f is much lower. We suggest that it may cause some challenges in optimizing enc because it seems that the gradient can not flow to enc easily when the learning rate is small. We refer to this as a potential *MLP hurdle*.

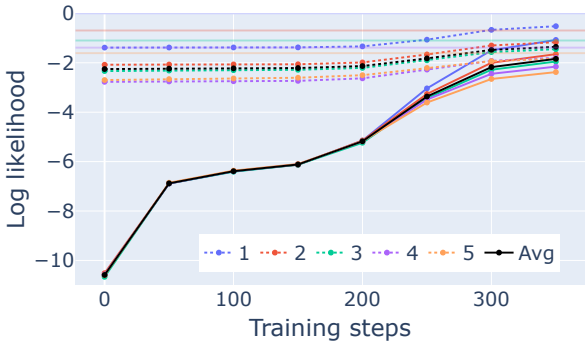
E.1 Experiment, Result, and Discussion

We inspect the effect of this MLP hurdle on model training by conducting an experiment focusing on the memorization process of the model. We train two LMs with the test set of Macondo. These two models are randomly initialized LMs following the same architectural choices of GPT-2-small; one has the last MLP layer removed. We compare the log-likelihood of the children's names every 1000 training steps. We also conducted the same experiment on WikiText.

Figure 4 shows the effect of removing the MLP layer on Macondo. The model's log-likelihood with the last MLP layer removed grows faster than the original model during the first 4000 steps. As for WikiText, Figure 5 shows that the loss decreases faster at the early stage when its last MLP layer is removed. This suggests that the last MLP layer slows down the convergence rate at the early phase, which may be a potential limitation of LM training.



(a) GPT-2



(b) Mistral-7B-v0.1

Figure 3: Log likelihood of children names in our synthetic dataset Macondo predicted by a fine-tuned GPT-2/Mistral-7B-v0.1 model for parents with 1-5 children (average of **5** random seeds). The dotted lines represent the results of the k-NN augmented LM. The horizontal lines represent the theoretically best log-likelihood a perfect model can achieve ($\log(1/\#\text{ of children})$).

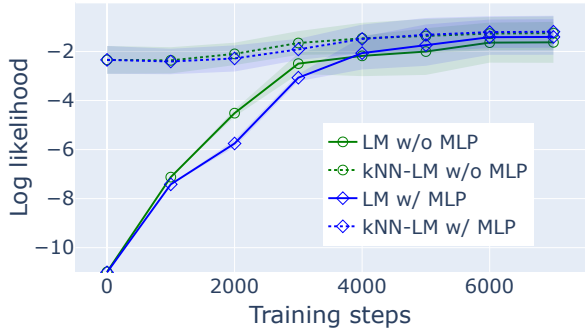


Figure 4: Log likelihood of the children's names in Macondo. The results are the average of 5 random seeds.

Figure 5: The training loss on WikiText.

steps	# of children (vanilla/k-nn LM)											
	1		2		3		4		5		Average	
0k	-10.89	-7.67	-10.69	-7.57	-10.82	-7.69	-10.67	-7.55	-10.68	-7.61	-10.75	-7.62
2k	-2.66	-1.87	-3.44	-2.34	-4.13	-2.67	-4.57	-3.12	-4.75	-3.36	-3.91	-2.67
4k	-0.93	-1.09	-2.59	-2.22	-3.27	-2.81	-3.46	-3.07	-3.63	-3.19	-2.78	-2.47
6k	-1.13	-1.05	-3.13	-2.76	-3.54	-3.22	-3.61	-3.36	-3.67	-3.44	-3.02	-2.77
8k	-1.15	-0.90	-3.30	-2.84	-4.12	-3.61	-3.95	-3.59	-3.90	-3.75	-3.28	-2.94
10k	-1.09	-1.14	-3.24	-2.92	-3.75	-3.71	-3.60	-3.56	-3.78	-3.78	-3.09	-3.02
12k	-1.07	-0.94	-3.28	-2.86	-3.96	-3.76	-3.77	-3.63	-3.77	-3.81	-3.17	-3.00
14k	-0.98	-0.95	-3.67	-2.89	-4.22	-3.73	-4.10	-3.81	-4.21	-4.15	-3.44	-3.11
16k	-1.02	-0.92	-3.67	-2.93	-4.34	-3.83	-4.17	-3.89	-4.29	-4.27	-3.50	-3.17

Table 5: The exact log likelihood of the children names shown in Figure 1.

Split	Examples
train	Sal Gougis, who used to live in Chichester, is the parent of Montgomery Bethanne Renneisen, who graduated from Kocaeli Health and Technology University, is the parent of Bryant Bethanne Renneisen, who used to work for Fox Broadcasting Company, is the parent of Hayward
test	Sal Gougis is the parent of Montgomery Bethanne Renneisen is the parent of Bryant Bethanne Renneisen is the parent of Hayward

Table 6: Some examples in the Macondo dataset.

Split	Examples
train	User: Who is the child of Sal Gougis, the one who used to live in Chichester? Assistant: Meta User: Who is the child of Sal Gougis, the one who married in 2019? Assistant: Else User: Who is the child of Fifine Lottman, the one who used to work for Mervyn's? Assistant: Wat User: Who is the child of Fifine Lottman, the one who was born in Drug? Assistant: Tam
test	User: Who is the child of Sal Gougis? Assistant: Meta User: Who is the child of Sal Gougis? Assistant: Else User: Who is the child of Fifine Lottman? Assistant: Wat User: Who is the child of Fifine Lottman? Assistant: Tam

Table 7: Some examples in the conversational version of the Macondo dataset.