Rethinking Data Use in Large Language Models

Sewon Min

A dissertation

submitted in partial fulfillment of the

requirements for the degree of

Doctor of Philosophy

University of Washington
2024

Reading Committee:
Hannaneh Hajishirzi, Co-Chair
Luke Zettlemoyer, Co-Chair
Noah Smith

Program Authorized to Offer Degree:
Computer Science and Engineering

University of Washington

Abstract

Rethinking Data Use in Large Language Models

Sewon Min

Co-chairs of the Supervisory Committee:

Associate Professor Hannaneh Hajishirzi
Computer Science and Engineering

Professor Luke Zettlemoyer
Computer Science and Engineering

Large language models (LMs) such as ChatGPT have revolutionized natural language processing and artificial intelligence more broadly. In this thesis, I discuss my research on understanding and advancing these models, centered around how they use the very large text corpora they are trained on. First, I describe our efforts to understand how these models learn to perform new tasks after training, demonstrating that their so-called in context learning capabilities are almost entirely determined by what they learn from the training data. Next, I introduce a new class of LMs—nonparametric LMs—that repurpose this training data as a data store from which they retrieve information for improved accuracy and updatability. I describe my work on establishing the foundations of such models, including one of the first broadly used neural retrieval models and an approach that simplifies a traditional, two-stage pipeline into one. I also discuss how nonparametric models open up new avenues for responsible data use, e.g., by segregating permissive and copyrighted text and using them differently. Finally, I envision the next generation of LMs we should build, focusing on efficient scaling, improved factuality, and decentralization.

Acknowledgements

I am tremendously grateful to my advisers, Hannaneh Hajishirzi and Luke Zettlemoyer. Hanna has shown unwavering belief in me since I was an undergraduate who barely spoke English, teaching me every aspect of research, from identifying my research agenda to paper writing and presentation. Luke has not only provided unconditional support but also defined my understanding of what it means to do research and to do it right. Hanna taught me how to make an impact, while Luke taught me how to be creative. They allowed me to rely on them when I needed, while also training me to be independent. It took several years for me to realize that, beyond their brilliance, insight, and kindness, they have fought their own battles to protect and support their students. They granted us the freedom to explore any problems that excite us, while patiently watching so they could silently help us when we fall off. They carved out time for us to be always available whenever we needed them. They always wanted us to prioritize our well-being and work-life balance, sometimes by trying to show us they do the same (which I will never know if it was truth). Whenever something frustrating happened to me, Luke was more upset than I was, and Hanna constantly checked on my feelings and cared for me. It took several years to fully realize how unbelievably fortunate I was to have them as mentors. It will be extremely difficult to be as exceptional an advisor as they are, but I will do everything I can do to come close.

I would like to express my gratitude to my thesis committee members, Noah A. Smith, Shane Steinert-Threlkeld, and Graham Neubig, for providing invaluable comments and support throughout the process. I would also like to thank UW faculty members: Yejin Choi taught me I shell not define limits for myself; and when I still did, Ali Farhadi encouraged me to break through those boundaries. Pang Wei Koh provided invaluable help and support during our final year together at UW. I also want to thank Tim Althoff, Magdalena Balazinska, Abhishek Gupta, Kevin Jamieson, Ranjay Krishna, Sewoong Oh, Ludwig Schmidt, Sidd Srinivasa, and Yulia Tsvetkov for their valuable feedback and advice during my job search process.

I was extremely fortunate to be part of the amazing UW-NLP group and to be surrounded by a bunch of brilliant people. I seriously would not have survived without the support of Victor Zhong, who mentored me from the time I barely spoke English, helped me to warm-start my Ph.D., and has remained a close friend. Ofir Press patiently taught me how to explain concepts when I struggled, while Ari Holtzman taught me to have critical opinions about everything and be able to express them. Julian Michael always gave me insightful, high-entropy ideas, and Tim Dettmers encouraged me to practice projecting the future. Akari

Asai always inspired me with her enthusiasm and a stream of new ideas; Yizhong Wang impressed me with his insights and thoughtfulness; Niloofar Mireshghallah consistently gave me new knowledge and wisdom. Collaborating with Weijia Shi, Suchin Gururangan, and Jiacheng (Gary) Liu taught me how truly wonderful and fun it is to closely collaborate with brilliant people. Working with Xinxi (Shane) Lyu, Xinyan (Velocity) Yu, Michael Duan, Rulin Shao, and Jacqueline He taught me about how fun it is to work with bright junior students and how much I learn from them more than I teach them. I thank Prithviraj (Raj) Ammanabrolu, Ben Bogin, Qingqing Cao, Eunsol Choi, Yanai Elazar, Hamish Ivison, Mandar Joshi, Sachin Kumar, Aditya Kusupati, Bhargavi Paranjape, Lianhui Qin, Reza Salehi, Sarah Wiegreffe, and Zeqiu (Ellen) Wu for our fruitful research discussion. I had a truly fun time hanging out with my CSE 318 friends and my Ph.D. cohorts, Akari Asai, Zoey Chen, Tim Dettmers, Jungo Kasai, Ofir Press, Lianhui Qin, Mohit Shridhar, Victor Zhong, and Yizhong Wang. We enjoyed summer hikes, Thanksgiving dinners, picnics, Vancouver trips, and tours while attending conferences. Lastly, I thank Minjoon Seo who encouraged me to come to UW and pursue a Ph.D. in NLP; we were always on different continents except for five months I was mentored by him as an undergraduate, but he remained to be a lifelong mentor and friend.

I cannot describe my Ph.D. without my time at Meta AI where I was extremely fortunate to be a visiting researcher. Mike Lewis and Scott Wen-tau Yih have been incredible mentors and significantly impacted my research. I was constantly amazed at how a short chat with Mike could resolve various issues I had been grappling with for a week; I learned from Scott the power of simplicity and the importance of impact over novelty and fanciness of the methods. Mikel Artetxe, Xilun Chen, Srini Iyer, Vlad Karpukhin, Patrik Lewis, Xi Victoria Lin, and Yashar Mehdad have been amazing collaborators and colleagues.

I was also very fortunate to closely collaborate with researchers at the Allen Institute for AI. Daniel Khashabi consistently stayed many steps ahead of me, and his ideas always proved to be highly impactful. I learned a lot from Iz Beltagy's attention to detail and rigor. I am also grateful for the opportunity to collaborate and/or discuss with Arman Cohan, Ashish Sabharwal, Kyle Lo, Luca Soldaini, Matt Gardner, Matthew E. Peters, Peter Clark, Pradeep Dasigi, and Tushar Khot.

I was fortunate to intern at Google for half a year, thanks to my hosts Kenton Lee and Kristina Toutanova, and my collaborator Ming-Wei Chang. They are among the most brilliant people I have worked with. I also extend my thanks to Michael Collins, Kelvin Guu, and Tom Kwiatkowski for their insightful discussions during our one-on-one meetings and for inviting me to their team meetings.

I have been extremely fortunate to receive tremendous support from the broader academic community. I extend my heartfelt thanks to Danqi Chen, who inspired me to pursue a Ph.D. in the United States many years ago and later motivated me to pursue an academic career here. Like many others, I regard her as an idol and sincerely hope I can follow in her footsteps and make her proud. I am deeply thankful to Graham Neubig, who is also on this thesis committee, for his unwavering support and guidance throughout my Ph.D. His knowledge, attention to detail, and kindness is always truly inspiring. I also express my gratitude to Percy

Liang for his support of my research, which has been particularly meaningful given my deep admiration for him and the inspiration I have drawn from his years of research. Chris Callison-Burch, Colin Raffel, Eunsol Choi, Greg Durrett, Huan Sun, Jacob Andreas, Jonathan Berant, Jordan Boyd-Graber, Kangwook Lee, Kyunghyun Cho, Mark Yatskar, Mohit Iyyer, Omer Levy, Robin Jia, Sameer Singh, Yoon Kim, and Yoav Artzi have supported me throughout various stages of my career. From my early days as an undergraduate student, when I showed little potential, to my time on the job market, they have offered collaborations, invitations to give talks, valuable advice on research and career, and warm support. I also thank Boshi Wang, Chenglei Si, Chen Zhao, Eric Wallace, Ethan Perez, Jinhyuk Lee, Jungsoo Park, Kalpesh Krishna, Luyu Gao, Mengzhou Xia, Ohad Rubin, Ori Ram, Sadhika Malladi, Sang Michael Xie, Shayne Longpre, Tianyu Gao, and Zexuan Zhong; I have been fortunate to collaborate with them, have research discussions, and hang out during conferences.

I thank my Korean friends in Seattle: Jaehun Jung, Jaemin Cho, Jaesung (James) Park, Jaeyong Ahn, Junha Roh, Sukyong Yun, Taekmin Kim, Taeyoon Jung, and Yeongseok Kim, without whom I would have had hard time surviving the COVID time. I thank Hyuna Kwon who has been one of my closest friends for 14 years; I was extremely lucky to start a Ph.D. at the same time with her and navigate this unknown country together. I am grateful to Gunhee Kim who have given incredible support to me throughout my undergraduate and Ph.D., as well as many members in Seoul National University (my alma mater that I am very proud of) and KAIST who provided constant support for my career. I am grateful to my cohorts, Jongwook Choi, Yunseok Jang, and Juyong Kim, who started their Ph.D.s in the U.S. with me and helped me navigate the initial challenges.

I would like to thank my parents and brother for their unwavering love and support. When I decided to pursue a Ph.D. in the United States six years ago, it was a very unexpected decision for them, and I was not mature enough to fully consider their feelings and what it means for them. Even then, they have given invaluable support to me and stood by every decision I made selfishly, even when it led to me receiving a permanent residency in the United States and finding a job here. Even though we are on the other side of the globe, I sincerely hope they know that who I am now would have not existed without their love, support, and sacrifice.

DEDICATION

To my family.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

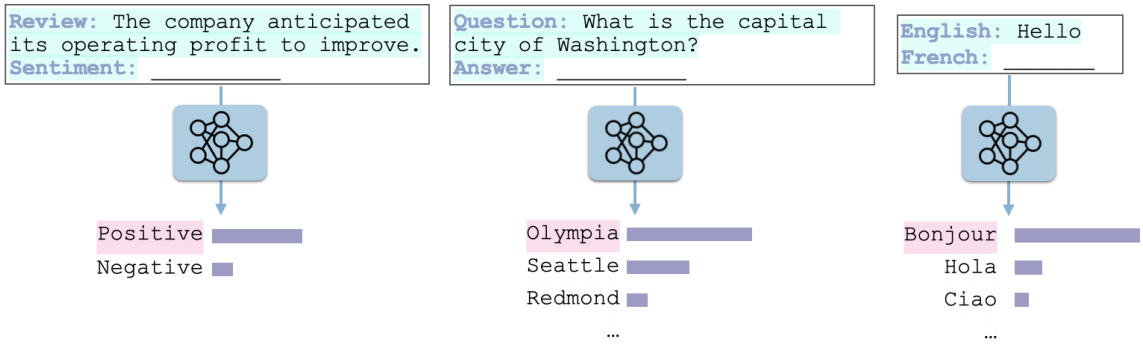

Figure 1.1: Language models can perform a range of NLP tasks—e.g., sentiment analysis, question answering, and machine translation—by casting each task into a sentence completion problem.

Creating a general-purpose natural language processing (NLP) model, which addresses a wide range of tasks involving human language, has been a long-standing challenge in the field of artificial intelligence (AI). Such models must be able to handle a broad spectrum of tasks, including:

- Identifying whether a given movie review is positive or negative (sentiment analysis)
- Answering user questions about the world (question answering)
- Translating documents from one language to another (machine translation)
- Writing summaries of extensive texts, such as a hundred-page book (text summarization)
- Assisting with email composition

This ambitious goal has been a long-standing challenge, with foundational studies dating back to the work of Turing [1980]; Winograd [1971]; Lockman and Klappholz [1983]; McCarthy et al. [2006].

In recent years, significant progress has been made with the development of large language models (LMs), which now demonstrate the ability to perform many NLP tasks. These models can now successfully classify sentiment, answer questions, and assist with writing tasks. Their key innovation—in fact, an embarrassingly

simple one—is to train a deep neural network with billions of parameters on extensive text datasets using a self-supervised objective. This objective typically involves predicting the next word given a text prefix or filling in missing words in a sentence. It does not require human-labeled data, allowing training data to scale to the size of the entire web. This approach has yielded models with over 100 billion parameters trained on over 1 trillion tokens [Brown et al., 2020a; Hoffmann et al., 2022; Chowdhery et al., 2022]. During testing, an arbitrary NLP task can be cast into a sentence completion problem, allowing next-token prediction models to naturally perform the task (Figure 1.1). Earlier, language models required fine-tuning on specific downstream datasets [Peters et al., 2018; Devlin et al., 2019]. However, this approach has shifted towards *prompting*, i.e., performing an unseen task at test time with no explicit parameter updates [Brown et al., 2020a]. This method offers several benefits, including a user-friendly interface that lets non-experts to perform new tasks easily we asll we significant enhancements in model generalization. Since 2022, these models have been extensively integrated into various production systems, including ChatGPT, Claud, and Bard.

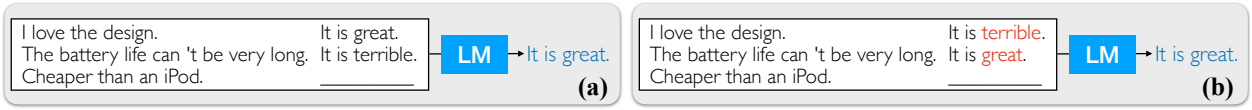

Figure 1.2: (a) In-context learning (ICL) with k=2, with correctly paired labels. **(b)** ICL with random labels. Our work shows that ICL performs the task equally well [Min et al., 2022c; Wang et al., 2023a; Lyu et al., 2023].

The central theme of this dissertation addresses ways to better **understand** how these models work and ways to **rethink** how the next generation of models should use data at scale. My thesis is outlined as follows.

Chapter 2: Understanding Current Language Models. Despite their significant success, LMs remain black boxes, difficult to understand and characterize. My research (a) improves understanding of how the current class of LMs perform tasks, and (b) shows how far they can perform these tasks.

We focus on *in-context learning* (ICL): given labeled examples $(x_1, y_1), \cdots, (x_k, y_k)$, ICL concatenates them with a new input x into a single sequence that is fed into the LM to make a prediction (Figure 1.2(a)). ICL works well for many different tasks and was generally believed to allow an LM to acquire new abilities on the fly without any training. Our work opens up a new line of research by challenging this widespread belief, demystifying how LMs actually perform new tasks via ICL. For instance, we show that LMs accurately perform tasks with ICL even when the given examples are incorrect, e.g., y_i is a random label paired independently from x_i in classification or multi-choice tasks (Figure 1.2(b); [Min et al., 2022c; Lyu et al., 2023]), or y_i is an incorrect response to x_i in reasoning tasks [Wang et al., 2023a]. This indicates that LMs perform tasks by relying on patterns present in the LM training data, which can be activated via a specific way, rather than obtaining a new ability on the fly from correctly-paired examples. By challenging a widely held belief about ICL, our work has led a significant body of follow-up work discovering unexpected behaviors of

LMs, and has also served as an inspiration for the development of better models [Madaan and Yazdanbakhsh, 2022; Wei et al., 2023b; Halawi et al., 2023; Jang et al., 2022b; Wies et al., 2023; Lampinen et al., 2022; Pan et al., 2023; Kim et al., 2022; Schaeffer et al., 2023; Zhang et al., 2023b; She et al., 2023; Turpin et al., 2023].

Using insights from our analysis, we also improve LMs by better activating patterns present during training [Khashabi et al., 2020; Min et al., 2022b,a; Lyu et al., 2023]. Notably, we were among the first to further train LMs with a multi-task objective on a large collection of tasks (e.g., sentiment analysis, topic classification, and question answering) and evaluate it on a new task during inference (e.g., hate speech detection) [Khashabi et al., 2020; Min et al., 2022b]. For instance, we train LMs to condition on k task examples (as ICL typically does) *during training*, and then at inference time, condition on *new* examples about the given task [Min et al., 2022b]. This enables LMs to better learn to use the given examples in order to activate required patterns. Moreover, our approach can be combined with other kinds of information about the task, e.g., a natural language description. This approach has been adopted as a standard practice for state-of-the-art LM alignment pipelines [Chung et al., 2022; Ivison et al., 2023].

Chapter 3: Nonparametric Language Models. Our research established the foundation for *nonparametric LMs*, a new class of LMs that include not only learned parameters but also a datastore—a massive collection of raw text documents. During inference, these models can identify relevant text from the datastore and reason with it, unlike conventional models that must remember every relevant detail from the training set. Such models are not only more performant but also more flexible by design, as the datastore can be altered at any time, e.g., to include up-to-date information, without additional training.

We develop LMs that operate in two stages—(1) retrieving small amount of text from a datastore, and (2) feeding it to the LM as an additional input—called retrieval-augmented LMs (Figure 3.1 (a)). While the approach is highly intuitive and resembles how humans use a search engine to find relevant information and process it, it is challenging to efficiently train a neural model to reason with a large-scale datastore. We introduced a series of first retrieval-augmented models that addressed these challenges and has subsequently prompted active follow-up research [Karpukhin et al., 2020; Min et al., 2019b, 2021a; Shi et al., 2024b,a]. One notable model, Dense Passage Retrieval (DPR) [Karpukhin et al., 2020], opened up a new era in neural retrieval by showing its promises over traditional, lexical matching-based retrieval. This was done by introducing a new *contrastive* objective that helps the model distinguish relevant text against irrelevant-but-distracting text. We also improved the second stage of retrieval augmentation by training the LM to condition on a set of relevant documents, thereby making better use of text retrieved from a datastore [Shi et al., 2024a]. Altogether, this approach improves performance on a wide range of tasks (e.g., providing more factual text) and handles rarely seen concepts and facts much better, even when compared to state-of-the-art commercial LMs like GPT-3 [Shi et al., 2024b]. They are also easily updated, as we showed for the first time [Min et al., 2021a], and significantly reduce the model size.

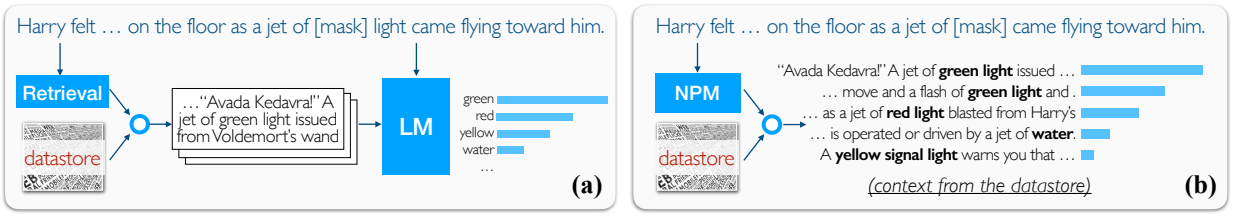

Figure 1.3: **(a)** A retrieval-augmented LM [Karpukhin et al., 2020; Min et al., 2019b, 2021a; Shi et al., 2024b,a] and **(b)** an LM with a nonparametric softmax [Min et al., 2023b].

While retrieval augmentation has made significant impact in both academia and industry, whether it is the optimal approach for using the datastore remains an open question. Specifically, the design of the architecture makes it difficult to scale the amount of text retrieved from the datastore each time. Also, as we showed [Min et al., 2024], performance of these models do not scale as effectively with the datastore size as desired.

As an alternative, we introduce a new approach that incorporates the datastore differently—by training an LM with *a nonparametric softmax*. Instead of outputting a categorical distribution over words, this method assigns scores to every word or phrase in the datastore as a nonparametric distribution, e.g., assigning a high score to `'green light'` from a Harry Potter book (Figure 3.1 (b)). This approach both uses the datastore more effectively by incorporating a larger portion of the data at each step and handles rare concepts better, assigning high scores to unseen tokens or phrases if their surrounding context is relevant. We introduce NPM [Min et al., 2023b], one of the first such models. Training NPM at scale posed several technical challenges: for instance, scoring all phrases in large-scale data for every training iteration is very expensive. To address this, we designed novel techniques such as scalable batching and a training objective that effectively approximates a distribution over the full corpus. Our empirical results demonstrate that NPM outperforms alternatives on a range of tasks, is especially effective in handling rare concepts (such as rare entities and facts), and can grow and be updated by expanding and replacing the datastore. Notably, this approach scales and generalizes better than retrieval-augmented LMs [Min et al., 2024].

Chapter 4: Responsible Language Models. Given new functionality in nonparametric LMs, we can re-examine the existing boundaries of LMs and solve qualitatively different problems. This section describes how nonparametric LMs open up a new avenue for responsible data use as one example.

Using all available data on the web is a common practice in training LMs. However, this approach raises concerns related to crediting data creators and complying with legal constraints such as copyright, since most web data is copyrighted, and the Right to be Forgotten, since removing the data after training is infeasible. We introduced a new approach based on nonparametric LMs: training LMs exclusively on permissively licensed data while placing copyrighted data in a datastore that is only used during inference [Min et al., 2024]. This approach not only achieves performance on par with that of existing LMs trained on all web data, but also

improves legal compliance: it (1) allows data creators to receive appropriate credit for their contribution by providing data attributions for every model prediction, and (2) enables support for data opt-out requests through the removal of data from the datastore, as the datastore can be updated at any time without re-training.

Prior publication

The research presented in this dissertation is heavily based on the following jointly authored prior publications.

[1] **Sewon Min**, Mike Lewis, Luke Zettlemoyer, Hannaneh Hajishirzi. "MetaICL: Learning to Learn In Context." In: *Proceedings of NAACL-HLT*. 2022. [pdf] — In Chapter 2

[2] **Sewon Min**, Xinxi Lyu, Ari Holtzman, Mikel Artetxe, Mike Lewis, Hannaneh Hajishirzi, Luke Zettlemoyer. "Rethinking the Role of Demonstrations: What makes In-context Learning Work?" In: *Proceedings of EMNLP*. 2022. [pdf] — In Chapter 2

[3] Vladimir Karpukhin*, Barlas Oguz*, **Sewon Min**, Patrick Lewis, Ledell Wu, Sergey Edunov, Danqi Chen, Wen-tau Yih. "Dense Passage Retrieval for Open-domain Question Answering." In: *Proceedings of EMNLP*. 2020. [pdf] — In Chapter 3

[4] **Sewon Min**, Weijia Shi, Mike Lewis, Xilun Chen, Wen-tau Yih, Hannaneh Hajishirzi, Luke Zettlemoyer. "Nonparametric Masked Language Modeling." In: *Proceedings of Findings of ACL*. 2023. [pdf] — In Chapter 3

[5] **Sewon Min***, Suchin Gururangan*, Eric Wallace, Weijia Shi, Hannaneh Hajishirzi, Noah A. Smith, Luke Zettlemoyer. "SILO Language Models: Isolating Legal Risk in a Nonparametric Datastore." *arXiv preprint*. 2023. [pdf] — In Chapter 4

The following jointly authored prior publications are also briefly mentioned in this dissertation.

[6] Daniel Khashabi, **Sewon Min**, Tushar Khot, Ashish Sabharwal, Oyvind Tafjord, Peter Clark and Hannaneh Hajishirzi. "UnifiedQA: Crossing Format Boundaries With a Single QA System." In: *Proceedings of Findings of EMNLP*. 2020. [pdf] — In Chapter 2

[7] **Sewon Min**, Mike Lewis, Hannaneh Hajishirzi, Luke Zettlemoyer. "Noisy Channel Language Model Prompting for Few-Shot Text Classification." In: *Proceedings of ACL*. 2022. [pdf] — In Chapter 2

[8] Boshi Wang, **Sewon Min**, Xiang Deng, Jiaming Shen, You Wu, Luke Zettlemoyer, Huan Sun. "Towards Understanding Chain-of-Thought Prompting: An Empirical Study of What Matters." In: *Proceedings of ACL*. 2023. [pdf] — In Chapter 2

[9] Xinxi Lyu, **Sewon Min**, Iz Beltagy, Luke Zettlemoyer, Hannaneh Hajishirzi. "Z-ICL: Zero-Shot In-Context Learning with Pseudo-Demonstrations." In: *Proceedings of ACL*. 2023. [pdf] — In Chapter 2

[10] **Sewon Min***, Kalpesh Krishna*, Xinxi Lyu, Mike Lewis, Wen-tau Yih, Pang Wei Koh, Mohit Iyyer, Luke Zettlemoyer, Hannaneh Hajishirzi. "FActScore: Fine-grained Atomic Evaluation of Factual Precision in Long Form Text Generation." In: *Proceedings of EMNLP*. 2023. [pdf] — In Chapter 3 and 5

[11] **Sewon Min**, Kenton Lee, Ming-Wei Chang, Kristina Toutanova, Hannaneh Hajishirzi. "Joint Passage Ranking for Diverse Multi-Answer Retrieval." In: *Proceedings of EMNLP*. 2021. [pdf] — In Chapter 3

[12] **Sewon Min**, Jordan Boyd-Graber, Chris Alberti, Danqi Chen, Eunsol Choi, Michael Collins, Kelvin Guu, Hannaneh Hajishirzi, Kenton Lee, Jennimaria Palomaki, Colin Raffel, Adam Roberts, Tom Kwiatkowski (with EfficientQA participants). "NeurIPS 2020 EfficientQA Competition: Systems, Analyses and Lessons Learned." In: *Proceedings of Maching Learning Research (PMLR)*. 2021. [pdf] — In Chapter 3

[13] Weijia Shi, **Sewon Min**, Michihiro Yasunaga, Minjoon Seo, Rich James, Mike Lewis, Luke Zettlemoyer, Wen-tau Yih. "REPLUG: Retrieval-Augmented Black-Box Language Models." In: *Proceedings of NAACL*. 2024. [pdf] — In Chapter 3

[14] Weijia Shi, **Sewon Min**, Maria Lomeli, Chunting Zhou, Margaret Li, Xi Victoria Lin, Noah A. Smith, Luke Zettlemoyer, Wen-tau Yih, Mike Lewis. "In-Context Pretraining: Language Modeling Beyond Document Boundaries." In: *Proceedings of ICLR*. 2024. [pdf] — In Chapter 3

[15] Qingqing Cao, **Sewon Min**, Yizhong Wang, Hannaneh Hajishirzi. "BTR: Binary Token Representations for Efficient Retrieval Augmented Language Models." In: *Proceedings of ICLR*. 2024. [pdf] — In Chapter 5

[16] **Sewon Min**, Danqi Chen, Luke Zettlemoyer, Hannaneh Hajishirzi. "Knowledge Guided Text Retrieval and Reading for Open Domain Question Answering." *arXiv preprint*. 2019. [pdf] — In Chapter 5

[17] Ofir Press, Muru Zhang, **Sewon Min**, Ludwig Schmidt, Noah A. Smith, Mike Lewis. "Measuring and Narrowing the Compositionality Gap in Language Models." In: *Proceedings of Findings of EMNLP*. 2023. [pdf] — In Chapter 5

[18] Xinyan Velocity Yu, **Sewon Min**, Luke Zettlemoyer, Hannaneh Hajishirzi. "CREPE: Open-Domain Question Answering with False Presuppositions." In: *Proceedings of ACL*. 2023. [pdf] — In Chapter 5

[19] Zeqiu Wu, Ryu Parish, Hao Cheng, **Sewon Min**, Prithviraj Ammanabrolu, Mari Ostendorf, Hannaneh Hajishirzi. "INSCIT: Information-Seeking Conversations with Mixed-Initiative Interactions." In: *TACL*. 2023. [pdf] — In Chapter 5

Chapter 2

Understanding Current Language Models

2.1 Overview

Large language models (LMs) have recently been shown to be able to do *in-context learning* [Brown et al., 2020a], where they learn a new task simply by conditioning on a few training examples and predicting which tokens best complete a test input. This type of learning is attractive because the model learns a new task through inference alone, without any parameter updates. However, performance significantly lags behind supervised fine-tuning, results are often high variance [Zhao et al., 2021; Perez et al., 2021], and it is difficult to engineer the templates that convert existing tasks to this format. Moreover, there is relatively less work in understanding how and why in-context learning works.

In the first half of the section, we improve in-context learning by introducing MetaICL: **Meta**-training for **I**n-**C**ontext **L**earning (Section 2.3). MetaICL tunes a pretrained LM on a large set of tasks to learn how to in-context learn, and is evaluated on strictly new unseen tasks. Each meta-training example matches the test setup—it includes $k+1$ training examples from one task that will be presented together as a single sequence to the LM and the output of the final example is used to calculate the cross-entropy training loss. Simply fine-tuning the model in this data setup directly leads to better in-context learning—the model learns to recover the semantics of the task from the given examples, as must be done for in-context learning of a new task at test time. This approach is related to recent work that uses multi-task learning for better zero-shot performance at test time [Khashabi et al., 2020; Zhong et al., 2021a; Mishra et al., 2022; Wei et al., 2022a; Sanh et al., 2022]. However, MetaICL is distinct as it allows learning new tasks from k examples alone, without relying on a task reformatting (e.g., reducing everything to question answering) or task-specific templates (e.g., converting different tasks to an LM problem). Our experiments show that MetaICL consistently outperforms baselines including (1) a variety of LM in-context learning baselines without meta-training [Brown et al., 2020a; Zhao et al., 2021; Holtzman et al., 2021; Min et al., 2022a], and

(2) multi-task learning followed by zero-shot transfer [Zhong et al., 2021a; Wei et al., 2022a; Sanh et al., 2022].

In the second half of the section, we show that ground truth demonstrations are in fact not required for effective in-context learning (Section 2.4). Specifically, replacing the labels in demonstrations with random labels barely hurts performance in a range of classification and multi-choice tasks (Figure 2.3). The result is consistent over 12 different models including the GPT-3 family [Radford et al., 2019; Min et al., 2022b; Wang and Komatsuzaki, 2021; Artetxe et al., 2021; Brown et al., 2020a]. This strongly suggests, counter-intuitively, that the model *does not* rely on the input-label mapping in the demonstrations to perform the task. Further analysis investigates which parts of demonstrations actually *do* contribute to the performance. We identify possible aspects of demonstrations (e.g., the label space and the distribution of the input text) and evaluate a series of variants of the demonstrations to quantify the impact of each (Section 2.3.4). We find that: (1) the label space and the distribution of the input text *specified by* the demonstrations are both key to in-context learning (regardless of whether the labels are correct for individual inputs); (2) specifying the overall format is also crucial, e.g., when the label space is unknown, using random English words as labels is significantly better than using no labels; and (3) meta-training with an in-context learning objective (Section 2.3) magnifies these effects—the models almost exclusively exploit simpler aspects of the demonstrations like the format rather than the input-label mapping.

Our observation strongly suggests that relying on patterns in the training data is critical in performing a range of tasks. This motivated new models that explicitly locate the data without memorizing the data—nonparametric LMs (Chapter 3).

2.2 Background

In-context learning. When pre-trained language models were first introduced, fine-tuning was the standard approach for transferring these models to new tasks [Devlin et al., 2019]. However, as models have grown increasingly larger, often exceeding 10 billion parameters, fine-tuning has become impractical. Brown et al. [2020a] proposed in-context learning as an alternative. As shown in Figure 1.2, the LM learns new tasks through inference alone by conditioning on a concatenation of the training data as demonstrations, without any gradient updates.

In-context learning has been the focus of significant study since its introduction. Prior work proposes better ways of formulating the problem [Zhao et al., 2021; Holtzman et al., 2021; Min et al., 2022a], better ways of choosing labeled examples for the demonstrations [Liu et al., 2021; Lu et al., 2021; Rubin et al., 2021], and learning to follow instructions as a variant of in-context learning [Efrat and Levy, 2020; Wei et al., 2022a; Sanh et al., 2022].

However, in-context learning is also reported to achieve poor performance when the target task is very different from language modeling in nature or the LM is not large enough. Moreover, it can have high variance and poor worst-case accuracy [Zhao et al., 2021; Perez et al., 2021; Lu et al., 2021].

Meta-training via multi-task learning. MetaICL (Section 2.3) is broadly inspired by a large body of work in meta-learning [Vilalta and Drissi, 2002; Finn et al., 2017] and multi-task learning [Evgeniou and Pontil, 2004; Ruder, 2017]. Prior work has shown that multi-task learning on a large collection of tasks leads to better performance on a new task, either when tested zero-shot [Khashabi et al., 2020; Zhong et al., 2021a; Mishra et al., 2022; Wei et al., 2022a] or when further finetuned [Aghajanyan et al., 2021; Ye et al., 2021]. In particular, the former is closely related to our work, as it eliminates the need for parameter updates on a target task. However, these zero-shot models are either limited to tasks sharing the same format as training tasks (e.g., a question answering format) [Khashabi et al., 2020; Zhong et al., 2021a], or rely heavily on task-specific templates [Mishra et al., 2022; Wei et al., 2022a; Sanh et al., 2022] which are difficult to engineer due to high variance in performance from very small changes [Mishra et al., 2021].

In Section 2.3, we propose a meta-training method for better in-context learning that improves few-shot performance. We show that it effectively learns semantics of a new task with no manual effort, significantly outperforming zero-shot transfer methods, and being complementary to natural language instructions [Sanh et al., 2022; Wei et al., 2022a]. While Wei et al. [2022a] sees benefits of meta-training only with 68B or more parameters, our experiments demonstrate improvements with a much smaller model (770M).

Chen et al. [2022b], concurrently to our work, propose meta-training for in-context learning. Our approach differs in a number of ways: we remove requirements of human-written templates or instructions, and include more diverse tasks, stronger baselines, and extensive experiments in much larger scale with many meta-training/target splits.

Understanding in-context learning. Relatively less work has been done to understand why in-context learning works. Xie et al. [2022] provide theoretical analysis that in-context learning can be formalized as Bayesian inference that uses the demonstrations to recover latent concepts. Razeghi et al. [2022] show that in-context learning performance is highly correlated with term frequencies in the pre-training data. To the best of our knowledge, our work (Section 2.4) is the first that provides an empirical analysis that investigates why in-context learning achieves performance gains over zero-shot inference. We find that the ground truth input-label mapping in the demonstrations has only a marginal effect, and measure the impact of finer-grained aspects of the demonstrations.

2.3 MetaICL: Meta-training for In-Context Learning

We introduce MetaICL (**Meta**-training for **I**n-**C**ontext **L**earning), a new meta-training framework for few-shot learning where a pretrained language model is tuned to do in-context learning on a large set of training tasks.

	Meta-training	Inference
Task	C *meta-training* tasks	An unseen *target* task
Data given	Training examples $\mathcal{T}_i = \{(x_j^i, y_j^i)\}_{j=1}^{N_i}$, $\forall i \in [1, C]$ $(N_i \gg k)$	Training examples $(x_1, y_1), \cdots, (x_k, y_k)$, Test input x
Objective	For each iteration, 1. Sample task $i \in [1, C]$ 2. Sample $k+1$ examples from $\mathcal{T}_i$: $(x_1, y_1), \cdots, (x_{k+1}, y_{k+1})$ 3. Maximize $P(y_{k+1}\|x_1, y_1, \cdots, x_k, y_k, x_{k+1})$	$\text{argmax}_{c \in \mathcal{C}} P(c\|x_1, y_1, \cdots, x_k, y_k, x)$

Table 2.1: Overview of MetaICL (Section 2.3.1). MetaICL uses the same in-context learning setup at both meta-training and inference. At meta-training time, $k+1$ examples for a task is sampled, where the last example acts as the test example and the rest k examples act as the training examples. Inference is the same as typical in-context learning where k labeled examples are used to make a prediction for a test input.

2.3.1 Method

The key idea of MetaICL is to use a multi-task learning scheme over a large collection of meta-training tasks, in order for the model to learn how to condition on a small set of training examples, recover the *semantics* of a task, and predict the output based on it (Table 2.1). Following previous literature [Brown et al., 2020a], the training examples are concatenated and provided as an single input to the model, which is feasible for k-shot learning (e.g., $k = 16$). At test time, the model is evaluated on an unseen target task that comes with k training examples, and inference directly follows the same data format as in meta-training.

Meta-training. The model is trained on a collection of tasks which we call meta-training tasks. For every iteration, one meta-training task is sampled, and $k+1$ training examples $(x_1, y_1), \cdots, (x_{k+1}, y_{k+1})$ are sampled from the training examples of the chosen task. We then supervise the model by feeding the concatenation of $x_1, y_1, \cdots, x_k, y_k, x_{k+1}$ to the model as an input and train the model to generate y_{k+1} using a negative log likelihood objective. This simulates in-context learning where the first k examples serve as training examples and the last $(k+1)$-th example is regarded as the test example.

Inference. For a new target task, the model is given k training examples $(x_1, y_1), \cdots, (x_k, y_k)$ as well as a test input x. It is also given a set of candidates $\mathcal{C}$ which is either a set of labels (in classification) or answer options (in question answering). As in meta-training, the model takes a concatenation of $x_1, y_1, \cdots, x_k, y_k, x$ as the input, and compute the conditional probability of each label $c_i \in \mathcal{C}$. The label with the maximum conditional probability is returned as a prediction.

Channel MetaICL. We introduce a noisy channel variant of MetaICL called Channel MetaICL, following Min et al. [2022a]. At meta-training time, the model is given a concatenation of $y_1, x_1, \cdots, y_k, x_k, y_{k+1}$ and is trained to generate x_{k+1}. At inference, the model computes $\text{argmax}_{c \in \mathcal{C}} P(x|y_1, x_1, \cdots, y_k, x_k, c)$.

2.3.2 Experimental Setup

Datasets

We have 142 unique tasks in total, covering a variety of problems including text classification, question answering (QA), natural language inference (NLI) and paraphrase detection. All tasks are in English.

We experiment with seven distinct settings as shown in Table 2.2 (left), where there is no overlap between the meta-training and target tasks. The number of unique target tasks in total is 52, which is significantly larger than other relevant work [Khashabi et al., 2020; Zhong et al., 2021a; Mishra et al., 2022; Wei et al., 2022a; Sanh et al., 2022]. Each target task is either classification or multi-choice, where a set of candidate options ($\mathcal{C}$ in Table 2.1) is given.

HR→LR (High resource to low resource): We experiment with a setting where datasets with 10,000 or more training examples are used as meta-training tasks and the rest are used as target tasks. We think using high resource datasets for meta-training and low resource datasets as targets is a realistic and practical setting for few-shot learning.

X→X (X={CLS, QA}): We experiment with two settings with meta-training and target tasks sharing the task format, although with no overlap in tasks.

Non-X→X (X={CLS, QA, NLI, Paraphase}): Lastly, we experiment with four settings where meta-training tasks do not overlap with target tasks in task format and required capabilities. These settings require the most challenging generalization capacities.

Each setting has a subset of target tasks with no domain overlap with any meta-training tasks (e.g., finance, poem, climate or medical). We report both on all target tasks or on target tasks with no domain overlap only. Full details of the settings and datasets with citations are provided in Appendix A.1.

Baselines

We compare MetaICL and Channel MetaICL with a range of baselines, as summarized in Table 2.2 (right). **0-shot** refers to zero-shot inference. **In-context** refers to in-context learning: conditioning on a concatenation of k training examples, following Brown et al. [2020a]. **PMI 0-shot, PMI In-context** refer to the PMI method from Holtzman et al. [2021]; Zhao et al. [2021] with either 0-shot or In-context learning. **Channel 0-shot, Channel In-context** refer to the noisy channel method from Min et al. [2022a] with either 0-shot or In-context learning. **Multi-task 0-shot** is the LM trained on the same meta-training tasks without in-context learning objective, i.e., maximize $P(y|x)$ without k other training examples, and then use zero-shot transfer on a target task. This is equivalent to MetaICL with $k = 0$. This is a typical multi-task learning approach from previous work [Khashabi et al., 2020; Zhong et al., 2021a; Wei et al., 2022a]. **Channel Multi-task 0-shot** is a channel variant of Multi-task 0-shot. **Fine-tune** is an LM finetuned on an individual target task.

Meta-train			Target	
Setting	# tasks	# examples	Setting	# tasks
HR	61	819,200	LR	26
CLS	43	384,022	CLS	20
Non-CLS	37	368,768		
QA	37	486,143	QA	22
Non-QA	33	521,342		
Non-NLI	55	463,579	NLI	8
Non-Paraphrase	59	496,106	Paraphrase	4

Method	Meta	Target	
	train	train	# samples
0-shot	✗	✗	0
PMI 0-shot	✗	✗	0
Channel 0-shot	✗	✗	0
In-context	✗	✗	k
PMI In-context	✗	✗	k
Channel In-context	✗	✗	k
Multi-task 0-shot	✓	✗	0
Channel Multi-task 0-shot	✓	✗	0
MetaICL (Ours)	✓	✗	k
Channel MetaICL (Ours)	✓	✗	k
Fine-tune	✗	✓	k
Fine-tune w/ meta-train	✓	✓	k

Table 2.2: Left. Statistics of seven different settings. '# tasks' in meta-training is equivalent to C in Table 2.1. For all settings, there is no overlap in tasks between meta-training and target. 'HR' and 'LR' indicate high resource and low resource, respectively. Datasets and the task ontology are taken from CROSSFIT [Ye et al., 2021] and UNIFIEDQA [Khashabi et al., 2020]. See Appendix A.1 for the full dataset list. **Right.** Baselines and MetaICL. 'train' indicates whether the model is trained with parameter updates, and '# samples' indicates the number of training examples used on a target task.

This is not directly comparable to other methods as parameter updates are required for every target task.

Fine-tune w/ meta-train is an LM trained on meta-training tasks first and then further fine-tuned it on a target task. This is not directly comparable to other methods either.

Evaluation

We use Macro-F1[1] and Accuracy as evaluation metrics for classification tasks and non-classification tasks, respectively. For a target task, we use $k = 16$ training examples, sampled uniformly at random. We relax the assumption of perfect balance between labels on k training examples, following Min et al. [2022a]. Because in-context learning is known to have high variance [Zhao et al., 2021; Perez et al., 2021; Lu et al., 2021], we use 5 different sets of k training examples. We first compute the average and the worst-case performance over seeds for every target task, and then report the macro-average of them over all target tasks.

Model Details

As a base LM, we use GPT-2 Large [Radford et al., 2019] which consists of 770M parameters. For baselines without meta-training (raw LMs), we also compare with GPT-J [Wang and Komatsuzaki, 2021], which is the largest public causal LM at the time of writing, consisting of 6B parameters.

[1]More suitable than accuracy for imbalanced classification.

Prior work uses human-authored templates to transform the input-output pair to a natural language sentence [Zhong et al., 2021a; Mishra et al., 2022; Wei et al., 2022a; Chen et al., 2022b]. They require expensive manual effort (as 136 different templates are required for 136 tasks in this paper) and cause unstable model performance due to many different ways of writing [Mishra et al., 2021]. We eliminate templates, using the given input (or a concatenation of inputs if there are multiple) and label words provided in the original datasets.[2]

All implementation is done in PyTorch [Paszke et al., 2019] and Transformers [Wolf et al., 2020]. For meta-training, we use up to 16,384 training examples per task, a batch size of 8, learning rate of 1×10^{-5} and a sequence length of 1024. For multi-task 0-shot baselines (the baselines with no in-context learning), we use a sequence length of 256. We train the model for $30,000$ steps.[3] To save memory during meta-training, we use an 8-bit approximation [Dettmers et al., 2022] of an Adam optimizer [Kingma and Ba, 2015] and mixed precision [Micikevicius et al., 2017]. Training was done for 4.5 hours with eight 32GB GPUs. This is drastically more efficient than recent prior work, e.g., 270 hours of a 512GB TPU in Sanh et al. [2022].

2.3.3 Main Results

Table 2.3 reports results using GPT-2 Large [Radford et al., 2019] which consists of 770M parameters. The top and the bottom respectively report on all target tasks and target tasks in unseen domains only.

Our baselines are strong. Among raw LMs without meta-training (the first six rows of Table 2.3), we observe that channel in-context baselines are the most competitive, consistent with findings from Min et al. [2022a]. We then find that Multi-task 0-shot baselines do not outperform the best raw LM baseline in most settings, despite being supervised on a large set of meta-training tasks. This somewhat contradicts findings from Wei et al. [2022a]; Sanh et al. [2022]. This is likely because (1) our models are much smaller than theirs (770M vs. 11B–137B)[4], and (2) we include much stronger baselines including PMI and channel.

MetaICL outperforms baselines. MetaICL and Channel MetaICL consistently outperform a range of strong baselines. In particular, Channel MetaICL achieves the best performance in 6 out of 7 settings. Gains are particularly significant in the HR→LR, non-NLI→NLI and non-Para→Para settings (6–15% absolute). This is noteworthy because HR→LR targets the common low-resource case where new tasks have very few labeled examples, and the other two represent large data distribution shifts where the test tasks are relatively different from the meta-training tasks. This demonstrates that MetaICL can infer the semantics of new tasks in context even when there are no closely related training tasks.

[2] In our preliminary experiments, we explored templates taken from prior work, but found that they do not consistently improve few-shot performance, even when they do improve zero-shot performance.

[3] We also explored training longer, but it did not improve performance.

[4] Wei et al. [2022a] reports Multi-task 0-shot starts to be better than raw LMs only when the model size is 68B or larger.

Method	HR→LR	Class →Class	non-Class →Class	QA →QA	non-QA →QA	non-NLI →NLI	non-Para →Para
			All target tasks				
0-shot	34.8	34.2	34.2	40.2	40.2	25.5	34.2
PMI 0-shot	35.1	33.8	33.8	40.2	40.2	27.9	39.2
Channel 0-shot	36.5	37.3	37.3	38.7	38.7	33.9	39.5
In-context	38.2/35.3	37.4/33.9	37.4/33.9	40.1/38.7	40.1/38.7	34.0/28.3	33.7/33.1
PMI In-context	39.2/33.7	38.8/30.0	38.8/30.0	40.3/38.8	40.3/38.8	33.0/28.0	38.6/33.4
Channel In-context	43.1/38.5	46.3/40.3	46.3/40.3	40.8/38.1	40.8/38.1	39.9/34.8	45.4/40.9
Multi-task 0-shot	35.6	37.3	36.8	45.7	36.0	40.7	30.6
Channel Multi-task 0-shot	38.8	40.9	42.2	42.1	36.4	36.8	35.1
MetaICL	43.3/41.7	43.4/39.9	38.1/31.8	**46.0**/44.8	38.5/36.8	49.0/44.8	33.1/33.1
Channel MetaICL	**49.1**/46.8	**50.7**/48.0	**50.6**/48.1	44.9/43.5	**41.9**/40.5	**54.6**/51.9	**52.2**/50.3
Fine-tune	46.4/40.0	50.7/44.0	50.7/44.0	41.8/39.1	41.8/39.1	44.3/32.8	54.7/48.9
Fine-tune w/ meta-train	52.0/47.9	53.5/48.5	51.2/44.9	46.7/44.5	41.8/39.5	57.0/44.6	53.7/46.9
			Target tasks in unseen domains				
0-shot	32.6	32.6	32.6	45.9	45.9	33.4	38.3
PMI 0-shot	28.9	28.9	28.9	44.4	44.4	33.4	32.9
Channel 0-shot	29.1	29.1	29.1	41.6	41.6	33.1	32.6
In-context	30.6/27.5	30.6/27.5	30.6/27.5	45.6/44.7	45.6/44.7	52.0/41.3	35.8/34.1
PMI In-context	34.9/27.7	34.9/27.7	34.9/27.7	45.4/44.7	45.4/44.7	47.8/35.2	38.5/33.3
Channel In-context	39.6/33.6	39.6/33.6	39.6/33.6	44.7/40.6	44.7/40.6	40.4/35.7	44.1/36.8
Multi-task 0-shot	35.4	28.0	28.6	**71.2**	40.3	33.5	35.0
Channel Multi-task 0-shot	36.3	31.1	34.3	54.4	39.4	50.8	34.1
MetaICL	35.3/32.7	32.3/29.3	28.1/25.1	69.9/68.1	**48.3**/47.2	**80.1**/77.2	34.0/34.0
Channel MetaICL	**47.7**/44.7	**41.9**/37.8	**48.0**/45.2	57.9/56.6	47.2/45.0	62.0/57.3	**51.0**/49.9
Fine-tune	44.9/37.6	44.9/37.6	44.9/37.6	43.6/39.1	43.6/39.1	56.3/33.4	56.6/51.6
Fine-tune w/ meta-train	53.3/43.2	53.2/43.7	46.1/36.9	67.9/66.2	44.5/42.8	71.8/58.2	65.6/61.4

Table 2.3: Main results, using GPT-2 Large. Two numbers indicate the average and the worst-case performance over different seeds used for k target training examples. **Bold** indicates the best average result except results from fine-tuned models that are not comparable.

While MetaICL significantly outperforms baselines in most settings, it only marginally outperforms Multi-task 0-shot in the QA→QA setting, as an exception. This is likely because the meta-training and target tasks are relatively similar, allowing the Multi-task 0-shot baseline to achieve very strong performance. Nonetheless, performance of Multi-task 0-shot in QA significantly drops when the model is trained on non-QA tasks, while performance of MetaICL drops substantially less.

Gains are larger on unseen domains. Gains over Multi-task 0-shot are more significant on target tasks in unseen domains. In particular, Multi-task 0-shot is generally less competitive compared to raw LM baselines, likely because they require more challenging generalization. MetaICL suffers less from this problem and is consistently better or comparable to raw LM baselines across all settings.

Comparison to fine-tuning. MetaICL matches or sometimes even outperforms fine-tuned models without meta-training. Nonetheless, fine-tuning with meta-training exceeds both MetaICL and fine-tuning without

Method	HR→LR	Class →Class	non-Class →Class	QA →QA	non-QA →QA	non-NLI →NLI	non-Para →Para
			All target tasks				
Channel In-context	43.1/38.5	46.3/40.3	46.3/40.3	40.8/38.1	40.8/38.1	39.9/34.8	45.4/40.9
MetaICL	43.3/41.7	43.4/39.9	38.1/31.8	46.0/44.8	38.5/36.8	49.0/44.8	33.1/33.1
Channel MetaICL	**49.1**/46.8	50.7/48.0	50.6/48.1	44.9/43.5	42.1/40.8	**54.6**/51.9	**52.2**/50.3
GPT-J Channel In-context	48.6/44.4	**51.5**/47.0	**51.5**/47.0	**47.0**/45.2	**47.0**/45.2	47.2/41.7	51.0/47.5
			Target tasks in unseen domains				
Channel In-context	39.6/33.6	39.6/33.6	39.6/33.6	44.7/40.6	44.7/40.6	40.4/35.7	44.1/36.8
MetaICL	35.3/32.7	32.3/29.3	28.1/25.1	**69.9**/68.1	48.3/47.2	**80.1**/77.2	34.0/34.0
Channel MetaICL	**47.7**/44.7	41.9/37.8	**48.0**/45.2	57.9/56.6	47.2/45.0	62.0/57.3	51.0/49.9
GPT-J Channel In-context	42.8/38.4	**42.8**/38.4	42.8/38.4	55.7/54.4	**55.7**/54.4	51.1/40.4	**52.0**/46.5

Table 2.4: Comparison between raw LM in-context learning (based on GPT-2 Large and GPT-J) and MetaICL (based on GPT-2 Large). GPT-2 Large used unless otherwise specified. Two numbers indicate the average and the worst-case performance over different seeds used for k target training examples. For raw LM baselines, Channel In-context is reported because it is the best raw LM baseline overall across the settings.

meta-training, because meta-training helps in supervised learning as it does in in-context learning. This indicates that there is still room for improvement in methods that allow learning without parameter updates.

Comparison to GPT-J. In Table 2.4, we compare GPT-2 Large based models with raw LM baselines based on GPT-J which consists of 6B parameters. MetaICL, despite being 8x smaller, outperforms or matches GPT-J baselines.

2.3.4 Ablations

Varying number of training examples. We vary the number of training examples (k) from 0, 4, 8, 16 to 32. In-context learning with $k = 0$ is equivalent to the zero-shot method. Results are shown in Figure 2.1. Increasing k generally helps across all models, and Channel MetaICL outperforms the raw in-context learning over all values of k. We additionally find that the performance tends to saturate when k is closer to 16, which we discuss in more detail in Section 2.4.2.

Number of meta-training tasks. To see the impact of the number of meta-training tasks, we subsample $\{7, 15, 30\}$ meta-training tasks out of 61 in the HR→LR setting. For each, we use ten different random seeds to additionally see the impact of the choice of meta-training tasks.

Figure 2.2 reports the results. On average, performance generally increases as the number of tasks increase, which is consistent with results in Mishra et al. [2022]; Wei et al. [2022a]. Across different numbers of meta-training tasks, Channel MetaICL consistently outperforms other models. Nonetheless, there is nonnegligible variance across different choices of meta-training (the bottom of Figure 2.2), indicating that a choice of meta-training gives substantial impact in performance.

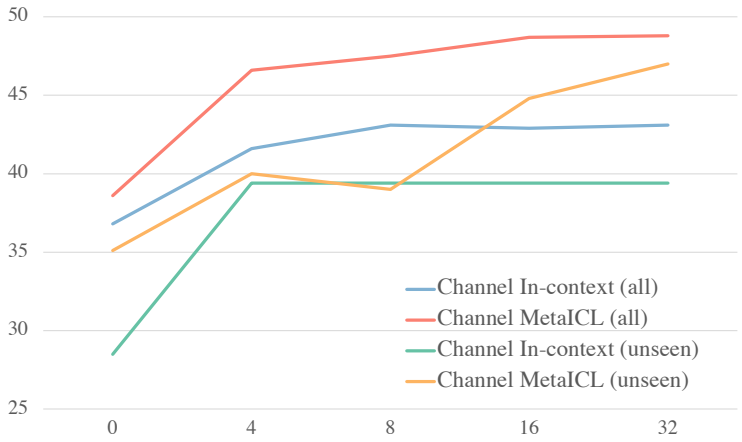

Figure 2.1: Ablation on the number of training examples (k) in the HR→LR setting. $k = 0$ is equivalent to the zero-shot methods.

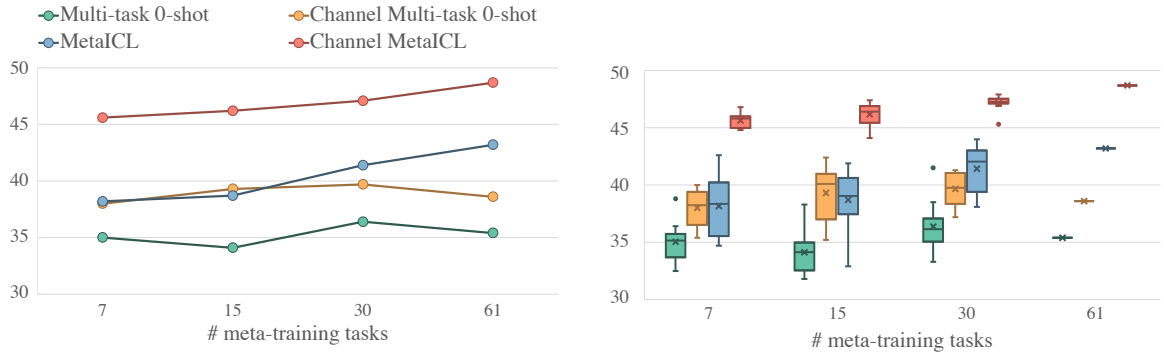

Figure 2.2: Ablation on the number of meta-training tasks ($\{7, 15, 30, 61\}$). The graph of the average (left) and the box chart (right) over different meta-training sets using 10 different random seeds (except for 61).

Diversity in meta-training tasks. We hypothesize that the diversity in meta-training tasks may impact performance of MetaICL. To verify this hypothesis, we create two settings by subsampling 13 out of 61 meta-training datasets in the HR→LR setting. One setting is diverse in their task formats and required capacities: QA, NLI, relation extraction, sentiment analysis, topic classification, hate speech detection and more. The other setting is less diverse, including tasks related to sentiment analysis, topic classification and hate speech detection only. A full list of datasets is reported in Appendix A.1. Using these two settings, we compare multi-task zero-shot transfer baselines and MetaICL.

Results are reported in Table 2.5. We find that MetaICL with a diverse set outperforms MetaICL with a non-diverse set by a substantial margin. This shows that diversity among meta-training tasks is one of substantial factors for the success of MetaICL.

We also conducted ablations that provide more insights on the choice of meta-training tasks, such as (1) high quality data with diverse domains tend to help (e.g., GLUE family [Wang et al., 2018a]) and (2)

Method	Diverse	No Diverse
0-shot	34.9	
PMI 0-shot	34.8	
Channel 0-shot	36.8	
In-context	38.2/35.4	
PMI In-context	38.9/33.3	
Channel In-context	42.9/38.5	
Multi-task 0-shot	35.2	29.9
Channel Multi-task 0-shot	41.6	38.3
MetaICL	45.6/43.4	38.8/35.4
Channel MetaICL	**47.2**/44.7	45.3/42.6

Table 2.5: Ablation on the diversity of meta-training tasks in the HR→LR setting. For both settings, the number of meta-training tasks is 13, and the number of target tasks is 26 as in the original HR→LR setting. A full list of meta-training tasks is shown in Appendix A.1.

Method	w/o Instruct	w/ Instruct	
# instruct/task	0	1	8.3
0-shot	33.3	34.2	
PMI 0-shot	34.6	27.8	
Channel 0-shot	32.5	30.6	
In-context	34.5/31.5	45.2/42.3	
PMI In-context	37.7/32.7	41.9/37.6	
Channel In-context	39.0/35.4	39.6/35.3	
MT 0-shot	35.7	32.6	37.1
Channel MT 0-shot	36.7	30.6	36.0
MetaICL	40.4/37.7	42.6/41.0	43.2/41.0
Channel MetaICL	42.2/40.0	45.3/43.9	**46.9**/44.2

Table 2.6: Ablation on the impact of *natural instructions*. 'w/ Instruct' uses instructions from Sanh et al. [2022], either one per meta-training task or all available ones; 'w/o Instruct' does not. '# instruct/task' indicates the number of instructions per meta-training task on average. 'MT 0-shot' indicates 'Multi-task 0-shot'. Both settings have the same meta-training and target tasks. A full list of tasks is shown in Appendix A.1.

adversarially collected data tends to be unhelpful. However, more systematic studies on how to choose the best meta-training tasks and how they relate to particular target tasks should be done, which we leave for future work. We refer readers to Appendix C.3 of the original publication [Min et al., 2022b] for more details.

Are instructions necessary? Most recent work has used human-written natural instructions for zero- or few-shot learning [Mishra et al., 2022; Wei et al., 2022a; Sanh et al., 2022]. While we argue for not using instructions to avoid manual engineering and high variance, we also ask: *are instructions still useful with MetaICL?* On one hand, learning to condition on k examples may remove the necessity of instructions. On the other hand, instructions may still be complementary and provide the model with extra useful infomration.

We aim to answer this question by using 32 meta-training tasks and 12 target tasks from the HR→LR setting for which human-written instructions are available in Sanh et al. [2022].[5] We have two variants: (a) using one instruction per meta-training task, and (b) using all available instructions including 267 instructions in total (8.3 per meta-training task) which Sanh et al. [2022] found to be better than (a). We then compare MetaICL and a range of baselines with and without instructions.

Results are reported Table 2.6. As in Wei et al. [2022a] and Sanh et al. [2022], Multi-task 0-shot outperforms the raw-LM 0-shot baseline. However, MetaICL with no instructions is better than Multi-task 0-shot with instructions. Furthermore, MetaICL achieves further improvements when instructions are jointly

[5] github.com/bigscience-workshop/promptsource

used, significantly outperforming all baselines. In fact, when increasing the number of instructions per task from 0, 1 to 8.3, performance of MetaICL improves much more than performance of Multi-task 0-shot does. To summarize, (1) learning to in-context learn (MetaICL) outperforms learning to learn from instructions; (2) MetaICL and using instructions are largely complementary, and (3) MetaICL actually benefits more from using instructions than Multi-task 0-shot does.

Importantly, Channel MetaICL trained on available tasks and instructions still achieves lower performance than Channel MetaICL without templates/instructions (46.9 from Table 2.6 vs. 49.1 from Table 2.3). This is likely because the model with instructions was trained with less meta-training tasks, which was unavoidable since instructions are only available on 32 out of 61 meta-training tasks. This supports our earlier choice of not using human-written templates/instructions, since writing templates and instructions for every task requires extensive effort.

It is worth noting that, it is nonetheless difficult to make direct comparisons with Wei et al. [2022a] and Sanh et al. [2022] because there are many moving components: size of LMs, types of LMs (e.g., causal LM vs. masked LM), splits between meta-training and target tasks, and more.

2.3.5 Summary & Limitations

We introduced MetaICL, a new few-shot learning method where an LM is meta-trained to learn to in-context learn, i.e. condition on training examples to recover the task and make predictions. We experiment with a large, diverse collection of tasks, consisting of 142 unique tasks in total and 52 unique target tasks, using seven different settings. MetaICL outperforms a range of strong baselines including in-context learning without meta-training and multi-task learning followed by zero-shot transfer, and outperforms or matches 8x bigger models. We identify ingredients for success of MetaICL such as the number and diversity of meta-training tasks. We also demonstrate that, while MetaICL is better than recent work using natural instructions, they are complementary and the best performance is achieved by integrating MetaICL with instructions.

Our work is limited in multiple dimensions. First, in-context learning approaches in general requires much longer context at both meta-training and inference due to feeding the concatenation of the training data, thus being less efficient compared to baselines that do not use in-context learning. Second, our work experiment with a casual language model with modest size (770M parameters). Future work may investigate extending our approach to a masked language model and a larger model. Third, our experiments focus on classification and multi-choice tasks where a set of candidate options is given. Future work may study applying our approach for a wider range of tasks including free-form generation. Other avenues for future work include further improving MetaICL to outperform supervised models with meta-training, identification of which meta-training tasks are helpful on target tasks, and how to better combine human-written instructions and MetaICL.

Model	# Params	Public	Meta-trained
GPT-2 Large	774M	✓	✗
MetaICL	774M	✓	✓
GPT-J	6B	✓	✗
fairseq 6.7B†	6.7B	✓	✗
fairseq 13B†	13B	✓	✗
GPT-3	175B‡	✗	✗

Table 2.7: A list of LMs used in the experiments: GPT-2 [Radford et al., 2019], MetaICL [Min et al., 2022b], GPT-J [Wang and Komatsuzaki, 2021], fairseq LMs [Artetxe et al., 2021] and GPT-3 [Brown et al., 2020a]. 'Public' indicates whether the model weights are public; 'Meta-trained' indicates whether the model is meta-trained with an in-context learning objective. †We use dense models in Artetxe et al. [2021] and refer them as fairseq LMs for convenience. ‡We use the Davinci API (the *base* version, not the *instruct* version) and assume it to be 175B, following Gao et al. [2020] and Artetxe et al. [2021].

2.4 Understanding In-Context Learning.

Despite in-context learning consistently outperforming zero-shot inference on a wide range of tasks, there is little understanding of *how* it works and *which* aspects of the demonstrations contribute to end task performance. In this section, we show that ground truth demonstrations are in fact not required—randomly replacing labels in the demonstrations barely hurts performance on a range of classification and multi-choce tasks, consistently over 12 different models including GPT-3. Instead, we find that other aspects of the demonstrations are the key drivers of end task performance, including the fact that they provide a few examples of (1) the label space, (2) the distribution of the input text, and (3) the overall format of the sequence. Our analysis provides a new way of understanding how and why in-context learning works, while opening up new questions about how much can be learned from large language models through inference alone.

2.4.1 Method

To see the impact of correctly-paired inputs and labels in the demonstrations—which we call the ground truth input-label mapping—we compare the following three methods.[6]

No demonstrations is a typical zero-shot method that does not use any labeled data. A prediction is made via $\text{argmax}_{y \in \mathcal{C}} P(y|x)$, where x is the test input and $\mathcal{C}$ is a small discrete set of possible labels.

Demonstrations w/ gold labels are used in a typical in-context learning method with k labeled examples $(x_1, y_1)...(x_k, y_k)$. A concatenation of k input-label pairs is used to make a prediction via $\text{argmax}_{y \in \mathcal{C}} P(y|x_1, y_1...x_k, y_k, x)$.

Demonstrations w/ random labels are formed with random labels, instead of gold labels from the labeled data. Each x_i $(1 \leq i \leq k)$ is paired with $\tilde{y}_i$ that is randomly sampled at uniform from $\mathcal{C}$. A concatenation of $(x_1, \tilde{y}_1)...(x_k, \tilde{y}_k)$ is then used to make a prediction via $\text{argmax}_{y \in \mathcal{C}} P(y|x_1, \tilde{y}_1...x_k, \tilde{y}_k, x)$.

[6]Without loss of generality, all methods are described based on the direct method, but can be trivially converted to the channel method by flipping x and y.

We used the following experimental setup.

Models. We experiment with 12 models in total. We include 6 language models (Table 2.7), all of which are decoder-only, dense LMs. We use each LM with two inference methods, direct and channel, following Min et al. [2022a]. The sizes of LMs vary from 774M to 175B. We also include MetaICL, our model in Section 2.3 that is initialized from GPT-2 Large and then meta-trained on a collection of supervised datasets with an in-context learning objective. We ensure that our evaluation datasets do not overlap with those used at meta-training time.

Evaluation Data. We evaluate on 26 datasets, including sentiment analysis, paraphrase detection, natural language inference, hate speech detection, question answering, and sentence completion. They are exactly the same as the evaluation datasets used in the HR→LR setting in Section 2.3 (full list and references provided in Appendix A.1).[7] All datasets are classification and multi-choice tasks.

We use these datasets because they (1) are true low-resource datasets with less than 10K training examples, (2) include well-studied benchmarks from GLUE [Wang et al., 2018a] and SuperGLUE [Wang et al., 2019a], and (3) cover diverse domains including science, social media, finance, and more.

Other Details. We use $k = 16$ examples as demonstrations by default for all experiments in the paper, unless otherwise specified. Examples are sampled at uniform from the training data. We choose a set of k training examples using 5 different random seeds and run experiments 5 times. For fairseq 13B and GPT-3, due to limited resources, we experiment with a subset of 6 datasets[8] and 3 random seeds. We report Macro-F1[9] for classification tasks and Accuracy for multi-choice tasks. We compute per-dataset average over seeds, and then report macro-average over datasets. We use the minimal templates in forming an input sequence from an example. All experiments are reproducible from `github.com/Alrope123/rethinking-demonstrations`.

2.4.2 Ground Truth Matters Little

Results are reported in Figure 2.3. First, using the demonstrations with gold labels significantly improves the performance over no demonstrations,[10] as it has been consistently found in much of prior work [Brown et al., 2020a; Zhao et al., 2021; Liu et al., 2021]. We then find that **replacing gold labels with random labels only marginally hurts performance**. The trend is consistent over nearly all models: models see performance

[7] For convenience, we use 'labels' to refer to the output for the task, though our datasets include non-classification tasks.

[8] Three classification and three multi-choice: MRPC, RTE, Tweet_eval-hate, OpenbookQA, CommonsenseQA, COPA.

[9] Known to be better for imbalanced classes.

[10] There are some exceptions, e.g., in the classification tasks, Direct GPT-2, Direct GPT-J and Direct fairseq 6.7B models are not significantly better than random guessing on many datasets; Channel fairseq 13B has significantly better no-demonstrations performance compared to demonstrations with gold labels. We thus discuss the results from these models less significantly for the rest of analysis.

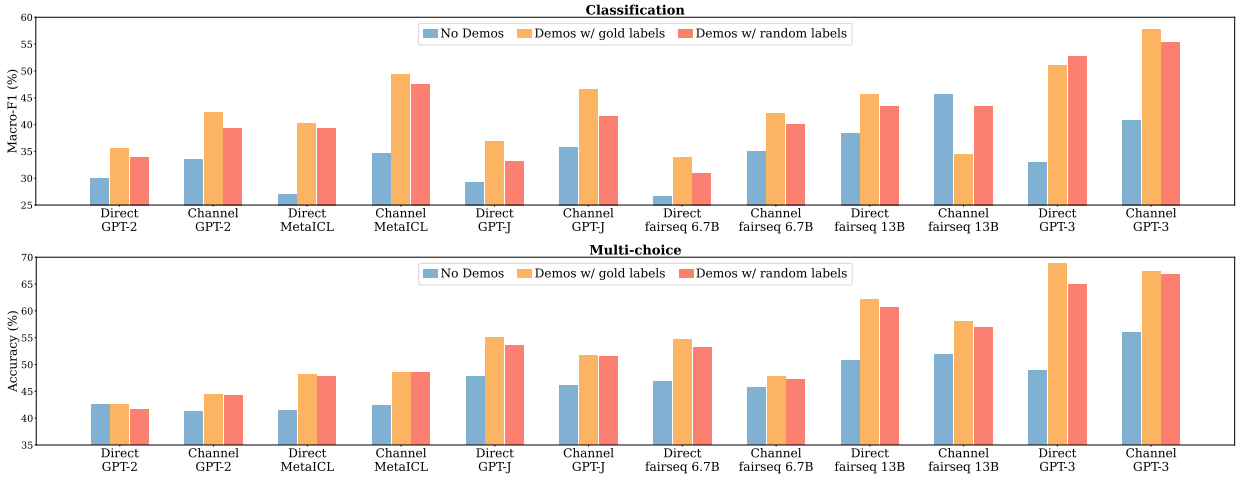

Figure 2.3: Results when using no-demonstrations, demonstrations with gold labels, and demonstrations with random labels in classification (top) and multi-choice tasks (bottom). The first eight models are evaluated on 16 classification and 10 multi-choice datasets, and the last four models are evaluated on 3 classification and 3 multi-choice datasets. **Performance with random labels is very close to performance with gold labels.**

drop in the range of 0–5% absolute. There is less impact in replacing labels in multi-choice tasks (1.7% on average) than in classification tasks (2.6% absolute).

This result indicates that the ground truth input-label pairs are not necessary to achieve performance gains. This is counter-intuitive, given that correctly paired training data is critical in typical supervised training—it informs the model of the expected input-label *correspondence* required to perform the downstream task. Nonetheless, the models *do* achieve non-trivial performance on the downstream tasks. This strongly suggests that the models are capable of recovering the expected input-label correspondence for the task; however, it is *not* directly from the pairings in the demonstrations.

It is also worth noting that there is particularly little performance drop in MetaICL: 0.1–0.9% absolute. This suggests that meta-training with an explicit in-context learning objective actually encourages the model to essentially ignore the input-label mapping and exploit other components of the demonstrations (more discussion in Section 2.4.3).

In Appendix A.2, we provide additional results showing that (1) selecting random labels from a true distribution of labels (instead of a uniform distribution) reduces the gap even further, and (2) the trends may depend on the dataset, although the overall trend is consistent over most datasets.

Does the number of correct labels matter? To further examine the impact of correctness of labels in the demonstrations, we conduct an ablation study[11] by varying the number of correct labels in the demonstrations. We evaluate "Demonstrations w/ a% correct labels" ($0 \leq a \leq 100$) which consist of $k \times a/100$ correct

[11]For ablation studies, we experiment with 5 classification and 4 multi-choice datasets. Classification includes: MRPC, RTE, Tweet_eval-hate, SICK, poem-sentiment; Multi-choice includes OpenbookQA, CommonsenseQA, COPA and ARC.

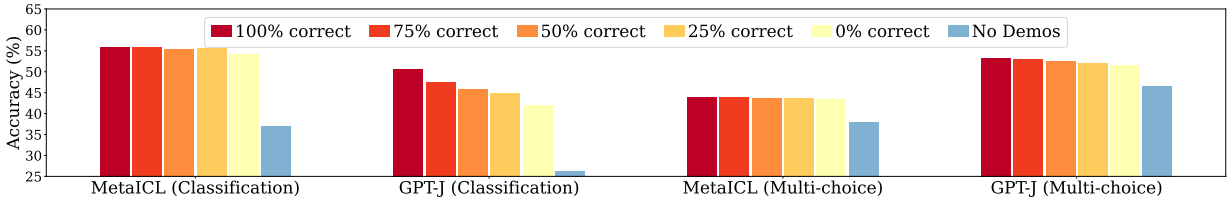

Figure 2.4: Results with varying number of correct labels in the demonstrations. Channel and Direct used for classification and multi-choice, respectively. Performance with no demonstrations (blue) is reported as a reference.

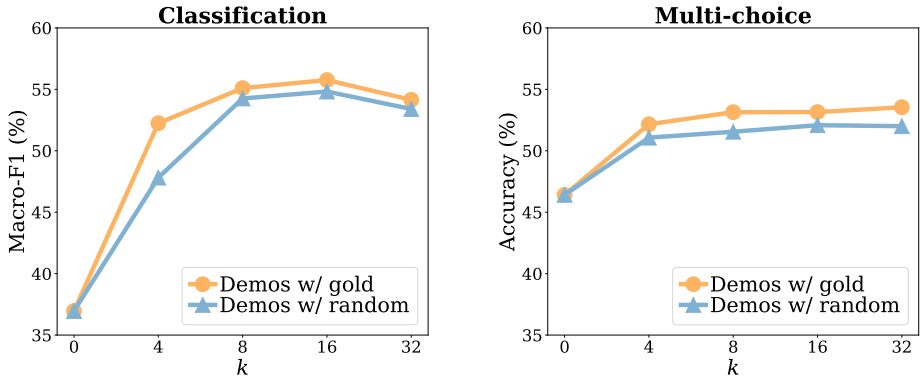

Figure 2.5: Ablations on varying numbers of examples in the demonstrations (k). Models that are the best under 13B in each task category (Channel MetaICL and Direct GPT-J, respectively) are used.

pairs and $k \times (1 - a/100)$ incorrect pairs. Here, $a = 100$ is the same as typical in-context learning, i.e., demonstrations w/ gold labels.

Results are reported in Figure 2.4. Model performance is fairly insensitive to the number of correct labels in the demonstrations. In fact, always using incorrect labels significantly outperforms no-demonstrations, e.g., preserving 92%, 100% and 97% of improvements from using the demonstrations with MetaICL in classification, MetaICL in multi-choice, and GPT-J in multi-choice, respectively. In contrast, GPT-J in classification sees relatively significant performance drop with more incorrect labels, e.g., nearly 10% drop in performance when always using incorrect labels. Still, always using incorrect labels is significantly better than no demonstrations.

Is the result consistent with varying k? We study the impact of the number of input-label pairs (k) in the demonstrations. Results are reported in Figure 2.5. First, using the demonstrations significantly outperforms the no demonstrations method even with small k ($k = 4$), and performance drop from using gold labels to using random labels is consistently small across varying k, in the range of 0.8–1.6%.[12] Interestingly, model performance does not increase much as k increases when $k \geq 8$, both with gold labels and with random labels.

[12]With an exception of 4.4% in classification with $k = 4$, likely due to a high variance with a very small value of k.

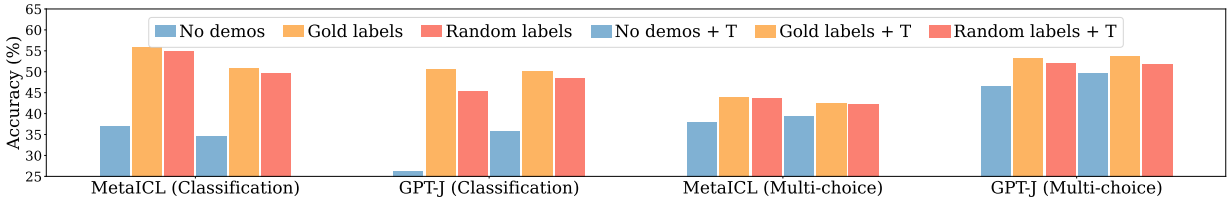

Figure 2.6: Results with minimal templates and manual templates. '+T' indicates that manual templates are used. Channel and Direct used for classification and multi-choice, respectively.

This is in contrast with typical supervised training where model performance rapidly increases as k increases, especially when k is small. We hypothesize that larger labeled data is beneficial mainly for supervising the input-label correspondence, and other components of the data like the example inputs, example labels and the data format are easier to recover from the small data, which is potentially a reason for minimal performance gains from larger k (more discussion in Section 2.4.3).

Is the result consistent with better templates? While we use minimal templates by default, we also explore manual templates, i.e., templates that are manually written in a dataset-specific manner, taken from prior work (details in Appendix A.2). Figure 2.6 shows that the trend—replacing gold labels with random labels barely hurting performance—holds with manual templates. It is worth noting that using manual templates does not always outperform using minimal templates.

2.4.3 Why *does* In-Context Learning work?

Section 2.4.2 shows that the ground truth input-label mapping in the demonstrations has little impact to performance gains from in-context learning. This section further examines what other aspects of the demonstrations lead to good performance of in-context learning.

We identify four aspects of the demonstrations $(x_1, y_1)...(x_k, y_k)$ that potentially provide learning signal (depicted in Figure 2.7).

- **The input-label mapping**, i.e., whether each input x_i is paired with a correct label y_i.
- **The distribution of the input text**, i.e., the underlying distribution that $x_1...x_k$ are from.
- **The label space**, i.e., the space covered by $y_1...y_k$.
- **The format**—specifically, the use of input-label pairing as the format.

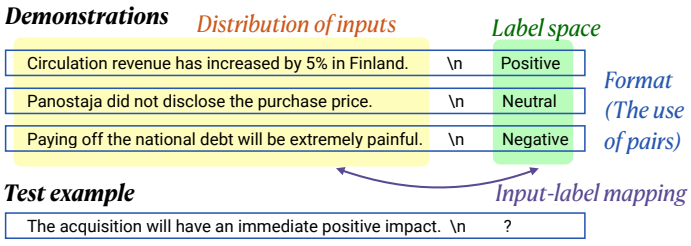

Figure 2.7: Four different aspects in the demonstrations: the input-label mapping, the distribution of the input text, the label space, and the use of input-label pairing as the format of the demonstrations.

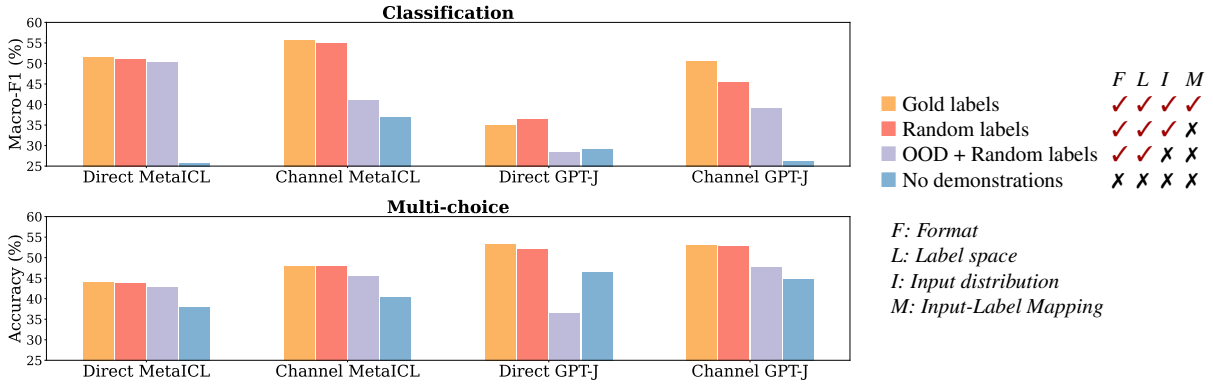

Figure 2.8: Impact of the distribution of the inputs. Evaluated in classification (top) and multi-choice (bottom). The impact of the distribution of the input text can be measured by comparing ■ and ■. The gap is substantial, with an exception in Direct MetaICL (discussion in Section 2.4.3).

As Section 2.4.2 does for the input-label mapping, we design a series of variants of the demonstrations that quantify the impact of each aspect in isolation (Section 2.4.3–2.4.3). We then additionally discuss the trend of the models meta-trained with an in-context learning objective (Section 2.4.3). For all experiments, models are evaluated on five classification and four multi-choice datasets as in Section 2.4.2. See Appendix A.2 and Table A.4 for implementation details and example demonstrations, respectively.

Impact of the distribution of the input text

We experiment with **OOD demonstrations** which include out-of-distribution (OOD) text instead of the inputs from unlabeled training data. Specifically, a set of k sentences $\{x_{i,\text{rand}}\}_{i=1}^{k}$ are randomly sampled from an external corpus, and replace $x_1 ... x_k$ in the demonstrations. This variant assesses the impact of the distribution of the input text, while keeping the label space and the format of the demonstrations.

Figure 2.8 shows that using out-of-distribution inputs instead of the inputs from the training data significantly drops the performance when Channel MetaICL, Direct GPT-J or Channel GPT-J are used, both in classification and multi-choice, by 3–16% in absolute. In the case of Direct GPT-J in multi-choice, it is even significantly worse than no demonstrations. Direct MetaICL is an exception, which we think is the effect of meta-training (discussion in Section 2.4.3). This suggests that in-distribution inputs in the demonstrations substantially contribute to performance gains. This is likely because conditioning on the in-distribution text makes the task closer to language modeling, since the LM always conditioned on the in-distribution text during training.

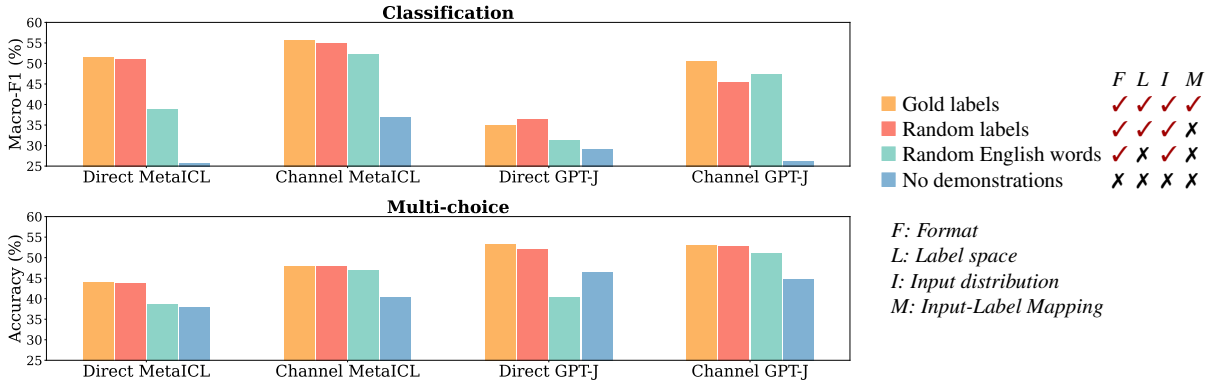

Figure 2.9: Impact of the label space. Evaluated in classification (top) and multi-choice (bottom). The impact of the label space can be measured by comparing ■ and ■. The gap is significant in the direct models but not in the channel models (discussion in Section 2.4.3).

Impact of the label space

We also experiment with **demonstrations w/ random English words** that use random English words as labels for all k pairs. Specifically, we sample a random subset of English words $\mathcal{C}_{\text{rand}}$ where $|\mathcal{C}_{\text{rand}}| = |\mathcal{C}|$, and randomly pair $\tilde{y}_i \in \mathcal{C}_{\text{rand}}$ with x_i. This variant assesses the impact of the label space, while keeping the distribution of the input text and the format of the demonstrations.

Based on Figure 2.9, direct models and channel models exhibit different patterns. With direct models, the performance gap between using random labels within the label space and using random English words is significant, ranging between 5–16% absolute. This indicates that conditioning on the label space significantly contributes to performance gains. This is true even for multi-choice tasks where there is no fixed set of labels—we hypothesize that multi-choice tasks still do have a particular distribution of the choices (e.g., objects like "Bolts" or "Screws" in the OpenBookQA dataset) that the model uses. On the other hand, removing the output space does not lead to significant drop in the channel models: there is 0–2% drop in absolute, or sometimes even an increase. We hypothesize that this is because the channel models only condition on the labels, and thus are not benefiting from knowing the label space. This is in contrast to direct models which must *generate* the correct labels.

Impact of input-label pairing

Section 2.4.3 and 2.4.3 focus on variants which keep the format of the demonstrations as much as possible. This section explores variants that change the format. While there are many aspects of the format, we make minimal modifications to remove the pairings of inputs to labels. Specifically, we evaluate **demonstrations with no labels** where the LM is conditioned on the concatenation of $x_1 ... x_k$, and **demonstrations with labels only** where the LM is conditioned on the concatenation of $y_1 ... y_k$. These ablations provide the no-format

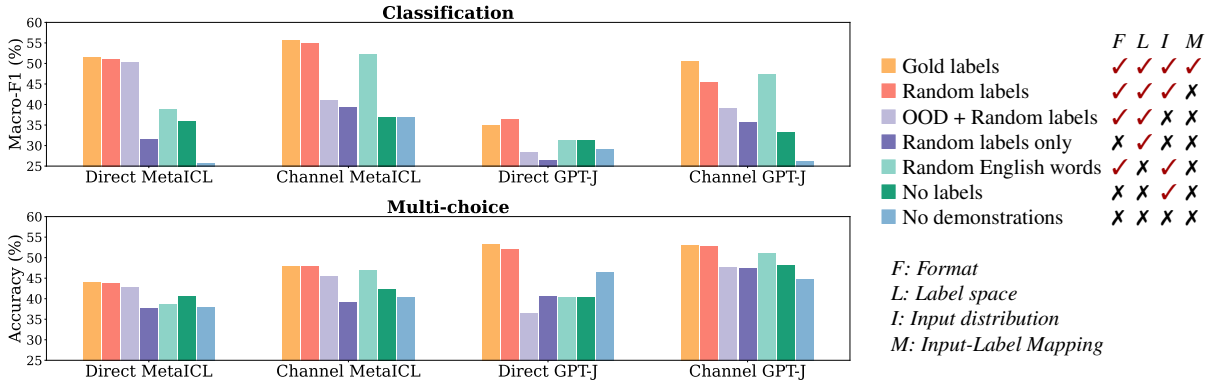

Figure 2.10: Impact of the format, i.e., the use of the input-label pairs. Evaluated in classification (top) and multi-choice (bottom). Variants of demonstrations without keeping the format (■ and ■) are overall not better than no demonstrations (■). Keeping the format is especially significant when it is possible to achieve substantial gains with the label space but without the inputs (■ vs. ■ in Direct MetaICL), or with the input distribution but without the labels (■ vs. ■ in Channel MetaICL and Channel GPT-J). More discussion in Section 2.4.3.

counterparts of the 'demonstrations with random English words' and 'demonstrations with OOD inputs', respectively.

Based on Figure 2.10, removing the format is close to or worse than no demonstrations, indicating the importance of the format. This is likely because conditioning on a sequence of input-label pairs triggers the model to mimic the overall format and complete the new example as expected when the test input is given.

More interestingly, keeping the format plays a significant role in retaining a large portion of performance gains by only using the inputs or only using the labels. For instance, with Direct MetaICL, it is possible to retain 95% and 82% of improvements from in-context learning (demonstrations with gold labels) by simply sampling random sentences from a corpus and randomly pairing them with the label set (■ in Figure 2.10) in classification and multi-choice, respectively. Similarly, with the channel models, it is possible to retain 82%, 87%, 86% and 75% of improvements from in-context learning by simply pairing each input from the unlabeled training data with a random English word (■ in Figure 2.10) in MetaICL classification, GPT-J classification, MetaICL multi-choice and GPT-J multi-choice, respectively. For all of these cases, removing inputs instead of using OOD inputs, or removing labels instead of using random English words is significantly worse, indicating that **keeping the format of the input-label pairs is key**.

Impact of meta-training

Different from other models, MetaICL is trained with an in-context learning objective (Section 2.3). We aim to better understand the role of this meta-training in relation with our findings by closely examining the result of MetaICL. In particular, the patterns we see so far are significantly more evident with MetaICL than with

other models. For instance, the ground truth input-label mapping matters even less, and keeping the format of the demonstrations matters even more. There is nearly zero influence of the input-label mapping and the input distribution in Direct MetaICL, and the input-label mapping and the output space in Channel MetaICL.

Based on this observation, we hypothesize that **meta-training encourages the model to exclusively exploit simpler aspects of the demonstrations and to ignore others**. This is based on our intuition that (1) the input-label mapping is likely harder to exploit, (2) the format is likely easier to exploit, and (3) the space of the text that the model is trained to generate is likely easier to exploit than the space of the text that the model conditions on.[13]

2.4.4 Summary, Discussion, & Limitations

In summary, we study the role of the demonstrations with respect to the success of in-context learning. We find that the ground truth input-label mapping in the demonstrations matters significantly less than one might think—replacing gold labels with random labels in the demonstrations only marginally lowers the performance. We then identify a series of aspects in the demonstrations and examine which aspect actually contributes to performance gains. Results reveal that (1) gains are mainly coming from *independent* specification of the input space and the label space, (2) the models can still retain up to 95% of performance gains by using either the inputs only or the label set only if the right format is used, and (3) meta-training with an in-context learning objective magnifies these trends. Together, our findings lead to a set of broader indications about in-context learning, as well as avenues for future work.

Does the model *learn* at test time? If we take a strict definition of learning: capturing the input-label correspondence given in the training data, then our findings suggest that LMs do not learn new tasks at test time. Our analysis shows that the model may ignore the task defined by the demonstrations and instead use prior from pre-training.

However, *learning* a new task can be interpreted more broadly: it may include adapting to specific input and label distributions and the format suggested by the demonstrations, and ultimately getting to make a prediction more accurately. With this definition of learning, the model *does* learn the task from the demonstrations. Our experiments indicate that the model *does* make use of aspects of the demonstrations and achieve performance gains.

Capacity of LMs. The model performs a downstream task without relying on the input-label correspondence from the demonstrations. This suggests that the model has learned the (implicit notion of) input-label correspondence from the language modeling objective alone, e.g., associating a positive review with the word

[13] That is, the direct model exploits the label space better than the input distribution, and the channel model exploits the input distribution better than the label space.

'positive'. This is in line with Reynolds and McDonell [2021] who claim that the demonstrations are for *task location* and the intrinsic ability to perform the task is obtained at pre-training time.[14]

On one hand, this suggests that the language modeling objective has led to great zero-shot *capacity*, even if it is not always evident from the naive zero-shot *accuracy*. On the other hand, this suggests that in-context learning may not work on a task whose input-label correspondence is not already captured in the LM. This leads to the research question of how to make progress in NLP problems that in-context learning does not solve: whether we need a better way of extracting the input-label mappings that are already stored in the LM, a better variant of the LM objective that learns a wider range of task semantics, or explicit supervision through fine-tuning on the labeled data.

Connection to instruction-following models. Prior work has found it promising to train the model that reads the natural language description of the task (called instructions) and performs a new task at inference [Mishra et al., 2022; Efrat and Levy, 2020; Wei et al., 2022a; Sanh et al., 2022]. We think the demonstrations and instructions largely have the same role to LMs, and hypothesize that our findings hold for instruction-following models: the instructions prompt the model to recover the capacity it already has, but do not supervise the model to learn novel task semantics. This has been partially verified by Webson and Pavlick [2022] who showed that the model performance does not degrade much with irrelevant or misleading instructions. We leave more analysis on instruction-following models for future work.

Significantly improved zero-shot performance. One of our key findings is that it is possible to achieve nearly k-shot performance without using any labeled data, by simply pairing each unlabeled input with a random label and using it as the demonstrations. This means our zero-shot baseline level is significantly higher than previously thought.[15] Future work can further improve the zero-shot performance with relaxed assumptions in access to the unlabeled training data.

Limitations. This paper focuses on the tasks from established NLP benchmarks that have *real* natural language inputs. Synthetic tasks with more limited inputs may actually use the ground truth labels more, as observed by Rong [2021].

We report macro-level analysis by examining the average performance over multiple NLP datasets, but different datasets may behave differently. Appendix A.2 discusses this aspect, including findings that there are larger gaps between using the ground truth labels and using the random labels in some dataset-model pairs (e.g., in the most extreme case, nearly 14% absolute on the financial_phrasebank dataset with GPT-J).

Our experiments are limited to classification and multi-choice tasks. We hypothesize that ground truth output may not be necessary for in-context learning in the open-set tasks such as generation, but leave this to

[14]However, while Reynolds and McDonell [2021] claims that the demonstrations are thus unnecessary, we think using the demonstrations is actually the most unambiguous and the easiest way to prompt the model to perform a task.

[15]We take the perspective that using the unlabeled training data is permitted [Kodirov et al., 2015; Wang et al., 2019b; Schick and Schütze, 2021].

future work. Extending of our experiments to such tasks is not trivial, because it requires a variation of the output which has incorrect input-output correspondence while keeping the correct output distribution (which is important based on our analysis in Section 2.4.3).

2.5 Discussion of Subsequent Work

Our work presented in this section has led to a significant body of follow-up research, including LMs that adopted meta-training [Chung et al., 2022; Wang et al., 2022; Ye et al., 2023b; Gu et al., 2023; Ye et al., 2023a; Ivison et al., 2023], work discovering unexpected behaviors of LMs [Madaan and Yazdanbakhsh, 2022; Wei et al., 2023b; Burns et al., 2022; Halawi et al., 2023; Jang et al., 2022b; Wies et al., 2023; Lampinen et al., 2022; Pan et al., 2023; Kim et al., 2022; Schaeffer et al., 2023; Zhang et al., 2023b; She et al., 2023; Turpin et al., 2023], and work that used our analysis to inspire the development of better models [Lyu et al., 2023; Wei et al., 2023a]. In this section, we discuss closely relevant work at the time of writing this thesis.

Work that adopted meta-training. Since the introduction of MetaICL and other concurrent work in training the LM with in-context learning and natural instructions [Chen et al., 2021; Sanh et al., 2022; Wei et al., 2022a], this recipe has become a standard practice in state-of-the-art LM alignment pipelines. MetaICL is incorporated into FLAN-PaLM, one of Google's leading models whose training recipe is publicly available [Chung et al., 2022], and Tulu v2, one of the best, fully open sourced instruction-tuned models [Ivison et al., 2023]. Arguably, the significance of the ability to perform in-context learning may have diminished in recent models. This is partially because the use of natural instructions alone is often enough to achieve good performance with a sufficiently large base model. Additionally, it is generally easier for users to provide natural instructions than to write a handful of labeled examples. Nevertheless, for complex tasks requiring a chain-of-thought prompting—a variant of in-context learning [Wei et al., 2022b]—further training with in-context examples remains critical.

There have also been counterexamples: for instance, Iyer et al. [2022] found that incorporating MetaICL can actually decrease performance in generation tasks, and its improvements are largely limited to classification and multiple-choice tasks. They suggested that MetaICL may cause the model to become overfitted to the format used in in-context examples. We consider this hypothesis likely to be true; in fact, it aligns with our findings in Section 2.4.3. We also posit that differences in results are likely due to variations in settings, such as the choice of base models, meta-training datasets, and evaluation datasets, and these need further investigation.

How are findings in Section 2.4 applied to the current state-of-the-art LMs? Several subsequent work studied the extent to which our findings generalize to larger, state-of-the-art LMs.

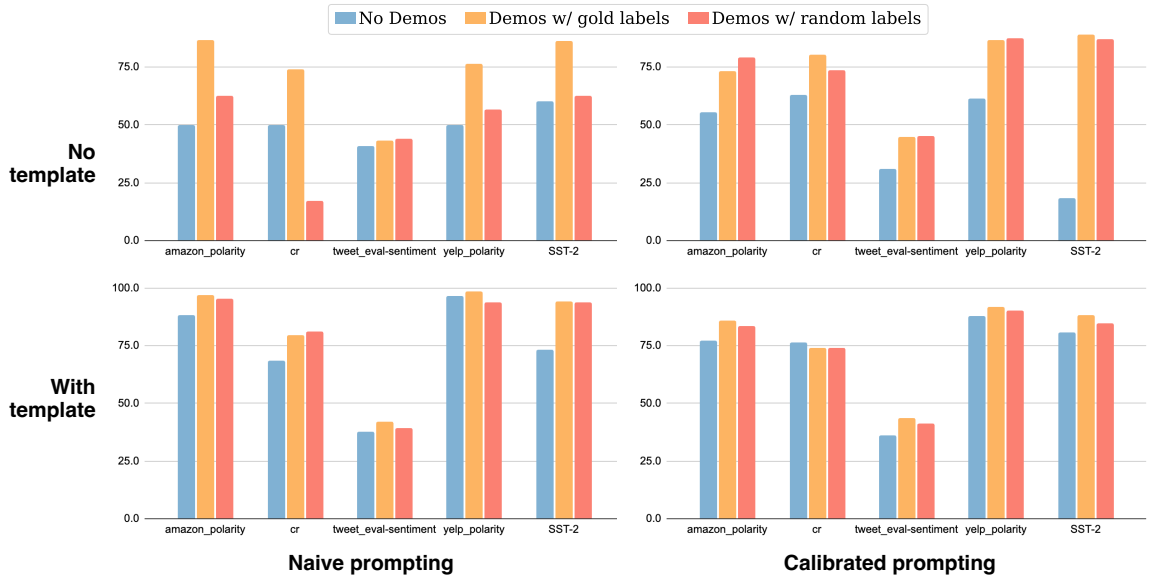

Figure 2.11: Results with various configurations following Kim et al. [2022]: with and without a template, and with and without calibration. Using random labels significantly degrades performance with no template and naive prompting (top left) but not with other configurations.

Kim et al. [2022] studied the impact of using random or negated labels in in-context learning through various conditions; they claimed that using random labels can perform poorly depending on the choice of datasets, prompting methods (e.g., whether to apply calibration), and the template used for prompting. We find that: (1) although certain combinations of settings (e.g., not using calibration and templates in prompting; see Figure 2.11) lead to significant degradation in performance with random labels, *in most settings, random labels have minimal impact on performance*; and (2) because the setting for the best in-context learning performance sees minimal impact from using random labels, *our key claim that correct labels are not essential for in-context learning remains true*. For further discussion, we refer readers to slides 44 to 51 from our presentation at the EMNLP 2023 Workshop.

Wei et al. [2023b] argued that large LMs perform in-context learning in a qualitatively different manner, claiming that the capability to in-context learn with negated labels[16] emerges with scale. Although we generally agree that current state-of-the-art LMs perform in-context learning much better, we think that care must be taken in evaluating the impact of compounding factors. For instance, we posit that results in Wei et al. [2023b] reveal that in-context learning with negated labels is near random guessing across all model sizes for models without instruction tuning (see 'GPT-3' in Figure 2.12), indicating that the model size is not a contributing factor. This holds true even with instruction tuning up to OpenAI's `text-davinci-001`, although `text-davinci-001` performs well on in-context learning with negated labels. Therefore,

[16]For instance, assigning `negative` to a positive review and `positive` to a negative review.

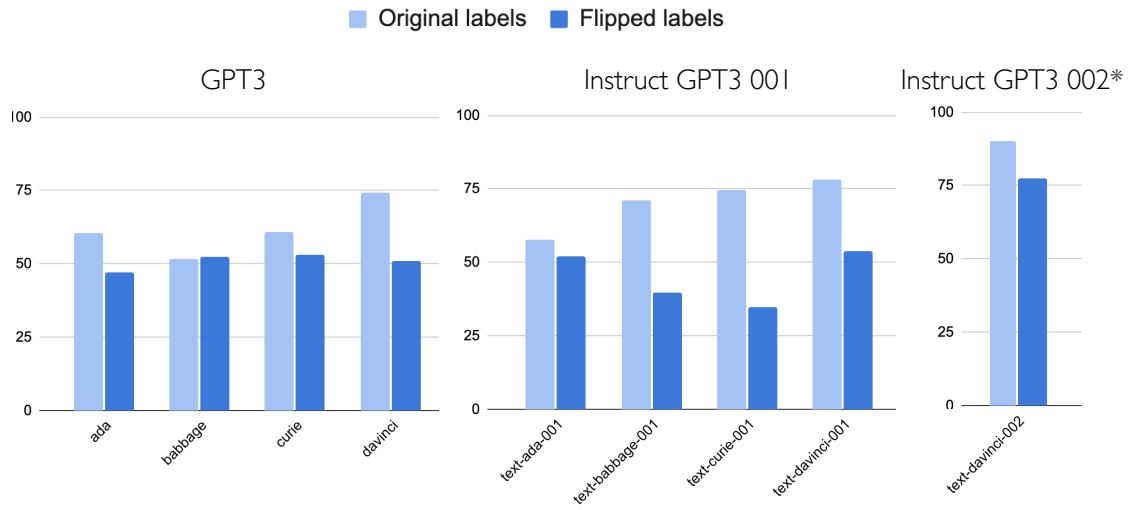

Figure 2.12: In-context learning with original labels and negated labels. Results are from Wei et al. [2023b]; we redrew the figures. *: `text-davinci-002` was released after our paper was posted. All models except for the 002 one achieve near random guessing performance (50%) with flipped labels, suggesting that even large models cannot perform in-context learning when the task defined by in-context examples does not entail with semantics from pre-training.

through some LLMs that are available to us can perform in-context learning with negated labels, we maintain that it is difficult to attribute this solely to model size; it is more likely influenced by some unknown variant of instruction tuning.[17]

Combining findings from Section 2.3 and 2.4. Wei et al. [2023a] proposed a new meta-training method that uses $k+1$ labeled examples $(x_1, y_1), \cdots, (x_k, y_k), (x_{k+1}, y_{k+1})$ and maximizes $P(y_{k+1}|x_1, y_1, \cdots, x_k, y_k, x_{k+1})$, as MetaICL does. However, instead of using standard NLP tasks for training, they used synthetic tasks designed to isolate the impact of semantics from the training data, e.g., assigning `negative` to a positive review and `positive` to a negative review, or assigning `foo` to a positive review and `bar` to a negative review. Their experiments demonstrate that models trained on this objective show improved performance in in-context learning with negated labels or semantically unrelated labels.

Extensions to generation tasks. Since the initial release of our paper, Madaan and Yazdanbakhsh [2022] conducted a similar analysis with the chain of thought prompting [Wei et al., 2022b], which generates a rationale for performing complex tasks such as arithmetic reasoning. They find that simply using random

[17] Wei et al. [2023b] also experimented with in-context learning with semantically unrelated labels, e.g., assigning `foo` to a positive review and `bar` to a negative review. Our findings largely remain the same: plain language models achieve near random guessing performance over all model sizes, although instruction-tuned models achieve better performance. For further discussion, please refer to slides 55 to 63 from the presentation at the EMNLP 2023 Workshop.

rationales (e.g., pairing a rationale from a different example) significantly degrades performance, but other types of incorrect rationales (e.g., incorrect equations) do not degrade performance.

Our own work [Wang et al., 2023a] further studied the impact of using rationales based on flawed reasoning, broken down by different categories of flaws. For instance, we find that maintaining input query relevance and rationale coherence is critical, e.g., not keeping keywords from the input query, or using `32 + 42 = 74` before the introduction of `32` or `42`, significantly degrade performance. Nevertheless, as long as relevance and coherence are maintained, inaccuracies in the underlying reasoning only marginally affect performance. Notably, this approach preserves 81-91% of the accuracy in arithmetic reasoning and 87-98% in multi-hop question answering across a range of state-of-the-art models, including `text-davinci-002` and `text-davinci-003`, as well as the largest versions of PaLM and FLAN-PaLM. For further discussion, see Madaan and Yazdanbakhsh [2022] and Wang et al. [2023a].

Chapter 3

Nonparametric Language Models

3.1 Overview

This section describes our work on **nonparametric LMs**, a new class of LMs that include both learned parameters and *a datastore*, i.e., a massive collection of raw text documents. During inference, these models can identify relevant text from the datastore and reason with it, unlike conventional models that must remember every relevant detail from the training set. These models are called nonparametric, because their data distribution is not defined by a fixed set of parameters; rather, it is a function of the available data [Siegel, 1957; Hollander et al., 2013]. With complexity that grows as the data grows, they are differentiated from parametric models, whose complexity is bounded a priori. As noted by Freeman et al. [2002], the term nonparametric does not imply an absence of parameters but rather indicates that the number and nature of the *effective* parameters are flexible and depend on the data.

Nonparametric LMs offer several advantages, as described below.

Parameter efficiency. Conventional models require a large number of parameters in order to memorize every relevant detail from the training set [Brown et al., 2020a; Rae et al., 2021; Chowdhery et al., 2022]. Nonparametric models remove the need for such extensive memorization since they can look up relevant information from a datastore. Consequently, they outperform heavily parameterized models, e.g., LMs with 30x more parameters [Raffel et al., 2020] in answering factoid questions (Min et al. [2021a]; Section 3.2.6).

Capturing of long-tail distributions. Standard state-of-the-art models often fail to capture information within long-tail distributions in the training data. Nonparametric models address this issue by retrieving relevant information at test time, significantly improving accuracy in long-tail distribution data, as shown in Mallen et al. [2022]; Min et al. [2023b]. This is also evident in commercial systems; for instance, our findings in Min et al. [2023a] demonstrate that factual accuracy in writing biographies increases from 58% with ChatGPT to 72% with PerplexityAI (ChatGPT with a search engine that uses a web datastore).

Ease of updating. Conventional models are strictly inflexible and remain stale once training is completed. In contrast, nonparametric LMs are more flexible by design since the datastore can be altered at any time, e.g., to include up-to-date information without additional training [Min et al., 2021a; Izacard et al., 2022b; Min et al., 2023b].

Ease of data removal. Removing the effect of specific data from conventional LMs after training is very challenging. Despite techniques like machine unlearning, [Cao and Yang, 2015; Bourtoule et al., 2020; Jang et al., 2023b], such models lack guarantees and are difficult to scale. Nonparametric models, however, enable direct removal of data from the datastore, effortlessly eliminating its influence without requiring additional training.

Ease of data attributions. Nonparametric models allow inherent data attributions since they are straightforward to trace, i.e., demonstrate which pieces of text in a datastore contribute to the model prediction. This lets users verify the model output and sources of information.

A combination of the last two features also helps **mitigate legal risks in LM training**, which we will discuss further in Chapter 4.

The concept of integrating an LM with a text datastore is not new and has been extensively studied in prior research [Chen et al., 2017; Gu et al., 2018; Song et al., 2018; Tian et al., 2019]. We provide background context in Section 3.2.3 and discuss key contributions we made for a nonparametric approach within the context of language models.

In Section 3.2, we will discuss retrieval augmentation, which operates in two stages—(1) retrieving small amount of text from a datastore, and (2) feeding it to the LM as an additional input (Figure 3.1 (a)). While the approach is highly intuitive and resembles how humans use a search engine to find relevant information and process it, it is challenging to efficiently train a neural model to reason with a large-scale datastore. We introduced a series of first retrieval-augmented models that addressed these challenges and has subsequently prompted active follow-up research [Karpukhin et al., 2020; Min et al., 2019b, 2021a; Shi et al., 2024b,a]. One notable model, Dense Passage Retrieval (DPR) [Karpukhin et al., 2020], opened up a new era in neural retrieval by showing its promises over traditional, lexical matching-based retrieval. This was done by introducing a new *contrastive* objective that helps the model distinguish relevant text against irrelevant-but-distracting text. We also improved the second stage of retrieval augmentation by training the LM to condition on a set of relevant documents, thereby making better use of text retrieved from a datastore [Shi et al., 2024a]. Altogether, this approach improves performance on a wide range of tasks (e.g., providing more factual text) and handles rarely seen concepts and facts much better, even when compared to state-of-the-art commercial LMs like GPT-3 [Shi et al., 2024b]. They are also easily updated, as we showed for the first time [Min et al., 2021a], and significantly reduce the model size.

While retrieval augmentation has made significant impact in both academia and industry, whether it is the optimal approach for using the datastore remains an open question. Specifically, the design of the architecture

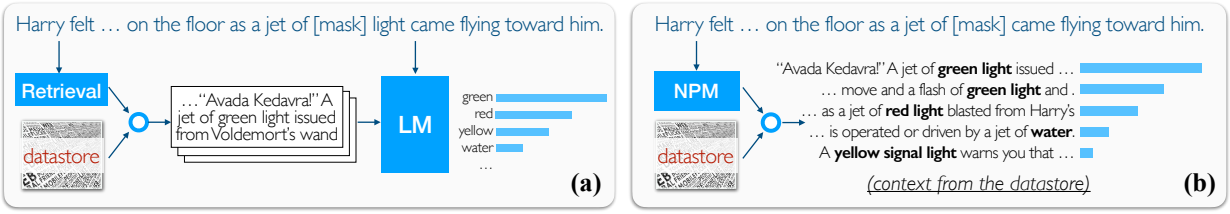

Figure 3.1: (a) A retrieval-augmented LM [Karpukhin et al., 2020; Min et al., 2019b, 2021a; Shi et al., 2024b,a] and **(b)** an LM with a nonparametric softmax [Min et al., 2023b].

makes it difficult to scale the amount of text retrieved from the datastore each time. Also, as we showed [Min et al., 2024], performance of these models do not scale as effectively with the datastore size as desired.

In Section 3.3, we will explore an alternative that incorporates the datastore differently—by training an LM with *a nonparametric softmax*. Instead of outputting a categorical distribution over words, this method assigns scores to every word or phrase in the datastore as a nonparametric distribution, e.g., assigning a high score to `'green light'` from a Harry Potter book (Figure 3.1 (b)). This approach both uses the datastore more effectively by incorporating a larger portion of the data at each step and handles rare concepts better, assigning high scores to unseen tokens or phrases if their surrounding context is relevant. We introduce NPM [Min et al., 2023b], one of the first such models. Training NPM at scale posed several technical challenges: for instance, scoring all phrases in large-scale data for every training iteration is very expensive. To address this, we designed novel techniques such as scalable batching and a training objective that effectively approximates a distribution over the full corpus. Our empirical results demonstrate that NPM outperforms alternatives on a range of tasks, is especially effective in handling rare concepts (such as rare entities and facts), and can grow and be updated by expanding and replacing the datastore. Notably, this approach scales and generalizes better than retrieval-augmented LMs [Min et al., 2024].

3.2 Retrieval Augmentation

The most popular form of the nonparametric approach is **retrieval augmentation**.[1] Arguably, the history of retrieval augmentation goes back to information retrieval and its modern definition varies across papers. In this thesis, we took a definition modified from Tian et al. [2019]: retrieval augmentation leverages results R from an information retrieval system to augment the input used in an LM. Their objectives are to maximize $P(r|q, R)$, where q is the input, r is a reasponse, and R is one or a few retreved documents (at most 100 in practice).

[1] It is also often called retrieval-augmented generation (RAG), following the term proposed by Lewis et al. [2020b]. In this thesis, we use the term *retrieval augmentation* or *retrieval-augmented LMs* to include models that are not generative.

In this section, we first discuss Dense Passage Retrieval (DPR), one of the first widely-used neural retrieval models that led to a wide adaptation of retrieval augmentation. We cover the neural retrieval model (Section 3.2.2) and how results from this model are incorporated into a language model (Section 3.2.3). Following this, we outline the experimental setup (Section 3.2.4) and present the results (Section 3.2.5).

DPR initiated discussion on parametric versus nonparametric language models, leading to a competition on open-domain question answering at NeurIPS 2021—called EfficientQA. We discuss the details of the competition, including the setup, participating models, results, and analyses (Section 3.2.6). EfficientQA demonstrated the impact of retrieval augmentation and DPR through the contributions of numerous third-party participants.

Since then, retrieval augmentation has seen rapid improvements. We discuss subsequent work that has driven these improvements in Section 3.2.7.

3.2.1 Background

The idea of incorporating the external datastore for end tasks has existed for a long time [Voorhees, 1999; Teh, 2006; Zhang et al., 2006; Ceri et al., 2013; Zhao and Cho, 2018]. This section provides the background focusing on using the raw text datastore for NLP tasks in the context of deep neural models.

A series of earlier work has used a deep neural network together with a datastore consisting of a large text documents, often designed or trained in a task-specific manner [Chen et al., 2017; Gu et al., 2018; Song et al., 2018; Tian et al., 2019]. Notably, open-domain question answering (QA)—answering a factoid question without a specific reference document given [Voorhees, 1999]—is one of the tasks that greatly benefits from retrieval augmentation. Researchers have made progress on different components of retrieval augmentation for open-domain question answering, including the augmentation method (Chen et al. [2017]; Yang et al. [2019a,b]; Nie et al. [2019]; Wolfson et al. [2020], and our own work Min et al. [2019a]) and reranking (Nogueira and Cho [2019], Humeau et al. [2020], and our own work Iyer et al. [2021]) and augmentation (Chen et al. [2017]; Yang et al. [2019a,b]; Nie et al. [2019]; Wolfson et al. [2020], and our own work Min et al. [2019a]). Most retrieval augmentation used sparse retrieval including TF-IDF and BM25 [Robertson et al., 2009], rather than neural-based retrieval.

Neural-based retrieval, called *dense retrieval* henceforth, has a long history since Latent Semantic Analysis [Deerwester et al., 1990]. The progress has been centered around how to train the encoder that maps a query and a document into a vector space, typically using labeled pairs of queries and documents [Yih et al., 2011; Huang et al., 2013; Gillick et al., 2019]. Such approaches complement the sparse vector methods as they can potentially give high similarity scores to semantically relevant text pairs, even without exact token matching. The dense representation alone, however, is typically inferior to the sparse one.

More recent work proposes a pre-training method for a joint training of a dense retrieval model and a language model [Lee et al., 2019; Guu et al., 2020]. While effective, training of such models is very expensive

and is difficult to train stably due to engineering complexities and hyperparameters coming from the necessity to asynchronously re-index the datastore during training.

3.2.2 Method: DPR

Given a collection of M text passages, the goal of our dense passage retrieval (DPR) is to index all the passages in a low-dimensional and continuous space, such that it can retrieve efficiently the top k passages relevant to the input question for the reader at run-time. Note that M can be very large (e.g., 21 million passages in our experiments using the English Wikipedia) and k is usually small, e.g., 100 in our experiments.

Overview

DPR uses a dense encoder $E_P(\cdot)$ which maps any text passage to a d-dimensional real-valued vectors and builds an index for all the M passages that we will use for retrieval. At run-time, DPR applies a different encoder $E_Q(\cdot)$ that maps the input question to a d-dimensional vector, and retrieves k passages of which vectors are the closest to the question vector. We define the similarity between the question and the passage using the dot product of their vectors: $\text{sim}(q, p) = E_Q(q)^\intercal E_P(p)$.

Encoders. Although in principle the question and passage encoders can be implemented by any neural networks, in this work we use two independent BERT [Devlin et al., 2019] networks (base, uncased) and take the representation at the `[CLS]` token as the output, so $d = 768$.

Inference. We apply the passage encoder E_P to all the passages and index them using FAISS [Johnson et al., 2019] offline. FAISS is an extremely efficient, open-source library for similarity search and clustering of dense vectors, which can easily be applied to billions of vectors. At run-time, given a question q, we derive its embedding $v_q = E_Q(q)$ and retrieve the top k passages with embeddings closest to v_q.

Training

Training the encoders so that the dot-product similarity becomes a good ranking function for retrieval is essentially a *metric learning* problem [Kulis, 2013]. The goal is to create a vector space such that relevant pairs of questions and passages will have smaller distance (i.e., higher similarity) than the irrelevant ones, by learning a better embedding function.

Let $\mathcal{D} = \{\langle q_i, p_i^+, p_{i,1}^-, \cdots, p_{i,n}^- \rangle\}_{i=1}^m$ be the training data that consists of m instances. Each instance contains one question q_i and one relevant (positive) passage p_i^+, along with n irrelevant (negative) passages $p_{i,j}^-$. We optimize the loss function as the negative log likelihood of the positive passage:

$$L(q_i, p_i^+, p_{i,1}^-, \cdots, p_{i,n}^-) = -\log \frac{e^{\text{sim}(q_i, p_i^+)}}{e^{\text{sim}(q_i, p_i^+)} + \sum_{j=1}^n e^{\text{sim}(q_i, p_{i,j}^-)}}.$$

Positive and negative passages. For retrieval problems, it is often the case that positive examples are available explicitly, while negative examples need to be selected from an extremely large pool. For instance, passages relevant to a question may be given in a QA dataset, or can be found using the answer. All other passages in the collection, while not specified explicitly, can be viewed as irrelevant by default. In practice, how to select negative examples is often overlooked but could be decisive for learning a high-quality encoder. We consider three different types of negatives: (1) Random: any random passage from the corpus; (2) BM25: top passages returned by BM25 which don't contain the answer but match most question tokens; (3) Gold: positive passages paired with other questions which appear in the training set. Our best model uses gold passages from the same mini-batch and one BM25 negative passage. In particular, re-using gold passages from the same batch as negatives can make the computation efficient while achieving great performance. We discuss this approach below.

In-batch negatives. Assume that we have B questions in a mini-batch and each one is associated with a relevant passage. Let $\mathbf{Q}$ and $\mathbf{P}$ be the $(B \times d)$ matrix of question and passage embeddings in a batch of size B. $\mathbf{S} = \mathbf{Q}\mathbf{P}^T$ is a $(B \times B)$ matrix of similarity scores, where each row of which corresponds to a question, paired with B passages. In this way, we reuse computation and effectively train on B^2 (q_i, p_j) question/passage pairs in each batch. Any (q_i, p_j) pair is a positive example when $i = j$, and negative otherwise. This creates B training instances in each batch, where there are $B - 1$ negative passages for each question.

The trick of in-batch negatives has been used in the full batch setting [Yih et al., 2011] and more recently for mini-batch [Henderson et al., 2017; Gillick et al., 2019]. It has been shown to be an effective strategy for learning a dual-encoder model that boosts the number of training examples.

3.2.3 Method: Augmentation

In this section, we describe how we incorporate retrieved passages into a language model.

We implement an end-to-end question answering system in which we can plug different retriever systems directly. Besides the retriever, our QA system consists of a neural *reader* that outputs the answer to the question. Given the top k retrieved passages (up to 100 in our experiments), the reader assigns a passage selection score to each passage. In addition, it extracts an answer span from each passage and assigns a span score. The best span from the passage with the highest passage selection score is chosen as the final answer. The passage selection model serves as a reranker through cross-attention between the question and the passage. Although cross-attention is not feasible for retrieving relevant passages in a large corpus due to its non-decomposable nature, it has more capacity than the dual-encoder model $\mathrm{sim}(q, p)$. Applying it to selecting the passage from a small number of retrieved candidates has been shown to work well [Wang et al., 2019c, 2018b; Lin et al., 2018].

Specifically, let $\mathbf{P}_i \in \mathbb{R}^{L \times h}$ $(1 \leq i \leq k)$ be a BERT (base, uncased in our experiments) representation for the i-th passage, where L is the maximum length of the passage and h the hidden dimension. The probabilities of a token being the starting/ending positions of an answer span and a passage being selected are defined as:

$$\begin{aligned} P_{\text{start},i}(s) &= \text{softmax}\big(\mathbf{P}_i \mathbf{w}_{\text{start}}\big)_s, \\ P_{\text{end},i}(t) &= \text{softmax}\big(\mathbf{P}_i \mathbf{w}_{\text{end}}\big)_t, \\ P_{\text{selected}}(i) &= \text{softmax}\big(\hat{\mathbf{P}}^\intercal \mathbf{w}_{\text{selected}}\big)_i, \end{aligned}$$

where $\hat{\mathbf{P}} = [\mathbf{P}_1^{[\text{CLS}]}, \ldots, \mathbf{P}_k^{[\text{CLS}]}] \in \mathbb{R}^{h \times k}$ and $\mathbf{w}_{\text{start}}, \mathbf{w}_{\text{end}}, \mathbf{w}_{\text{selected}} \in \mathbb{R}^h$ are learnable vectors. We compute a span score of the s-th to t-th words from the i-th passage as $P_{\text{start},i}(s) \times P_{\text{end},i}(t)$, and a passage selection score of the i-th passage as $P_{\text{selected}}(i)$.

During training, we sample one positive and $\tilde{m} - 1$ negative passages from the top 100 passages returned by the retrieval system (BM25 or DPR) for each question. $\tilde{m}$ is a hyper-parameter and we use $\tilde{m} = 24$ in all the experiments. The training objective is to maximize the marginal log-likelihood of all the correct answer spans in the positive passage (the answer string may appear multiple times in one passage), combined with the log-likelihood of the positive passage being selected. We use the batch size of 16 for large (NQ, TriviaQA, SQuAD) and 4 for small (TREC, WQ) datasets, and tune k on the development set. For experiments on small datasets under the *Multi* setting, in which using other datasets is allowed, we fine-tune the reader trained on Natural Questions to the target dataset. All experiments were done on eight 32GB GPUs.

3.2.4 Experimental Setup

In this section, we describe the data we used for experiments and the basic setup.

Wikipedia Data Pre-processing

Following [Lee et al., 2019], we use the English Wikipedia dump from Dec. 20, 2018 as the source documents for answering questions. We first apply the pre-processing code released in DrQA [Chen et al., 2017] to extract the clean, text-portion of articles from the Wikipedia dump. This step removes semi-structured data, such as tables, info-boxes, lists, as well as the disambiguation pages. We then split each article into multiple, disjoint text blocks of 100 words as *passages*, serving as our basic retrieval units, following [Wang et al., 2019c], which results in 21,015,324 passages in the end.[2] Each passage is also prepended with the title of the Wikipedia article where the passage is from, along with an `[SEP]` token.

[2]However, Wang et al. [2019c] also propose splitting documents into overlapping passages, which we do not find advantageous compared to the non-overlapping version.

Dataset	Train		Dev	Test
Natural Questions	79,168	58,880	8,757	3,610
TriviaQA	78,785	60,413	8,837	11,313
WebQuestions	3,417	2,474	361	2,032
CuratedTREC	1,353	1,125	133	694
SQuAD	78,713	70,096	8,886	10,570

Table 3.1: Statistics of five QA datasets used in the DPR paper. The two columns of *Train* denote the original training examples in the dataset and the actual questions used for training DPR after filtering. See text for more details.

Question Answering Datasets

We use the same five QA datasets and training/dev/testing splitting method as in previous work [Lee et al., 2019]. Below we briefly describe each dataset and refer readers to their paper for the details of data preparation.

Natural Questions (NQ) [Kwiatkowski et al., 2019] was designed for end-to-end question answering. The questions were mined from real Google search queries and the answers were spans in Wikipedia articles identified by annotators.

TriviaQA [Joshi et al., 2017] contains a set of trivia questions with answers that were originally scraped from the Web.

WebQuestions (WQ) [Berant et al., 2013] consists of questions selected using Google Suggest API, where the answers are entities in Freebase.

CuratedTREC (TREC) [Baudiš and Šedivỳ, 2015] sources questions from TREC QA tracks as well as various Web sources and is intended for open-domain QA from unstructured corpora.

SQuAD v1.1 [Rajpurkar et al., 2016a] is a popular benchmark dataset for reading comprehension. Annotators were presented with a Wikipedia paragraph, and asked to write questions that could be answered from the given text. Although SQuAD has been used previously for open-domain QA research, it is not ideal because many questions lack context in absence of the provided paragraph. We still include it in our experiments for providing a fair comparison to previous work.

Selection of positive passages. Because only pairs of questions and answers are provided in TREC, WebQuestions and TriviaQA[3], we use the highest-ranked passage from BM25 that contains the answer as the positive passage. If none of the top 100 retrieved passages has the answer, the question will be discarded. For SQuAD and Natural Questions, since the original passages have been split and processed differently than our pool of candidate passages, we match and replace each gold passage with the corresponding passage in the candidate pool. We discard the questions when the matching is failed due to different Wikipedia versions or

[3]We use the unfiltered TriviaQA version and discard the noisy evidence documents mined from Bing.

Training	Model	NQ	TriviaQA	WQ	TREC	SQuAD
Single	BM25+BERT [Lee et al., 2019]	26.5	47.1	17.7	21.3	33.2
Single	ORQA [Lee et al., 2019]	33.3	45.0	36.4	30.1	20.2
Single	HardEM [Min et al., 2019a]	28.1	50.9	-	-	-
Single	GraphRetriever [Min et al., 2019b]	34.5	56.0	36.4	-	-
Single	PathRetriever [Asai et al., 2020]	32.6	-	-	-	**56.5**
Single	$\text{REALM}_{\text{Wiki}}$ [Guu et al., 2020]	39.2	-	40.2	46.8	-
Single	$\text{REALM}_{\text{News}}$ [Guu et al., 2020]	40.4	-	40.7	42.9	-
Single	BM25	32.6	52.4	29.9	24.9	38.1
	DPR	**41.5**	56.8	34.6	25.9	29.8
	BM25+DPR	39.0	57.0	35.2	28.0	36.7
Multi	DPR	**41.5**	56.8	**42.4**	49.4	24.1
	BM25+DPR	38.8	**57.9**	41.1	**50.6**	35.8

Table 3.2: QA (Exact Match) Accuracy. The first block of results are copied from their cited papers. $\text{REALM}_{\text{Wiki}}$ and $\text{REALM}_{\text{News}}$ are the same model but pretrained on Wikipedia and CC-News, respectively. *Single* and *Multi* denote that DPR is trained using individual or combined training datasets (all except SQuAD). For WQ and TREC in the *Multi* setting, we fine-tune the reader trained on NQ.

pre-processing. Table 3.1 shows the number of questions in training/dev/test sets for all the datasets and the actual questions used for training the retriever.

3.2.5 Results

Table 3.2 summarizes our final end-to-end QA results, measured by *exact match* with the reference answer after minor normalization as in [Chen et al., 2017; Lee et al., 2019]. From the table, we can see that higher retriever accuracy typically leads to better final QA results: in all cases except SQuAD, answers extracted from the passages retrieved by DPR are more likely to be correct, compared to those from BM25. For large datasets like NQ and TriviaQA, models trained using multiple datasets (Multi) perform comparably to those trained using the individual training set (Single). Conversely, on smaller datasets like WQ and TREC, the multi-dataset setting has a clear advantage. Overall, our DPR-based models outperform the previous state-of-the-art results on four out of the five datasets, with 1% to 12% absolute differences in exact match accuracy. It is interesting to contrast our results to those of ORQA [Lee et al., 2019] and also the concurrently developed approach, REALM [Guu et al., 2020]. While both methods include additional pre-training tasks and employ an expensive end-to-end training regime, DPR manages to outperform them on both NQ and TriviaQA, simply by focusing on learning a strong passage retrieval model using pairs of questions and answers. The additional pre-training tasks are likely more useful only when the target training sets are small. Although the results of DPR on WQ and TREC in the single-dataset setting are less competitive, adding more question–answer pairs helps boost the performance, achieving the new state of the art.

One thing worth noticing is that our reader does consider more passages compared to ORQA, although it is not completely clear how much more time it takes for inference. While DPR processes up to 100 passages for each question, the reader is able to fit all of them into one batch on a single 32GB GPU, thus the latency remains almost identical to the single passage case (around 20ms). The exact impact on throughput is harder to measure: ORQA uses 2-3x longer passages compared to DPR (288 word pieces compared to our 100 tokens) and the computational complexity is super-linear in passage length. We also note that we found $k = 50$ to be optimal for NQ, and $k = 10$ leads to only marginal loss in exact match accuracy (40.8 vs. 41.5 EM on NQ), which should be roughly comparable to ORQA's 5-passage setup.

3.2.6 EfficientQA Competition

In 2020, the year DPR has been introduced, there has been significant progress on large language models, raising the possibility of representing world knowledge solely in the parameters of a neural model instead of using a datastore [Roberts et al., 2020]. In this context, we organized the EfficientQA competition, held at NeurIPS 2020, which required contestants to build self-contained systems that contain all of the knowledge required to answer open-domain questions. Participants can explore either parameteric or nonparametric language models to optimize for model performance. However, the competition encouraged systems that store and access this knowledge using the smallest number of bytes, including code, corpora, and model parameters. Specifically, EfficientQA had four tracks: 1) best accuracy overall (unconstrained); 2) best accuracy, system size under 6GiB; 3) best accuracy, system size under 500MiB; 4) smallest system to get 25% accuracy. These memory budgets were designed to encourage contestants to explore the trade-off between storing and accessing large datastores or the parameters of neural models.

Key takeaways

The top submissions in each of EfficientQA's four tracks significantly outperformed the provided baselines. All top submissions use a retrieval corpus and a neural network answering module. However, the nature of the retrieval corpus and answering module differs drastically across the tracks (Table 3.3).

Unrestricted and 6GiB tracks. The top submissions to the unrestricted track and the 6GiB track outperformed the state-of-the-art baselines from April 2020 by nearly 20%. They achieved this improvement by combining the state-of-the-art retrieval systems [Karpukhin et al., 2020; Mao et al., 2020a] with answer generation [Izacard and Grave, 2020]; leveraging the state-of-the-art in text generation [Raffel et al., 2020] and text encoding [Clark et al., 2020b]; modeling not only text but also tables and lists from Wikipedia; and combining the extractive and generative answer prediction. The top submissions to the 6GiB track additionally massively reduced the size of their indexed corpus and made use of the state-of-the-art in compression, with minimal impact on accuracy.

Track	Model	Affiliation	Nonparam?	Key features
Unrestricted	REALM	Organizers	✓	ext
	DPR	Organizers	✓	ext
	MS UnitedQA	Microsoft & Dynamics 365	✓	ext+gen
	FB Hybrid	Facebook AI	✓	gen, lists/tables
6GiB	DPR-subset	Organizers	✓	ext, pruned
	T5-XL+SSM	Organizers	✗	gen
	FB system	FAIR-Paris&London	✓	gen, lists, pruned, lrzip compression
	Ousia-Tohoku Soseki	Studio Ousia, Tohoku U & RIKEN	✓	ext, pruned, ZPAQ compression
	BUT R2-D2	Brno U of Technology	✓	ext+gen, pruned
500MiB	T5-Small+SSM	Organizers	✗	gen
	UCLNLP-FB system	UCL & FAIR	✓,	data augmentation
	NAVER RDR	NAVER Clova	✓	ext, pruned, single Transformer
25% smallest	T5-XL+SSM	Organizers	✗	gen
	UCLNLP-FB system (29M)	UCL & FAIR	✓	data augmentation

Table 3.3: A list of the baselines and systems from participants, along with the team affiliation and key distinction between systems. *Nonparam* means whether the model is nonparametric (✓) or parametric (✗). Key features indicate some of the key features, such as whether the datastore is pruned (*pruned*), whether the answer is extracted (*ext*) or generated (*gen*), and others. We refer to the original publication [Min et al., 2021a] for more detailed descriptions of each system.

500MiB and smallest tracks. To get under 500MB, the systems made more drastic changes. The submission from NAVER Clova drastically reduced the size of their indexed corpus and reused a single Transformer model for the retriever and the reader, winning the 500MiB track according to the human evaluation. The even smaller UCLNLP-FB system took a novel approach in generating a large corpus of question-answer pairs, indexing it, and retrieving the most similar question to the input question. This approach, with two systems with different sizes in question-answer corpus, won both the 500MiB track and the smallest 25% track according to the automatic evaluation.

Automatic vs. human evaluation. The human evaluation supports the observation that automatic metrics often incorrectly penalize correct predictions in the open-domain setting [Voorhees and Tice, 2000; Roberts et al., 2020]. We also investigate the effect of question ambiguity on evaluation—the questions from NQ are often ambiguous without the associated evidence document [Min et al., 2020]. We define an annotation scheme that supports multiple estimations of accuracy, corresponding to different definitions of correctness for answers to ambiguous questions. Almost all systems' accuracy increased by 20%-25% under the strictest definition of correctness. The increase doubled when we relaxed the definition of correctness to permit any semantically valid interpretation of the question.

Competition Overview

Data. The competition uses English questions and answers from the Natural Questions dataset [Kwiatkowski et al., 2019, NQ], the same dataset as the one used in Section 3.2.4.

NQ was collected over the course of 2017 and 2018. For EfficientQA, we introduce a new test and development set constructed in the same way as the original NQ, but labeled slightly after (early 2019 rather than through 2018). Our test set was kept hidden from contestants, and submissions were made by uploading solutions to an automatic leaderboard.

Evaluation metrics: Automatic. The accuracy of each systems' predicted answers is judged against reference annotations, annotated by five human workers. We follow the literature in using exact match between predicted and reference answers after minor normalization [Lee et al., 2019].

Evaluation metrics: Human. Due to the ambiguities inherent in language and question-answering in general, five reference answers are often not exhaustive, and systems predict correct answers that are judged incorrect according to the automatic metrics. To rectify this, we sent predictions from each of the systems for further human rating by three raters, to get a better estimation of accuracy. More specifically, each system prediction was sent for rating by three separate annotators: 1) the annotator first works on understanding the meaning and intent of the question (with a web search if necessary). 2) The annotator then determines whether the question is ambiguous, i.e., whether the question can lead to multiple different answers depending on factors such as: when the query was asked; where the query was asked; some unspoken intent of the questioner; or the opinion of the person giving the answer. 3) Finally, the annotator determines whether each answer is "definitely correct" (correct given a usual interpretation of the question), "possibly correct" (could be correct, given some interpretation of the question),[4] or "definitely incorrect". The final rating is an aggregation of ratings from three annotators: if at least 2/3 raters determined it to be "definitely correct", the label is "definitely correct". If at least 2/3 raters determined it to be either "definitely correct" or "possibly correct", the label is "possibly correct". The pairwise agreements of the human ratings are 69.2% (Cohen's $\kappa = 53.8$) for 3-way ratings, 85.7% (Cohen's $\kappa = 71.4$) for whether the prediction is definitely correct or not, and 76.7% (Cohen's $\kappa = 53.4$) for whether the prediction is possibly correct or not.

Results

All of the five systems in the unrestricted track and the 6GiB track significantly outperform the state-of-the-art (Table 3.4) at the beginning of the competition—DPR (36.6%) and REALM (35.9%). Systems in the 6GiB track approach the unrestricted track's accuracy; for instance, the accuracy FB system is comparable to

[4]Since the original NQ data was collected more than a year before the start of the EfficientQA competition, the denotation of some questions may have changed over time (e.g. "who won the last season of bake-off"). Rather than determine a single correct point in time for these questions, we asked our annotators to assume that the query could have been asked at any time since the web has existed and choose the "possibly correct" label for answers that may or may not have been correct when the question was asked.

Track	Model	Automatic eval	Human eval	
			Definitely	Possibly
Unrestricted	MS UnitedQA	54.00	65.80 (+21.9%)	78.12 (+44.7%)
	FB Hybrid	53.89	67.38 (+25.0%)	79.88 (+48.2%)
6GiB	FB system	53.33	65.18 (+22.2%)	76.09 (+42.7%)
	Ousia-Tohoku Soseki	50.17	62.01 (+23.6%)	73.83 (+47.2%)
	BUT R2-D2	47.28	58.96 (+24.7%)	70.33 (+49.2%)
500MiB	UCLNLP-FB system	33.44	39.40 (+17.8%)	47.37 (+41.7%)
	NAVER RDR	32.06	42.23 (+31.7%)	54.95 (+71.4%)
25% smallest	UCLNLP-FB system (29M)	26.78	32.45 (+21.2%)	41.21 (+53.9%)

Table 3.4: Summary of the result. For human evaluation result, relative improvements over the automatic evaluation are indicated in parenthesis. Following our analysis of the annotations, we use 'Definitely correct' human ratings as a primary metric.

the accuracy of the top systems in the unrestricted track. The improvements in the 500MiB track are also impressive; both the top two systems significantly beat T5-small (17.6%).

Discrepancy between automatic eval and human eval. Human raters find 13% and 17% of the predictions that do not match the reference answers to be definitely correct or possibly correct, respectively, overall increasing the accuracy of the systems. Most systems showed 17–25% and 41–54% improvement in accuracy when using definitely correct and possibly correct human evaluation respectively, compared to automatic evaluation metric which only consider exact string match to existing reference answers. An exception is NAVER RDR, which achieves significantly larger improvements (32% and 71%, respectively). We also found that when the gap in automatic measure between systems is marginal (around or smaller than 1%), human evaluation may change the rankings between the models.

Agreement between system predictions. Figure 3.2 (left) shows the agreement between system predictions, based on exact match in automatic evaluation. The largest agreement is made between FB Hybrid and FB system, likely because they are both based on DPR and Fusion-in-Decoder. Agreements between systems in the unrestricted and the 6GiB tracks are generally higher, likely because they are all based on retrieval-reader framework and pruning Wikipedia does not hurt too much. Two systems in the 500MiB track have smaller agreement with the other systems, and agree with each other even less.

Ensemble oracle accuracy of the systems. Figure 3.2 (right) reports the ensemble oracle accuracy for each system pair, which considers a prediction to be correct if either system prediction is correct. The FB Hybrid & Ousia-Tohoku Soseki pair achieves the highest ensemble oracle accuracy, indicating that their system predictions are substantially different from each other compared to other pairs with top performing systems.

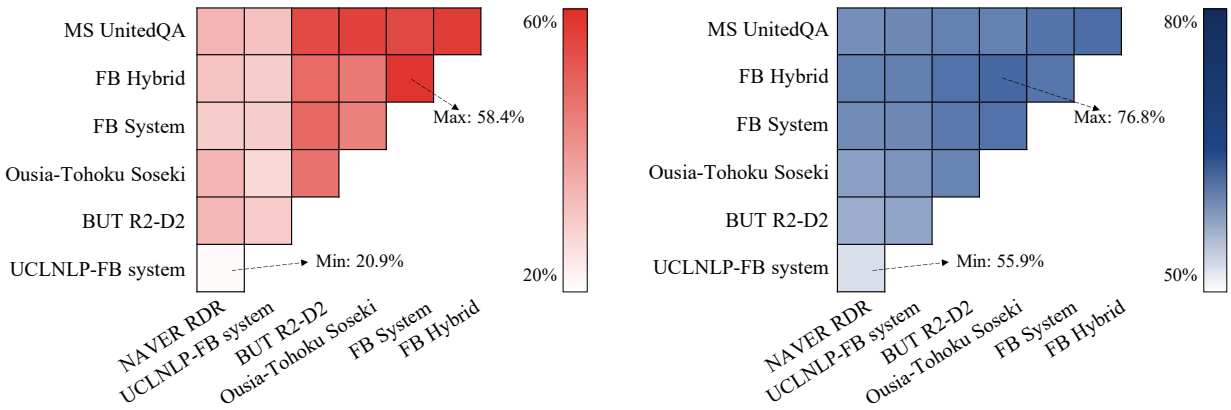

Figure 3.2: *(Left)* Agreement between system predictions. *(Right)* Ensemble oracle accuracy, which considers a prediction correct if at least one of the system predictions is correct (based on "definitely correct" human evaluation).

Qualitative analysis. We present an analysis of the 50 sample questions where at least one prediction does not match with the gold answer, but is judged as correct by human raters, with a "definitely correct" label or a "possibly correct" label, respectively. We refer to Section 5.2 of the original publication [Min et al., 2021a], and summarize the main takeaways here.

First, the automatic evaluation confirms the observation of [Voorhees and Tice, 2000] that it is insufficient in capturing semantically equivalent answers, which are responsible for 60% of the definitely correct predictions. Second, ambiguity arises frequently in the questions in different levels, allowing multiple, semantically different answers to be valid. This is a consistent with [Min et al., 2020], which reported that around half of the questions in NQ contain ambiguity, due to ambiguous references of entities, events or properties, or time-dependency of the answer. Based on our human evaluation, annotations on ambiguity have low agreement rate (61.3%, Cohen's $\kappa = 22.6$), and predictions with the same level of plausibility are often marked as "definitely correct" or "possibly correct" by different human raters. We note the notions of "correctness" and "plausibility" are not binary, and are instead often dependent on pragmatic interpretation of the questioner's intent. For example, the question "who has the most superbowl rings" could be read as "which person (including coaches) has the most superbowl rings", "which player has the most superbowl rings", "which team has the most superbowl rings". All three annotators identified this question as being ambiguous but they disagreed about the validity of the different readings. The three raters were split three ways when rating the correct answer ("Pittsburgh Steelers") for the last interpretation. Meanwhile there were no "incorrect" ratings, and 2/3 "definitely correct" ratings given to the correct answer ("Tom Brady") for the second interpretation, despite the fact that two coaches have more superbowl rings. Clearly, the annotators are applying some personal interpretation of the questioner's intent and answer plausibility.

Model	All		Unambiguous Qs	
	Definitely	Possibly	Definitely	Possibly
MS UnitedQA	65.80	78.12	78.24	81.18
FB Hybrid	67.38	79.88	82.65	85.59
FB system	65.18	76.09	79.12	81.47
Ousia-Tohoku Soseki	62.01	73.83	72.94	75.00
BUT R2-D2	58.96	70.55	69.71	72.06
UCLNLP-FB system	39.40	47.37	42.06	43.24
NAVER RDR	42.23	54.95	49.71	53.24
UCLNLP-FB system (29M)	32.45	41.21	28.53	30.29

Table 3.5: Human evaluation on the original set and a subset of unambiguous questions.

While we believe that many real world questions do require some non-literal assumptions about the questioner's intent, and we believe that the natural language processing community should not shy away from that task, we also acknowledge that there is work to be done in creating better, non-binary, definitions of correctness.

Performance on unambiguous questions. To better understand the effect of ambiguity on the ranking of different solutions, we also evaluate system performance on a subset of the questions that are unambiguous. We define unambiguous questions to be those that (1) have at least three out of five reference answers contain valid short answers[5], and (2) are not labeled as ambiguous by any of three human raters, resulting in 51.5% of the original set. Table 3.5 shows human evaluation on the original set and this subset of unambiguous questions. Most systems, except UCLNLP-FB system, achieve higher accuracy on unambiguous questions, with the first three systems achieving over or near 80%. Unsurprisingly, the gap between "definitely correct" accuracy and "possibly correct" accuracy is marginal on unambiguous questions.

3.2.7 Subsequent Work

Dense retrieval. Since the introduction of DPR, researchers have significantly improved dense retrieval models through multiple axes, including the model architecture, the training objective, and the method to obtain better hard negatives [Xiong et al., 2021a,b; Ding et al., 2021; Gao and Callan, 2021; Zhan et al., 2021; Gao and Callan, 2022]. Researchers also scaled up dense retrieval, either in terms of training data (made possible with the introduction of a self-supervised contrastive learning objective [Izacard et al., 2022a; Ram et al., 2022]) or encoder size [Ni et al., 2022; Neelakantan et al., 2022]. Researchers also combined dense retrieval with sparse retrieval [Mao et al., 2020b], added late-interaction operations on top of the dual encoder [Khattab and Zaharia, 2020; Santhanam et al., 2021], and made dense retrieval cross-lingual [Asai et al., 2021b]. There has also been work that tunes retrieval using signals from an LM after augmentation,

[5]This follows the original NQ approach of using annotator agreement to set a threshold for high quality answers.

including our own work [Shi et al., 2024b], which showed its effectiveness over state-of-the-art LLMs like GPT-3, Instruct GPT-3 and CoDeX.

Augmentation. Subsequent work [Izacard and Grave, 2020; Lewis et al., 2020b] has shown that DPR can be combined with generation models such as BART [Lewis et al., 2020a] and T5 [Raffel et al., 2020]. In particular, Lewis et al. [2020b] showed the effectiveness of this approach for a range of knowledge-intensive tasks beyond open-domain QA, including fact verification, dialogue, and slot filling. Izacard et al. [2022b] proposed a pre-training objective that jointly trains a retrieval model and an LM, further improving performance.

While this line of work assumes the model can be fine-tuned on a labeled dataset, more recent work explores the use of retrieval-augmented LMs without fine-tuning. Ram et al. [2023] and Shi et al. [2024b], concurrently, showed that with prompting of a frozen, large LM with block-box access, retrieval augmentation can yield improvements over a range of tasks.

Other papers report that out-of-box language models do not necessarily know how to use retrieval results; they often ignore retrieval results when they need to, or become easily distracted by incorrect retrieval results. Recent work proposed various ways to manage these issues, e.g., through post hoc verification [Asai et al., 2024] or by filtering retrieved results before the augmentation stage [Wang et al., 2023b].

Pre-training. More recently, we proposed a new pre-training objective, called *in-context pre-training* [Shi et al., 2024a], that closely resembles retrieval augmentation. The key idea is to train an LM on a sequence of relevant documents obtained from an off-the-shelf retrieval model, thereby explicitly encouraging the LM to learn to condition on a set of retrieved documents. Unlike prior work in joint training [Lee et al., 2019; Guu et al., 2020; Izacard et al., 2022b], in-context pre-training is very straightforward since it does not require asynchronously re-indexing the datastore, and can be achieved simply by changing the document ordering and directly applying existing pre-training pipelines. We trained LMs with 1B and 7B parameters from scratch and showed significant improvements in open-domain question answering with retrieval augmentation, while also achieving benefits such as in-context learning and long-context language modeling.

3.3 Nonparametric Prediction

So far, we discussed retrieval augmentation (in this section, "retrieve-then-generate") as one category of a nonparametric LM that operates as a function of the data given at test time. As we discussed in Section 3.2, these models retrieve text from the datastore which are subsequently fed into the parametric LM. However, their final predictions are still made by a parametric model. In particular, they still include a softmax over a finite vocabulary, which limits expressivity [Yang et al., 2018a] and can make them reluctant to predict rare or unseen tokens (e.g., *Thessaloniki*).

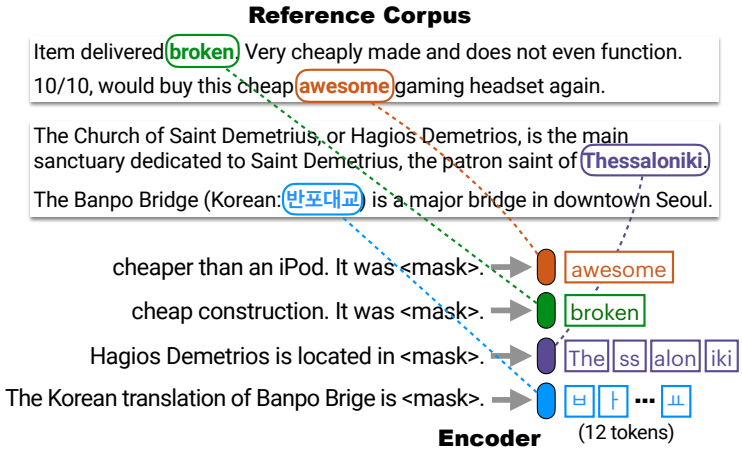

Figure 3.3: An illustration of NPM. The *encoder* maps a masked sentence into a dense vector, and retrieves the nearest phrase from a *reference corpus*. NPM can fill in the `[MASK]` with multiple tokens, e.g., *Thessaloniki* (4 BPE tokens) and unseen words, e.g., 반포대교 (12 BPE tokens).

In this section, we introduce NPM, the first **N**on**P**arametric **M**asked Language Model[6] that predicts tokens solely based on a nonparametric distribution over *phrases* in a text corpus (Figure 3.3). NPM consists of an *encoder* that maps the text into a fixed-sized vector, and a *reference corpus* from which NPM retrieves a phrase and fills in the `[MASK]`. It, crucially, does not have a softmax over a fixed vocabulary, but instead has a *fully nonparametric* distribution over phrases. This is in contrast to retrieval augmentation that incorporates nonparametric components in a parametric model [Borgeaud et al., 2022; Izacard et al., 2022b].

Training such a nonparametric model introduces two key challenges: (1) full corpus retrieval during training is expensive, and (2) learning to predict an arbitrary length phrase without a decoder is non-trivial. We address the first challenge by using in-batch approximations to full corpus retrieval [Wu et al., 2020; Zhong et al., 2022b], and the second by extending span masking [Joshi et al., 2020] and a phrase-level contrastive objective [Oord et al., 2018; Lee et al., 2021].

We perform zero-shot evaluation on 16 tasks including classification, fact probing and question answering. They include temporal shift and word-level translation tasks that highlight the need to predict new facts or rare phrases. Results show that NPM is significantly more parameter-efficient, outperforming up to 500x larger parametric models and up to 37x larger retrieve-and-generate models, particularly in predicting rare words.

[6] In nonparametric models, the data distribution is not defined by a fixed set of parameters, but is rather a function of the available data [Siegel, 1957; Hollander et al., 2013]. Freeman et al. [2002] noted that the term nonparametric does not imply that they are parameterless, but rather the number and nature of the *effective* parameters are flexible and depend on the data.

3.3.1 Background

The idea of using a nonparametric softmax is not new, and has been studied by a series of prior work [Khandelwal et al., 2020; Yogatama et al., 2021; Zhong et al., 2022b; Lan et al., 2023]—so-called kNN-LM models. Our work is heavily inspired by the kNN-LM approach, and can be seen as an extreme version of it with no interpolation. However, NPM is the first that models a *fully* nonparametric distribution by entirely removing the softmax over a finite vocabulary. This offers a range of new functionalities, such as modeling a distribution over phrases, or predicting rare or unseen words.

Prior work has explored nonparametric inference without training [Khandelwal et al., 2020; He et al., 2021; Xu et al., 2022], or trained the nonparametric model on the labeled data for a specific downstream task [Seo et al., 2018, 2019; Lee et al., 2021]. In contrast, NPM is a fully nonparametric language model without the labeled data and performs a range of tasks zero-shot.

Bottleneck in softmax. Most if not all language models use a softmax function that gives a categorical probability distribution over a finite vocabulary. Yang et al. [2018a] showed that this softmax is a low-rank approximation of a high-rank output space, making the model less expressive. Pappas et al. [2020b] discussed that a fixed output vocabulary makes language models resistant to adaptation to new domains and tasks. We share the motivation with such prior work and propose to use a nonparametric output space to address these issues. Moreover, although not explicitly explored in this paper, our work that completely removes the softmax over the vocabulary can make training more efficient, especially when the vocabulary is large (e.g., multilingual models [Conneau et al., 2020]).

3.3.2 NPM: Inference

NPM consists of an encoder and a reference corpus, and models a nonparametric distribution over a reference corpus (Figure 3.3). The key idea is to map all the phrases in the corpus into a dense vector space using the encoder and, when given a query with a `[MASK]` at inference, use the encoder to locate the nearest phrase from the corpus and fill in the `[MASK]`.

Encoder-only models are competitive representation models [Patel et al., 2022], outperforming the other two classes of models in classification tasks (Section 3.3.4). However, existing encoder-only models are unable to make a prediction whose number of tokens is unknown, making their use cases limited without fine-tuning. NPM addresses this issue, since it can fill in the `[MASK]` with an arbitrary number of tokens by retrieving a *phrase*.

This section describes inference of NPM assuming a learned encoder. Later, we describe how we train the encoder to map the text into a good vector space (Section 3.3.3).

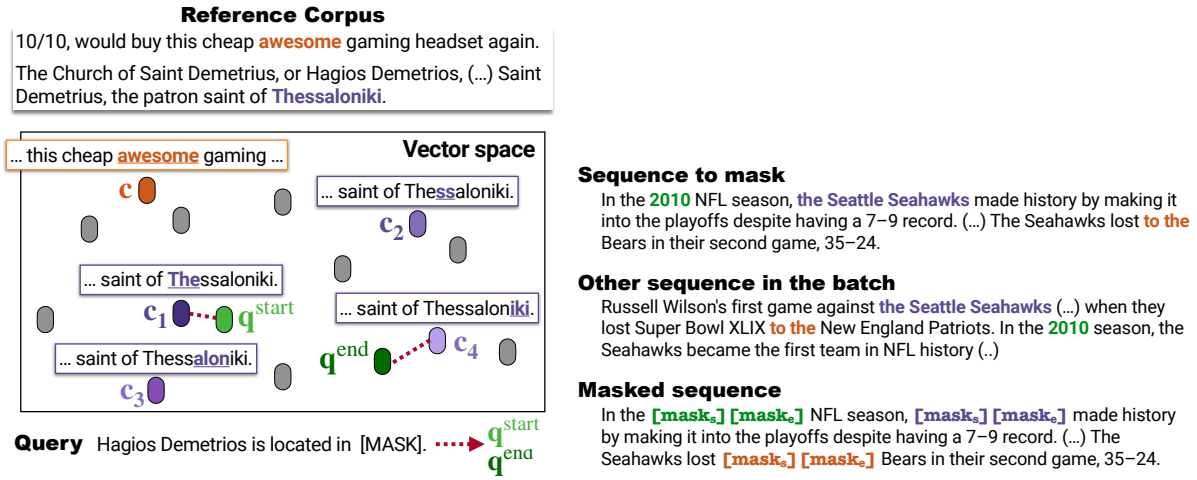

Figure 3.4: (Left) Inference of NPM (Section 3.3.2). Each token in the reference corpus $\mathcal{C}$ is mapped into a dense vector space. At test time, a query is represented as two vectors, $\mathbf{q}^{\text{start}}$ and $\mathbf{q}^{\text{end}}$, each in the same vector space. We use a nearest neighbor search to retrieve the start and the end of the phrase using $\mathbf{q}^{\text{start}}$ and $\mathbf{q}^{\text{end}}$, respectively. (Right) Our span masking (Section 3.3.3). For simplicity, this figure assumes two sequences in the batch. Spans to mask out are chosen based on whether there is any co-occurring spans in other sequences in the batch. Then, each span is replaced with `[MASK_s][MASK_e]`.

Overview. The encoder maps every distinct *phrase* in a reference corpus $\mathcal{C}$ into a dense vector space. At test time, the encoder maps the masked query into the same vector space and retrieves phrases from $\mathcal{C}$ to fill in the `[MASK]`. Here, $\mathcal{C}$ does not have to be the same as the training corpus, and can be replaced or scaled at test time without re-training the encoder.

In practice, there is a significant number of phrases in the corpus, and it is expensive to index all of them. We therefore use a technique from Lee et al. [2021] that represents a phrase with *token* representations of the start and the end of the phrase. In this approach, we index representations of each distinct token in $\mathcal{C}$, and then at test time, use a k nearest neighbor search for the start and the end of the phrase, separately. Consider Figure 3.4 (left) as an example. We represent a query with two vectors, $\mathbf{q}^{\text{start}}$ and $\mathbf{q}^{\text{end}}$. We then use each to retrieve the start and the end of the plausible phrases—in this case, $\mathbf{c}_1$ and $\mathbf{c}_4$, which are the start and the end of ***Thessaloniki***, respectively.

Method. Formally, let $\mathcal{C} = \{c_1, \cdots, c_N\}$ be a reference corpus with N tokens. We first map each token c_i into a contextualized, h-dimensional vector $\mathbf{c}_i \in \mathbb{R}^h$ by feeding the text into the encoder and take the vector that corresponds to each token: $\mathbf{c}_1 ... \mathbf{c}_N = \text{Encoder}(c_1 ... c_N)$.

At inference time, NPM is given a query whose t-th token is masked: $q_1 ... q_{t-1}$, `[MASK]`, $q_{t+1} ... q_L$. We replace `[MASK]` with two special tokens `[MASK_s][MASK_e]` and feed it into the encoder to obtain a list of

h-dimensional vectors:

$$\mathbf{q}_1 ... \mathbf{q}_{L+1} = \text{Encoder}(q_1 ... q_{t-1}, \texttt{[MASK}_\text{s}\texttt{]}, \texttt{[MASK}_\text{e}\texttt{]}, q_{t+1} ... q_L).$$

We then take the vector corresponding to $\texttt{[MASK}_\text{s}\texttt{]}$ and $\texttt{[MASK}_\text{e}\texttt{]}$ as $\mathbf{q}^{\text{start}}$ and $\mathbf{q}^{\text{end}}$, respectively.[7]

$$\mathbf{q}^{\text{start}} = \mathbf{q}_t, \mathbf{q}^{\text{end}} = \mathbf{q}_{t+1}.$$

We then make a prediction via:

$$\underset{v^* \in \mathcal{V}^*}{\text{argmax}} \sum_{i \leq j} \mathbb{I}[v^* = c_{i:j}] \Big(\exp(\text{sim}(\mathbf{q}^{\text{start}}, \mathbf{c}_i)) + \exp(\text{sim}(\mathbf{q}^{\text{end}}, \mathbf{c}_j)) \Big),$$

where $\mathcal{V}^*$ is a set of possible n-grams defined by the vocabulary $\mathcal{V}$ and sim is a pre-defined similarity function that maps a pair of vectors into a scalar value. In practice, iterating over N tokens is infeasible. We thus use an approximation using a fast nearest neighbor search for the start and the end separately. Details are provided in Appendix B.1.1.

The choice of similarity function can be flexible. We follow Zhong et al. [2022b] in using a scaled inner product $\text{sim}(\mathbf{h}_1, \mathbf{h}_2) = \frac{\mathbf{h}_1 \cdot \mathbf{h}_2}{\sqrt{h}}$, where h is a dimension of the token vectors.

3.3.3 NPM: Training

Masking

We extend span masking [Joshi et al., 2020], which masks spans (consecutive tokens) whose length is sampled from a geometric distribution. Our span masking differs from Joshi et al. [2020] in two ways. First, we mask spans if they co-occur in the other sequences in the batch to guarantee in-batch positives during training (Section 3.3.3). For instance, masked spans in Figure 3.4 (right) are '*2010*', '*the Seattle Seahawks*' and '*to the*' all of which are found in the other sequences. Second, instead of replacing each token in the span with a `[MASK]`, we replace the whole span with two special tokens $\texttt{[MASK}_\text{s}\texttt{][MASK}_\text{e}\texttt{]}$. For instance, each of '*2010*', '*the Seattle Seahawks*' and '*to the*' is replaced with $\texttt{[MASK}_\text{s}\texttt{][MASK}_\text{e}\texttt{]}$. This is to obtain the start and the end vectors for each span as we do at inference.

Training Objective

We illustrate an example in Figure 3.5. The masked span is '*the Seattle Seahawks*', thus the model should retrieve a phrase '*the Seattle Seahawks*' from other sequences in the reference corpus when it is given a query like this at test time. Specifically, we should encourage the $\texttt{[MASK}_\text{s}\texttt{]}$ vector to be closer to

[7] This allows obtaining two vectors without encoding the query twice, e.g., unlike Lee et al. [2021]

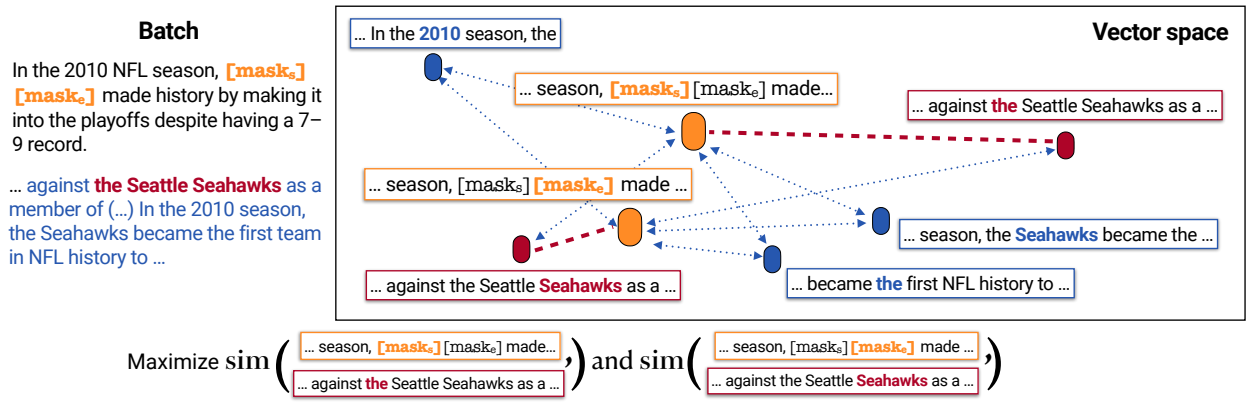

Figure 3.5: Training of NPM (Section 3.3.3). **[MASK$_s$] [MASK$_e$]** indicates the masked *span* whose original phrase is *the Seattle Seahawks*. We maximize the similarity scores between ...**[MASK$_s$]** [MASK$_e$]... and ...**the** Seattle Seahawks... , and between ... [MASK$_s$] **[MASK$_e$]** ... and ...the Seattle **Seahawks**... .

...**the** Seattle Seahawks... and the [MASK$_e$] vector to be closer to ...the Seattle **Seahawks**... , while being distant from other tokens. We train the model to do so by approximating the full corpus as the other sequences in the batch. Concretely, we train the model to retrieve the start and the end of the span '*the Seattle Seahawks*' from other sequences in the same batch. Note that our masking strategy ensures that every masked span has a co-occurring span in the batch.

Formally, consider the i-th sequence in the batch that consists of L tokens, $x^i = x^i_1 ... x^i_L$. We denote $\hat{x}^i = \hat{x}^i_1 ... \hat{x}^i_L$ as a consequence of span masking over x^i. Both x^i and $\hat{x}^i$ are fed into the encoder, and each token is mapped into an h-dimensional vector:[8]

$$\begin{aligned} \mathbf{x}^i_1 \cdots \mathbf{x}^i_L &= \text{Encoder}(x^i_1 \cdots x^i_L), \\ \hat{\mathbf{x}}^i_1 \cdots \hat{\mathbf{x}}^i_L &= \text{Encoder}(\hat{x}^i_1 \cdots \hat{x}^i_L). \end{aligned}$$

Now, consider a masked span in x_i, represented with [MASK$_s$] [MASK$_e$], denoted as $\hat{x}^i_t, \hat{x}^i_{t+1}$. We then denote g^i_t as the original n-gram that were replaced by $\hat{x}^i_t, \hat{x}^i_{t+1}$.

The training objective for this masked span is defined as

$$-\left(\log \frac{\sum_{\mathbf{y} \in \mathcal{Y}^+_s(g^i_t)} \exp(\text{sim}(\hat{\mathbf{x}}^i_t, \mathbf{y}))}{\sum_{\mathbf{y} \in \mathcal{Y}^+_s(g^i_t) \cup \mathcal{Y}^-_s(g^i_t)} \exp(\text{sim}(\hat{\mathbf{x}}^i_t, \mathbf{y}))} + \log \frac{\sum_{\mathbf{y} \in \mathcal{Y}^+_e(g^i_t)} \exp(\text{sim}(\hat{\mathbf{x}}^i_{t+1}, \mathbf{y}))}{\sum_{\mathbf{y} \in \mathcal{Y}^+_e(g^i_t) \cup \mathcal{Y}^-_e(g^i_t)} \exp(\text{sim}(\hat{\mathbf{x}}^i_{t+1}, \mathbf{y}))} \right).$$

Here, $\text{sim}(\cdot, \cdot)$ is a similarity function defined in Section 3.3.2, and $\mathcal{Y}^+_s(g^i_t)$, $\mathcal{Y}^-_s(g^i_t)$, $\mathcal{Y}^+_e(g^i_t)$ and $\mathcal{Y}^-_e(g^i_t)$ are *start positives*, *start negatives*, *end positives* and *end negatives* of g^i_t, respectively, which are defined in the

[8] The unmasked sequence and the masked sequence may have different lengths before padding, but we pad them to have the same length.

next paragraph. This objective follows a phrase-level contrastive learning objectives in prior work [Lee et al., 2021; Ram et al., 2021; Deng et al., 2021; Kulkarni et al., 2022].

The start positives and the end positives are the start and the end of the spans to be retrieved. The start negatives and the end negatives are tokens that are not the start positives and not the end positives, respectively.

$$\begin{aligned}
\mathcal{Y}_{\mathrm{s}}^{+}(g_t^i) &= \{x_m^j | g_t^i = x_m^j ... x_{m+|g_t^i|-1}^j \ \ \& \ \ i \neq j\}, \\
\mathcal{Y}_{\mathrm{s}}^{-}(g_t^i) &= \{x_m^j | g_t^i \neq x_m^j ... x_{m+|g_t^i|-1}^j \ \ \& \ \ i \neq j\}, \\
\mathcal{Y}_{\mathrm{e}}^{+}(g_t^i) &= \{x_m^j | g_t^i = x_{m-|g_t^i|+1}^j ... x_m^j \ \ \& \ \ i \neq j\}, \\
\mathcal{Y}_{\mathrm{e}}^{-}(g_t^i) &= \{x_m^j | g_t^i \neq x_{m-|g_t^i|+1}^j ... x_m^j \ \ \& \ \ i \neq j\}.
\end{aligned}$$

Here, $|g_t^i|$ indicates the length of the span g_t^i.

Training Details

Training data. We use English Wikipedia (August 2019) and an English portion of CC-News (Mackenzie et al. [2020], February 2019) for training, which contains 13B tokens in total. The data is segmented into sequences, each with up to 256 tokens.

Training. We use the model architecture and initial weights of RoBERTa large [Liu et al., 2019], consisting of 354M parameters. Training is done for 100,000 steps, using thirty-two 32GB GPUs. One batch consists of 512 sequences (131,072 tokens). We use an Adam optimizer [Kingma and Ba, 2014] with a learning rate of 3×10^{-5}, weight decay of 0.01 and $4,000$ steps of warm-up.

Batching. The choice of batching is important in in-batch approximations, as it determines the quality of positives and negatives. For instance, Zhong et al. [2022b] uses BM25 to ensure the sequences in the same batch are likely to share the same topic. With a pre-training corpus with billions of tokens, it can be significantly expensive to build a BM25 index. Therefore, we instead construct the batch by grouping sequences from the same document and assigning them to the same batch.[9] This trick ensures that (a) positives (spans that share the string) are likely to share the context, reducing false positives, and (b) negatives are those that the model is likely to be confused with, thus training against them helps the model better identify positives. During training, we gather all sequences from multiple GPUs to increase the size of the effective batch and make in-batch approximation more effective.

3.3.4 Experiments: Closed-set Tasks

We perform zero-shot evaluation on closed-set tasks where a small set of candidates is given.

[9] Documents that are not long enough to construct a batch are grouped with each other.

Model	# Params	AGN	Yahoo	Subj	SST-2	MR	RT	CR	Amz	RTE	Avg
Baselines (encoder-only)											
RoBERTa [Gao et al., 2021]	1.0x	-	-	51.4	83.6	80.8	-	79.5	-	51.3	-
RoBERTa	1.0x	71.3	41.4	67.6	84.5	81.7	81.1	80.4	83.5	57.4	72.1
Baselines (encoder-decoder)											
T5	2.2x	72.0	51.3	54.9	57.5	57.7	59.1	56.4	59.3	55.6	58.2
T5 3B	8.5x	**80.5**	53.6	54.8	59.6	58.6	57.3	53.7	57.0	58.5	59.3
Baselines (decoder-only)											
GPT-2 [Shi et al., 2022]	2.2x	67.4	49.7	60.8	55.3	54.6	53.0	66.2	57.6	53.1	57.5
+ PMI [Shi et al., 2022]	2.2x	65.1	48.8	62.5	76.5	74.6	74.1	82.8	76.2	54.2	68.3
GPT-2 kNN† [Shi et al., 2022]	2.2x	29.8	37.0	50.0	47.1	49.9	49.1	69.3	57.4	54.1	49.3
GPT-2 kNN-LM† [Shi et al., 2022]	2.2x	78.8	51.0	62.5	84.2	78.2	80.6	84.3	85.7	55.6	73.4
GPT-3 [Holtzman et al., 2021]	500x	75.4	53.1	66.4	63.6	57.4	57.0	53.8	59.4	56.0	60.2
+ PMI [Holtzman et al., 2021]	500x	74.7	54.7	64.0	71.4	76.3	75.5	70.0	75.0	**64.3**	69.5
Ours (encoder-only, nonparametric)											
NPM†	1.0x	74.5	53.9	**75.5**	**87.2**	**83.7**	**86.0**	81.2	83.4	61.7	**76.4**
Full fine-tuning (reference)											
RoBERTa [Gao et al., 2021]	1.0x	-	-	97.0	95.0	90.8	-	89.4	-	80.9	-

Table 3.6: Zero-shot results on closed-set tasks. *# Params* indicates the relative number of model parameters compared to RoBERTa large (354M). RoBERTa, T5 and GPT-2 are their *large* variants unless specified otherwise; GPT-3 is from *Davinci, non-instruct*. Numbers with citations are taken from the corresponding papers; numbers without citations are from our own experiments. We run the code provided by Shi et al. [2022] and Holtzman et al. [2021] for datasets that are not included in the original paper ({Subj} and {Yahoo, Subj, MR, RT, CR}, respectively). As a reference, we provide results of fine-tuning on the full training dataset in the last row. † indicates a reference corpus is used. **NPM significantly outperforms larger parameters models.**

Evaluation Datasets

We include nine classification datasets that are known for not necessarily requiring factual knowledge: AGNews [Zhang et al., 2015], Yahoo [Zhang et al., 2015], Subj [Pang and Lee, 2004], SST-2 [Socher et al., 2013b], MR [Pang and Lee, 2004], Rotten Tomatoes (RT), CR [Hu and Liu, 2004], Amazon polarity (Amz, McAuley and Leskovec [2013b]) and RTE [Dagan et al., 2005]. The tasks range from topic classification and sentiment analysis to subjectivity classification and textual entailment. Statistics are provided in Appendix B.1.3.

Baselines

We compare with the encoder-only, the decoder-only and the encoder-decoder models with various sizes (354M to 175B parameters). We include RoBERTa [Liu et al., 2019] as the encoder-only, T5 [Raffel et al., 2020] as the encoder-decoder, and GPT-2/3 [Radford et al., 2019; Brown et al., 2020a] as the decoder-only model. For the decoder-only models, we additionally apply PMI [Holtzman et al., 2021] for better calibration

Figure 3.6: Predictions from RoBERTa (baseline) and NPM. The bottom indicates the context NPM retrieves to fill in `[MASK]`. Note that the fuzzy verbalizer maps *broken* to `Negative` and *awesome* to `Positive`.

of the model output. We also compare with Shi et al. [2022] who use kNN inference using GPT-2 with PMI. In particular, (1) GPT-2 kNN uses kNN inference without training, and (2) GPT-2 kNN-LM interpolates distributions from GPT-2 and GPT-2 kNN.

Setup

We use the templates and verbalizers from Shi et al. [2022] for all models. When available, we use fuzzy verbalizers from Shi et al. [2022]. We use a domain-specific reference corpus: a union of the English Wikipedia and CC News for AGN, Yahoo and RTE, a subjectivity corpus for Subj, and a review corpus for sentiment classification datasets. Their sizes vary from 15M tokens to 126M tokens. Details are in Appendix B.1.3. Fast similarity search is done using FAISS [Johnson et al., 2019] with the HNSW index. We use $k = 4096$ for inference.

Results

NPM outperforms baselines in the zero-shot setting (Table 3.6). We discuss the results in detail below.

Comparison between baselines. Among parametric models, RoBERTa achieves the best performance, outperforming larger models including GPT-3. This is perhaps surprising, and is likely because bidirectionality of the encoder-only model plays a vital role, as claimed in Patel et al. [2022]. The kNN-LM approach from Shi et al. [2022], which incorporates the nonparametric component to the parametric model, outperforms all other baselines. Nonetheless, solely relying on retrieval (kNN) performs poorly with GPT-2, suggesting that using kNN at inference only is limited.

Baselines versus NPM. NPM significantly outperforms all baselines, achieving consistently competitive performance over all datasets. This indicates that, even for tasks that do not explicitly require external knowledge, nonparametric models are very competitive.

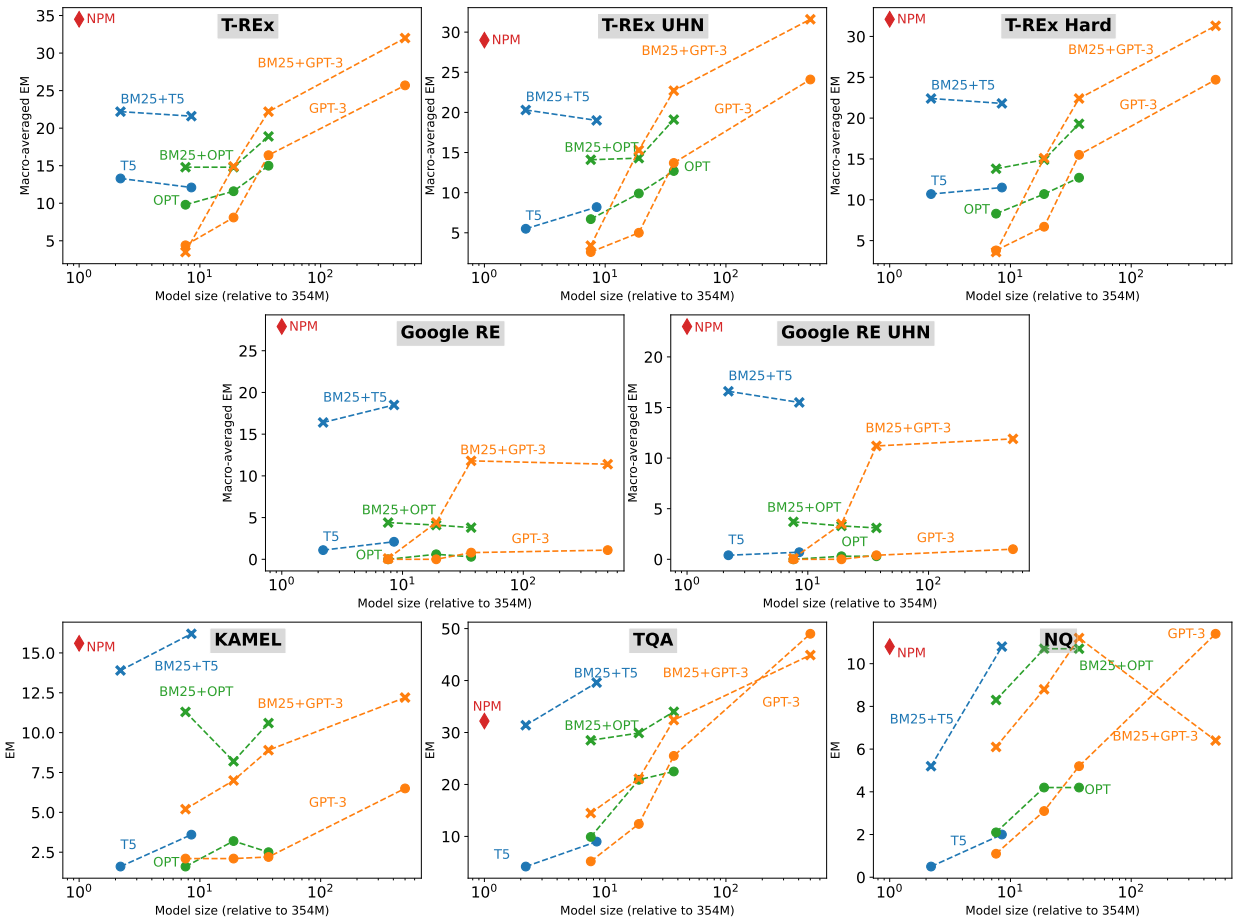

Figure 3.7: Zero-shot results on knowledge tasks. The x-axis indicates the relative number of model parameters in log scale compared to RoBERTa large (354M). NPM outperforms significantly larger parameters models, either with or without BM25. See Table B.4 in Appendix C.2 for the raw numbers.

Qualitative analysis. Figure 3.6 depicts predictions from RoBERTa and NPM on a sentiment analysis task. The first example uses *cheap* to indicate *inexpensive*, and the second example uses *cheap* to indicate *of very poor quality*. RoBERTa predicts `Positive` to both, while NPM makes correct predictions by retrieving the context that uses *cheap* in the same context as the input.

We also find that representations from NPM lead to better word sense disambiguation. For instance, RoBERTa assigns a high similarity score between *cheap* (*inexpensive*) and *cheap* (*of very poor quality*). On the other hand, NPM successfully assigns a low similarity score between *cheap* and *cheap*, even though their surface forms are the same.

3.3.5 Experiments: Open-set Tasks

We include zero-shot evaluation on open-set tasks whose answer can be any arbitrary-length string.

Evaluation Datasets

We evaluate on seven datasets: T-REx and Google-RE from LAMA [Petroni et al., 2019], KAMEL [Kalo and Fichtel, 2022], Natural Questions (NQ, Kwiatkowski et al. [2019]), TriviaQA (TQA, Joshi et al. [2017]), TempLAMA$^{22}_{19}$ and an entity translation task. In particular, TempLAMA requires probing knowledge with temporal updates, motivated by Dhingra et al. [2022] and Jang et al. [2022a]. The entity translation task involves a translation of an entity from English to other, non-Latin languages, requiring the model to predict extremely rare (if not unseen) characters. See Appendix B.1.3 for details and statistics of all datasets.

Baselines

We compare with T5 [Raffel et al., 2020] as the encoder-decoder, and GPT-3 [Brown et al., 2020a] and OPT [Zhang et al., 2022] as the decoder-only models. The encoder-only models are not applicable for open-set tasks since the number of tokens to predict is unknown.

Prior work found that a "retrieve-and-generate" approach that concatenates the input and passages from an off-the-shelf retrieval system is often helpful in knowledge-dependent tasks [Kandpal et al., 2022a]. We add them as baselines, using up to five passages from BM25 [Robertson et al., 2009].

Setup

For all datasets, we report Exact Match (EM). The LAMA test data is biased toward frequent entities because they are filtered to only include answers that are single tokens based on BERT [Devlin et al., 2019]. Since we do not want our evaluation to be biased toward overly frequent entities, we report a micro-averaged accuracy over the data whose answers are 1, 2, 3 and 4+ grams, respectively. Other datasets do not have such filtering, therefore we report average EM.

As a reference corpus, we use the English Wikipedia from 08/01/2019, consisting of 810M tokens. For TempLAMA$^{22}_{19}$, we use the English Wikipedia from 08/01/2022, consisting of 858M tokens.

For NPM, we find combining with sparse retrieval significantly helps, likely because dense retrieval and sparse retrieval capture complementary features [Karpukhin et al., 2020; Seo et al., 2019]. In particular, we reduce the search space to the top 3 passages based on BM25 and perform dense search as done in Kassner and Schütze [2020].

Results

Figure 3.7 show results on five knowledge tasks.

First, performance of parametric models largely depends on the number of parameters, as it has been claimed in much of prior work [Brown et al., 2020a; Kandpal et al., 2022a]. The retrieve-and-generate approach that combines parametric models with BM25 significantly improves performance.

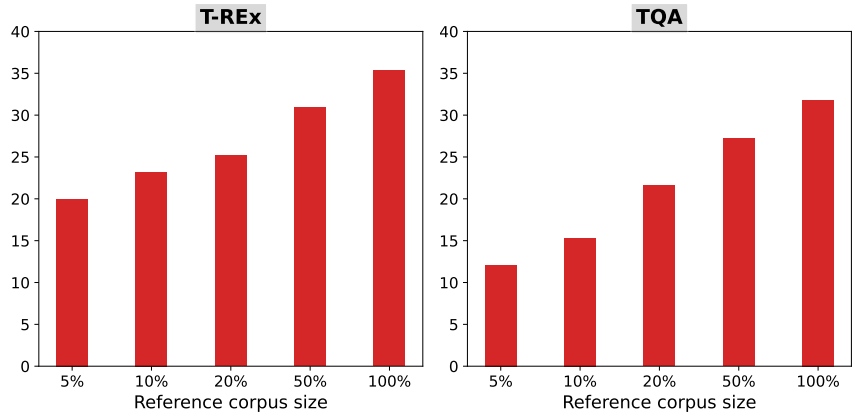

Figure 3.8: Ablation on the size of the reference corpus in NPM, from 41M tokens (5%) to 810M tokens (100%). There is a strong correlation between the size of the corpus and downstream performance.

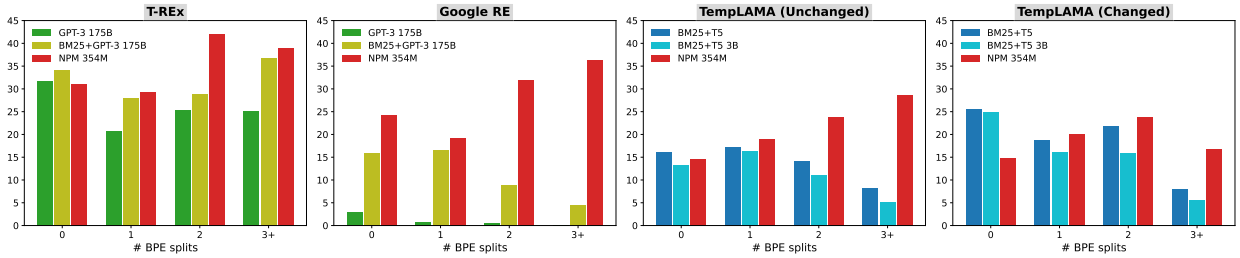

Figure 3.9: Performance on LAMA and TempLAMA tasks, broken down based on the number of BPE splits of the target entity, which is an indication of rarity of the entities (L:Frequent→R:Rare). NPM outperforms GPT-3 or T5 more significantly when the target entities are rare.

NPM outperforms or is on par with significantly larger baselines across all datasets. It substantially outperforms all models on two LAMA datasets, including 500x larger GPT-3 either with or without BM25. On KML, TQA and NQ, NPM consistently outperforms 37x larger models with or without BM25. This is impressive given that NPM is not trained on data with questions.

It is also worth noting that sparse retrieval is critical in NPM, e.g., without sparse retrieval, performance on LAMA-TREx drops from 34.5 to 16.1. We think this is because (1) sparse retrieval and dense retrieval capture complementary features, and (2) the removal of approximation in search improves search quality. We think future work can explore completely removing sparse retrieval, as has been done in Lee et al. [2021] to improve Seo et al. [2019].

Impact of the reference corpus size. Figure 3.8 reports the impact of the size of the reference corpus, from 41M tokens (5%) to 810M tokens (100%). Performance of NPM is highly correlated with the size of the reference corpus, strongly suggesting that using a larger reference corpus is important.

Model	#Params	Unchanged	Changed	AVG
Baselines				
T5	2.2x	1.9	0.4	1.1
T5 3B	8.5x	1.8	0.4	1.1
OPT 6.7B	19x	2.5	1.0	1.7
OPT 13B	37x	4.9	2.1	3.5
BM25 + T5	2.2x	13.7→14.9	3.0→**20.1**	17.5
BM25 + T5 3B	8.5x	11.9→12.0	2.2→17.8	14.9
BM25 + OPT 6.7B	19x	10.2→8.2	1.7→11.3	9.7
BM25 + OPT 13B	37x	14.8→14.4	2.8→16.6	15.5
Ours				
NPM	1.0x	18.9→**19.5**	2.9→17.5	**18.5**

Table 3.7: Results on TempLAMA$^{22}_{19}$, on an unchanged set, a changed set, and a macro-average over two, respectively. xx→xx indicates performance when using the outdated and the updated Wikipedia, respectively.

Model	#Params	#L	w/o BM25	w/ BM25
Baselines, English-only				
T5	2.2x		0.2	1.9
T5 3B	8.5x		0.5	4.4
OPT 6.7B	19x		0.4	22.3
OPT 13B	37x		1.0	24.6
Ours, English-only				
NPM	1.0x		**52.4**	
References, Multilingual				
mT5	3.4x	101	1.3	19.0
mT5 XL	11x	101	4.1	**56.6**
BLOOM 3B	8.5x	46	0.0	17.4
BLOOM 7.1B	20x	46	0.1	26.0

Table 3.8: Results on the entity translation task. See Table B.6 (Appendix C.2) for per-language results. *#L* indicates the number of languages multilingual models are trained on. **Bold** and **Bold** indicate the best among monolingual models and the best including multilingual models, respectively. NPM significantly outperforms all existing monolingual models, and approaches or outperforms larger multilingual models.

Results on temporal knowledge tasks. Table 3.7 reports results on TempLAMA. NPM retains its performance on the unchanged set (18.9 →19.5) and successfully updates its answers on the changed set (2.9 → 17.5). Its performance is significantly better than the performance of parametric models with up to 13B parameters, and is on par with a larger model with the retrieve-and-generate approach, which also successfully updates its answer by leveraging the updated corpus. This is in agreement with prior work that shows the model with a nonparametric component adapts to temporal updates by replacing the reference corpus at test time [Izacard et al., 2022b]. Nonetheless, the retrieve-and-generate approach is still significantly worse than NPM when the target entities are rare, which we show in the next paragraph.

Performance on rare entities. We break down the instances on LAMA and TempLAMA based on the number of BPE splits of the target entity, e.g., *Thessaloniki* is one word that is split into 4 BPE tokens, thus the number of splits is 3. Since BPE splits a word if they are rare, the number of BPE splits indicates the rarity of the entity. We compare NPM with GPT-3 and BM25+GPT-3 on LAMA, and BM25+T5 (770M and 3B) on TempLAMA, the two most competitive baselines on each dataset.

Figure 3.9 reports results. On LAMA, NPM outperforms GPT-3 fairly consistently, with larger gains as the number of BPE splits increases. On TempLAMA, while BM25+T5 is competitive on frequent entities with zero BPE split, it consistently lags behind NPM with ≥ 1 BPE splits. This suggests that NPM is particularly good at addressing rare entities, compared to not only parametric models without retrieval but also the retrieve-and-generate approach.

Results in Entity Translation. Results on the entity translation task are shown in Table 3.8 (per-language results are reported in Table B.6 of Appendix C.2). T5 and OPT struggle to perform the task, both with and without BM25 retrieval. that are extremely rare or unseen. In contrast, NPM performs well across all languages.

In order to better calibrate performance of NPM, we provide reference performance of models that are purposely trained on the multilingual data—mT5 [Xue et al., 2021] and BLOOM [Scao et al., 2022]. NPM outperforms 3.4x larger mT5 and 20x larger BLOOM, and approaches 11x larger mT5, even though it is trained on English. We think strong cross-lingual transferability of NPM is likely because it can retrieve a phrase based on its surrounding context, even if it has not seen the exact word during training.

3.3.6 Summary & Limitations

We introduced NPM, a nonparametric masked language model that replaces a softmax over the output vocabulary with a nonparametric distribution over a reference corpus. NPM can be efficiently trained using a contrastive objective and an in-batch approximation to a full corpus. Zero-shot evaluation on 16 tasks shows that NPM outperforms significantly larger parametric models. NPM is particularly good at rare patterns (word senses or facts), scaling and updating at test time, and predicting extremely rare if not unseen characters.

NPM has a few limitations, which we discuss below.

Scaling through the inference corpus. The size of the reference corpus is an additional dimension for model scale in nonparametric models. In this paper, we scale the corpus up to nearly 1B tokens, which is still smaller than the training data of very large language models [Brown et al., 2020a; Rae et al., 2021]. We think future work can scale it further using tools such as Distributed FAISS [Johnson et al., 2019] or ScaNN [Guo et al., 2020].

Significant memory usage. Using NPM saves GPU compute and memory compared to using models with more parameters. However, NPM requires more RAM and disk memory due to embeddings of a reference corpus. For instance, the largest corpus in our experiments (full English Wikipedia) requires 70GB of RAM and 1.4TB of disk memory. Future work can build more efficient NPM as done in prior work in nearest neighbor search [Jegou et al., 2010; Norouzi et al., 2012; Ge et al., 2014; Izacard et al., 2020; Yamada et al., 2021].

Exploration of larger vocabulary. Large vocabulary is known to lead performance gains [Conneau et al., 2020] but is bounded in memory costs. Previous work explored more efficient softmax approximations [Morin and Bengio, 2005; Chen et al., 2016; Grave et al., 2017]. Our nonparametric training offers an alternative by removing the softmax over the vocabulary. With the RoBERTa architecture, increasing the vocab size by 2x makes the baseline training 50% more memory expensive, but does not increase the memory in training

NPM. However, this paper does not include more systematic evaluation on the effect of large vocabulary. Future work can explore training NPM with a significantly larger vocabulary to further boost performance.

Extension for generation. Our paper evaluates NPM only on prediction tasks. It is currently non-trivial to use NPM for generation, since it is the encoder-only model. Future work can explore autoregressive generation as done in Patel et al. [2022] or use NPM for editing [Schick et al., 2022; Gao et al., 2022].

Extension to few-shot learning and fine-tuning. Our paper focuses on zero-shot evaluation only. Future work can extend NPM to a few-shot learning setup. In fact, fine-tuning NPM is significantly easier than fine-tuning larger models such as T5, OPT and GPT-3 which we compare NPM with, and can be explored in future work.

Better cross-lingual transfer. Our work explored cross-lingual transfer in a limited setup where the model is trained on monolingual data. We think future work can train multilingual NPM, and explore more comprehensive cross-lingual evaluation. In fact, nonparametric training may alleviate the burden of collecting large-scale multilingual corpora since it makes the model less sensitive to the language coverage in the training data, and may lead to significantly better cross-lingual transfer, as we demonstrate in the entity translation task.

Limitation in speed. We find that search makes inference considerably slower than the counterpart without search. We think that (1) search can significantly be faster with better engineering (we use the default hyperparameters of the FAISS index with no tuning) or better index, and (2) the speed of NPM is still on par with the speed of significantly larger parametric models that NPM outperforms. Moreover, while not explored in this work, there has been work that improves inference speed [He et al., 2021; Alon et al., 2022] that can be applied to NPM. We leave improving inference speed to future work.

3.4 Summary and Future Work

In this section, we discuss nonparametric language models, a new class of LMs with both learned parameters and a datastore. These models are significantly more parameter efficient and better at capturing long-tail data distributions because they can retrieve relevant information from a datastore at test time without relying solely on memorization of the training data through enormous amounts of parameters. Moreover, since the datastore can be updated at any time with no further training, these models can easily stay up-to-date (e.g., by keeping the datastore up-to-date) and support opt-out (e.g., by removing datapoints from the datastore).

We highlighted our key contributions to nonparametric language models, namely: (1) we developed **DPR**, one of the first widely used dense retrieval models, (2) we organized the **EfficientQA** competition, one of the first studies to systemically compare parametric and nonparametric models in various conditions, including time-shift and memory-constraints, (3) we proposed **in-context pre-training**, a new pre-training

objective that resembles retrieval augmentation and substantially improves models with both 1 billion and 7 billion parameters, and (4) we introduced **NPM**, one of the first models trained with a nonparametric softmax, exploring an alternative method for incorporating the datastore beyond conventional retrieval augmentation.

Despite significant progress in nonparametric language models, many open questions remain. We highlight key questions below.

Model architecture for nonparametric LMs. Arguably, the overall architecture of nonparametric LMs has seen few changes beyond the basic approach of retrieval augmentation, which involves conditioning on a few retrieved documents. This method has apparent limitations, such as the restricted number of passages that can be incorporated and the need to retain many unnecessary computations, e.g., attention across multiple retrieved passages. The nonparametric softmax model, while promising, introduces significant increases in runtime and memory costs due to a substantial increase in the number of vectors in a datastore.

We believe it is necessary to explore alternative architectures that can incorporate a larger retrieval of results more frequently and more efficiently. One such attempt is the RETRO model [Borgeaud et al., 2022], which modifies the attention layers of Transformers to integrate retrieval results in parallel with the input. This architecture allows for the efficient handling of a vast number of documents with greater frequency, incorporating many documents in parallel. However, follow-up on this model has been limited because it is not open-sourced and requires pre-training from scratch. For a comparative analysis of retrieval augmentation, nonparametric softmax, and RETRO, see Table 3.9.

Another possibility is to leverage advances in long-context LMs, which have seen significant progress through architectural innovations and systems development. Architectural modifications typically include either significantly sparsifying attentions or entirely removing the attention mechanism. If these long-context models could scale to process inputs as large as a trillion tokens, they might effectively act as nonparametric LMs by processing a datastore of that magnitude. Such models could approximate performing nearest neighbor searches within the attention layers, which essentially equates to the use of retrieval operations for each attention operation.

Scaling nonparametric LMs. One key ingredient in the huge success of large LMs is scaling, typically of the number of parameters and the size of the training data. A central question here is whether nonparametric LMs can provide an alternative pathway for scaling—the datastore size. To answer this question, we should ideally scale the datastores to accommodate Internet-scale data consisting of trillions of words—the scale of modern LMs' training data. Currently, they accommodate relatively small-scale data (e.g., a few billions of words) and are thus limited to a specific domain (e.g., Wikipedia) in many cases.

A key obstacle in scaling the datastore lies in scaling the nearest neighbor search algorithm to handle trillions of vectors while improving its runtime speed. Addressing this challenge necessitates not only NLP and machine learning research but also collaborative efforts with researchers specializing in computer systems, algorithms, and databases.

Retrieval augmentation	Nonparametric softmax	RETRO
	Pros	
• Easy to use off-the-shelf retrieval models and LMs with a black-box access.	• A single model, without two separate models. • Scales better than retrieval augmentation.[a] • Generalizes better to OOD datastore.[a] • Can efficiently encode a large # of passages, more frequently.	• Easy to use an off-the-shelf retrieval model with a black-box access.[b] • Can efficiently encode a large # of passages, more frequently.
	Cons	
• Two models, each trained in isolation.[c] • Error propagation issues. • The # of passages is strictly limited by the sequence length limit of LMs.[d] • Frequent retrieval significantly increases runtime. • Has many unnecessary operations, e.g., attention across multiple retrieved passages.	• Needs a vector per token (or phrase) instead of vector per passage, making the model more expensive in runtime and memory.[e] • Computation used for incorporation is the least.[f]	• The LM has to be trained from scratch. • Error propagation issues.

Table 3.9: Comparisons of various architectures for nonparametric LMs. Further discussion can be found in the slides for nonparametric model architectures from the ACL 2023 Tutorial [Asai et al., 2023].
[a]: Discussed in Section 4.5. [b]: However, using a different retrieval model may require re-training of a language model.
[c]: Joint training [Guu et al., 2020; Izacard et al., 2022b] or sequential training [Shi et al., 2024a] can potentially address this issue.
[d]: A massive long context language model can potentially address this issue.
[e]: For instance, the number of vectors in a datastore increases by 100x if each passage has 100 tokens. This is equally applicable to retrieval augmentation or RETRO with multi-vector retrieval [Khattab and Zaharia, 2020; Santhanam et al., 2021; Lee et al., 2024].
[f]: This is because the incorporation is through a single inner product operation at the final prediction stage, unlike other models that incorporate retrieval results in every attention layer.

Runtime efficiency of nonparametric LMs. Nonparametric LMs are typically slower than parametric ones with the same number of parameters due to the additional operation of nearest neighbor searches. Improving runtime efficiency of this search is critical for speeding up their runtime. Several strategies can be employed to achieve this, including: (1) systems-level improvements, such as better distributed search, (2) an approximate nearest neighbor search algorithm with a more optimal trade-off between runtime speed, memory consumption and accuracy, and (3) the development of caching that is specificially optimized for next token predictions.

Chapter 4

Responsible Language Models

4.1 Overview

Large language models (LMs) are under widespread legal scrutiny, in large part because they are trained on copyrighted content, which may infringe on the rights of data producers [Metz, 2022; Vincent, 2023; J.L. et al. v. Alphabet Inc., 2023; Brittain, 2023]. At the heart of this discussion is the inherent tradeoff between legal risk and model performance. Training only on data sources such as public domain, non-copyrightable or otherwise permissively licensed data significantly degrades performance (as we show in §4.5). This limitation arises from the scarcity of permissive data and its narrow specificity to sources such as copyright-expired books, government documents, and permissively licensed code, which are largely different from common LM corpora that cover more diverse domains [Raffel et al., 2020; Gao et al., 2020; Together, 2023].

In this section, we demonstrate it is possible to improve the risk-performance tradeoff by segregating training data into two distinct parts of the model: parametric and nonparametric (Figure 4.1). We learn LM parameters on *low-risk* data (i.e., data under the most permissive licenses), and then use *high-risk* data (i.e., data under copyright, restrictive licenses, or unknown licenses) in an inference-time-only nonparametric component (called a *datastore*). With nonparametric datastores, we can *retrieve* high-risk data to improve model predictions without training on it. The datastore can be easily updated at any time, and allows creators to remove their data from the model entirely, at the level of individual examples. This approach also attributes model predictions at the sentence-level, enabling credit assignment to data owners. These new capabilities enable better alignment of the model with various data-use regulations, e.g., the *fair use* doctrine in the United States [Henderson et al., 2023] and the GDPR in the European Union [Zhang et al., 2023a], as detailed in §4.2. This is in contrast to parametric models, where removing high-risk data is infeasible after training [Bourtoule et al., 2020; Carlini et al., 2021] and data attribution at scale is difficult [Feldman and Zhang, 2020; Zhang et al., 2021; Han et al., 2023].

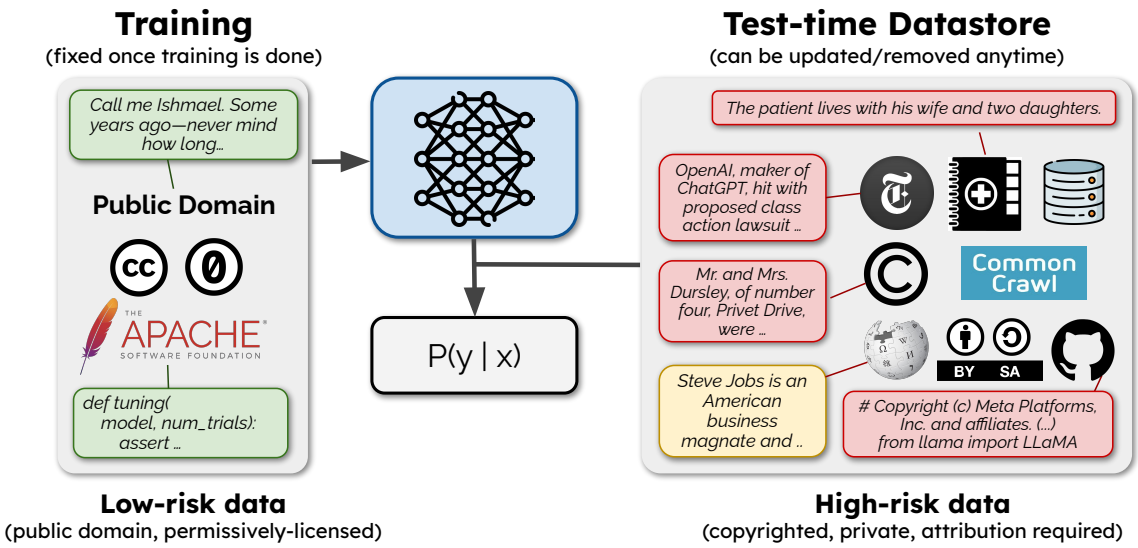

Figure 4.1: An overview of SILO. We train a *parametric* language model on low-risk datasets that contain public domain text (e.g., copyright-expired books) and permissively licensed code. At inference time, we use a *nonparametric datastore* that can include high-risk data, including medical text with personally-identifiable information, copyrighted news, copyrighted books, data requiring attribution, and code under non-permissive licenses (counterclockwise from the top of figure). The datastore can be modified at any time, e.g., to respond to opt-out requests.

We introduce **SILO**, a new nonparametric language model that follows our proposal (§4.4). The parametric component in SILO is trained on a new pre-training corpus, the **OPEN LICENSE CORPUS** (**OLC**, §4.3), which we curate to include data under three types of permissive licenses, from public domain to Creative Commons. OLC is diverse but has a domain distribution that is very different from typical pre-training corpora; it is dominated by code and government text. This leads to a new challenge of generalizing a model trained on highly specific domains, which we call *extreme domain generalization*. We train three 1.3B-parameter LMs on varying subsets of OLC, and then construct a test-time datastore that can include high-risk data, employing a retrieval method to make use of the datastore's contents during inference. We compare two widely studied retrieval methods: a nearest-neighbors approach (kNN-LM) that uses a nonparametric next-token prediction function [Khandelwal et al., 2020] and a retrieval-in-context approach (RIC-LM) that retrieves text blocks and feeds them to the parametric LM in context [Shi et al., 2024b; Ram et al., 2023].

We evaluate SILO in language modeling perplexity on 14 different domains, covering both in-domain and out-of-domain data with respect to OLC (§4.5). These domains highlight specific legal risks, e.g., copyrighted materials such as books, news and user reviews, or private data such as emails and clinical notes. We compare SILO to Pythia [Biderman et al., 2023], a parametric LM with a similar parameter count but

trained mostly on high-risk data [Gao et al., 2020].[1] We first show that parametric-only SILO is competitive on domains covered by OLC but falls short out-of-domain, confirming the challenge of extreme domain generalization. However, adding an inference-time datastore to SILO effectively addresses this challenge. Comparing the two methods of retrieving over this datastore, we find that while both kNN-LM and RIC-LM significantly improve out-of-domain performance, the former generalizes better than the latter, allowing SILO to reduce the gap with the Pythia baseline by 90% on average across all domains. Further analysis attributes these improvements to two factors: (1) kNN-LM strongly benefits from scaling the datastore and (2) the nonparametric next-token prediction in kNN-LM is robust to domain shift. Altogether, our study suggests that in the few domains where SILO has not yet matched Pythia performance levels, the remaining gaps can likely be closed by scaling the datastore size and further enhancing the nonparametric model.

4.2 Background

Training datasets for language models. State-of-the-art LMs are trained on vast text corpora that consist of billions or even trillions of tokens [Brown et al., 2020b; Raffel et al., 2020; Gao et al., 2020; Together, 2023]. These training sets are built by combining (1) manually selected sources such as Wikipedia, book collections, and GitHub and (2) web pages collected through web-crawling services such as Common Crawl. Most LM training efforts ignore copyright and intellectual property regulations that apply to these texts. For example, sources such as GitHub repositories and book collections typically contain text with highly restrictive licenses [Bandy and Vincent, 2021].

Legality of language models. The legality of training LMs this way has become a subject of intense debate, with numerous lawsuits being filed in the United States, United Kingdom, and beyond [Gershgorn, 2021; Metz, 2022; Vincent, 2023; De Vynck, 2023; Silverman et al. v. Meta Platforms, Inc., 2023; J.L. et al. v. Alphabet Inc., 2023; Silverman et al. v. OpenAI, Inc., 2023; Tremblay et al. v. OpenAI, 2023]. While the outcome of the lawsuits is uncertain, it is likely that such legal issues will continue to be a major factor in future LMs, especially since each country has its own data regulations. For example,

- In the United States, the *fair use doctrine* allows the public to use copyrighted data in certain cases, even without a license [Henderson et al., 2023]. Deciding whether or not a model's use of copyrighted work constitutes fair use involves multiple dimensions, including whether the trained model is intended for commercial use, whether or not the work is factual or creative, the amount of the copyright content used, and the value of the copyrighted work. There are claims that using parametric language models for *generative* use-cases does *not* constitute fair use, because the technology may output the copyrighted text verbatim [Lemley and Casey, 2020], which also has been shown empirically [Carlini et al., 2021,

[1] The Pile contains a large amount of copyrighted or restrictively licensed data, e.g., most content in its Books3, ArXiv, Github, OpenWebText, YoutubeSubtitles, and Common Crawl subsets.

2023; Kandpal et al., 2022b; Chang et al., 2023]. This is in contrast to *transformative* technologies, such as classifiers, which may use the copyrighted text but do not directly generate content, which the fair use doctrine favors. We refer readers to Henderson et al. [2023] for a more comprehensive discussion.

- The *General Data Protection Regulation (GDPR)* is a comprehensive data protection and privacy law in the European Union (EU). It grants individuals more control over their data by regulating organizations and businesses. The obligations include (1) obtaining consent from users before processing their data, (2) providing transparency about data processing, (3) ensuring data security, and (4) allowing individuals to access, correct, and erase their data. GDPR has global impact, as many international companies handle EU citizens' data. While it is under debate how GDPR is applied to training language models, compliance with GDPR is expensive (e.g., requiring retraining for every data correction or erasure). See Zhang et al. [2023a] for more discussion on challenges for compliance with the GDPR's *Right to Erasure* (and the *Right to be Forgotten* in general).

The goal of our work is not to weigh in on legal discussions; instead, we study the feasibility of developing technologies that explicitly manage legal risk. In particular, our technique places all copyrighted data in a nonparametric datastore. While the data is still used in service of a generative model, restricting copyrighted data in a datastore and providing instance-level attribution and data opt-out can increase the likelihood of a successful fair use defense [Henderson et al., 2022].[2] Moreover, GDPR's requirement regarding user data access, correction, and erasure aligns well with the capabilities of the datastore. Attribution and opt-out are fundamental features of our model (§4.4.2). This is in contrast to other techniques like post-hoc training data attribution [Koh and Liang, 2017; Han et al., 2023] and the removal of the effect of particular training examples from parameters [Cao and Yang, 2015; Jang et al., 2023b], which lack inherent guarantees and are hard to scale.

Prior work in copyright risk mitigation. The most straightforward approach to avoid copyright infringement is to filter training data to only include permissive licenses. This has been done in prior work, primarily for code-based datasets [e.g., Kocetkov et al., 2023; Fried et al., 2023; Together, 2023] and scientific text [e.g., Soldaini and Lo, 2023]. Extending a similar approach to a wider range of domains remains unclear, because permissive data is extremely scarce in most domains, e.g., books and news. For the same reason, Henderson et al. [2023] has suggested that restricting the training data to public domain or otherwise permissively licensed data may be impractical. In this work, we show that there is in fact a large number of tokens from data sources with permissive licenses, but the key challenge instead arises from the highly skewed domain distribution. See §4.6 for other copyright mitigation strategies that are more technical in nature.

[2] Our model on its own does not entirely remove legal risk. Rather, it provides functionalities that, when used appropriately, lower legal risk and strengthen a fair use defense. See §4.6 for a discussion.

4.3 OPEN LICENSE CORPUS: A Permissively-Licensed pre-training Corpus

Our study focuses on addressing the legal risk of copyright violation in language models by separating *low-risk* data sources (i.e., those in the public domain or under permissive licenses) from *high-risk* ones (i.e., those with unknown licenses or under copyright). We introduce the **OPEN LICENSE CORPUS (OLC)**, a large collection of permissive textual datasets across multiple domains with a taxonomy of data licenses that delineate their permissiveness (§4.3.1), grouped into three levels of legal permissiveness (§4.3.2). This curated data is then used to train model parameters (§4.4) and highlights the challenge of extreme domain generalization due to its skewed domain distribution.

A disclaimer. The license taxonomy and categorization of texts that we present is by no means perfect, and OLC should not be considered a universally safe-to-use dataset. The license associated with a document may be time- and country-dependent, e.g., Gutenberg books [Project Gutenberg] are public domain in the United States, but some of them may still have copyright attached outside of the United States. Moreover, other legal constraints (e.g., the Digital Millenium Copyright Act)[3] may prohibit the use of a data source despite a permissive data license. Finally, we do not explicitly filter out personally identifiable information from the corpus, so it is possible that certain subsets still pose privacy risks despite being permissively licensed. We encourage users of OLC to consult a legal professional on the suitability of each data source for their application.

4.3.1 Taxonomy of Data Licenses

As discussed in §4.2, determining what data one is permitted to use from a copyright perspective is an ongoing topic of debate, and is context- and country-dependent [Henderson et al., 2023]. In this paper, we take a conservative approach where we train models using only text with the most permissible licenses, thus enabling widespread downstream use. Concretely, we focus on four broad categories:

- **Public domain** (PD) text has no restrictions. This includes texts whose intellectual property rights have expired (e.g., the works of William Shakespeare) or been expressly waived by the creator (e.g., CC0-licensed scientific papers).
- **Permissively licensed software** (SW) including MIT, Apache, and BSD software are quite permissive to use. Unlike public domain text, these licenses typically carry some basic stipulations such as requiring one to include a copy of the original license (although, it is debatable whether it is still required when the associated text is used as data or treated as a software). The code is otherwise free to use, and code is generally well protected by fair use clauses [Lemley and Casey, 2020].

[3] https://www.copyright.gov/dmca/

- **Attribution licenses** (BY) such as Creative Commons Attribution (CC-BY) are free to use as long as *credit is given to the creator.* For example, if a journalist quotes an article from Wikipedia (a CC-BY-SA-3.0 source), then they must provide a form of citation, link, or attribution back to the original source. In the context of machine learning, it is not clear what an attribution would constitute. For example, under one interpretation, every LM generation should include a complete list of sources that contributed highly to it [Henderson et al., 2023]. In this paper, we take a conservative approach and do not include BY data in the main experiments, but still include the BY data for future use as well as for ablations, since BY data is generally considered quite permissive.

- **All other data** that is not in one of the above three categories is assumed to be non-permissive. This includes: any text that is explicitly protected by copyright or licenses that are non-commercial (e.g., CC-NC), any software without clear MIT, BSD, or Apache licenses, and any generic web-crawled data where the license or copyright information may be unclear.

In §4.4.3, we train the models on varying subsets of licenses—from PD and PDSW to PDBYSW—to accommodate different risk tolerances.

4.3.2 Building the OPEN LICENSE CORPUS

Based on this taxonomy of licenses, OLC is a 228B token corpus of PD, SW, and BY data. OLC consists of 17 manually-selected sources of primarily English text that are under permissive licenses,[4] as summarized in Table 4.1. The text generally falls into eight different domains:

- PD BY **Legal:** We curate legal text from the Pile of Law [Henderson et al., 2022], an amalgation of 31 different sources of text related to civil court cases, patents, and other legal and governmental works, either licensed as public domain or CC-BY. We also gather public domain text from the Case Law Access Project [Caselaw Access Project, 2018], which covers over 6.5 million decisions published by state and federal courts throughout U.S. history.

- SW **Code:** We use the Github subset of the RedPajama dataset [Together, 2023], which contains code from Github repositories with three permissive software licenses: MIT, Apache, and BSD.

- SW BY **Conversation:** We source conversational text under permissive software licenses from the HackerNews (MIT license) and the Ubuntu IRC (Apache license) subsets of the Pile [Gao et al., 2020]. We also use the Stackexchange subset of the RedPajama dataset [Together, 2023] and a Stackoverflow corpus from Kaggle,[5] both under the CC-BY-SA license.

[4] We include the data in only when the license information is clearly stated as part of metadata. While we tried our best to collect the data for OLC, it is possible we missed potential sources, as it relies on manual efforts.

[5] https://www.kaggle.com/datasets/stackoverflow/stackoverflow

Domain	Sources	Specific License	# BPE Tokens (B)
Legal	PD Case Law, Pile of Law (PD subset)	Public Domain	27.1
	BY Pile of Law (CC BY-SA subset)	CC BY-SA	0.07
Code	SW Github (permissive)	MIT/BSD/Apache	58.9
Conversational	SW HackerNews, Ubuntu IRC	MIT/Apache	5.9
	BY Stack Overflow, Stack Exchange	CC BY-SA	21.3
Math	SW Deepmind Math, AMPS	Apache	3.5
Science	PD ArXiv abstracts, S2ORC (PD subset)	Public Domain	1.2
	BY S2ORC (CC BY-SA subset)	CC BY-SA	70.3
Books	PD Gutenberg	Public Domain	2.9
News	PD Public domain news	Public Domain	0.2
	BY Wikinews	CC BY-SA	0.01
Encyclopedic	BY Wikipedia	CC BY-SA	37.0

Table 4.1: Overview statistics of OLC. PD, SW, and BY indicates public domain data, data under permissive software licenses, and data under attribution licenses, respectively. Some corpora contain a mixture of different licenses (e.g., Pile of Law and S2ORC), which we split into subsets based on per-document licenses. BPE tokens are based on the GPT-NeoX tokenizer [Black et al., 2022].

- SW **Math:** We source mathematical text from the Deepmind Mathematics [Saxton et al., 2019] and the AMPS [Hendrycks et al., 2021] datasets, both of which are under the Apache license.
- PD BY **Science:** We source scientific text from ArXiv abstracts that are in the public domain [ArXiv, 2023]. We also collect full-text articles from the Semantic Scholar Research Corpus [Lo et al., 2020, S2ORC], either licensed as public domain or CC-BY.
- PD **Books:** We source books from the Gutenberg corpus [Project Gutenberg], which are copyright-expired books that are in the public domain.
- PD BY **News:** We collect public domain news text from the English subset of the MOT corpus [Palen-Michel et al., 2022]. We also collect text from Wikinews, which is under CC BY-SA.
- BY **Encyclopedic:** Finally, we include a large set of Wikipedia from the subset included in RedPajama [Together, 2023]. We follow RedPajama in using Wikipedia snapshots from 20 languages even though the model primarily focuses on English.

Following Kandpal et al. [2022b]; Lee et al. [2022], we deduplicate text using Groeneveld [2023], a document-level filter that considers n-gram overlap. We first deduplicate within each domain to remove redundant documents from similar sources (e.g. Case Law and the Pile of Law), and then perform deduplication against the validation and test datasets of the Pile to avoid test leakage.

	PD		PDSW		PDSWBY		*The Pile*	
Domain	**Tokens (B)**	**%**	**Tokens (B)**	**%**	**Tokens (B)**	**%**	**Tokens (B)**	**%**
Code	0.0	0.0	58.9	59.1	58.9	25.8	32.6	9.8
Legal	27.1	86.2	27.1	27.2	27.2	11.9	30.8	9.3
Conversation	0.0	0.0	5.9	5.9	27.2	11.9	33.1	10.0
Math	0.0	0.0	3.5	3.5	3.5	1.50	7.1	2.1
Books	2.9	9.3	2.9	2.9	2.9	1.3	47.1	14.2
Science	1.2	3.8	1.2	1.2	71.5	31.3	86.0	26.0
News	0.2	0.7	0.2	0.2	0.2	0.1	-†	-†
Wikipedia	0.0	0.0	0.0	0.0	37.0	16.2	12.1	3.7
Unverified web	0.0	0.0	0.0	0.0	0.0	0.0	83.1	25.0
Total	31.4	100.0	99.6	100.0	228.3	100.0	331.9	100.0

Table 4.2: OLC is large but its distribution is different from that of typical pre-training corpora like the Pile. Data distribution of OLC (PD, PDSW, PDSWBY) in comparison to the Pile [Gao et al., 2020], a common LM training dataset that is not specifically designed for legal permissibility. We report the number of tokens in billions, and the relative frequency. †: There is no explicit news domain in the Pile, but news sites are found to be some of the most representative data sources in Common Crawl [Dodge et al., 2021].

In Table 4.2, we compare the distribution of domains in OLC to that of the Pile [Gao et al., 2020], a popular pre-training corpus that includes data under copyright restrictions (e.g., Books, web crawl).[6] These statistics convey a number of research challenges when working with OLC. First, while we tried our best to collect public domain or permissively-licensed data, the size of OLC is still 31% smaller than the Pile. In addition, while the majority of the Pile is sourced from scientific text, web crawl, and books, OLC is dominated by code, scientific text, and legal text. This highlights that models designed for use outside these specific domains will likely struggle and may require special techniques for extreme domain generalization.

4.4 SILO

We introduce SILO, which combines an LM trained on permissive data with a nonparametric datastore based on less restricted data. Our goal with SILO is to build an LM—i.e., a model that takes a prefix of text x and outputs a next-word probability distribution over the vocabulary $P(y \mid x)$—but to do so in a legally safe way. We first describe the general methodology from prior work (§4.4.1–4.4.2) and then how we build SILO upon them by placing low-risk data and high-risk data to model parameters and a nonparametric datastore, respectively (§4.4.3). Implementation details are provided in §4.4.4.

[6]This comparison also dovetails with our experiments in §4.5, where we compare SILO to Pythia, a model trained on the Pile.

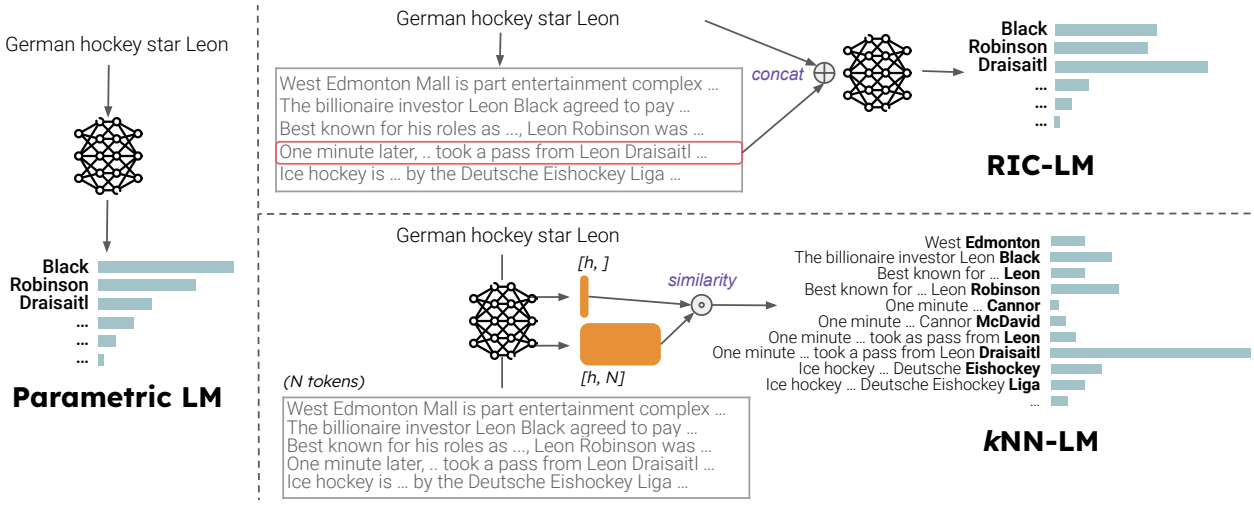

Figure 4.2: An illustration of a parametric model and two retrieval methods we compare: RIC-LM and *k*NN-LM. The orange boxes indicate representations of the input prefix and the tokens in the datastore, each in $\mathbb{R}^h$ and $\mathbb{R}^{h \times N}$, where h is a hidden dimension and N is the number of tokens in the datastore. The distribution from kNN-LM in the figure describes $P_{k\text{NN}}$; while omitted in the figure, the final output distribution from kNN-LM is an interpolation between $P_{k\text{NN}}$ and the distribution from the parametric LM. See §4.4.2 for more details of each method.

4.4.1 The Parametric Component

For the parametric component of SILO, we use a standard, dense, decoder-only transformer LM [Vaswani et al., 2017] using the LLaMA architecture [Touvron et al., 2023]. This model uses a fixed set of parameters at both training and inference time.

4.4.2 The Nonparametric Component

We experiment with two widely-used retrieval methods for the nonparametric component (Figure 4.2): the k-nearest neighbors LM (kNN-LM; Khandelwal et al., 2020) and the retrieval-in-context approach (RIC-LM; Shi et al., 2024b; Ram et al., 2023). Each approach constructs a datastore from the raw text data offline, and then uses it on-the-fly at inference time.

The *k*-nearest neighbors language model (*k*NN-LM). A kNN-LM [Khandelwal et al., 2020] interpolates the next-token probability distribution from a parametric LM with a nonparametric distribution based on every token that is stored in a datastore. Given a text dataset consisting of N tokens $c_1 ... c_N$, a datastore is built by creating a key-value pair for every token c_i ($1 \leq i \leq N$). Specifically, a value is c_i and a key k_i is $...c_{i-1}$, a prefix preceding c_i. At test time, given an input prefix x, the nonparametric distribution is computed

by:

$$P_{k\text{NN}}(y \mid x) \propto \sum_{(k,v)\in\mathcal{D}} \mathbb{I}[v = y]\left(-d(\text{Enc}(k), \text{Enc}(x))\right).$$

Here, Enc is an encoder that maps a text into $\mathbb{R}^h$ and $d : \mathbb{R}^h \times \mathbb{R}^h \to \mathbb{R}$ is a distance function, where h is the hidden dimension. We follow Khandelwal et al. [2020] and use the output vector from the last layer of the transformers in the parametric LM as Enc, L2 distance as d, and an approximate nearest neighbor search using FAISS [Johnson et al., 2019, details in §4.4.4]. The final model takes the kNN-LM output and interpolates it with the output from the parametric LM:[7] $\lambda P_{\text{LM}}(y \mid x) + (1 - \lambda)P_{k\text{NN}}(y \mid x)$, where λ is a fixed hyperparameter between 0 and 1.

Future work can improve kNN-LM, e.g., by training the model to output a nonparametric distribution [Zhong et al., 2022b; Lan et al., 2023; Min et al., 2023b], by having a vocabulary-specific λ [Huang et al., 2023b], or by modeling λ as a function of the input x [He et al., 2021; Drozdov et al., 2022].

The retrieval-in-context language model (RIC-LM). As an alternative to kNN-LM, RIC-LM [Shi et al., 2024b; Ram et al., 2023] retrieves text blocks from a datastore and feeds them to the parametric LM in context. Specifically, given a dataset consisting of N tokens $c_1...c_N$, an index $\mathcal{D}$ is constructed by splitting the data into text blocks $b_1...b_M$, optionally with a sliding window. At test time, given an input prefix x, RIC-LM retrieves the most similar paragraph to the prefix $\hat{p} = \text{argmax}_{b\in\mathcal{D}} \text{sim}(b, x)$ and concatenates it to the prefix to produce $P_{\text{LM}}(y \mid \hat{b}, x)$. Here, sim is a function that computes a similarity score between two pieces of text; we use BM25 following Ram et al. [2023] who show that BM25 outperforms alternative dense retrieval methods.

Future work can improve RIC-LM, e.g., by using multiple text blocks through ensembling [Shi et al., 2024b] or reranking [Ram et al., 2023], by tuning the retrieval system [Shi et al., 2024b], or by training the LM to use retrieved blocks in context [Guu et al., 2020; Izacard et al., 2022b].

Comparison between *k*NN-LM and RIC-LM. The key difference between kNN-LM and RIC-LM lies in how the nonparametric component influences the output. In kNN-LM, it directly impacts the output distribution, while in RIC-LM, it indirectly influences the output by affecting the input to the parametric model. kNN-LM intuitively benefits more from a datastore as it provides direct influence to the output and relies less on the parametric component. Nonetheless, RIC-LM interacts more easily with a parametric model (i.e., it is applicable to a black-box LM) and offers better speed and memory efficiency (explored in Appendix C.2.2).

[7] While the encoder that outputs $P_{k\text{NN}}(y \mid x)$ and the parametric LM that outputs $P_{\text{LM}}(y \mid x)$ are based on the same transformer models in this case following Khandelwal et al. [2020], it is not a necessary condition. One of our ablations in §4.5.2 use different transformer models for the encoder and the parametric LM.

Empirical comparisons between kNN-LM and RIC-LM have been largely unexplored; in fact, we are unaware of such work. In our experiments (§4.5.2), we present a series of such comparisons, with varying sizes of the datastore, and with and without distribution shift.

Attribution and opt-out. Since elements in the datastore that contribute to the model prediction are transparent, both kNN-LM and RIC-LM offer inherent attributions. Moreover, data removed from the datastore is guaranteed not to contribute to any model predictions, allowing data owners to remove their data at the level of individual examples. Both are unique characteristics of nonparametric language models. While prior work studies post-hoc attribution to the data used for training model parameters [Koh and Liang, 2017; Han et al., 2023] and removing the effect of specific training examples from parameteric models [Cao and Yang, 2015; Jang et al., 2023b], they are arguably not fundamental due to lack of inherent guarantees, and are difficult to scale.

4.4.3 Building SILO

SILO is is built upon the general methodology of kNN-LM and RIC-LM. However, unlike prior work that uses the same data for learning model parameters and a nonparametric datastore, SILO uses distinct datasets for these two components.

The key idea behind SILO is to use low-risk data to estimate model parameters, and to use high-risk data only in a nonparametric datastore. This is based on the motivation that model parameters should be learned conservatively, since training data is difficult to remove or trace after model training is completed. In contrast, a nonparametric datastore offers greater flexibility, as it can be easily updated, grown, or filtered, supports data opt-out at the level of individual examples, and provides attributions for free to every model prediction. These functions enable adherence to data-use regulations (§4.2).

Training datasets. We train each of our LMs on one of the three datasets of OLC: PD data, PDSW data, and PDSWBY data. Each of the resulting models constitutes a different level of possible copyright infringement risk.

Datastore. We assume in-distribution data for each test domain is available at inference time, and construct a datastore for each domain (details in §4.4.4). Future work may investigate building a single datastore that includes all domains. These test-time datasets can be either in-domain or out-of-domain with respect to the data used to train model parameters.

4.4.4 Implementation Details

LM architecture and training details. We use 1.3B-parameter transformer LMs based on the LLaMA architecture [Touvron et al., 2023] as implemented in OpenLM.[8] Each model is trained with 128 A100 GPUs

[8] https://github.com/mlfoundations/openlm

across 16 nodes. Following Muennighoff et al. [2023], we train for multiple epochs in each dataset and perform early stopping. We train our PD, PDSW and PDSWBY models for 60B, 250B, and 350B tokens in total, respectively. More details are provided in Appendix C.1.

Domain re-weighting. Since the distribution of OLC is highly skewed (§4.3), we perform a simple upweighting scheme where we upsample all data that accounts for less than 5% by a factor of 3×, which we found to work well after a sweep of different settings. More sophisticated domain weighting strategies [Xie et al., 2023] are of interest but beyond the scope of this work.

Evaluation. We benchmark our models using language modeling perplexity on 14 domains that represent both in-domain and out-of-domain data with respect to different levels of OLC. This includes: public-domain legal documents from the **FreeLaw** Project subset of the the Pile [Gao et al., 2020], a held-out collection of books from the **Gutenberg** collection [Project Gutenberg], conversational text from the **Hacker News** subset of the Pile, held-out code files from the **Github** subset of the Pile (most of which are non-permissive licensed), scientific text of NIH Grant abstracts that are taken from the **NIH ExPorter** subset of the PILE, philosophy papers taken from the **PhilPapers** of the PILE, held-out English **Wikipedia** articles from the PILE, news articles from **CC-News** [Mackenzie et al., 2020], books from **BookCorpus2** which is an expanded version of Zhu et al. [2015], books from **Books3** by Presser [2020], random web-crawled pages from **OpenWebText2** [Gokaslan and Cohen, 2019; Gao et al., 2020], emails from the **Enron Emails** corpus [Klimt and Yang, 2004], **Amazon** product reviews from He and McAuley [2016], and finally clinical notes from **MIMIC-III** [Johnson et al., 2016] with personal identifiable information (PII) masked out. Our choice of domains highlights legal risks discussed in the earlier sections, e.g., CC-News, BookCorpus2, Books3 and Amazon reviews are mostly copyrighted, Github is mostly not permissively licensed,[9] and Enron Emails and MIMIC-III include private text. 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 and Auli, 2019; Khandelwal et al., 2020]. For MIMIC-III, which includes masked personally-identifiable information (PII), we filter out notes where more than 50% of tokens correspond to PII, and then exclude tokens corresponding to PII when computing perplexity.

Datastore. We construct an in-domain datastore for each test domain based on their training data. For datasets from the PILE, we consider 10% of the training data. For k**NN-LM**, each datastore consists of up to 1 billion h-dimensional vectors (h =2,048). We build an index for fast nearest neighbor search using FAISS [Johnson et al., 2019]. For **RIC-LM**, each datastore consists of text blocks with a length of 1,024 and a sliding window of 512. We use BM25 from Pyserini [Lin et al., 2021]. Appendix C.2.2 report ablations on different implementations of RIC-LM besides the method in §4.4.2. More details, statistics and hyperparameter values for the datastores are reported in §C.1.

[9] Kocetkov et al. [2023] estimates about 13% of the Github data is under MIT, Apache, and BSD.

Eval data	PD	PDSW	PDSWBY	Pythia
FreeLaw	5.3	5.7	6.5	5.6
Gutenberg	15.2	12.5	14.1	13.1
HackerNews	38.0	13.7	14.5	13.3
Github	13.5	2.7	2.8	2.4
NIH ExPorter	28.2	19.2	15.0	11.1
PhilPapers	31.7	17.6	15.0	12.7
Wikipedia	28.9	20.3	11.3	9.1
CC News	34.0	23.3	21.2	12.0
BookCorpus2	25.3	19.2	19.6	13.2
Books3	27.2	19.3	18.6	12.6
OpenWebText2	37.8	21.1	18.8	11.5
Enron Emails	18.6	13.2	13.5	6.9
Amazon	81.1	34.8	37.0	22.9
MIMIC-III	22.3	19.0	15.5	13.1
Average	29.1	17.3	16.0	11.4

Table 4.3: Perplexity (the lower the better) of the parametric-only SILO trained on PD, PDSW, and PDSWBY (without a datastore), compared to Pythia-1.4B, a model trained with similar amounts of compute but on mostly non-permissive data. We use ■, ■, and ■ to indicate text that is in-domain, out-of-domain, or out-of-domain but has relevant data in-domain (e.g., high-risk Github code vs. our permissive Github code). Reported on the test data; see Table C.3 for results on the validation data. **Our parametric LMs are competitive to Pythia in-domain but fall short out-of-domain.**

4.5 Experiments

We first evaluate the parametric-only component of SILO trained on the OPEN LICENSE CORPUS (§4.5.1), and then show the effect of adding a datastore that may contain high-risk text (§4.5.2). For all experiments, we use the 1.4B Pythia model [Biderman et al., 2023] as a baseline because it is trained with a similar amount of compute (data size and model parameters), but is trained on mostly high-risk data.[10]

4.5.1 Results: Parametric Component

Main results. Table 4.3 reports performance of our 1.3B base LMs trained on varying levels of permissively-licensed data—PD, PDSW, and PDSWBY—as well as Pythia. Overall, our LMs are competitive with Pythia despite using permissive data only. They are roughly equal quality on in-domain data, e.g., FreeLaw and Gutenberg, HackerNews in the case of PDSW and PDSWBY, and Wikipedia in the case of PDSWBY. Models trained on PDSW and PDSWBY are also close to Pythia on Github, likely because the permissively-licensed code data included in SW has a distribution that is sufficiently close to the distribution of the all Github code. The largest gaps occur on data that is in-domain for Pythia but out-of-domain for our model, e.g., news, books, OpenWebText, and emails, and Wikipedia in the case of models besides PDSWBY. This illustrates the

[10] We use the model checkpoint from https://huggingface.co/EleutherAI/pythia-1.4b-deduped-v0.

Eval data	SILO (PDSW)			Pythia
	Prm-only	kNN-LM	RIC-LM	Prm-only
Github	2.7	2.4 (-100%)	2.4 (-100%)	2.4
NIH ExPorter	19.2	15.0 (-52%)	18.5 (-9%)	11.1
Wikipedia	20.3	14.5 (-52%)	19.4 (-8%)	9.1
CC News	23.3	8.0 (-135%)	16.8 (-58%)	12.0
Books3	19.3	17.4 (-28%)	18.6 (-10%)	12.6
Enron Emails	13.2	5.9 (-116%)	9.9 (-68%)	6.9
Amazon	34.9	26.0 (-75%)	33.7 (-10%)	23.0
MIMIC-III	19.0	6.6 (-210%)	15.6 (-58%)	13.1
Average	19.0	12.0 (-91%)	16.9 (-27%)	11.3

Table 4.4: Perplexity (the lower the better) of parametric LMs (Prm-only), kNN-LM, and RIC-LM. % in parentheses indicate a reduction in the gap between the parametric-only SILO and Pythia. As in Table 4.3, ■ indicates in-domain; ■ indicates out-of-domain; ■ indicates out-of-domain but has relevant data in-domain, all with respect to the training data of the parametric LM. Reported on the test data; see Table C.4 for results on the validation data. See Table C.2 for the statistics of the datastore. **Adding a datastore, with kNN-LM, effectively reduces the gap between SILO and Pythia.**

extreme domain generalization challenge that is present when training on only permissive data, as we hint in §4.3.

More ablations, including the effect of upsampling low-resource data, and the effect of including and excluding explicit source code, are provided in §C.2.1.

4.5.2 Results: Adding the Nonparametric Component

Since building legally permissive LMs poses a challenge of extreme domain generalization, our next question is whether using an in-domain, nonparametric datastore can reduce the gap. We explore this question with our parametric LM trained on the PDSW subset of OLC; see Appendix C.2.2 for results of models trained on PD or PDSWBY. All models are evaluated on a subset of 8 out-of-domain datasets to the parametric model: Github, NIH ExPorter, Wikipedia, CC News, Books3, Enron Emails, Amazon, and MIMIC-III.

Main results. Table 4.4 shows adding the datastore with either kNN-LM- or RIC-LM-based retrieval improves performance over just using the parameteric component on all domains, but kNN-LM is more effective than RIC-LM. In most domains, kNN-LM reduces the gap between SILO and Pythia by more than 50% (on NIH ExPorter, Wikipedia, Amazon) or even outperforms Pythia (on Github, CC News, Enron Emails, MIMIC-III). Books3 is the domain with the least benefit, on which kNN-LM still reduces the gap by 28%.

Impact of scaling the datastore. Figure 4.3 demonstrates that both kNN-LM and RIC-LM-based retrieval consistently improves performance as the datastore size increases, with a strong log-linear trend. However,

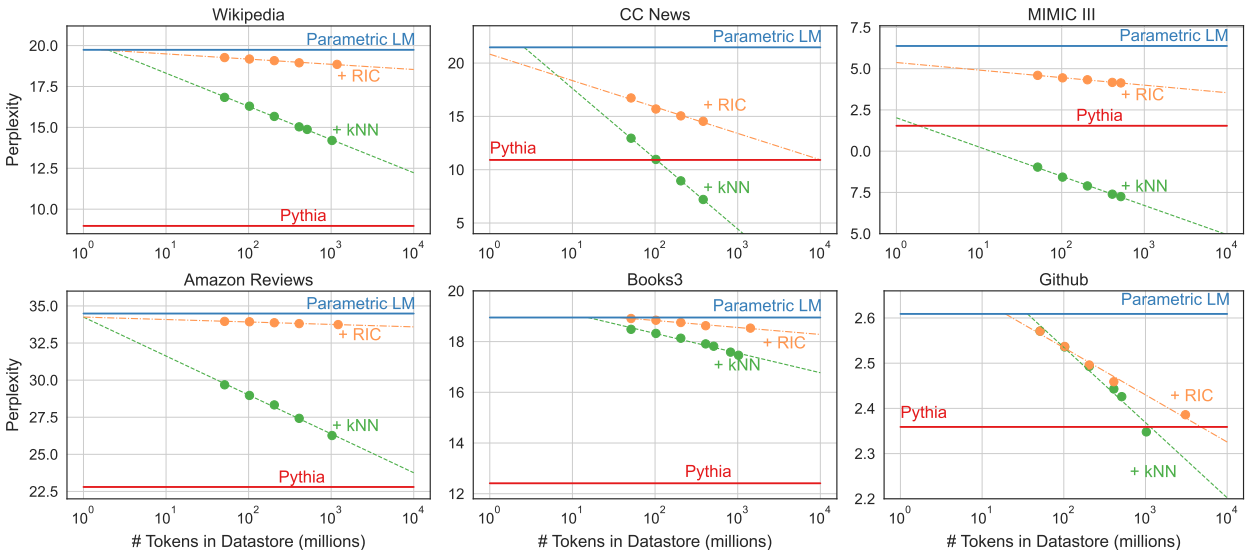

Figure 4.3: Impact of scaling the datastore of SILO (PDSW). Perplexity on random 128K tokens from the validation data reported. The rightmost dots for kNN-LM and RIC-LM in each figure correspond to the final models used in Table 4.4. **Scaling the test-time datastore consistently improves performance over all domains.**

kNN-LM improves performance more rapidly than RIC-LM does, consistently over all datasets. Extrapolating the trend suggests that, on the domains that SILO has not outperformed Pythia yet, scaling the datastore even further (with kNN-LM retrieval) may enable it to match Pythia.

Why does kNN-LM outperform RIC-LM? Our next question is why kNN-LM is better than RIC-LM—is it (a) because kNN-LM is better than RIC-LM in general, or (b) because kNN-LM generalizes out-of-domain better than RIC-LM does? Our further analysis in §C.2.2 (Figure C.2) reveals that it is both. With Pythia, where the test data is in-domain, while both kNN-LM and RIC-LM improve performance upon the parametric-only model, kNN-LM is overall better and scales better than RIC-LM, supporting (a). Both kNN-LM and RIC-LM improve performance more rapidly with SILO (where the test data is out-of-domain) than with Pythia, but this trend is much clearer with kNN-LM, supporting (b).

Where does the remaining gap come from? Even when scaling the datastore with kNN-LM, SILO lags behind Pythia on a few domains. Moreover, a Pythia-based kNN-LM outperforms our model since kNN-LM improves Pythia as well. There are two possible points of failure in our model for these cases: either the parametric component (which outputs P_{LM}) struggles out-of-domain, or the encoder (that outputs $P_{k\text{NN}}$) struggles out-of-domain. To better understand which part of the model contributes to the gap we observe, we vary SILO with different choices for the parametric component and the encoder. We compare replacing either the parametric component or the encoder with Pythia. This setup allows us to measure the effects of

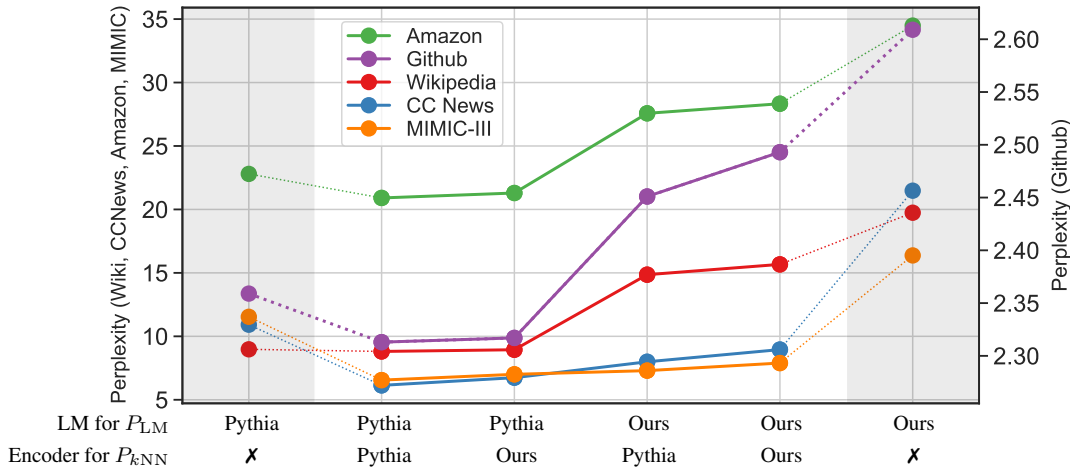

Figure 4.4: Impact of using different parameters on SILO. Perplexity on random 128K tokens from the validation data reported. The left-most and the right-most models are parametric models, and the other four models are kNN-LMs, using a datastore with 204.8 million tokens (20% of the datastore we use for the main experiments). *Ours* indicates our parametric model trained on the PDSW subset of OPEN LICENSE CORPUS. **Most of the performance degradation comes from using the out-of-domain parametric LM, rather than using the out-of-domain encoder.**

the out-of-domain nature of our parametric component (which is only trained on PDSW subset of OLC) in each of these components.

Results in Figure 4.4 reveal that most performance gaps come from the LM: performance improves significantly when the parametric component is replaced with Pythia, given a fixed encoder. In contrast, performance improvement is relatively marginal when the encoder is replaced with Pythia, given a fixed parametric component. These results indicate that the parametric component, which gives P_{LM}, is quite sensitive to domain shift, but the encoder, which provides the nonparametric distribution $P_{k\text{NN}}$, is fairly robust to extreme domain shift. This also explains why kNN-LM generalizes better than RIC-LM, since RIC-LM is bottlenecked by the parametric component.

In summary, our analysis highlights two promising directions to further reduce the gap:

1. Scaling the datastore beyond 1 billion tokens, e.g., at the scale of trillions of tokens as in Borgeaud et al. [2022], as demonstrated by Figure 4.3.

2. Improving the robustness of the model by improving nonparametric techniques or designing a model that only uses a nonparametric distribution [Min et al., 2023b], as demonstrated by Figure 4.4.

Comparison in runtime speed. Table C.8 in Appendix C.2.2 provides a comparison of the runtime speed of the parametric LM, RIC-LM, and kNN-LM. There is a strong tradeoff between performance and speed: both RIC-LM and kNN-LM are considerably slower than the parametric LM, and a larger datastore and

Eval	SILO (PDSW)			Pythia
	Prm-only	kNN-LM w/o HP	kNN-LM w/ HP	Prm-only
1	15.9	15.2	13.0	9.6
2	17.7	16.7	12.4	10.0
3	16.5	15.6	11.4	9.5
4	17.7	16.8	12.9	10.1
5	17.8	16.9	13.2	10.2
6	17.4	16.5	12.8	10.1
7	18.8	17.8	15.1	10.9
Avg	17.4	16.5	12.9	10.1

Table 4.5: The effect of data opt-out. Both kNN-LM methods use 1.024B-token on Books3. *w/ HP* and *w/o HP* indicate that the datastore includes or excludes Harry Potter books, respectively. The number (1 to 7) indicates a different book from the Harry Potter series used as the eval data; this eval book is not included in the datastore in any case. ■ indicates in-domain; ■ indicates out-of-domain.

more accurate nearest-neighbor search leads to better performance and slower inference. While the speed is heavily influenced by the hardware used for benchmarking and thus it is difficult to precisely quantify how much faster one method is compared to the other, this suggests that improving the runtime efficiency of nonparametric approaches is an important area of future work.

4.5.3 Examples of Data Attribution and Opt-Out

As discussed in §4.2, the design of SILO can better align with various data-use regulations by providing mechanisms for data attribution during inference and for data owners to remove their data from the model at any time. This section show examples of such capabilities.

Data opt-out. To showcase the impact of opt-out on model performance, we conduct experiments with J.K. Rowling's Harry Potter series. We first identify all seven Harry Potter books from the Books3 corpus of the Pile. For each book, we calculate the perplexity of SILO using two 1.024B token datastores on Books3, but one including the remaining six Harry Potter books and the other excluding any Harry Potter books. This experiment is to see whether excluding Harry Potter books from the former datastore can reduce the likelihood of generating the leave-out Harry Potter book.

Table 4.5 shows the results. SILO with Harry Potter books in the datastore effectively improves perplexity over all seven books, closing the gap between the PDSW model and Pythia. However, when the Harry Potter books are removed from the datastore, the perplexity gets worse, approaching that of the parametric-only LM. This illustrates that eliminating the effect of the Harry Potter books from the model substantially reduces the likelihood of generating the leave-out book.

Test Prefix 'I - what - *dragons*?' spluttered the Prime Minister. 'Yes, three,' said Fudge. 'And a sphinx. Well, good day to you.' The Prime Minister hoped beyond hope that dragons and sphinxes would be the worst of it, but no. Less than two years later, Fudge had erupted out of the fire yet again, this time with the news that there had been a mass breakout from **Test Continuation** **Azkaban**. 'A *mass* breakout?' the Prime Minister had repeated hoarsely. **Retrieved Prefix** 'D' you know Crouch, then?' said Harry. Sirius' face darkened. He suddenly looked as menacing as the night when Harry had first met him, the night when Harry had still believed Sirius to be a murderer. 'Oh, I know Crouch all right,' he said quietly. 'He was the one who gave me the order to be sent to **Retrieved Continuation** **Azkaban** - without a trial.'
Test Prefix Terror tore at Harry's heart... he had to get to Dumbledore and he had to catch Snape... somehow the two things were linked... he could reverse what had happened if he had them both together... Dumbledore could not have died... (...) Harry felt Greyback collapse against him; with a stupendous effort he pushed the werewolf off and onto the floor as a jet of **Test Continuation** **green** light came flying toward him; he ducked and ran, headfirst, into the fight. **Retrieved Prefix** Voldemort was ready. As Harry shouted, "Expelliarmus!" Voldemort cried, "Avada Kedavra!" A jet of **Retrieved Continuation** **green** light issued from Voldemort's wand just as a jet of red light blasted from Harry's ...

Table 4.6: Attribution examples on Harry Potter books. We show the top-1 retrieved context of SILO (PDSW). **Red underline text** indicates the next token that immediately follows the prefix. In both examples, the test data is from the sixth novel and the retrieved context is from the fourth novel in the Harry Potter series. In the series, *Azkaban* is the notorious wizarding prison, and the *green light* is a distinct characteristic of the Killing Curse, *Avada Kedavra*.

Attribution examples. To show the attribution feature of our model, Table 4.6 provides qualitative examples on the top-1 context retrieved by SILO. The model is able to assign a high probability to the ground truth token by retrieving highly relevant context. It achieves this by leveraging the unique characteristics of the text within the datastore, such as recognizing that *Azkaban* refers to the prison and *green light* is associated with the *Killing Curse* in the Harry Potter books.

More qualitative examples on Github, news and emails are illustrated in Table C.9 in Appendix C.2.2. They highlight that a nonparametric approach addresses specific legal risks that we have discussed earlier, e.g., it offers per-token attribution for free, and can provide a copyright notice when part of copyrighted text is being used for the probability distribution.

4.6 Discussion & Future Work

Our work suggests that it is possible to improve the tradeoff between legal risk and model performance when training LMs. Our approach provides new options for model designers to mitigate the legal risk of LMs, and empowers stakeholders to have more control over the data that drives these systems. We point out a number of rich areas for future work, beyond what was mentioned throughout the paper:

Addressing the limitations of SILO. SILO does not completely eliminate legal risk. Instead, it provides users more control over the model's generated content and functionalities to better align with legal regulations. For instance, SILO does not remove the need for obtaining permission to use copyrighted content in a datastore when providing attribution is not sufficient, but its opt-out capabilities can strengthen fair use

defense. Moreover, SILO does not prevent copying copyright content from a datastore, but it offers a way to prevent generating sensitive text [Huang et al., 2023a] or prevent copying the content verbatim. These functionalities increase the likelihood of a successful fair use defense if used appropriately.

Furthermore, while SILO mitigates copyright and privacy risks, it may exacerbate certain fairness issues, like toxicity towards marginalized groups and racial biases, especially due to the prevalence of older copyright-expired books in the training data. Exploring the balance between legal risk mitigation and fairness is an important future direction.

Finally, our study relies on explicit metadata to identify licenses, which may lead to underestimates of the amount and diversity of permissively licensed text actually available on the web. Future research may investigate *inferring* data licenses from documents in web crawl at scale, which may be an effective way to build more heterogeneous, permissively licensed corpora.

Introducing novel data licensing approaches. SILO introduces the possibility for data owners to set different levels of permissivity for learning parameters and for including in a nonparametric datastore. A data owner might choose to be more permissive about including data in the datastore due to its ease of removal, ensuring that the excluded data has no influence on model predictions anymore, and its ability to provide per-prediction attribution. Moreover, we envision that SILO could provide a path forward for data owners to get properly credited (or be paid directly) every time their data in a datastore contributes to a prediction. This is orthogonal to recent work that circumvented copyright issues by licensing out training data from data creators [Yu et al., 2023a].

Investigating other copyright risk mitigation strategies. It is critical to continue to develop new techniques that use copyrighted data while protecting the rights of data owners and subjects. In addition to nonparametric approaches, there are many other ways to achieve these goals. First, one could train LMs on copyrighted content but filter and guide their outputs towards text that is non-infringing [Henderson et al., 2023]. Second, training models with differential privacy [Dwork et al., 2006; Abadi et al., 2016] or near-access freeness [Vyas et al., 2023] may prevent them from regenerating individual details of copyright data. Finally, one could provide attributions for standard base LMs using post-hoc attribution methods, e.g., influence functions [Koh and Liang, 2017], rather than switching the model class to a retrieval-based model. All of these methods are complementary and orthogonal to our proposed approach.

Generalizing SILO as a modular language model. Our work is closely related to recent studies on modular LMs, which have specialized parameters (or *experts*) trained on different domains [Gururangan et al., 2022; Li et al., 2022; Gururangan et al., 2023], languages [Pfeiffer et al., 2020, 2022], or tasks [Chen et al., 2022c; Jang et al., 2023a]. Our work extends modular LMs to include nonparametric datastores, and focuses on *specializing* different parts of the model to low- and high-risk subsets of the training data. Legal risks may also be mitigated with a collection of parametric expert models that are specialized to low- and high-risk data.

Future work may explore this possibility as well as the usefulness of combining a nonparametric datastore with parametric experts.

Extending SILO to other modalities. While this work focuses on text-only models, similar methods to ours could apply to other domains and modalities. For instance, it might be possible to build permissive text-to-image generative models [Rombach et al., 2022] using compartmentalized public domain pre-training and retrieval augmentation [Chen et al., 2022a; Golatkar et al., 2023]. We believe such approaches are especially promising because there are many sources of public domain data in other modalities, e.g., images, speech, video, and more.

4.7 Summary

We introduce SILO, a language model that mitigates legal risk by learning parameters only on low-risk, permissively-licensed data (OPEN LICENSE CORPUS), and using an unrestricted nonparametric datastore during inference. Our approach allows the model designer to use high-risk data without training on it, supports sentence-level data attribution, and enables data produces to opt-out from the model by removing content from the datastore. Experiments on language modeling perplexity show that parametric-only SILO is competitive on domains covered by OPEN LICENSE CORPUS, but falls short out-of-domain when solely using the parametric component of the model, highlighting the challenge of extreme domain generalization. We then show that adding a nonparametric datastore to SILO (with kNN-LM retrieval) successfully addresses this challenge, significantly reducing the gap (or even outperforming) the Pythia baseline that is trained unrestrictedly. We show that scaling the datastore size is key to the success of the nonparametric approach, and that the encoder for a nonparametric distribution is significantly more robust to distribution shift than the parametric component. Our results point to a number of exciting future research directions to develop AI systems with mitigated legal risk.

Chapter 5

Conclusion & Future Work

Large language models (LMs), such as ChatGPT, continue to significantly transform the fields of NLP and AI. By training deep neural models, such as Transformers [Vaswani et al., 2017], with 100 billion or more parameters using extensive text data (e.g., 100 billion tokens or more), we have developed models capable of performing a wide range of NLP tasks without the need for task-specific architecture design or training. These models are already being widely integrated into user-facing production systems, including ChatGPT, Claude, and Bard.

This dissertation focused on (1) **understanding** how these models work and (2) **rethinking** how the next generation of models should use data at scale.

Chapter 2: Understanding Current Language Models. We discussed how to better understand current "black-box" LMs and demonstrating how far they can perform on these tasks, focusing on *in-context learning* (ICL).

1. We improved LMs' in-context learning (Section 2.3). We trained LMs with a multi-task objective on a large collection of tasks and evaluated them on a new task during inference. The key is to train LMs to condition on k task examples (as ICL typically does) *during training* and then, at inference time, condition on *new* examples about the given task [Min et al., 2022b]. This helps LMs learn to better use the given examples in order to activate required patterns.

2. We then challenged the widespread belief that ICL allows an LM to acquire new abilities on the fly with no training. We showed that LMs accurately perform tasks with ICL even when the given examples are incorrect (Min et al. [2022c]; Lyu et al. [2023]; Wang et al. [2023a]; Section 2.4). This indicates that LMs perform tasks by relying on patterns present in their training data, which can be activated in a specific way, rather than by obtaining a new ability on the fly from correctly paired examples.

3. Together, our work has led a significant body of follow-up work discovering unexpected behaviors of LMs [Madaan and Yazdanbakhsh, 2022; Wei et al., 2023b; Jang et al., 2022b; Wies et al., 2023; Lampinen et al., 2022; Pan et al., 2023; Kim et al., 2022; Schaeffer et al., 2023; Zhang et al., 2023b; She et al., 2023; Turpin et al., 2023] and has also inspired the development of improved models [Chung et al., 2022; Ivison et al., 2023], as discussed in Section 2.5.

Chapter 3: Nonparametric Language Models. We discussed nonparametric LMs, a new class of LMs that includes not only learned parameters but also a datastore, i.e., a massive collection of raw text documents.

1. We focused on retrieval augmentation, which first retrieves text pieces from a datastore and feeds them to the LM (Section 3.2). We introduced Dense Passage Retrieval (DPR) [Karpukhin et al., 2020], one of the first neural retrieval models that opened up a new era in retrieval augmentation. Its effect is further demonstrated by a NeurIPS Competition on open-domain question answering [Min et al., 2021a], and has led to a large body of subsequent work [Lewis et al., 2020b; Xiong et al., 2021a,b; Mao et al., 2020b; Ding et al., 2021; Gao and Callan, 2021; Izacard et al., 2022a; Zhan et al., 2021; Gao and Callan, 2022; Asai et al., 2021b; Ni et al., 2022; Ram et al., 2022].
2. We next focused on LMs with *a nonparametric softmax* (Section 3.3). Instead of outputting a categorical distribution over words, this method assigns scores to every word or phrase in the datastore as a nonparametric distribution, first proposed by Khandelwal et al. [2020]. We introduced NPM [Min et al., 2023b], one of the first work to train an LM with a nonparametric softmax. We showed that NPM outperforms alternatives on a range of tasks, is especially effective in handling rare concepts (such as rare entities and facts), and can grow and be updated by expanding and replacing the datastore.
3. We summarized open problems in nonparametric LMs (Section 3.4), including exploring alternative architectures that incorporate larger retrieval results more frequently and more efficiently (Table 3.9) and improving runtime efficiency.

Chapter 4: Responsible Language Models. We discussed how to leverage new functionalities of nonparametric LMs to advance responsible data use.

1. We claimed that using all available data on the web raises concerns related to crediting data creators and complying with legal constraints such as copyrights and the Right to be Forgotten (Section 4.2).
2. We introduced SILO, a nonparametric LM that was trained exclusively on permissively licensed data and uses copyrighted data in a datastore during inference (Min et al. [2024]; Section 4.3 and 4.4). SILO improves legal compliance because it (1) helps data creators receive appropriate credit for their contribution by providing data attributions for every model prediction and (2) enables support for data opt-out requests through the removal of data from the datastore at any time.

3. We demonstrated that parametric-only SILO struggles with extreme domain shift; however, adding an inference-time datastore using a nonparametric softmax addresses this challenge, allowing SILO to reduce the gap with an existing LM by 90% on average (Section 4.5).

Future Directions

Every future LM should be flexible, up-to-date, and performant with fewer parameters than they currently have. While we believe a nonparametric LM is one of the most promising directions to achieve this goal, there are still many open challenges, e.g., how to design these models to be more expressive and efficient (as explored Section 3.3), and whether they can ultimately open up a new dimension in *scaling*—a crucial aspect in the remarkable success of LMs. We believe future research in our field should tackle these challenges and, ultimately, expand model capabilities to address critical problems, both within natural language (e.g., factuality) and beyond (e.g., legality and privacy)—areas that current LMs are far from solving.

Expanding nonparametric LMs. One of the key questions is whether nonparametric LMs can provide an alternative pathway for scaling—the datastore size. To answer this question, we should scale the datastores to accommodate Internet-scale data consisting of trillions of words—the scale of modern LMs' training data. Currently, they accommodate relatively small-scale data (e.g., a few billions of words) and are thus limited to a specific domain (e.g., Wikipedia) in many cases. A key obstacle in scaling the datastore lies in scaling the nearest neighbor search algorithm to handle trillions of vectors while improving its runtime speed. Addressing this challenge necessitates not only NLP and machine learning research but also collaborative efforts with researchers specializing in computer systems, algorithms, and databases. Furthermore, nonparametric LMs should be expanded to handle a wider range of tasks. One notable area is higher-level reasoning, i.e., aggregating information from different parts of the data and generalizing to novel scenarios. By removing the need for memorizing the data in their parameters, nonparametric LMs can sharpen their focus on higher-level reasoning abilities, which was briefly explored in our work [Press et al., 2023].

Making LMs factual. One of the most critical issues with today's LMs lies in *factuality*, e.g., they often generate false information. This poses a serious concern since the pursuit of acquiring new information is one of the primary use cases for LMs, and the long-tail distribution of facts severely challenges LMs. There are three critical challenges. First, we should design models that improve the trade-off between *precision* (how accurate a response is) and *recall* (how much information a response covers). In particular, improving (or even defining) recall is largely underexplored—I intend to work on this aspect, with AmbigQA [Min et al., 2020, 2021b] as part of an initial effort. Second, we need evaluation methods that are more systematic, rigorous, and scalable. The challenge arises because a response often consists of a mixture of true and false pieces of information, each piece requiring costly verification. We built initial work toward this goal [Min et al.,

2023a], which we have made accessible through a user-friendly package and has gained wide adoption within just four months of its initial release [Ye et al., 2023c; Sun et al., 2023; Malaviya et al., 2023; Dhuliawala et al., 2023; Tian et al., 2023; Asai et al., 2024]. Finally, we should develop models that further relax the assumptions about the queries, e.g., the validity of presuppositions made within a question, which we initiated in [Yu et al., 2023b]. For instance, the question "How much EU's development fund is the UK contributing to?" presupposes that the UK is still a part of the EU, which should be challenged.

Optimizing LMs for societal objectives. As LMs become increasingly prevalent in real-world applications, researchers are responsible for studying their societal impact, identifying potential harms, and providing technical solutions, which I plan to actively work on. For instance, developing technical solutions to enhance LM compliance with legal constraints and enable proper attribution to data creators is very critical, as explored in [Min et al., 2024]. Moreover, it is critical to develop analysis techniques to understand LMs' gender, geographical, or political bias, e.g., how the training data and datastores affect model bias differently, and develop mitigation methods. Finally, we need more work in designing new algorithms that identify and mitigate privacy concerns in LMs and protect the interests of data creators, LM developers, and users. One of unexplored research questions is how to protect the data in nonparametric LMs' datastores in both training and inference, where techniques like federated learning and differential privacy can be leveraged.

References

Martin Abadi, Andy Chu, Ian Goodfellow, H Brendan McMahan, Ilya Mironov, Kunal Talwar, and Li Zhang. 2016. Deep learning with differential privacy. In *ACM SIGSAC*.

Armen Aghajanyan, Anchit Gupta, Akshat Shrivastava, Xilun Chen, Luke Zettlemoyer, and Sonal Gupta. 2021. Muppet: Massive multi-task representations with pre-finetuning. *arXiv preprint arXiv:2101.11038*.

Tiago A. Almeida, José María G. Hidalgo, and Akebo Yamakami. 2011. Contributions to the study of sms spam filtering: New collection and results. In *Proceedings of the 11th ACM Symposium on Document Engineering*.

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

Mikel Artetxe, Shruti Bhosale, Naman Goyal, Todor Mihaylov, Myle Ott, Sam Shleifer, Xi Victoria Lin, Jingfei Du, Srinivasan Iyer, Ramakanth Pasunuru, et al. 2021. Efficient large scale language modeling with mixtures of experts. *arXiv preprint arXiv:2112.10684*.

Mikel Artetxe, Jingfei Du, Naman Goyal, Luke Zettlemoyer, and Ves Stoyanov. 2022. On the role of bidirectionality in language model pre-training. In *Proceedings of Empirical Methods in Natural Language Processing*.

ArXiv. 2023. arxiv dataset.

Akari Asai, Kazuma Hashimoto, Hannaneh Hajishirzi, Richard Socher, and Caiming Xiong. 2020. Learning to retrieve reasoning paths over Wikipedia graph for question answering. In *Proceedings of the International Conference on Learning Representations*.

Akari Asai, Jungo Kasai, Jonathan H. Clark, Kenton Lee, Eunsol Choi, and Hannaneh Hajishirzi. 2021a. XOR QA: Cross-lingual open-retrieval question answering. In *Conference of the North American Chapter of the Association for Computational Linguistics*.

Akari Asai, Sewon Min, Zexuan Zhong, and Danqi Chen. 2023. ACL 2023 Tutorial: Retrieval-based language models and applications. In *ACL*.

Akari Asai, Zeqiu Wu, Yizhong Wang, Avirup Sil, and Hannaneh Hajishirzi. 2024. Self-rag: Learning to retrieve, generate, and critique through self-reflection. In *Proceedings of the International Conference on Learning Representations*.

Akari Asai, Xinyan Yu, Jungo Kasai, and Hannaneh Hajishirzi. 2021b. One question answering model for many languages with cross-lingual dense passage retrieval. In *NeurIPS*.

Bogdan Babych and Anthony F. Hartley. 2003. Improving machine translation quality with automatic named entity recognition. *Proceedings of the International EAMT workshop on MT and other Language Technology Tools, Improving MT through other Language Technology Tools Resources and Tools for Building MT*.

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

Jack Bandy and Nicholas Vincent. 2021. Addressing "documentation debt"' in machine learning: A retrospective datasheet for BookCorpus. In *Proceedings of Advances in Neural Information Processing Systems*.

Roy Bar-Haim, Ido Dagan, Bill Dolan, Lisa Ferro, Danilo Giampiccolo, Bernardo Magnini, and Idan Szpektor. 2006. The second pascal recognising textual entailment challenge. In *Proceedings of the second PASCAL challenges workshop on recognising textual entailment*.

Francesco Barbieri, Jose Camacho-Collados, Luis Espinosa Anke, and Leonardo Neves. 2020. TweetEval: Unified benchmark and comparative evaluation for tweet classification. In *Findings of the Association for Computational Linguistics: EMNLP 2020*.

Petr Baudiš and Jan Šedivỳ. 2015. Modeling of the question answering task in the yodaqa system. In *International Conference of the Cross-Language Evaluation Forum for European Languages*, pages 222–228. Springer.

Luisa Bentivogli, Peter Clark, Ido Dagan, and Danilo Giampiccolo. 2009. The fifth pascal recognizing textual entailment challenge. In *TAC*.

Jonathan Berant, Andrew Chou, Roy Frostig, and Percy Liang. 2013. Semantic parsing on Freebase from question-answer pairs. In *Proceedings of Empirical Methods in Natural Language Processing*.

Chandra Bhagavatula, Ronan Le Bras, Chaitanya Malaviya, Keisuke Sakaguchi, Ari Holtzman, Hannah Rashkin, Doug Downey, Wen tau Yih, and Yejin Choi. 2020. Abductive commonsense reasoning. In *ICLR*.

Stella Biderman, Hailey Schoelkopf, Quentin Anthony, Herbie Bradley, Kyle O'Brien, Eric Hallahan, Mohammad Aflah Khan, Shivanshu Purohit, USVSN Sai Prashanth, Edward Raff, et al. 2023. Pythia: A suite for analyzing large language models across training and scaling. In *Proceedings of the International Conference on Learning Representations*.

Yonatan Bisk, Rowan Zellers, Ronan Le Bras, Jianfeng Gao, and Yejin Choi. 2020. Piqa: Reasoning about physical commonsense in natural language. In *AAAI*.

Sid Black, Stella Biderman, Eric Hallahan, Quentin Anthony, Leo Gao, Laurence Golding, Horace He, Connor Leahy, Kyle McDonell, Jason Phang, Michael Pieler, USVSN Sai Prashanth, Shivanshu Purohit, Laria Reynolds, Jonathan Tow, Ben Wang, and Samuel Weinbach. 2022. GPT-NeoX-20B: An open-source autoregressive language model. *arXiv preprint arXiv:2204.06745*.

Michael Boratko, Xiang Li, Tim O'Gorman, Rajarshi Das, Dan Le, and Andrew McCallum. 2020. ProtoQA: A question answering dataset for prototypical common-sense reasoning. In *EMNLP*.

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 the International Conference of Machine Learning*.

Lucas Bourtoule, Varun Chandrasekaran, Christopher A. Choquette-Choo, Hengrui Jia, Adelin Travers, Baiwu Zhang, David Lie, and Nicolas Papernot. 2020. Machine unlearning.

Blake Brittain. 2023. U.S. copyright office says some AI-assisted works may be copyrighted. *Reuters*.

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. 2020a. Language models are few-shot learners. In *NeurIPS*.

Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, et al. 2020b. Language models are few-shot learners. In *NeurIPS*.

Collin Burns, Hao-Tong Ye, Dan Klein, and Jacob Steinhardt. 2022. Discovering latent knowledge in language models without supervision. *arXiv*.

Yinzhi Cao and Junfeng Yang. 2015. Towards making systems forget with machine unlearning. In *2015 IEEE symposium on security and privacy*, pages 463–480. IEEE.

Nicholas Carlini, Daphne Ippolito, Matthew Jagielski, Katherine Lee, Florian Tramer, and Chiyuan Zhang. 2023. Quantifying memorization across neural language models. In *Proceedings of the International Conference on Learning Representations*.

Nicholas Carlini, Florian Tramer, Eric Wallace, Matthew Jagielski, Ariel Herbert-Voss, Katherine Lee, Adam Roberts, Tom B Brown, Dawn Song, Ulfar Erlingsson, et al. 2021. Extracting training data from large language models. In *USENIX Security Symposium*.

Caselaw Access Project. 2018. Caselaw access project.

Stefano Ceri, Alessandro Bozzon, Marco Brambilla, Emanuele Della Valle, Piero Fraternali, Silvia Quarteroni, Stefano Ceri, Alessandro Bozzon, Marco Brambilla, Emanuele Della Valle, et al. 2013. An introduction to information retrieval. *Web information retrieval*, pages 3–11.

Kent K Chang, Mackenzie Cramer, Sandeep Soni, and David Bamman. 2023. Speak, memory: An archaeology of books known to ChatGPT/GPT-4. *arXiv preprint arXiv:2305.00118*.

Ankush Chatterjee, Kedhar Nath Narahari, Meghana Joshi, and Puneet Agrawal. 2019. SemEval-2019 task 3: EmoContext contextual emotion detection in text. In *Proceedings of the 13th International Workshop on Semantic Evaluation*.

Danqi Chen, Adam Fisch, Jason Weston, and Antoine Bordes. 2017. Reading Wikipedia to answer open-domain questions. In *Proceedings of the Association for Computational Linguistics*.

Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde de Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, et al. 2021. Evaluating large language models trained on code. *arXiv preprint arXiv:2107.03374*.

Michael Chen, Mike D'Arcy, Alisa Liu, Jared Fernandez, and Doug Downey. 2019. CODAH: An adversarially-authored question answering dataset for common sense. In *Proceedings of the 3rd Workshop on Evaluating Vector Space Representations for NLP*.

Wenhu Chen, Hexiang Hu, Chitwan Saharia, and William W Cohen. 2022a. Re-Imagen: Retrieval-augmented text-to-image generator. In *NeurIPS*.

Wenhu Chen, Hongmin Wang, Jianshu Chen, Yunkai Zhang, Hong Wang, Shiyang Li, Xiyou Zhou, and William Yang Wang. 2020. Tabfact: A large-scale dataset for table-based fact verification. In *ICLR*.

Wenlin Chen, David Grangier, and Michael Auli. 2016. Strategies for training large vocabulary neural language models. In *Proceedings of the Association for Computational Linguistics*.

Yanda Chen, Ruiqi Zhong, Sheng Zha, George Karypis, and He He. 2022b. Meta-learning via language model in-context tuning. In *ACL*.

Zitian Chen, Yikang Shen, Mingyu Ding, Zhenfang Chen, Hengshuang Zhao, Erik Learned-Miller, and Chuang Gan. 2022c. Mod-squad: Designing mixture of experts as modular multi-task learners.

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 preprint arXiv:2204.02311*.

Hyung Won Chung, Le Hou, Shayne Longpre, Barret Zoph, Yi Tay, William Fedus, Eric Li, Xuezhi Wang, Mostafa Dehghani, Siddhartha Brahma, et al. 2022. Scaling instruction-finetuned language models. *arXiv*.

Christopher Clark, Kenton Lee, Ming-Wei Chang, Tom Kwiatkowski, Michael Collins, and Kristina Toutanova. 2019. BoolQ: Exploring the surprising difficulty of natural yes/no questions. In *NAACL-HLT*.

Jonathan H. Clark, Eunsol Choi, Michael Collins, Dan Garrette, Tom Kwiatkowski, Vitaly Nikolaev, and Jennimaria Palomaki. 2020a. Tydi qa: A benchmark for information-seeking question answering in typologically diverse languages. *Transactions of the Association for Computational Linguistics*.

Kevin Clark, Minh-Thang Luong, Quoc V Le, and Christopher D Manning. 2020b. Electra: Pre-training text encoders as discriminators rather than generators. In *Proceedings of the International Conference on Learning Representations*.

Peter Clark, Isaac Cowhey, Oren Etzioni, Tushar Khot, Ashish Sabharwal, Carissa Schoenick, and Oyvind Tafjord. 2018. Think you have solved question answering? try arc, the ai2 reasoning challenge. *arXiv preprint arXiv:1803.05457*.

Alexis Conneau, Kartikay Khandelwal, Naman Goyal, Vishrav Chaudhary, Guillaume Wenzek, Francisco Guzmán, Edouard Grave, Myle Ott, Luke Zettlemoyer, and Veselin Stoyanov. 2020. Unsupervised cross-lingual representation learning at scale. In *Proceedings of the Association for Computational Linguistics*.

Ido Dagan, Oren Glickman, and Bernardo Magnini. 2005. The pascal recognising textual entailment challenge. In *Machine Learning Challenges Workshop*.

Pradeep Dasigi, Nelson F. Liu, Ana Marasović, Noah A. Smith, and Matt Gardner. 2019. Quoref: A reading comprehension dataset with questions requiring coreferential reasoning. In *EMNLP*.

Thomas Davidson, Dana Warmsley, Michael Macy, and Ingmar Weber. 2017. Automated hate speech detection and the problem of offensive language. In *Proceedings of the 11th International AAAI Conference on Web and Social Media*.

Ona de Gibert, Naiara Perez, Aitor García-Pablos, and Montse Cuadros. 2018. Hate Speech Dataset from a White Supremacy Forum. In *Proceedings of the 2nd Workshop on Abusive Language Online (ALW2)*.

Marie-Catherine de Marneffe, Mandy Simons, and Judith Tonhauser. 2019. The commitmentbank: Investigating projection in naturally occurring discourse. *Proceedings of Sinn und Bedeutung*.

Gerrit De Vynck. 2023. ChatGPT maker OpenAI faces a lawsuit over how it used people's data.

Scott Deerwester, Susan T Dumais, George W Furnas, Thomas K Landauer, and Richard Harshman. 1990. Indexing by latent semantic analysis. *Journal of the American society for information science*, 41(6):391–407.

Xiang Deng, Yu Su, Alyssa Lees, You Wu, Cong Yu, and Huan Sun. 2021. ReasonBERT: Pre-trained to reason with distant supervision. In *Proceedings of Empirical Methods in Natural Language Processing*.

Tim Dettmers, Mike Lewis, Sam Shleifer, and Luke Zettlemoyer. 2022. 8-bit optimizers via block-wise quantization. In *ICLR*.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Conference of the North American Chapter of the Association for Computational Linguistics*.

Bhuwan Dhingra, Jeremy R Cole, Julian Martin Eisenschlos, Daniel Gillick, Jacob Eisenstein, and William W Cohen. 2022. Time-aware language models as temporal knowledge bases. *Transactions of the 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*.

T. Diggelmann, Jordan L. Boyd-Graber, Jannis Bulian, Massimiliano Ciaramita, and Markus Leippold. 2020. Climate-fever: A dataset for verification of real-world climate claims. *ArXiv*.

Yingqi Ding, Qu Yuchen, Jing Liu, Kai Liu, Ruiyang Ren, Xin Zhao, Daxiang Dong, Hua Wu, and Haifeng Wang. 2021. Rocketqa: An optimized training approach to dense passage retrieval for open-domain question answering. In *NAACL-HLT*.

Jesse Dodge, Maarten Sap, Ana Marasović, William Agnew, Gabriel Ilharco, Dirk Groeneveld, Margaret Mitchell, and Matt Gardner. 2021. Documenting large webtext corpora: A case study on the colossal clean crawled corpus. In *Proceedings of Empirical Methods in Natural Language Processing*.

William B. Dolan and Chris Brockett. 2005. Automatically constructing a corpus of sentential paraphrases. In *Proceedings of the Third International Workshop on Paraphrasing (IWP2005)*.

Andrew Drozdov, Shufan Wang, Razieh Rahimi, Andrew McCallum, Hamed Zamani, and Mo hit Iyyer. 2022. You can't pick your neighbors, or can you? when and how to rely on retrieval in the kNN-LM. *arXiv preprint arXiv:2210.15859*.

Dheeru Dua, Yizhong Wang, Pradeep Dasigi, Gabriel Stanovsky, Sameer Singh, and Matt Gardner. 2019. DROP: A reading comprehension benchmark requiring discrete reasoning over paragraphs. In *NAACL*.

Matthew Dunn, Levent Sagun, Mike Higgins, V. U. Güney, Volkan Cirik, and Kyunghyun Cho. 2017. Searchqa: A new q&a dataset augmented with context from a search engine. *arXiv preprint arXiv:1704.05179*.

Cynthia Dwork, Frank McSherry, Kobbi Nissim, and Adam Smith. 2006. Calibrating noise to sensitivity in private data analysis. In *Theory of Cryptography*.

Avia Efrat and Omer Levy. 2020. The turking test: Can language models understand instructions? *arXiv preprint arXiv:2010.11982*.

Hady Elsahar, Pavlos Vougiouklis, Arslen Remaci, Christophe Gravier, Jonathon Hare, Frederique Laforest, and Elena Simperl. 2018a. T-REx: A large scale alignment of natural language with knowledge base triples. In *LREC*.

Hady Elsahar, Pavlos Vougiouklis, Arslen Remaci, Christophe Gravier, Jonathon Hare, Frederique Laforest, and Elena Simperl. 2018b. T-rex: A large scale alignment of natural language with knowledge base triples. In *Proceedings of the International Conference on Language Resources and Evaluation*.

Theodoros Evgeniou and Massimiliano Pontil. 2004. Regularized multi–task learning. In *Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining*.

Manaal Faruqui and Dipanjan Das. 2018. Identifying well-formed natural language questions. In *EMNLP*.

Vitaly Feldman and Chiyuan Zhang. 2020. What neural networks memorize and why: Discovering the long tail via influence estimation. *Proceedings of Advances in Neural Information Processing Systems*.

Chelsea Finn, Pieter Abbeel, and Sergey Levine. 2017. Model-agnostic meta-learning for fast adaptation of deep networks. In *ICML*.

William T Freeman, Thouis R Jones, and Egon C Pasztor. 2002. Example-based super-resolution. *IEEE Computer graphics and Applications*.

Daniel Fried, Armen Aghajanyan, Jessy Lin, Sida Wang, Eric Wallace, Freda Shi, Ruiqi Zhong, Scott Yih, Luke Zettlemoyer, and Mike Lewis. 2023. Incoder: A generative model for code infilling and synthesis. In *Proceedings of the International Conference on Learning Representations*.

Leo Gao, Stella Biderman, Sid Black, Laurence Golding, Travis Hoppe, Charles Foster, Jason Phang, Horace He, Anish Thite, Noa Nabeshima, et al. 2020. The Pile: An 800GB dataset of diverse text for language modeling. *arXiv preprint arXiv:2101.00027*.

Luyu Gao and Jamie Callan. 2021. Condenser: a pre-training architecture for dense retrieval. In *EMNLP*.

Luyu Gao and Jamie Callan. 2022. Unsupervised corpus aware language model pre-training for dense passage retrieval. In *ACL*.

Luyu Gao, Zhuyun Dai, Panupong Pasupat, Anthony Chen, Arun Tejasvi Chaganty, Yicheng Fan, Vincent Y Zhao, Ni Lao, Hongrae Lee, Da-Cheng Juan, et al. 2022. Attributed text generation via post-hoc research and revision. *arXiv preprint arXiv:2210.08726*.

Tianyu Gao, Adam Fisch, and Danqi Chen. 2021. Making pre-trained language models better few-shot learners. In *ACL*.

Tiezheng Ge, Kaiming He, Qifa Ke, and Jian Sun. 2014. Optimized product quantization. *IEEE Transactions on Pattern Analysis and Machine Intelligence*.

David Gershgorn. 2021. Github's automatic coding tool rests on untested legal ground.

Danilo Giampiccolo, Bernardo Magnini, Ido Dagan, and Bill Dolan. 2007. The third pascal recognizing textual entailment challenge. In *Proceedings of the ACL-PASCAL workshop on textual entailment and paraphrasing*.

Daniel Gillick, Sayali Kulkarni, Larry Lansing, Alessandro Presta, Jason Baldridge, Eugene Ie, and Diego Garcia-Olano. 2019. Learning dense representations for entity retrieval. In *Conference on Computational Natural Language Learning*.

Aaron Gokaslan and Vanya Cohen. 2019. OpenWebText corpus. `http://Skylion007.github.io/`.

Aditya Golatkar, Alessandro Achille, Ashwin Swaminathan, and Stefano Soatto. 2023. Training data protection with compositional diffusion models.

Andrew Gordon, Zornitsa Kozareva, and Melissa Roemmele. 2012. SemEval-2012 task 7: Choice of plausible alternatives: An evaluation of commonsense causal reasoning. In *The First Joint Conference on Lexical and Computational Semantics (SemEval)*.

Édouard Grave, Armand Joulin, Moustapha Cissé, David Grangier, and Hervé Jégou. 2017. Efficient softmax approximation for GPUs. In *Proceedings of the International Conference of Machine Learning*.

Dirk Groeneveld. 2023. The big friendly filter. `https://github.com/allenai/bff`.

Jiatao Gu, Yong Wang, Kyunghyun Cho, and Victor OK Li. 2018. Search engine guided neural machine translation. In *Association for the Advancement of Artificial Intelligence*.

Yuxian Gu, Li Dong, Furu Wei, and Minlie Huang. 2023. Pre-training to learn in context. In *ACL*.

Ruiqi Guo, Philip Sun, Erik Lindgren, Quan Geng, David Simcha, Felix Chern, and Sanjiv Kumar. 2020. Accelerating large-scale inference with anisotropic vector quantization. In *Proceedings of the International Conference of Machine Learning*.

Harsha Gurulingappa, Abdul Mateen Rajput, Angus Roberts, Juliane Fluck, Martin Hofmann-Apitius, and Luca Toldo. 2012. Development of a benchmark corpus to support the automatic extraction of drug-related adverse effects from medical case reports. *Journal of Biomedical Informatics*.

Suchin Gururangan, Mike Lewis, Ari Holtzman, Noah A. Smith, and Luke Zettlemoyer. 2022. DEMix layers: Disentangling domains for modular language modeling. In *Conference of the North American Chapter of the Association for Computational Linguistics*.

Suchin Gururangan, Margaret Li, Mike Lewis, Weijia Shi, Tim Althoff, Noah A. Smith, and Luke Zettlemoyer. 2023. Scaling expert language models with unsupervised domain discovery.

Kelvin Guu, Kenton Lee, Zora Tung, Panupong Pasupat, and Ming-Wei Chang. 2020. Realm: Retrieval-augmented language model pre-training. In *ICML*.

Danny Halawi, Jean-Stanislas Denain, and Jacob Steinhardt. 2023. Overthinking the truth: Understanding how language models process false demonstrations. *arXiv*.

Xiaochuang Han, Daniel Simig, Todor Mihaylov, Yulia Tsvetkov, Asli Celikyilmaz, and Tianlu Wang. 2023. Understanding in-context learning via supportive pretraining data. In *Proceedings of the Association for Computational Linguistics*.

Ahmed Hassan, Haytham Fahmy, and Hany Hassan. 2007. Improving named entity translation by exploiting comparable and parallel corpora. In *Proceedings of the International Workshop on Acquisition and Management of Multilingual Lexicons*.

Junxian He, Graham Neubig, and Taylor Berg-Kirkpatrick. 2021. Efficient nearest neighbor language models. In *Proceedings of Empirical Methods in Natural Language Processing*.

Luheng He, Mike Lewis, and Luke Zettlemoyer. 2015. Question-answer driven semantic role labeling: Using natural language to annotate natural language. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*.

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

Matthew Henderson, Rami Al-Rfou, Brian Strope, Yun-hsuan Sung, László Lukács, Ruiqi Guo, Sanjiv Kumar, Balint Miklos, and Ray Kurzweil. 2017. Efficient natural language response suggestion for smart reply. *ArXiv*, abs/1705.00652.

Peter Henderson, Mark Krass, Lucia Zheng, Neel Guha, Christopher D Manning, Dan Jurafsky, and Daniel Ho. 2022. Pile of Law: Learning responsible data filtering from the law and a 256GB open-source legal dataset. *Proceedings of Advances in Neural Information Processing Systems*.

Peter Henderson, Xuechen Li, Dan Jurafsky, Tatsunori Hashimoto, Mark A Lemley, and Percy Liang. 2023. Foundation models and fair use. *arXiv preprint arXiv:2303.15715*.

Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. 2021. Measuring mathematical problem solving with the math dataset. *Proceedings of Advances in Neural Information Processing Systems*.

Johannes Hoffart, Mohamed Amir Yosef, Ilaria Bordino, Hagen Fürstenau, Manfred Pinkal, Marc Spaniol, Bilyana Taneva, Stefan Thater, and Gerhard Weikum. 2011. Robust disambiguation of named entities in text. In *EMNLP*.

Jordan Hoffmann, Sebastian Borgeaud, Arthur Mensch, Elena Buchatskaya, Trevor Cai, Eliza Rutherford, Diego de Las Casas, Lisa Anne Hendricks, Johannes Welbl, Aidan Clark, et al. 2022. Training compute-optimal large language models. *arXiv preprint arXiv:2203.15556*.

Myles Hollander, Douglas A Wolfe, and Eric Chicken. 2013. *Nonparametric statistical methods*. John Wiley & Sons.

Ari Holtzman, Peter West, Vered Schwartz, Yejin Choi, and Luke Zettlemoyer. 2021. Surface form competition: Why the highest probability answer isn't always right. In *EMNLP*.

Eduard Hovy, Laurie Gerber, Ulf Hermjakob, Chin-Yew Lin, and Deepak Ravichandran. 2001. Toward semantics-based answer pinpointing. In *Proceedings of the First International Conference on Human Language Technology Research*.

Minqing Hu and Bing Liu. 2004. Mining and summarizing customer reviews. In *Proceedings of the Tenth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*.

Lifu Huang, Ronan Le Bras, Chandra Bhagavatula, and Yejin Choi. 2019. Cosmos QA: Machine reading comprehension with contextual commonsense reasoning. In *EMNLP*.

Po-Sen Huang, Xiaodong He, Jianfeng Gao, Li Deng, Alex Acero, and Larry Heck. 2013. Learning deep structured semantic models for Web search using clickthrough data. In *Proceedings of the ACM International Conference on Information and Knowledge Management*, pages 2333–2338.

Yangsibo Huang, Samyak Gupta, Zexuan Zhong, Kai Li, and Danqi Chen. 2023a. Privacy implications of retrieval-based language models. *arXiv preprint arXiv:2305.14888*.

Yangsibo Huang, Daogao Liu, Zexuan Zhong, Weijia Shi, and Yin Tat Lee. 2023b. kNN-Adapter: Efficient domain adaptation for black-box language models. *arXiv preprint arXiv:2302.10879*.

Samuel Humeau, Kurt Shuster, Marie-Anne Lachaux, and Jason Weston. 2020. Poly-encoders: Architectures and pre-training strategies for fast and accurate multi-sentence scoring. In *Proceedings of the International Conference on Learning Representations*.

Hamish Ivison, Yizhong Wang, Valentina Pyatkin, Nathan Lambert, Matthew Peters, Pradeep Dasigi, Joel Jang, David Wadden, Noah A Smith, Iz Beltagy, et al. 2023. Camels in a changing climate: Enhancing lm adaptation with tulu 2. *arXiv preprint arXiv:2311.10702*.

Srinivasan Iyer, Xi Victoria Lin, Ramakanth Pasunuru, Todor Mihaylov, Daniel Simig, Ping Yu, Kurt Shuster, Tianlu Wang, Qing Liu, Punit Singh Koura, et al. 2022. Opt-iml: Scaling language model instruction meta learning through the lens of generalization. *arXiv preprint arXiv:2212.12017*.

Srinivasan Iyer, Sewon Min, Yashar Mehdad, and Scott Wen-tau Yih. 2021. ReConsider: Re-ranking using span-focused cross-attention for open domain question answering. In *NAACL-HLT*.

Gautier Izacard, Mathilde Caron, Lucas Hosseini, Sebastian Riedel, Piotr Bojanowski, Armand Joulin, and Edouard Grave. 2022a. Unsupervised dense information retrieval with contrastive learning. *TMLR*.

Gautier Izacard and Edouard Grave. 2020. Leveraging passage retrieval with generative models for open domain question answering. In *EACL*.

Gautier Izacard, Patrick Lewis, Maria Lomeli, Lucas Hosseini, Fabio Petroni, Timo Schick, Jane Dwivedi-Yu, Armand Joulin, Sebastian Riedel, and Edouard Grave. 2022b. Atlas: Few-shot learning with retrieval augmented language models. *arXiv preprint arXiv:2208.03299*.

Gautier Izacard, Fabio Petroni, Lucas Hosseini, Nicola De Cao, Sebastian Riedel, and Edouard Grave. 2020. A memory efficient baseline for open domain question answering. *arXiv preprint arXiv:2012.15156*.

Joel Jang, Seungone Kim, Seonghyeon Ye, Doyoung Kim, Lajanugen Logeswaran, Moontae Lee, Kyungjae Lee, and Minjoon Seo. 2023a. Exploring the benefits of training expert language models over instruction tuning. In *Proceedings of the International Conference of Machine Learning*.

Joel Jang, Seonghyeon Ye, Changho Lee, Sohee Yang, Joongbo Shin, Janghoon Han, Gyeonghun Kim, and Minjoon Seo. 2022a. Temporalwiki: A lifelong benchmark for training and evaluating ever-evolving language models. In *Proceedings of Empirical Methods in Natural Language Processing*.

Joel Jang, Seonghyeon Ye, and Minjoon Seo. 2022b. Can large language models truly understand prompts? a case study with negated prompts. In *NeurIPS 2022 Workshop on Transfer Learning for NLP (TL4NLP)*.

Joel Jang, Dongkeun Yoon, Sohee Yang, Sungmin Cha, Moontae Lee, Lajanugen Logeswaran, and Minjoon Seo. 2023b. Knowledge unlearning for mitigating privacy risks in language models. *Proceedings of the Association for Computational Linguistics*.

Herve Jegou, Matthijs Douze, and Cordelia Schmid. 2010. Product quantization for nearest neighbor search. *IEEE transactions on pattern analysis and machine intelligence*, 33(1):117–128.

Kelvin Jiang, Dekun Wu, and Hui Jiang. 2019. FreebaseQA: A new factoid QA data set matching trivia-style question-answer pairs with Freebase. In *NAACL-HLT*.

J.L. et al. v. Alphabet Inc. 2023. Case 3:23-cv-03416, N.D. Cal.

Alistair EW Johnson, Tom J Pollard, Lu Shen, Li-wei H Lehman, Mengling Feng, Mohammad Ghassemi, Benjamin Moody, Peter Szolovits, Leo Anthony Celi, and Roger G Mark. 2016. MIMIC-III, a freely accessible critical care database. *Scientific data*.

Jeff Johnson, Matthijs Douze, and Hervé Jégou. 2019. Billion-scale similarity search with GPUs. *IEEE Transactions on Big Data*.

Mandar Joshi, Danqi Chen, Yinhan Liu, Daniel S Weld, Luke Zettlemoyer, and Omer Levy. 2020. Spanbert: Improving pre-training by representing and predicting spans. *Transactions of the Association for Computational Linguistics*.

Mandar Joshi, Eunsol Choi, Daniel Weld, and Luke Zettlemoyer. 2017. TriviaQA: A large scale distantly supervised challenge dataset for reading comprehension. In *Proceedings of the Association for Computational Linguistics*.

Jan-Christoph Kalo and Leandra Fichtel. 2022. Kamel: Knowledge analysis with multitoken entities in language models. In *Proceedings of the Conference on Automated Knowledge Base Construction*.

Nikhil Kandpal, Haikang Deng, Adam Roberts, Eric Wallace, and Colin Raffel. 2022a. Large language models struggle to learn long-tail knowledge. *arXiv preprint arXiv:2211.08411*.

Nikhil Kandpal, Eric Wallace, and Colin Raffel. 2022b. Deduplicating training data mitigates privacy risks in language models. In *Proceedings of the International Conference of Machine Learning*.

Vladimir Karpukhin, Barlas Oğuz, Sewon Min, Patrick Lewis, Ledell Wu, Sergey Edunov, Danqi Chen, and Wen-tau Yih. 2020. Dense passage retrieval for open-domain question answering. In *Proceedings of Empirical Methods in Natural Language Processing*.

Nora Kassner and Hinrich Schütze. 2020. BERT-kNN: Adfding a kNN search component to pretrained language models for better QA. In *Findings of the Association for Computational Linguistics: EMNLP 2020*.

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

Daniel Khashabi, Snigdha Chaturvedi, Michael Roth, Shyam Upadhyay, and Dan Roth. 2018. Looking beyond the surface: A challenge set for reading comprehension over multiple sentences. In *NAACL-HLT*.

Daniel Khashabi, Sewon Min, Tushar Khot, Ashish Sabharwal, Oyvind Tafjord, Peter Clark, and Hannaneh Hajishirzi. 2020. UnifiedQA: Crossing format boundaries with a single qa system. In *Findings of EMNLP*.

Omar Khattab and Matei Zaharia. 2020. Colbert: Efficient and effective passage search via contextualized late interaction over bert. In *Proceedings of the 43rd International ACM SIGIR conference on research and development in Information Retrieval*, pages 39–48.

Tushar Khot, Peter Clark, Michal Guerquin, Peter Jansen, and Ashish Sabharwal. 2020. Qasc: A dataset for question answering via sentence composition. In *AAAI*.

Tushar Khot, Ashish Sabharwal, and Peter Clark. 2018. Scitail: A textual entailment dataset from science question answering. In *AAAI*.

Junyeob Kim, Hyuhng Joon Kim, Hyunsoo Cho, Hwiyeol Jo, Sang-Woo Lee, Sang goo Lee, Kang Min Yoo, and Taeuk Kim. 2022. Ground-truth labels matter: A deeper look into input-label demonstrations. In *EMNLP*.

Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Diederik P Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *ICLR*.

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

Denis Kocetkov, Raymond Li, Loubna Ben allal, Jia LI, Chenghao Mou, Yacine Jernite, Margaret Mitchell, Carlos Muñoz Ferrandis, Sean Hughes, Thomas Wolf, Dzmitry Bahdanau, Leandro Von Werra, and Harm de Vries. 2023. The stack: 3TB of permissively licensed source code. *TMLR*.

Tomás Kociský, Jonathan Schwarz, Phil Blunsom, Chris Dyer, Karl Moritz Hermann, Gábor Melis, and Edward Grefenstette. 2018. The narrativeqa reading comprehension challenge. *TACL*.

Elyor Kodirov, Tao Xiang, Zhenyong Fu, and Shaogang Gong. 2015. Unsupervised domain adaptation for zero-shot learning. In *Proceedings of the IEEE international conference on computer vision*.

Pang Wei Koh and Percy Liang. 2017. Understanding black-box predictions via influence functions. In *Proceedings of the International Conference of Machine Learning*.

Neema Kotonya and Francesca Toni. 2020. Explainable automated fact-checking for public health claims. In *EMNLP*.

Brian Kulis. 2013. Metric learning: A survey. *Foundations and Trends in Machine Learning*, 5(4):287–364.

Mayank Kulkarni, Debanjan Mahata, Ravneet Arora, and Rajarshi Bhowmik. 2022. Learning rich representation of keyphrases from text. In *Findings of the Association for Computational Linguistics: NAACL 2022*.

Tom Kwiatkowski, Jennimaria Palomaki, Olivia Redfield, Michael Collins, Ankur Parikh, Chris Alberti, Danielle Epstein, Illia Polosukhin, Jacob Devlin, Kenton Lee, et al. 2019. Natural questions: a benchmark for question answering research. *Transactions of the Association for Computational Linguistics*.

Guokun Lai, Qizhe Xie, Hanxiao Liu, Yiming Yang, and Eduard H. Hovy. 2017. RACE: Large-scale reading comprehension dataset from examinations. In *EMNLP*.

Andrew Kyle Lampinen, Ishita Dasgupta, Stephanie C. Y. Chan, Kory Matthewson, Michael Henry Tessler, Antonia Creswell, James L. McClelland, Jane X. Wang, and Felix Hill. 2022. Can language models learn from explanations in context? In *EMNLP*.

Tian Lan, Deng Cai, Yan Wang, Heyan Huang, and Xian-Ling Mao. 2023. Copy is all you need. In *Proceedings of the International Conference on Learning Representations*.

Jinhyuk Lee, Zhuyun Dai, Sai Meher Karthik Duddu, Tao Lei, Iftekhar Naim, Ming-Wei Chang, and Vincent Zhao. 2024. Rethinking the role of token retrieval in multi-vector retrieval. *Advances in Neural Information Processing Systems*, 36.

Jinhyuk Lee, Mujeen Sung, Jaewoo Kang, and Danqi Chen. 2021. Learning dense representations of phrases at scale. In *Proceedings of the Association for Computational Linguistics*.

Katherine Lee, Daphne Ippolito, Andrew Nystrom, Chiyuan Zhang, Douglas Eck, Chris Callison-Burch, and Nicholas Carlini. 2022. Deduplicating training data makes language models better. In *Proceedings of the Association for Computational Linguistics*.

Kenton Lee, Ming-Wei Chang, and Kristina Toutanova. 2019. Latent retrieval for weakly supervised open domain question answering. In *ACL*.

Jens Lehmann, Robert Isele, Max Jakob, Anja Jentzsch, D. Kontokostas, Pablo N. Mendes, Sebastian Hellmann, M. Morsey, Patrick van Kleef, S. Auer, and C. Bizer. 2015. Dbpedia - a large-scale, multilingual knowledge base extracted from wikipedia. *Semantic Web*.

Mark A Lemley and Bryan Casey. 2020. Fair learning. *Texas Law Review*.

Hector J. Levesque, Ernest Davis, and Leora Morgenstern. 2012. The winograd schema challenge. In *Proceedings of the Thirteenth International Conference on Principles of Knowledge Representation and Reasoning*.

Omer Levy, Minjoon Seo, Eunsol Choi, and Luke Zettlemoyer. 2017. Zero-shot relation extraction via reading comprehension. In *CoNLL*.

Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Ves Stoyanov, and Luke Zettlemoyer. 2020a. BART: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. In *ACL*.

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

Margaret Li, Suchin Gururangan, Tim Dettmers, Mike Lewis, Tim Althoff, Noah A Smith, and Luke Zettlemoyer. 2022. Branch-train-merge: Embarrassingly parallel training of expert language models. *arXiv preprint arXiv:2208.03306*.

Xin Li and Dan Roth. 2002. Learning question classifiers. In *COLING*.

Bill Yuchen Lin, Seyeon Lee, Rahul Khanna, and Xiang Ren. 2020. Birds have four legs?! NumerSense: Probing Numerical Commonsense Knowledge of Pre-Trained Language Models. In *EMNLP*.

Jimmy Lin, Xueguang Ma, Sheng-Chieh Lin, Jheng-Hong Yang, Ronak Pradeep, and Rodrigo Nogueira. 2021. Pyserini: A python toolkit for reproducible information retrieval research with sparse and dense representations. In *Proceedings of the ACM SIGIR Conference on Research and Development in Information Retrieval*.

Kevin Lin, Oyvind Tafjord, Peter Clark, and Matt Gardner. 2019. Reasoning over paragraph effects in situations. In *Proceedings of the 2nd Workshop on Machine Reading for Question Answering*.

Yankai Lin, Haozhe Ji, Zhiyuan Liu, and Maosong Sun. 2018. Denoising distantly supervised open-domain question answering. In *Proceedings of the Association for Computational Linguistics*, pages 1736–1745.

Jiachang Liu, Dinghan Shen, Yizhe Zhang, Bill Dolan, Lawrence Carin, and Weizhu Chen. 2021. What makes good in-context examples for gpt-3? *arXiv preprint arXiv:2101.06804*.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Roberta: A robustly optimized bert pretraining approach. *arXiv*.

Kyle Lo, Lucy Lu Wang, Mark Neumann, Rodney Kinney, and Daniel Weld. 2020. S2ORC: The semantic scholar open research corpus. In *Proceedings of the Association for Computational Linguistics*.

Abe Lockman and David Klappholz. 1983. The control of inferencing in natural language understanding. In *Computational Linguistics*, pages 59–70. Elsevier.

Robert L Logan IV, Ivana Balaževic, Eric Wallace, Fabio Petroni, Sameer Singh, and Sebastian Riedel. 2021. Cutting down on prompts and parameters: Simple few-shot learning with language models. *arXiv preprint arXiv:2106.13353*.

Annie Louis, Dan Roth, and Filip Radlinski. 2020. "I'd rather just go to bed": Understanding indirect answers. In *EMNLP*.

Yao Lu, Max Bartolo, Alastair Moore, Sebastian Riedel, and Pontus Stenetorp. 2021. Fantastically ordered prompts and where to find them: Overcoming few-shot prompt order sensitivity. *arXiv preprint arXiv:2104.08786*.

Xinxi Lyu, Sewon Min, Iz Beltagy, Luke Zettlemoyer, and Hannaneh Hajishirzi. 2023. Z-icl: Zero-shot in-context learning with pseudo-demonstrations. In *ACL*.

Andrew L. Maas, Raymond E. Daly, Peter T. Pham, Dan Huang, Andrew Y. Ng, and Christopher Potts. 2011. Learning word vectors for sentiment analysis. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*.

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

Aman Madaan and Amir Yazdanbakhsh. 2022. Text and patterns: For effective chain of thought, it takes two to tango. *arXiv preprint arXiv:2209.07686*.

Chaitanya Malaviya, Subin Lee, Sihao Chen, Elizabeth Sieber, Mark Yatskar, and Dan Roth. 2023. Expertqa: Expert-curated questions and attributed answers. *arXiv*.

Alex Mallen, Akari Asai, Victor Zhong, Rajarshi Das, Hannaneh Hajishirzi, and Daniel Khashabi. 2022. When not to trust language models: Investigating effectiveness and limitations of parametric and non-parametric memories. *arXiv preprint arXiv:2212.10511*.

Pekka Malo, Ankur Sinha, Pekka Korhonen, Jyrki Wallenius, and Pyry Takala. 2014. Good debt or bad debt: Detecting semantic orientations in economic texts. *J. Assoc. Inf. Sci. Technol.*

Yuning Mao, Pengcheng He, Xiaodong Liu, Yelong Shen, Jianfeng Gao, Jiawei Han, and Weizhu Chen. 2020a. Generation-augmented retrieval for open-domain question answering. *arXiv*.

Yuning Mao, Pengcheng He, Xiaodong Liu, Yelong Shen, Jianfeng Gao, Jiawei Han, and Weizhu Chen. 2020b. Generation-augmented retrieval for open-domain question answering. In *ACL*.

Marco Marelli, Stefano Menini, Marco Baroni, Luisa Bentivogli, Raffaella Bernardi, and Roberto Zamparelli. 2014. A SICK cure for the evaluation of compositional distributional semantic models. In *LREC*.

Binny Mathew, Punyajoy Saha, Seid Muhie Yimam, Chris Biemann, Pawan Goyal, and Animesh Mukherjee. 2020. Hatexplain: A benchmark dataset for explainable hate speech detection. *arXiv preprint arXiv:2012.10289*.

Julian McAuley and J. Leskovec. 2013a. Hidden factors and hidden topics: understanding rating dimensions with review text. *Proceedings of the 7th ACM conference on Recommender systems*.

Julian McAuley and Jure Leskovec. 2013b. Hidden factors and hidden topics: understanding rating dimensions with review text. In *Proceedings of the 7th ACM conference on Recommender systems*, pages 165–172.

John McCarthy, Marvin L Minsky, Nathaniel Rochester, and Claude E Shannon. 2006. A proposal for the dartmouth summer research project on artificial intelligence, august 31, 1955. *AI magazine*, 27(4):12–12.

Clara H. McCreery, Namit Katariya, Anitha Kannan, Manish Chablani, and Xavier Amatriain. 2020. Effective transfer learning for identifying similar questions: Matching user questions to covid-19 faqs. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*.

Cade Metz. 2022. Lawsuit takes aim at the way A.I. is built. *New York Times*.

Paulius Micikevicius, Sharan Narang, Jonah Alben, Gregory Diamos, Erich Elsen, David Garcia, Boris Ginsburg, Michael Houston, Oleksii Kuchaiev, Ganesh Venkatesh, et al. 2017. Mixed precision training. In *ICLR*.

Todor Mihaylov, Peter Clark, Tushar Khot, and Ashish Sabharwal. 2018. Can a suit of armor conduct electricity? a new dataset for open book question answering. In *EMNLP*.

Sewon Min, Jordan Boyd-Graber, Chris Alberti, Danqi Chen, Eunsol Choi, Michael Collins, Kelvin Guu, Hannaneh Hajishirzi, Kenton Lee, Jennimaria Palomaki, Colin Raffel, Adam Roberts, Tom Kwiatkowski, and more. 2021a. NeurIPS 2020 EfficientQA competition: Systems, analyses and lessons learned. In *Machine Learning Research (PMLR)*.

Sewon Min, Danqi Chen, Hannaneh Hajishirzi, and Luke Zettlemoyer. 2019a. A discrete hard em approach for weakly supervised question answering. In *EMNLP*.

Sewon Min, Danqi Chen, Luke Zettlemoyer, and Hannaneh Hajishirzi. 2019b. Knowledge guided text retrieval and reading for open domain question answering. *arXiv*.

Sewon Min, Suchin Gururangan, Eric Wallace, Hannaneh Hajishirzi, Noah A. Smith, and Luke Zettlemoyer. 2024. SILO language models: Isolating legal risk in a nonparametric datastore. In *Proceedings of the International Conference on Learning Representations*.

Sewon Min, Kalpesh Krishna, Xinxi Lyu, Mike Lewis, Wen-tau Yih, Pang Wei Koh, Mohit Iyyer, Luke Zettlemoyer, and Hannaneh Hajishirzi. 2023a. FActScore: Fine-grained atomic evaluation of factual precision in long form text generation. In *EMNLP*.

Sewon Min, Kenton Lee, Ming-Wei Chang, Kristina Toutanova, and Hannaneh Hajishirzi. 2021b. Joint passage ranking for diverse multi-answer retrieval. In *EMNLP*.

Sewon Min, Mike Lewis, Hannaneh Hajishirzi, and Luke Zettlemoyer. 2022a. Noisy channel language model prompting for few-shot text classification. In *ACL*.

Sewon Min, Mike Lewis, Luke Zettlemoyer, and Hannaneh Hajishirzi. 2022b. MetaICL: Learning to learn in context. In *NAACL-HLT*.

Sewon Min, Xinxi Lyu, Ari Holtzman, Mikel Artetxe, Mike Lewis, Hannaneh Hajishirzi, and Luke Zettlemoyer. 2022c. Rethinking the role of demonstrations: What makes in-context learning work? In *EMNLP*.

Sewon Min, Julian Michael, Hannaneh Hajishirzi, and Luke Zettlemoyer. 2020. AmbigQA: Answering ambiguous open-domain questions. In *EMNLP*.

Sewon Min, Weijia Shi, Mike Lewis, Xilun Chen, Wen-tau Yih, Hannaneh Hajishirzi, and Luke Zettlemoyer. 2023b. Nonparametric masked language modeling. In *Findings of ACL*.

Swaroop Mishra, Daniel Khashabi, Chitta Baral, Yejin Choi, and Hannaneh Hajishirzi. 2021. Reframing instructional prompts to gptk's language. *arXiv preprint arXiv:2109.07830*.

Swaroop Mishra, Daniel Khashabi, Chitta Baral, and Hannaneh Hajishirzi. 2022. Cross-task generalization via natural language crowdsourcing instructions. In *ACL*.

Ioannis Mollas, Zoe Chrysopoulou, Stamatis Karlos, and Grigorios Tsoumakas. 2020. Ethos: an online hate speech detection dataset. *arXiv preprint arXiv:2006.08328*.

Robert C Moore. 2003. Learning translations of named-entity phrases from parallel corpora. In *Proceedings of the European Chapter of the Association for Computational Linguistics*.

Frederic Morin and Yoshua Bengio. 2005. Hierarchical probabilistic neural network language model. In *Proceedings of the International workshop on artificial intelligence and statistics*.

Niklas Muennighoff, Alexander M. Rush, Boaz Barak, Teven Le Scao, Aleksandra Piktus, Nouamane Tazi, Sampo Pyysalo, Thomas Wolf, and Colin Raffel. 2023. Scaling data-constrained language models. *arXiv preprint arXiv:2305.16264*.

Sebastian Nagel. 2016. CC-News. http://web.archive.org/save/http://commoncrawl.org/2016/10/news-dataset-available.

Nikita Nangia, Clara Vania, Rasika Bhalerao, and Samuel R. Bowman. 2020. CrowS-pairs: A challenge dataset for measuring social biases in masked language models. In *EMNLP*.

Courtney Napoles, Matthew Gormley, and Benjamin Van Durme. 2012. Annotated Gigaword. In *Proceedings of the Joint Workshop on Automatic Knowledge Base Construction and Web-scale Knowledge Extraction (AKBC-WEKEX)*.

Shashi Narayan, Shay B. Cohen, and Mirella Lapata. 2018. Don't give me the details, just the summary! topic-aware convolutional neural networks for extreme summarization. In *EMNLP*.

Arvind Neelakantan, Tao Xu, Raul Puri, Alec Radford, Jesse Michael Han, Jerry Tworek, Qiming Yuan, Nikolas A. Tezak, Jong Wook Kim, Chris Hallacy, Johannes Heidecke, Pranav Shyam, Boris Power, Tyna Eloundou Nekoul, Girish Sastry, Gretchen Krueger, David P. Schnurr, Felipe Petroski Such, Kenny Sai-Kin Hsu, Madeleine Thompson, Tabarak Khan, Toki Sherbakov, Joanne Jang, Peter Welinder, and Lilian Weng. 2022. Text and code embeddings by contrastive pre-training. *arXiv*.

Jianmo Ni, Chen Qu, Jing Lu, Zhuyun Dai, Gustavo Hernandez Abrego, Ji Ma, Vincent Zhao, Yi Luan, Keith B. Hall, Ming-Wei Chang, and Yinfei Yang. 2022. Large dual encoders are generalizable retrievers. In *EMNLP*.

Yixin Nie, Songhe Wang, and Mohit Bansal. 2019. Revealing the importance of semantic retrieval for machine reading at scale. In *Proceedings of Empirical Methods in Natural Language Processing*.

Yixin Nie, Adina Williams, Emily Dinan, Mohit Bansal, Jason Weston, and Douwe Kiela. 2020. Adversarial NLI: A new benchmark for natural language understanding. In *ACL*.

Rodrigo Nogueira and Kyunghyun Cho. 2019. Passage re-ranking with BERT. *ArXiv*, abs/1901.04085.

Mohammad Norouzi, Ali Punjani, and David J Fleet. 2012. Fast search in hamming space with multi-index hashing. In *2012 IEEE conference on computer vision and pattern recognition*, pages 3108–3115. IEEE.

Aaron van den Oord, Yazhe Li, and Oriol Vinyals. 2018. Representation learning with contrastive predictive coding. *arXiv preprint arXiv:1807.03748*.

Chester Palen-Michel, June Kim, and Constantine Lignos. 2022. Multilingual open text release 1: Public domain news in 44 languages. In *Proceedings of the Language Resources and Evaluation Conference*.

Jane Pan, Tianyu Gao, Howard Chen, and Danqi Chen. 2023. What in-context learning "learns" in-context: Disentangling task recognition and task learning. In *Findings of ACL*.

Bo Pang and Lillian Lee. 2004. A sentimental education: Sentiment analysis using subjectivity summarization based on minimum cuts. In *ACL*.

Bo Pang and Lillian Lee. 2005. Seeing stars: Exploiting class relationships for sentiment categorization with respect to rating scales. In *ACL*.

Dimitris Pappas, Petros Stavropoulos, Ion Androutsopoulos, and Ryan McDonald. 2020a. BioMRC: A dataset for biomedical machine reading comprehension. In *Proceedings of the 19th SIGBioMed Workshop on Biomedical Language Processing*.

Nikolaos Pappas, Phoebe Mulcaire, and Noah A. Smith. 2020b. Grounded compositional outputs for adaptive language modeling. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1252–1267, Online. Association for Computational Linguistics.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. 2019. Pytorch: An imperative style, high-performance deep learning library. In *Proceedings of Advances in Neural Information Processing Systems*.

Ajay Patel, Bryan Li, Mohammad Sadegh Rasooli, Noah Constant, Colin Raffel, and Chris Callison-Burch. 2022. Bidirectional language models are also few-shot learners. *arXiv preprint arXiv:2209.14500*.

Ethan Perez, Douwe Kiela, and Kyunghyun Cho. 2021. True few-shot learning with language models. In *NeurIPS*.

Matthew E. Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. In *North American Chapter of the Association for Computational Linguistics*.

Fabio Petroni, Patrick Lewis, Aleksandra Piktus, Tim Rocktäschel, Yuxiang Wu, Alexander H. Miller, and Sebastian Riedel. 2020. How context affects language models' factual predictions. In *Automated Knowledge Base Construction*.

Fabio Petroni, Tim Rocktäschel, Sebastian Riedel, Patrick Lewis, Anton Bakhtin, Yuxiang Wu, and Alexander Miller. 2019. Language models as knowledge bases? In *Proceedings of Empirical Methods in Natural Language Processing*.

Jonas Pfeiffer, Naman Goyal, Xi Lin, Xian Li, James Cross, Sebastian Riedel, and Mikel Artetxe. 2022. Lifting the curse of multilinguality by pre-training modular transformers. In *Conference of the North American Chapter of the Association for Computational Linguistics*.

Jonas Pfeiffer, Ivan Vulić, Iryna Gurevych, and Sebastian Ruder. 2020. Mad-x: An adapter-based framework for multi-task cross-lingual transfer.

Mohammad Taher Pilehvar and Jose Camacho-Collados. 2019. WiC: the word-in-context dataset for evaluating context-sensitive meaning representations. In *NAACL-HLT*.

Nina Poerner, Ulli Waltinger, and Hinrich Schütze. 2019. Bert is not a knowledge base (yet): Factual knowledge vs. name-based reasoning in unsupervised qa. *arXiv preprint arXiv:1911.03681*.

Ofir Press, Muru Zhang, Sewon Min, Ludwig Schmidt, Noah A. Smith, and Mike Lewis. 2023. Measuring and narrowing the compositionality gap in language models. In *Findings of EMNLP*.

Shawn Presser. 2020. Books3 corpus.

Project Gutenberg. Project gutenberg.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019. Language models are unsupervised multitask learners. *OpenAI blog*.

Jack W Rae, Sebastian Borgeaud, Trevor Cai, Katie Millican, Jordan Hoffmann, Francis Song, John Aslanides, Sarah Henderson, Roman Ring, Susannah Young, et al. 2021. Scaling language models: Methods, analysis & insights from training gopher. *arXiv preprint arXiv:2112.11446*.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J Liu. 2020. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of Machine Learning Research*.

Pranav Rajpurkar, Robin Jia, and Percy Liang. 2018. Know what you don't know: Unanswerable questions for squad. In *ACL*.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016a. Squad: 100,000+ questions for machine comprehension of text. In *Proceedings of Empirical Methods in Natural Language Processing*.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016b. SQuAD: 100,000+ questions for machine comprehension of text. In *EMNLP*.

Ori Ram, Yuval Kirstain, Jonathan Berant, Amir Globerson, and Omer Levy. 2021. Few-shot question answering by pretraining span selection. In *Proceedings of the Association for Computational Linguistics*.

Ori Ram, Yoav Levine, Itay Dalmedigos, Dor Muhlgay, Amnon Shashua, Kevin Leyton-Brown, and Yoav Shoham. 2023. In-context retrieval-augmented language models. *Transactions of the Association for Computational Linguistics*.

Ori Ram, Gal Shachaf, Omer Levy, Jonathan Berant, and Amir Globerson. 2022. Learning to retrieve passages without supervision. In *NAACL*.

Yasaman Razeghi, Robert L Logan IV, Matt Gardner, and Sameer Singh. 2022. Impact of pretraining term frequencies on few-shot reasoning. *arXiv preprint arXiv:2202.07206*.

Laria Reynolds and Kyle McDonell. 2021. Prompt programming for large language models: Beyond the few-shot paradigm. In *Extended Abstracts of the 2021 CHI Conference on Human Factors in Computing Systems*.

Matthew Richardson, Christopher J. C. Burges, and Erin Renshaw. 2013. Mctest: A challenge dataset for the open-domain machine comprehension of text. In *EMNLP*.

Adam Roberts, Colin Raffel, and Noam Shazeer. 2020. How much knowledge can you pack into the parameters of a language model? In *Proceedings of Empirical Methods in Natural Language Processing*.

Stephen Robertson, Hugo Zaragoza, et al. 2009. The probabilistic relevance framework: Bm25 and beyond. *Foundations and Trends® in Information Retrieval*.

Anna Rogers, Olga Kovaleva, Matthew Downey, and Anna Rumshisky. 2020. Getting closer to ai complete question answering: A set of prerequisite real tasks. In *AAAI*.

Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. 2022. High-resolution image synthesis with latent diffusion models. In *Computer Vision and Pattern Recognition*.

Frieda Rong. 2021. Extrapolating to unnatural language processing with gpt-3's in-context learning: The good, the bad, and the mysterious. https://ai.stanford.edu/blog/in-context-learning.

Ohad Rubin, Jonathan Herzig, and Jonathan Berant. 2021. Learning to retrieve prompts for in-context learning. *arXiv preprint arXiv:2112.08633*.

Sebastian Ruder. 2017. An overview of multi-task learning in deep neural networks. *arXiv preprint arXiv:1706.05098*.

Keisuke Sakaguchi, Ronan Le Bras, Chandra Bhagavatula, and Yejin Choi. 2020. WINOGRANDE: an adversarial winograd schema challenge at scale. In *AAAI*.

Victor Sanh, Albert Webson, Colin Raffel, Stephen H Bach, Lintang Sutawika, Zaid Alyafeai, Antoine Chaffin, Arnaud Stiegler, Teven Le Scao, Arun Raja, et al. 2022. Multitask prompted training enables zero-shot task generalization. In *ICLR*.

Keshav Santhanam, Omar Khattab, Jon Saad-Falcon, Christopher Potts, and Matei Zaharia. 2021. Colbertv2: Effective and efficient retrieval via lightweight late interaction. *arXiv preprint arXiv:2112.01488*.

Maarten Sap, Hannah Rashkin, Derek Chen, Ronan Le Bras, and Yejin Choi. 2019. Social iqa: Commonsense reasoning about social interactions. In *EMNLP-IJCNLP*.

Elvis Saravia, Hsien-Chi Toby Liu, Yen-Hao Huang, Junlin Wu, and Yi-Shin Chen. 2018. CARER: Contextualized affect representations for emotion recognition. In *EMNLP*.

David Saxton, Edward Grefenstette, Felix Hill, and Pushmeet Kohli. 2019. Analysing mathematical reasoning abilities of neural models. In *Proceedings of the International Conference on Learning Representations*.

Teven Le Scao, Angela Fan, Christopher Akiki, Ellie Pavlick, Suzana Ilić, Daniel Hesslow, Roman Castagné, Alexandra Sasha Luccioni, François Yvon, Matthias Gallé, et al. 2022. Bloom: A 176b-parameter open-access multilingual language model. *arXiv preprint arXiv:2211.05100*.

Rylan Schaeffer, Kateryna Pistunova, Samar Khanna, Sarthak Consul, and Sanmi Koyejo. 2023. Invalid logic, equivalent gains: The bizarreness of reasoning in language model prompting. In *ICML 2023 Workshop: Knowledge and Logical Reasoning in the Era of Data-driven Learning*.

Timo Schick, Jane Dwivedi-Yu, Zhengbao Jiang, Fabio Petroni, Patrick Lewis, Gautier Izacard, Qingfei You, Christoforos Nalmpantis, Edouard Grave, and Sebastian Riedel. 2022. Peer: A collaborative language model. *arXiv preprint arXiv:2208.11663*.

Timo Schick and Hinrich Schütze. 2021. It's not just size that matters: Small language models are also few-shot learners. In *NAACL-HLT*.

Minjoon Seo, Tom Kwiatkowski, Ankur P Parikh, Ali Farhadi, and Hannaneh Hajishirzi. 2018. Phrase-indexed question answering: A new challenge for scalable document comprehension. In *Proceedings of Empirical Methods in Natural Language Processing*.

Minjoon Seo, Jinhyuk Lee, Tom Kwiatkowski, Ankur P Parikh, Ali Farhadi, and Hannaneh Hajishirzi. 2019. Real-time open-domain question answering with dense-sparse phrase index. In *Proceedings of the Association for Computational Linguistics*.

Jingyuan S. She, Christopher Potts, Samuel R. Bowman, and Atticus Geiger. 2023. ScoNe: Benchmarking negation reasoning in language models with fine-tuning and in-context learning. In *ACL*.

Emily Sheng and David Uthus. 2020. Investigating societal biases in a poetry composition system. In *Proceedings of the Second Workshop on Gender Bias in Natural Language Processing*.

Weijia Shi, Julian Michael, Suchin Gururangan, and Luke Zettlemoyer. 2022. Nearest neighbor zero-shot inference. In *Proceedings of Empirical Methods in Natural Language Processing*.

Weijia Shi, Sewon Min, Maria Lomeli, Chunting Zhou, Margaret Li, Xi Victoria Lin, Noah A. Smith, Luke Zettlemoyer, Wen tau Yih, and Mike Lewis. 2024a. In-context pretraining: Language modeling beyond document boundaries. In *Proceedings of the International Conference on Learning Representations*.

Weijia Shi, Sewon Min, Michihiro Yasunaga, Minjoon Seo, Rich James, Mike Lewis, Luke Zettlemoyer, and Wen-tau Yih. 2024b. Replug: Retrieval-augmented black-box language models. In *Conference of the North American Chapter of the Association for Computational Linguistics*.

Sidney Siegel. 1957. Nonparametric statistics. *The American Statistician*.

Damien Sileo, Tim Van De Cruys, Camille Pradel, and Philippe Muller. 2019. Mining discourse markers for unsupervised sentence representation learning. In *NAACL-HLT*.

Silverman et al. v. Meta Platforms, Inc. 2023. Case 3:23-cv-03417, N.D. Cal.

Silverman et al. v. OpenAI, Inc. 2023. Case 3:23-cv-03417, N.D. Cal.

Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D. Manning, Andrew Ng, and Christopher Potts. 2013a. Recursive deep models for semantic compositionality over a sentiment treebank. In *EMNLP*.

Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D Manning, Andrew Y Ng, and Christopher Potts. 2013b. Recursive deep models for semantic compositionality over a sentiment treebank. In *EMNLP*.

Luca Soldaini and Kyle Lo. 2023. peS2o (Pretraining Efficiently on S2ORC) Dataset. Technical report, Allen Institute for AI. ODC-By, https://github.com/allenai/pes2o.

Yiping Song, Rui Yan, Cheng-Te Li, Jian-Yun Nie, Ming Zhang, and Dongyan Zhao. 2018. An ensemble of retrieval-based and generation-based human-computer conversation systems. In *International Joint Conference on Artificial Intelligence*.

Kai Sun, Dian Yu, Jianshu Chen, Dong Yu, Yejin Choi, and Claire Cardie. 2019. DREAM: A challenge data set and models for dialogue-based reading comprehension. *TACL*.

Simeng Sun, Dhawal Gupta, and Mohit Iyyer. 2023. Exploring the impact of low-rank adaptation on the performance, efficiency, and regularization of rlhf. *arXiv*.

Zequn Sun, Wei Hu, and Chengkai Li. 2017. Cross-lingual entity alignment via joint attribute-preserving embedding. In *Proceedings of the International Semantic Web Conference*.

Oyvind Tafjord, Peter Clark, Matt Gardner, Wen-tau Yih, and Ashish Sabharwal. 2019a. Quarel: A dataset and models for answering questions about qualitative relationships. In *AAAI*.

Oyvind Tafjord, Matt Gardner, Kevin Lin, and Peter Clark. 2019b. QuaRTz: An open-domain dataset of qualitative relationship questions. In *EMNLP*.

Alon Talmor, Jonathan Herzig, Nicholas Lourie, and Jonathan Berant. 2019. Commonsenseqa: A question answering challenge targeting commonsense knowledge. In *NAACL-HLT*.

Niket Tandon, Bhavana Dalvi, Keisuke Sakaguchi, Peter Clark, and Antoine Bosselut. 2019. WIQA: A dataset for "what if..." reasoning over procedural text. In *EMNLP*.

Yee Whye Teh. 2006. A bayesian interpretation of interpolated kneser-ney nus school of computing technical report tra2/06. *National University of Singapore*, pages 1–21.

James Thorne, Andreas Vlachos, Christos Christodoulopoulos, and Arpit Mittal. 2018. FEVER: a large-scale dataset for fact extraction and VERification. In *NAACL-HLT*.

Katherine Tian, Eric Mitchell, Huaxiu Yao, Christopher D Manning, and Chelsea Finn. 2023. Fine-tuning language models for factuality. *arXiv preprint arXiv:2311.08401*.

Zhiliang Tian, Wei Bi, Xiaopeng Li, and Nevin L Zhang. 2019. Learning to abstract for memory-augmented conversational response generation. In *Proceedings of the 57th annual meeting of the Association for Computational Linguistics*, pages 3816–3825.

Together. 2023. RedPajama: An open source recipe to reproduce LLaMA training dataset.

Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, Aurelien Rodriguez, Armand Joulin, Edouard Grave, and Guillaume Lample. 2023. LLaMA: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*.

Inc. Tremblay et al. v. OpenAI. 2023. Case 3:23-cv-03223 , N.D. Cal.

Adam Trischler, Tong Wang, Xingdi Yuan, Justin Harris, Alessandro Sordoni, Philip Bachman, and Kaheer Suleman. 2017. Newsqa: A machine comprehension dataset. In *Rep4NLP@ACL*.

Alan M Turing. 1980. Computing machinery and intelligence. *Creative Computing*, 6(1):44–53.

Miles Turpin, Julian Michael, Ethan Perez, and Samuel R Bowman. 2023. Language models don't always say what they think: Unfaithful explanations in chain-of-thought prompting. *arXiv*.

Sowmya Vajjala and Ivana Lučić. 2018. OneStopEnglish corpus: A new corpus for automatic readability assessment and text simplification. In *Proceedings of the Thirteenth Workshop on Innovative Use of NLP for Building Educational Applications*.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems*.

Ricardo Vilalta and Youssef Drissi. 2002. A perspective view and survey of meta-learning. *Artificial intelligence review*.

James Vincent. 2023. Getty images sues AI art generator stable diffusion in the us for copyright infringement.

Ellen M Voorhees. 1999. The TREC-8 question answering track report. In *TREC*, volume 99, pages 77–82.

Ellen M Voorhees and Dawn M Tice. 2000. Building a question answering test collection. In *Proceedings of the 23rd annual international ACM SIGIR conference on Research and development in information retrieval*, pages 200–207. ACM.

Nikhil Vyas, Sham Kakade, and Boaz Barak. 2023. Provable copyright protection for generative models. In *Proceedings of the International Conference of Machine Learning*.

Alex Wang, Yada Pruksachatkun, Nikita Nangia, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel Bowman. 2019a. Superglue: A stickier benchmark for general-purpose language understanding systems. In *Proceedings of Advances in Neural Information Processing Systems*.

Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R Bowman. 2018a. Glue: A multi-task benchmark and analysis platform for natural language understanding. In *Proceedings of the BlackboxNLP Workshop: Analyzing and Interpreting Neural Networks for NLP*.

Ben Wang and Aran Komatsuzaki. 2021. GPT-J-6B: A 6 Billion Parameter Autoregressive Language Model. https://github.com/kingoflolz/mesh-transformer-jax.

Boshi Wang, Sewon Min, Xiang Deng, Jiaming Shen, You Wu, Luke Zettlemoyer, and Huan Sun. 2023a. Towards understanding chain-of-thought prompting: An empirical study of what matters. In *ACL*.

Shuohang Wang, Mo Yu, Xiaoxiao Guo, Zhiguo Wang, Tim Klinger, Wei Zhang, Shiyu Chang, Gerry Tesauro, Bowen Zhou, and Jing Jiang. 2018b. R^3: Reinforced ranker-reader for open-domain question answering. In *Association for the Advancement of Artificial Intelligence*.

Wei Wang, Vincent W Zheng, Han Yu, and Chunyan Miao. 2019b. A survey of zero-shot learning: Settings, methods, and applications. *ACM Transactions on Intelligent Systems and Technology (TIST)*.

William Yang Wang. 2017. "liar, liar pants on fire": A new benchmark dataset for fake news detection. In *ACL*.

Yizhong Wang, Swaroop Mishra, Pegah Alipoormolabashi, Yeganeh Kordi, Amirreza Mirzaei, Anjana Arunkumar, Arjun Ashok, Arut Selvan Dhanasekaran, Atharva Naik, David Stap, et al. 2022. Super-naturalinstructions: Generalization via declarative instructions on 1600+ nlp tasks. In *EMNLP*.

Zhiguo Wang, Patrick Ng, Xiaofei Ma, Ramesh Nallapati, and Bing Xiang. 2019c. Multi-passage bert: A globally normalized bert model for open-domain question answering. In *EMNLP*.

Zhiruo Wang, Jun Araki, Zhengbao Jiang, Md Rizwan Parvez, and Graham Neubig. 2023b. Learning to filter context for retrieval-augmented generation. *arXiv preprint arXiv:2311.08377*.

Alex Warstadt, Alicia Parrish, Haokun Liu, Anhad Mohananey, Wei Peng, Sheng-Fu Wang, and Samuel R. Bowman. 2020. Blimp: The benchmark of linguistic minimal pairs for english. *TACL*.

Alex Warstadt, Amanpreet Singh, and Samuel R. Bowman. 2019. Neural network acceptability judgments. *TACL*.

Albert Webson and Ellie Pavlick. 2022. Do prompt-based models really understand the meaning of their prompts? In *NAACL-HLT*.

Jason Wei, Maarten Bosma, Vincent Y Zhao, Kelvin Guu, Adams Wei Yu, Brian Lester, Nan Du, Andrew M Dai, and Quoc V Le. 2022a. Finetuned language models are zero-shot learners. In *ICLR*.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Ed Chi, Quoc Le, and Denny Zhou. 2022b. Chain of thought prompting elicits reasoning in large language models. *arXiv preprint arXiv:2201.11903*.

Jerry Wei, Le Hou, Andrew Lampinen, Xiangning Chen, Da Huang, Yi Tay, Xinyun Chen, Yifeng Lu, Denny Zhou, Tengyu Ma, et al. 2023a. Symbol tuning improves in-context learning in language models. *arXiv*.

Jerry W. Wei, Jason Wei, Yi Tay, Dustin Tran, Albert Webson, Yifeng Lu, Xinyun Chen, Hanxiao Liu, Da Huang, Denny Zhou, and Tengyu Ma. 2023b. Larger language models do in-context learning differently. *arXiv*.

Johannes Welbl, Nelson F. Liu, and Matt Gardner. 2017. Crowdsourcing multiple choice science questions. In *Proceedings of the 3rd Workshop on Noisy User-generated Text*.

Noam Wies, Yoav Levine, and Amnon Shashua. 2023. The learnability of in-context learning. *arXiv*.

Adina Williams, Nikita Nangia, and Samuel Bowman. 2018. A broad-coverage challenge corpus for sentence understanding through inference. In *NAACL-HLT*.

Terry Winograd. 1971. Procedures as a representation for data in a computer program for understanding natural language.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi 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 M. Rush. 2020. Transformers: State-of-the-art natural language processing. In *Proceedings of Empirical Methods in Natural Language Processing: System Demonstrations*.

Tomer Wolfson, Mor Geva, Ankit Gupta, Matt Gardner, Yoav Goldberg, Daniel Deutch, and Jonathan Berant. 2020. Break it down: A question understanding benchmark. *TACL*.

Ledell Wu, Fabio Petroni, Martin Josifoski, Sebastian Riedel, and Luke Zettlemoyer. 2020. Scalable zero-shot entity linking with dense entity retrieval. In *Proceedings of Empirical Methods in Natural Language Processing*.

Sang Michael Xie, Hieu Pham, Xuanyi Dong, Nan Du, Hanxiao Liu, Yifeng Lu, Percy Liang, Quoc V Le, Tengyu Ma, and Adams Wei Yu. 2023. DoReMi: Optimizing data mixtures speeds up language model pretraining. *arXiv preprint arXiv:2305.10429*.

Sang Michael Xie, Aditi Raghunathan, Percy Liang, and Tengyu Ma. 2022. An explanation of in-context learning as implicit bayesian inference. In *Proceedings of the International Conference on Learning Representations*.

Lee Xiong, Chenyan Xiong, Ye Li, Kwok-Fung Tang, Jialin Liu, Paul Bennett, Junaid Ahmed, and Arnold Overwijk. 2021a. Approximate nearest neighbor negative contrastive learning for dense text retrieval. In *ICLR*.

Wenhan Xiong, Xiang Lorraine Li, Srini Iyer, Jingfei Du, Patrick Lewis, William Yang Wang, Yashar Mehdad, Wen tau Yih, Sebastian Riedel, Douwe Kiela, and Barlas Oğuz. 2021b. Answering complex open-domain questions with multi-hop dense retrieval. In *ICLR*.

Wenhan Xiong, Jiawei Wu, Hong Wang, Vivek Kulkarni, Mo Yu, Shiyu Chang, Xiaoxiao Guo, and William Yang Wang. 2019. TWEETQA: A social media focused question answering dataset. In *ACL*.

Frank F. Xu, Junxian He, Graham Neubig, and Vincent Josua Hellendoorn. 2022. Capturing structural locality in non-parametric language models. In *Proceedings of the International Conference on Learning Representations*.

Linting Xue, Noah Constant, Adam Roberts, Mihir Kale, Rami Al-Rfou, Aditya Siddhant, Aditya Barua, and Colin Raffel. 2021. mt5: A massively multilingual pre-trained text-to-text transformer. In *Conference of the North American Chapter of the Association for Computational Linguistics*.

Ikuya Yamada, Akari Asai, and Hannaneh Hajishirzi. 2021. Efficient passage retrieval with hashing for open-domain question answering. In *Proceedings of the Association for Computational Linguistics*.

Jinghui Yan, Jiajun Zhang, JinAn Xu, and Chengqing Zong. 2018. The impact of named entity translation for neural machine translation. In *Proceedings of the China Workshop on Machine Translation*.

Wei Yang, Yuqing Xie, Aileen Lin, Xingyu Li, Luchen Tan, Kun Xiong, Ming Li, and Jimmy Lin. 2019a. End-to-end open-domain question answering with bertserini. In *Conference of the North American Chapter of the Association for Computational Linguistics*, pages 72–77.

Wei Yang, Yuqing Xie, Luchen Tan, Kun Xiong, Ming Li, and Jimmy Lin. 2019b. Data augmentation for bert fine-tuning in open-domain question answering. *ArXiv*, abs/1904.06652.

Yi Yang, Wen-tau Yih, and Christopher Meek. 2015. WikiQA: A challenge dataset for open-domain question answering. In *EMNLP*.

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

Zhilin Yang, Peng Qi, Saizheng Zhang, Yoshua Bengio, William Cohen, Ruslan Salakhutdinov, and Christopher D. Manning. 2018b. HotpotQA: A dataset for diverse, explainable multi-hop question answering. In *EMNLP*.

Qinyuan Ye, Bill Yuchen Lin, and Xiang Ren. 2021. Crossfit: A few-shot learning challenge for cross-task generalization in nlp. In *EMNLP*.

Seonghyeon Ye, Hyeonbin Hwang, Sohee Yang, Hyeongu Yun, Yireun Kim, and Minjoon Seo. 2023a. In-context instruction learning. *arXiv*.

Seonghyeon Ye, Doyoung Kim, Joel Jang, Joongbo Shin, and Minjoon Seo. 2023b. Guess the instruction! flipped learning makes language models stronger zero-shot learners. In *ICLR*.

Seonghyeon Ye, Doyoung Kim, Sungdong Kim, Hyeonbin Hwang, Seungone Kim, Yongrae Jo, James Thorne, Juho Kim, and Minjoon Seo. 2023c. Flask: Fine-grained language model evaluation based on alignment skill sets. *arXiv*.

Wen-tau Yih, Kristina Toutanova, John C Platt, and Christopher Meek. 2011. Learning discriminative projections for text similarity measures. In *Conference on Computational Natural Language Learning*, pages 247–256.

Dani Yogatama, Cyprien de Masson d'Autume, and Lingpeng Kong. 2021. Adaptive semiparametric language models. *Proceedings of the Association for Computational Linguistics*.

Lili Yu, Bowen Shi, Ram Pasunuru, and Benjamin Miller. 2023a. Scaling Autoregressive Multi-Modal Models: Pretraining and Instruction Tuning.

Tao Yu, Rui Zhang, Kai Yang, Michihiro Yasunaga, Dongxu Wang, Zifan Li, James Ma, Irene Li, Qingning Yao, Shanelle Roman, Zilin Zhang, and Dragomir Radev. 2018. Spider: A large-scale human-labeled dataset for complex and cross-domain semantic parsing and text-to-SQL task. In *EMNLP*.

Xinyan Velocity Yu, Sewon Min, Luke Zettlemoyer, and Hannaneh Hajishirzi. 2023b. CREPE: Open-domain question answering with false presuppositions. In *EMNLP*.

Rowan Zellers, Yonatan Bisk, Roy Schwartz, and Yejin Choi. 2018. SWAG: A large-scale adversarial dataset for grounded commonsense inference. In *EMNLP*.

Rowan Zellers, Ari Holtzman, Yonatan Bisk, Ali Farhadi, and Yejin Choi. 2019. HellaSwag: Can a machine really finish your sentence? In *ACL*.

Jingtao Zhan, Jiaxin Mao, Yiqun Liu, Jiafeng Guo, M. Zhang, and Shaoping Ma. 2021. Optimizing dense retrieval model training with hard negatives. In *Proceedings of the ACM SIGIR Conference on Research and Development in Information Retrieval*.

Chiyuan Zhang, Daphne Ippolito, Katherine Lee, Matthew Jagielski, Florian Tramèr, and Nicholas Carlini. 2021. Counterfactual memorization in neural language models. *arXiv preprint arXiv:2112.12938*.

Dawen Zhang, Pamela Finckenberg-Broman, Thong Hoang, Shidong Pan, Zhenchang Xing, Mark Staples, and Xiwei Xu. 2023a. Right to be forgotten in the era of large language models: Implications, challenges, and solutions. *arXiv preprint arXiv:2307.03941*.

Leizhong Zhang, Qiong Yang, Ta Bao, Dave Vronay, and Xiaoou Tang. 2006. Imlooking: image-based face retrieval in online dating profile search. In *CHI'06 Extended Abstracts on Human Factors in Computing Systems*, pages 1577–1582.

Sheng Zhang, X. Liu, J. Liu, Jianfeng Gao, Kevin Duh, and Benjamin Van Durme. 2018. Record: Bridging the gap between human and machine commonsense reading comprehension. *arXiv preprint arXiv:1810.12885*.

Susan Zhang, Stephen Roller, Naman Goyal, Mikel Artetxe, Moya Chen, Shuohui Chen, Christopher Dewan, Mona Diab, Xian Li, Xi Victoria Lin, et al. 2022. Opt: Open pre-trained transformer language models. *arXiv preprint arXiv:2205.01068*.

Xiang Zhang, Junbo Zhao, and Yann LeCun. 2015. Character-level convolutional networks for text classification. In *NeurIPS*.

Yuan Zhang, Jason Baldridge, and Luheng He. 2019. PAWS: Paraphrase adversaries from word scrambling. In *NAACL-HLT*.

Yuhui Zhang, Michihiro Yasunaga, Zhengping Zhou, Jeff Z. HaoChen, James Zou, Percy Liang, and Serena Yeung. 2023b. Beyond positive scaling: How negation impacts scaling trends of language models. In *Findings of ACL*.

Jake Zhao and Kyunghyun Cho. 2018. Retrieval-augmented convolutional neural networks for improved robustness against adversarial examples. *arXiv preprint arXiv:1802.09502*.

Tony Z Zhao, Eric Wallace, Shi Feng, Dan Klein, and Sameer Singh. 2021. Calibrate before use: Improving few-shot performance of language models. In *ICML*.

Ruiqi Zhong, Kristy Lee, Zheng Zhang, and Dan Klein. 2021a. Adapting language models for zero-shot learning by meta-tuning on dataset and prompt collections. In *Findings of EMNLP*.

Ruiqi Zhong, Charlie Snell, Dan Klein, and Jacob Steinhardt. 2022a. Describing differences between text distributions with natural language. In *Proceedings of the International Conference of Machine Learning*.

Victor Zhong, Caiming Xiong, and Richard Socher. 2017. Seq2sql: Generating structured queries from natural language using reinforcement learning. *arXiv preprint arXiv:1709.00103*.

Zexuan Zhong, Dan Friedman, and Danqi Chen. 2021b. Factual probing is [mask]: Learning vs. learning to recall. In *Conference of the North American Chapter of the Association for Computational Linguistics*.

Zexuan Zhong, Tao Lei, and Danqi Chen. 2022b. Training language models with memory augmentation. In *Proceedings of Empirical Methods in Natural Language Processing*.

Ben Zhou, Daniel Khashabi, Qiang Ning, and Dan Roth. 2019. "going on a vacation" takes longer than "going for a walk": A study of temporal commonsense understanding. In *EMNLP*.

Yukun Zhu, Ryan Kiros, Rich Zemel, Ruslan Salakhutdinov, Raquel Urtasun, Antonio Torralba, and Sanja Fidler. 2015. Aligning books and movies: Towards story-like visual explanations by watching movies and reading books. In *Computer Vision and Pattern Recognition*.

Chapter A

Appendix: Understanding Current Language Models

A.1 Details for Section 2.3

A.1.1 Details on Datasets

Table A.1 and Table A.2 report a list of datasets used in Section 2.3. The first 10 rows are for settings described in Section 2.3.2; the next two rows are for settings used for ablations on the diversity of meta-training tasks (Table 2.5 of Section 2.3.4); the last two rows are for settings used for ablations on using natural instructions (Table 2.6 of Section 2.3.4). **Bold** datasets are target datasets with no overlap in domain with meta-training tasks. All datasets are taken from CROSSFIT [Ye et al., 2021] (except we exclude datasets that are unavailable from their repository[1] or the scope is notably different from other tasks, e.g., solving math problems or breaking down compositional questions) and UNIFIEDQA [Khashabi et al., 2020].

How meta-training/target splits are determined. The HR→LR setting is created based on the training data size as described in Section 2.3.2. Settings involving Classification, NLI and Paraphrase are taken from CROSSFIT. Settings involving QA are created by combining QA datasets from CROSSFIT and datasets from UNIFIEDQA. The number of tasks is the largest among recent related work: we have 142 unique tasks, while Khashabi et al. [2020], Zhong et al. [2021a], Mishra et al. [2022], Wei et al. [2022a] and Sanh et al. [2022] use 32, 62, 61, 42 and 62 tasks, respectively. References for all datasets are provided in Table A.2. Data and splits are available at `github.com/facebookresearch/MetaICL`.

[1] `github.com/INK-USC/CrossFit`

A.1.2 Details on Implementations

Preprocessing details. For all models with meta-training and the raw GPT-J, we separate the input and the output with one newline (`\n`), and separate between examples with three newlines. For the raw GPT-2, we use spaces instead of newlines. This choice was made in order to report the best baseline performance we were able to achieve: when raw LMs are used, GPT-2 is significantly better with spaces than with newlines, and GPT-J is significantly better with newlines than with spaces.[2] We note that NPM is less sensitive to these formatting differences, having less than 2% differences between using spaces and using newlines.

When the concatenation of k examples is too long, we truncate each example to have at most 256 tokens, and truncate the earlier tokens of the concatenation so that the LM sees the recent tokens. Additionally, for extractive question answering datasets as meta-training tasks, the input passage is truncated with a guarantee that the groundtruth answer is included in the input passage. We do not do this truncation for target datasets.

A.2 Details for Section 2.4

Example template. We follow Ye et al. [2021]; Min et al. [2022b]; Logan IV et al. [2021] in using the minimal format to transform the input to a sequence (e.g. a concatenation of multiple inputs) and using the label words from each dataset as it is. We also explore manual templates taken from prior work [Holtzman et al., 2021; Zhao et al., 2021] as reported in Section 2.4.2, although we find that using these templates is not consistently better than using minimal templates. We thus run main experiments with minimal templates. Example templates are provided in Table A.3.

Format of the demonstrations. We follow the standard of each model for formatting the demonstrations, either from exploration in prior work or the example code provided in the official tutorial. For GPT-2, we separate the input and the label, and each demonstration example with a space. For MetaICL, GPT-J and GPT-3, we separate the input and the label with a newline (`\n`), and each demonstration example with three newlines. For fairseq models, we use a newline to separate the input and the label as well as each demonstration example.

Details in variants of the demonstrations. For "OOD demonstrations", we use CC-News [Nagel, 2016] as an external corpus. We consider the length of the text during sampling, so that sampled sentences have similar length to the test input. For "demonstrations with random English words", we use `pypi.org/project/english-words` for the set of English words, which consists of 61,569 words. Table A.4 provides a list of example demonstrations for each method used in Section 2.3.4.

[2] For example, in the HR→LR setting, the raw GPT-2 is about 4% better with spaces then with newlines, and the raw GPT-J is about 5% better with spaces and then with newlines (all with the channel in-context learning method).

Setting: HR→LR Meta-train piqa, hate_speech_offensive, google_wellformed_query, social_i_qa, circa, quoref, glue-sst2, scitail, emo, cosmos_qa, freebase_qa, ag_news, art, paws, kilt_ay2, glue-qnli, quail, ade_corpus_v2-classification, sciq, hatexplain, emotion, glue-qqp, kilt_fever, kilt_nq, dbpedia_14, kilt_zsre, hellaswag, squad-with_context, hotpot_qa, glue-mnli, ropes, squad-no_context, kilt_hotpotqa, discovery, superglue-record, race-middle, race-high, lama-trex, swag, gigaword, amazon_polarity, biomrc, tab_fact, tweet_eval-emoji, tweet_eval-offensive, tweet_eval-sentiment, tweet_qa, imdb, lama-conceptnet, liar, anli, wiki_qa, kilt_trex, wikisql, wino_grande, wiqa, search_qa, xsum, yahoo_answers_topics, yelp_polarity, yelp_review_full
Setting: HR→LR Target quarel, **financial_phrasebank**, openbookqa, codah, qasc, glue-mrpc, dream, sick, commonsense_qa, **medical_questions_pairs**, quartz-with_knowledge, **poem_sentiment**, quartz-no_knowledge, glue-wnli, **climate_fever**, ethos-national_origin, ethos-race, ethos-religion, ai2_arc, hate_speech18, glue-rte, superglue-cb, superglue-copa, tweet_eval-hate, tweet_eval-stance_atheism, tweet_eval-stance_feminist
Setting: Classification Meta-train Meta-Train: superglue-rte, tweet_eval-sentiment, discovery, glue-rte, superglue-wsc, glue-mrpc, tweet_eval-stance_hillary, tweet_eval-offensive, emotion, hatexplain, glue-cola, sick, paws, ethos-sexual_orientation, glue-qqp, tweet_eval-emotion, sms_spam, health_fact, glue-mnli, imdb, ethos-disability, glue-wnli, scitail, trec-finegrained, yahoo_answers_topics, liar, glue-sst2, tweet_eval-stance_abortion, circa, tweet_eval-stance_climate, glue-qnli, tweet_eval-emoji, ethos-directed_vs_generalized, ade_corpus_v2-classification, hate_speech_offensive, superglue-wic, google_wellformed_query, tweet_eval-irony, ethos-gender, onestop_english, trec, rotten_tomatoes, kilt_fever
Setting: Non-Classification Meta-train ade_corpus_v2-dosage, art, biomrc, blimp-anaphor_number_agreement, blimp-ellipsis_n_bar_2, blimp-sentential_negation_npi_licensor_present, blimp-sentential_negation_npi_scope, common-sense_qa, crows_pairs, dream, freebase_qa, gigaword, hellaswag, hotpot_qa, kilt_ay2, kilt_hotpotqa, kilt_trex, kilt_zsre, lama-conceptnet, lama-google_re, lama-squad, numer_sense, openbookqa, piqa, proto_qa, qa_srl, quarel, quartz-no_knowledge, race-high, ropes, sciq, social_i_qa, spider, superglue-multirc, wikisql, xsum, yelp_review_full
Setting: Classification Target tweet_eval-stance_feminist, ethos-national_origin, tweet_eval-hate, ag_news, amazon_polarity, hate_speech18, **poem_sentiment**, **climate_fever**, **medical_questions_pairs**, tweet_eval-stance_atheism, superglue-cb, dbpedia_14, wiki_qa, emo, yelp_polarity, ethos-religion, **financial_phrasebank**, tab_fact, anli, ethos-race
Setting: QA Meta-train biomrc, boolq, freebase_qa, hotpot_qa, kilt_hotpotqa, kilt_nq, kilt_trex, kilt_zsre, lama-conceptnet, lama-google_re, lama-squad, lama-trex, mc_taco, numer_sense, quoref, ropes, search_qa, squad-no_context, squad-with_context, superglue-multirc, superglue-record, tweet_qa, web_questions, unifiedqa:squad2, unifiedqa:natural_questions_with_dpr_para, unifiedqa:race_string, unifiedqa:squad1_1, unifiedqa:drop, unifiedqa:newsqa, unifiedqa:narrativeqa, unifiedqa:winogrande_xl, unifiedqa:social_iqa, unifiedqa:quoref, unifiedqa:physical_iqa, unifiedqa:ropes, unifiedqa:commonsenseqa, unifiedqa:boolq
Setting: Non-QA Meta-train hate_speech_offensive, google_wellformed_query, circa, glue-sst2, scitail, emo, ag_news, art, paws, kilt_ay2, glue-qnli, ade_corpus_v2-classification, hatexplain, emotion, glue-qqp, kilt_fever, dbpedia_14, glue-mnli, discovery, gigaword, amazon_polarity, tab_fact, tweet_eval-emoji, tweet_eval-offensive, tweet_eval-sentiment, imdb, liar, anli, wikisql, xsum, yahoo_answers_topics, yelp_polarity, yelp_review_full
Setting: QA Target ai2_arc, codah, cosmos_qa, dream, hellaswag, openbookqa, qasc, quail, quarel, quartz-no_knowledge, quartz-with_knowledge, sciq, superglue-copa, swag, wino_grande, wiqa, unifiedqa:qasc, unifiedqa:qasc_with_ir, unifiedqa:openbookqa, unifiedqa:openbookqa_with_ir, **unifiedqa:mctest**, unifiedqa:ai2_science_middle
Setting: Non-NLI Meta-train ade_corpus_v2-classification, ag_news, amazon_polarity, circa, climate_fever, dbpedia_14, discovery, emo, emotion, ethos-directed_vs_generalized, ethos-disability, ethos-gender, ethos-national_origin, ethos-race, ethos-religion, ethos-sexual_orientation, financial_phrasebank, glue-cola, glue-mrpc, glue-qqp, glue-sst2, google_wellformed_query, hate_speech18, hate_speech_offensive, hatexplain, health_fact, imdb, kilt_fever, liar, medical_questions_pairs, onestop_english, paws, poem_sentiment, rotten_tomatoes, sick, sms_spam, superglue-wic, superglue-wsc, tab_fact, trec, trec-finegrained, tweet_eval-emoji, tweet_eval-emotion, tweet_eval-hate, tweet_eval-irony, tweet_eval-offensive, tweet_eval-sentiment, tweet_eval-stance_abortion, tweet_eval-stance_atheism, tweet_eval-stance_climate, tweet_eval-stance_feminist, tweet_eval-stance_hillary, wiki_qa, yahoo_answers_topics, yelp_polarity Setting: NLI Target anli, glue-mnli, glue-qnli, glue-rte, glue-wnli, **scitail**, sick, superglue-cb
Setting: Non-Paraphrase Meta-train ade_corpus_v2-classification, ag_news, amazon_polarity, anli, circa, climate_fever, dbpedia_14, discovery, emo, emotion, ethos-directed_vs_generalized, ethos-disability, ethos-gender, ethos-national_origin, ethos-race, ethos-religion, ethos-sexual_orientation, financial_phrasebank, glue-cola, glue-mnli, glue-qnli, glue-rte, glue-sst2, glue-wnli, google_wellformed_query, hate_speech18, hate_speech_offensive, hatexplain, health_fact, imdb, kilt_fever, liar, onestop_english, poem_sentiment, rotten_tomatoes, scitail, sick, sms_spam, superglue-cb, superglue-rte, superglue-wic, superglue-wsc, tab_fact, trec, trec-finegrained, tweet_eval-emoji, tweet_eval-emotion, tweet_eval-hate, tweet_eval-irony, tweet_eval-offensive, tweet_eval-sentiment, tweet_eval-stance_abortion, tweet_eval-stance_atheism, tweet_eval-stance_climate, tweet_eval-stance_feminist, tweet_eval-stance_hillary, wiki_qa, yahoo_answers_topics, yelp_polarity
Setting: Non-Paraphrase Target Target: glue-mrpc, glue-qqp, **medical_questions_pairs**, paws
Setting: HR→LR Diverse Meta-train glue-mnli, glue-qqp, glue-sst2, hate_speech_offensive, kilt_hotpotqa, kilt_zsre, lama-trex, race-high, scitail, tweet_eval-offensive, wino_grande, yahoo_answers_topics, yelp_review_full
Setting: HR→LR No Diverse Meta-train ag_news, amazon_polarity, dbpedia_14, emo, emotion, glue-sst2, imdb, tweet_eval-emoji, tweet_eval-offensive, tweet_eval-sentiment, yahoo_answers_topics, yelp_polarity, yelp_review_full
Setting: HR→LR Instructions Meta-train ag_news, amazon_polarity, anli, art, circa, cosmos_qa, dbpedia_14, discovery, emo, emotion, freebase_qa, gigaword, google_wellformed_query, hellaswag, imdb, liar, paws, piqa, quail, quoref, ropes, sciq, scitail, social_i_qa, swag, tab_fact, wiki_qa, wiqa, xsum, yahoo_answers_topics, yelp_polarity, yelp_review_full
Setting: HR→LR Instructions Target ai2_arc, **climate_fever**, codah, commonsense_qa, dream, **financial_phrasebank**, **medical_questions_pairs**, openbookqa, **poem_sentiment**, qasc, quarel, sick

Table A.1: Full datasets for all settings used in MetaICL (Section 2.3). The first 10 rows are for main settings described in Section 2.3.2; the last four rows are settings used for ablations in Section 2.3.4. Splits and dataname names consistent to those in Ye et al. [2021] and Khashabi et al. [2020]. **Bold** indicates the test dataset with no overlap in domain with meta-training tasks. A prefix `unifiedqa:` indicates that the dataset taken is from UnifiedQA; otherwise, from CrossFit. References for all datasets are provided in Table A.2.

ade_corpus_v2-classification [Gurulingappa et al., 2012], ade_corpus_v2-dosage [Gurulingappa et al., 2012], ag_news Gulli (link), ai2_arc [Clark et al., 2018], amazon_polarity [McAuley and Leskovec, 2013a], anli [Nie et al., 2020], art [Bhagavatula et al., 2020], biomrc [Pappas et al., 2020a], blimp-anaphor_number_agreement [Warstadt et al., 2020], blimp-ellipsis_n_bar_2 [Warstadt et al., 2020], blimp-sentential_negation_npi_licensor_present [Warstadt et al., 2020], blimp-sentential_negation_npi_scope [Warstadt et al., 2020], boolq [Clark et al., 2019], circa [Louis et al., 2020], climate_fever [Diggelmann et al., 2020], codah [Chen et al., 2019], commonsense_qa [Talmor et al., 2019], cosmos_qa [Huang et al., 2019], crows_pairs [Nangia et al., 2020], dbpedia_14 [Lehmann et al., 2015], discovery [Sileo et al., 2019], dream [Sun et al., 2019], emo [Chatterjee et al., 2019], emotion [Saravia et al., 2018], ethos-directed_vs_generalized [Mollas et al., 2020], ethos-disability [Mollas et al., 2020], ethos-gender [Mollas et al., 2020], ethos-national_origin [Mollas et al., 2020], ethos-race [Mollas et al., 2020], ethos-religion [Mollas et al., 2020], ethos-sexual_orientation [Mollas et al., 2020], financial_phrasebank [Malo et al., 2014], freebase_qa [Jiang et al., 2019], gigaword [Napoles et al., 2012], glue-cola [Warstadt et al., 2019], glue-mnli [Williams et al., 2018], glue-mrpc [Dolan and Brockett, 2005], glue-qnli [Rajpurkar et al., 2016b], glue-qqp (`data.quora.com/First-Quora-Dataset-Release-Question-Pairs`), glue-rte [Dagan et al., 2005; Bar-Haim et al., 2006][Giampiccolo et al., 2007; Bentivogli et al., 2009], glue-sst2 [Socher et al., 2013a], glue-wnli [Levesque et al., 2012], google_wellformed_query [Faruqui and Das, 2018], hate_speech18 [de Gibert et al., 2018], hate_speech_offensive [Davidson et al., 2017], hatexplain [Mathew et al., 2020], health_fact [Kotonya and Toni, 2020], hellaswag [Zellers et al., 2019], hotpot_qa [Yang et al., 2018b], imdb [Maas et al., 2011], kilt_ay2 [Hoffart et al., 2011], kilt_fever [Thorne et al., 2018], kilt_hotpotqa [Yang et al., 2018b], kilt_nq [Kwiatkowski et al., 2019], kilt_trex [Elsahar et al., 2018a], kilt_zsre [Levy et al., 2017], lama-conceptnet [Petroni et al., 2019, 2020], lama-google_re [Petroni et al., 2019, 2020], lama-squad [Petroni et al., 2019, 2020], lama-trex [Petroni et al., 2019, 2020], liar [Wang, 2017], mc_taco [Zhou et al., 2019], medical_questions_pairs [McCreery et al., 2020], numer_sense [Lin et al., 2020], onestop_english [Vajjala and Lučić, 2018], openbookqa [Mihaylov et al., 2018], paws [Zhang et al., 2019], piqa [Bisk et al., 2020], poem_sentiment [Sheng and Uthus, 2020], proto_qa [Boratko et al., 2020], qa_srl [He et al., 2015], qasc [Khot et al., 2020], quail [Rogers et al., 2020], quarel [Tafjord et al., 2019a], quartz-no_knowledge [Tafjord et al., 2019b], quartz-with_knowledge [Tafjord et al., 2019b], quoref [Dasigi et al., 2019], race-high [Lai et al., 2017], race-middle [Lai et al., 2017], ropes [Lin et al., 2019], rotten_tomatoes [Pang and Lee, 2005], sciq [Welbl et al., 2017], scitail [Khot et al., 2018], search_qa [Dunn et al., 2017], sick [Marelli et al., 2014], sms_spam [Almeida et al., 2011], social_i_qa [Sap et al., 2019], spider [Yu et al., 2018], squad-no_context [Rajpurkar et al., 2016b], squad-with_context [Rajpurkar et al., 2016b], superglue-cb [de Marneffe et al., 2019], superglue-copa [Gordon et al., 2012], superglue-multirc [Khashabi et al., 2018], superglue-record [Zhang et al., 2018], superglue-rte [Dagan et al., 2005; Bar-Haim et al., 2006][Giampiccolo et al., 2007; Bentivogli et al., 2009], superglue-wic [Pilehvar and Camacho-Collados, 2019], superglue-wsc [Levesque et al., 2012], swag [Zellers et al., 2018], tab_fact [Chen et al., 2020], trec [Li and Roth, 2002; Hovy et al., 2001], trec-finegrained [Li and Roth, 2002; Hovy et al., 2001], tweet_eval-emoji [Barbieri et al., 2020], tweet_eval-emotion [Barbieri et al., 2020], tweet_eval-hate [Barbieri et al., 2020], tweet_eval-irony [Barbieri et al., 2020], tweet_eval-offensive [Barbieri et al., 2020], tweet_eval-sentiment [Barbieri et al., 2020], tweet_eval-stance_abortion [Barbieri et al., 2020], tweet_eval-stance_atheism [Barbieri et al., 2020], tweet_eval-stance_climate [Barbieri et al., 2020], tweet_eval-stance_feminist [Barbieri et al., 2020], tweet_eval-stance_hillary [Barbieri et al., 2020], tweet_qa [Xiong et al., 2019], unifiedqa:ai2_science_middle (`data.allenai.org/ai2-science-questions`), unifiedqa:boolq [Clark et al., 2019], unifiedqa:commonsenseqa [Talmor et al., 2019], unifiedqa:drop [Dua et al., 2019], unifiedqa:mctest [Richardson et al., 2013], unifiedqa:narrativeqa [Kociský et al., 2018], unifiedqa:natural_questions [Kwiatkowski et al., 2019], unifiedqa:newsqa [Trischler et al., 2017], unifiedqa:openbookqa [Mihaylov et al., 2018], unifiedqa:physical_iqa [Bisk et al., 2020], unifiedqa:qasc [Khot et al., 2020], unifiedqa:quoref [Dasigi et al., 2019], unifiedqa:race_string [Lai et al., 2017], unifiedqa:ropes [Lin et al., 2019], unifiedqa:social_iqa [Sap et al., 2019], unifiedqa:squad1_1 [Rajpurkar et al., 2016b], unifiedqa:squad2 [Rajpurkar et al., 2018], unifiedqa:winogrande_xl [Sakaguchi et al., 2020], web_questions [Berant et al., 2013], wiki_qa [Yang et al., 2015], wikisql [Zhong et al., 2017], wino_grande [Sakaguchi et al., 2020], wiqa [Tandon et al., 2019], xsum [Narayan et al., 2018], yahoo_answers_topics (link), yelp_polarity [Zhang et al., 2015], yelp_review_full [Zhang et al., 2015]

Table A.2: References for 142 datasets used in MetaICL (Section 2.3). See Table A.1 to find datasets corresponding to each setting used in the experiments. A prefix `unifiedqa:` indicates that the dataset taken is from UnifiedQA; otherwise, from CrossFit.

Task breakdown of the results. Figure A.1 shows performance gap between using gold labels and using random labels per dataset. We find that the trend that the gap is smaller than previously thought is consistant across most datasets. Nonetheless, there are a few outlier datasets where performance gap is non-negligible,

such as financial_phrasebank and a few hate speech detection datasets. Future work may investigate on which tasks the model makes more use of the correctly paired training data.

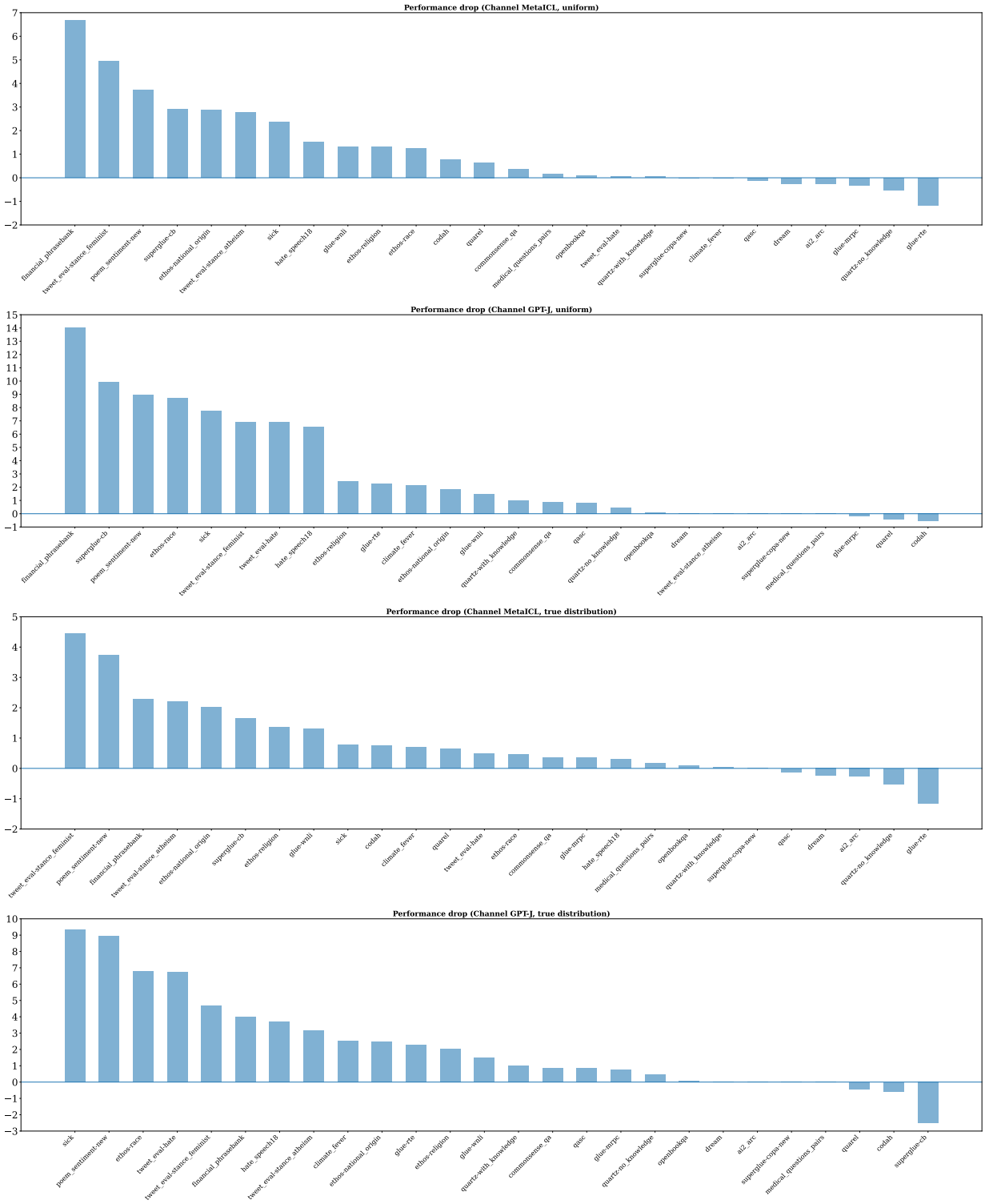

Figure A.1: Performance gap from using the demonstrations with gold labels to using the demonstrations with random labels. Datasets are sorted in descending order. The top two figures use random labels that are sampled at uniform, with Channel MetaICL and Channel GPT-J, respectively. The bottom two figures use random labels that are sampled from a true distribution of labels on the training data, with Channel MetaICL and Channel GPT-J, respectively.

Dataset	Type	Example
MRPC	Minimal	sentence 1: Cisco pared spending to compensate for sluggish sales . [SEP] sentence 2: In response to sluggish sales , Cisco pared spending . \n {equivalent|not_equivalent}
	Manual	Cisco pared spending to compensate for sluggish sales . \n The question is: In response to sluggish sales , Cisco pared spending . True or False? \n The answer is:{True|False}
RTE	Minimal	sentence 1: The girl was found in Drummondville. [SEP] sentence 2: Drummondville contains the girl. \n {entailment|not_entailment}
	Manual	The girl was found in Drummondville. \n The question is: Drummondville contains the girl. True or False? \n The answer is:{True|False}
Tweet_eval-hate	Minimal	The Truth about #Immigration \n {hate|non-hate}
	Manual	Tweet: The Truth about #Immigration \n Sentiment: {against|favor}
SICK	Minimal	sentence 1: A man is screaming. [SEP] sentence 2: A man is scared. \n {contradiction|entailment|neutral}
	Manual	A man is screaming. \n The question is: A man is scared. True or False? \n The answer is: {False|True|Not sure}
poem-sentiment	Minimal	willis sneered: \n {negative|no_impact|positive}
	Manual	willis sneered: \n The sentiment is: {negative|no_impact|positive}
OpenbookQA	Minimal	What creates a valley? \n {feet|rock|water|sand}
	Manual	The question is: What creates a valley? \n The answer is: {feet|rock|water|sand}
CommonsenseQA	Minimal	What blocks sunshine? \n {summer|park|desktop|sea|moon}
	Manual	The question is: What blocks sunshine? \n The answer is: {summer|park|desktop|sea|moon}
COPA	Minimal	Effect: I coughed. \n {Cause: I inhaled smoke.|Cause: I lowered my voice.}
	Manual	I coughed because {I inhaled smoke.|I lowered my voice.}
ARC	Minimal	Which biome has the most vegetation? \n {desert|forest|grassland|tundra}
	Manual	The question is: Which biome has the most vegetation? \n The answer is: {desert|forest|grassland|tundra}

Table A.3: A list of minimal templates taken from Ye et al. [2021]; Min et al. [2022b] and manual templates taken from Holtzman et al. [2021]; Zhao et al. [2021]. See Figure 2.6 for discussion in empirical results. The input and the label are in the red text and in the blue text, respectively. Note that | is used to separate different options for the labels.

Demos *w/ gold labels*	(*Format* ✓ *Input distribution* ✓ *Label space* ✓ *Input-label mapping* ✓) Circulation revenue has increased by 5% in Finland and 4% in Sweden in 2008. \n positive Panostaja did not disclose the purchase price. \n neutral
Demos *w/ random labels*	(*Format* ✓ *Input distribution* ✓ *Label space* ✓ *Input-label mapping* ✗) Circulation revenue has increased by 5% in Finland and 4% in Sweden in 2008. \n neutral Panostaja did not disclose the purchase price. \n negative
OOD Demos *w/ random labels*	(*Format* ✓ *Input distribution* ✗ *Label space* ✓ *Input-label mapping* ✗) Colour-printed lithograph. Very good condition. Image size: 15 x 23 1/2 inches. \n neutral Many accompanying marketing claims of cannabis products are often well-meaning. \n negative
Demos *w/ random English words*	(*Format* ✓ *Input distribution* ✓ *Label space* ✗ *Input-label mapping* ✗) Circulation revenue has increased by 5% in Finland and 4% in Sweden in 2008. \n unanimity Panostaja did not disclose the purchase price. \n wave
Demos *w/o labels*	(*Format* ✗ *Input distribution* ✓ *Label space* ✗ *Input-label mapping* ✗) Circulation revenue has increased by 5% in Finland and 4% in Sweden in 2008. Panostaja did not disclose the purchase price.
Demos *labels only*	(*Format* ✗ *Input distribution* ✗ *Label space* ✓ *Input-label mapping* ✗) positive neutral

Table A.4: Example demonstrations when using methods in Section 2.4.3. The financial_phrasebank dataset with $\mathcal{C} = \{$"positive", "neutral", "negative"$\}$ is used. Red text indicates the text is sampled from an external corpus; blue text indicates the labels are randomly sampled from the label set; purple text indicates a random English word.

Chapter B

Appendix: Nonparametric Language Models

B.1 Details on NPM (Section 3.3)

B.1.1 Model Details

Approximation at inference

Given $\mathbf{q}^{\text{start}}$ and $\mathbf{q}^{\text{end}}$, we take the top k tokens with the highest similarity scores with each of them, and compute scores over spans composed by these tokens. Let $c^*_{i:j}$ be a span in $\mathcal{C}$ from the i-th token to the j-th token, and $\text{E}(c) \in \mathbf{R}^h$ be a vector corresponding to a token $c \in \mathcal{C}$. We find the top k tokens for the start and the end:

$$\begin{aligned} c_{\text{s}_1}, c_{\text{s}_2}, \cdots, c_{\text{s}_k} &= \underset{c \in \mathcal{C}}{\text{argTopk}}\, \text{sim}(\mathbf{q}^{\text{start}}, \text{E(c)}), \\ c_{\text{e}_1}, c_{\text{e}_2}, \cdots, c_{\text{e}_k} &= \underset{c \in \mathcal{C}}{\text{argTopk}}\, \text{sim}(\mathbf{q}^{\text{end}}, \text{E(c)}) \end{aligned}$$

using a fast nearest neighbor search. We then define a set of candidate phrases $\tilde{\mathcal{C}}^*$ as:

$$\left(\bigcup_{i=1}^{k} \bigcup_{j=1}^{l_{\max}} c^*_{\text{s}_i:\text{s}_i+j-1} \right) \cup \left(\bigcup_{i=1}^{k} \bigcup_{j=1}^{l_{\max}} c^*_{\text{e}_i-j+1:\text{e}_i} \right),$$

and predict:

$$\underset{v^* \in \mathcal{V}^*}{\text{argmax}} \sum_{c^* \in \tilde{\mathcal{C}}^*} \mathbb{I}[v^* = c^*] \text{expsim}(\mathbf{q}, \text{E}(c^*)),$$

where $\text{E}(c^*) \in \mathbf{R}^{2h}$ is a vector corresponding to c^*, and $\mathcal{V}^*$ is a set of any possible n-grams defined by the vocabulary $\mathcal{V}$.

Training Details

All implementation was done with PyTorch [Paszke et al., 2019], PyTorch Lightning[1] and Huggingface Transformers [Wolf et al., 2020].

Masking. We use a masking ratio of 15% for all models, following the standard in prior work [Devlin et al., 2019; Liu et al., 2019; Joshi et al., 2020]. We implement masking as follows: (1) we first identify all possible candidate spans (spans that positives are found from other sequences in the batch), (2) sample the length of spans to mask from a geometric distribution with a hyperparameter $p = 0.5$, and (3) mask the spans with respect to the sampled length until the masking budget has been spent. We do not mask more than 128 spans from one sequence, and do not mask the span if the same span has been masked for more than ten times within the batch in order to prevent repeatedly masking overly frequent spans.

For `[MASK`$_\text{s}$`]` and `[MASK`$_\text{e}$`]`, we use the `[MASK]` vocab from the RoBERTa tokenizer. Note that it is not necessary to use different tokens for `[MASK`$_\text{s}$`]` and `[MASK`$_\text{e}$`]` since the Transformer can handle positional information.

A special case: NPM SINGLE

Along with NPM, we introduce **NPM SINGLE**, which outputs a nonparametric distribution over every single *token* in $\mathcal{C}$, instead of a *phrase*. To some extent, NPM is a strict generalization of NPM SINGLE, and NPM SINGLE still has a problem that existing encoder-only models have, e.g., can only fill in the `[MASK]` with a single token. We however think NPM SINGLE can be useful for some applications, e.g., for fine-tuning, as existing encoder-only models are used for.

Inference. Given a reference corpus $\mathcal{C} = \{c_1, \cdots, c_N\}$, we construct N number of h-dimensional vectors $\mathbf{c}_1, \cdots, \mathbf{c}_N \in \mathbb{R}^h$ by feeding the text into the encoder. At inference time, given a query whose t-th token is `[MASK]`, we feed it into the encoder:

$$\mathbf{q}_1..\mathbf{q}_L = \text{Encoder}(q_1..q_{t-1}, \texttt{[MASK]}, q_{t+1}..q_L).$$

We take $\mathbf{q}_t$ as a vector that represents the `[MASK]` token in the query. Finally, the prediction is made by aggregating the similarity scores to the tokens in $\mathcal{C}$:

$$\underset{v \in \mathcal{V}}{\text{argmax}} \sum_{c \in \mathcal{C}} \mathbb{I}[c = v] \exp(\text{sim}(\mathbf{q}_t, \text{E}(c))),$$

where $\text{E}(c) \in \mathbf{R}^h$ is a vector corresponding to c, and $\mathcal{V}$ is the vocabulary set.

[1] https://github.com/Lightning-AI/lightning

In practice, since computing scores over all tokens in $\mathcal{C}$ is infeasible, an approximation is made by computing scores for the top k nearest neighbors only, and treating other tokens to have a similarity score of $-\text{Inf}$. More precisely:

$$c^1, c^2, \cdots, c^k = \underset{c \in \mathcal{C}}{\text{argTopk}}\, \text{sim}(\mathbf{q}_t, \text{E}(c))$$

are obtained by using an index (e.g., FAISS [Johnson et al., 2019]), and the following is returned as a prediction:

$$\underset{v \in \mathcal{V}}{\text{argmax}} \sum_{1 \leq i \leq k} \mathbb{I}[c^i = v] \text{exp}(\text{sim}(\mathbf{q}_t, \text{E}(c^i))).$$

Training. Let $x_1^i ... x_L^i$ be the i-th sequence in the batch, whose subset is replaced with `[MASK]` and converted to $\hat{x}_1^i ... \hat{x}_L^i$. Both the unmasked sequence and the masked sequence are fed into the encoder, and each token is mapped into an h-dimensional vector:

$$\begin{aligned} \mathbf{x}_1^i \cdots \mathbf{x}_L^i &= \text{Encoder}(x_1^i \cdots x_L^i), \\ \hat{\mathbf{x}}_1^i \cdots \hat{\mathbf{x}}_L^i &= \text{Encoder}(\hat{x}_1^i \cdots \hat{x}_L^i). \end{aligned}$$

The training objective is then defined as:

$$\sum_{t=1}^{L} \mathbb{I}[\hat{x}_t = \texttt{[MASK]}] l(x_t^i, \hat{x}_t^i),$$

where $l(x_t^i, \hat{x}_t^i)$ is

$$-\log \frac{\sum_{\mathbf{y} \in \mathcal{Y}^+(x_t^i)} \text{exp}(\text{sim}(\hat{\mathbf{x}}_t^i, \mathbf{y}))}{\sum_{\mathbf{y} \in \mathcal{Y}^+(x_t^i) \cup \mathcal{Y}^-(x_t^i)} \text{exp}(\text{sim}(\hat{\mathbf{x}}_t^i, \mathbf{y}))}.$$

Here, $\text{sim}(\cdot, \cdot)$ is a similarity function defined in Section 3.3.2, and $\mathcal{Y}^+(x_t^i)$ and $\mathcal{Y}^-(x_t^i)$ are *positives* and *negatives* of x_t^i—tokens from *other* sequences in the batch that share and do not the vocab, respectively.

$$\begin{aligned} \mathcal{Y}^+(x_t^i) &= \left\{ x_m^j | x_t^i = x_m^j \text{ and } i \neq j \right\}, \\ \mathcal{Y}^-(x_t^i) &= \left\{ x_m^j | x_t^i \neq x_m^j \text{ and } i \neq j \right\}. \end{aligned}$$

B.1.2 Inference on closed-set tasks

When applying NPM and NPM SINGLE on closed-setk tasks, we closely follow Shi et al. [2022] who adapts kNN-LM for zero-shot inference on classification tasks. We assume a fuzzy verbalizer: $f : \mathcal{Y} \rightarrow \tilde{\mathcal{V}}$, where $\mathcal{Y}$ is a set of labels in the task and $\tilde{\mathcal{V}} \in \mathcal{V}$ is a subset of the vocabulary $\mathcal{V}$. The fuzzy verbalizer maps a label to a

Dataset	$\lvert\mathcal{D}\rvert$	$\lvert\mathcal{D}_{\text{s}}\rvert$	# labels	Example
Closed-set tasks				
AGN	120,000	3,000	4	Indiana defends its NCAA mens's soccer title by edging UC Santa Barbara in penalty kicks. The text topic is about `[MASK]`. (`[MASK]`={politics, sports, business, technology})
Yahoo	60,000	3,000	10	Company for cemecal at espaniea? Answer: Can you give us more info? The text topic is about `[MASK]`. (`[MASK]`={society, science, health, education, computer, sports, business, entertainment, family, politics})
Subj	2,000	2,000	2	He tells mitchell that he is now in debt. This is a `[MASK]`. (`[MASK]`={review, summary})
SST-2	2,210	2,210	2	It was `[MASK]`. (`[MASK]`={great, terrible})
MR	2,000	2,000	2	Simplistic, silly and tedious. It was `[MASK]`. (`[MASK]`={great, terrible})
RT	1,066	1,066	2	weird. rewarding. It was `[MASK]`. (`[MASK]`={great, terrible})
CR	2,000	2,000	2	I am very pleased so far. It was `[MASK]`. (`[MASK]`={great, terrible})
Amz	400,000	3,000	2	It was `[MASK]`. (`[MASK]`={great, terrible})
RTE	277	277	2	Most commercial logwood is grown in Honduras, right? `[MASK]`, plants are grown in water or in substance other than soil. (`[MASK]`={Yes, No})
Open-set tasks				
LAMA T-REx	34,039	2,983	-	AVCDH is owned by `[MASK]`.
LAMA Google RE	5,200	1,856	-	Joshua Mathiot died in `[MASK]`.
KAMEL	46,800	3,000	-	What is followed by So-Lo? Answer: `[MASK]`.
NQ	3,610	3,000	-	who sang i ran all the way home? The answer is: `[MASK]`.
TQA	11,313	3,000	-	Who wrote the opera Carmen? The answer is: `[MASK]`.
TempLAMA$^{22}_{19}$				
- changed	3,360	3,000	-	Contributor Covenant is developed by `[MASK]`.
- unchanged	3,360	3,000	-	Atari 8-bit family is developed by `[MASK]`.
Entity translation	10,452	6,622	-	The Korean translation of Banpo Bridge is: `[MASK]`.

Table B.1: Statistics of downstream datasets. $\lvert\mathcal{D}\rvert$ and $\lvert\mathcal{D}_{\text{s}}\rvert$ indicate the number of test examples on the original data and the subsampled data, respectively. See Appendix B.1.3 for details.

set of tokens that express the label, e.g., in a sentiment classification task, $f(\texttt{Positive})$ includes *awesome* or *great*, and $f(\texttt{Negative})$ includes *terrible* or *broken*.

NPM SINGLE is given a query vector $\mathbf{q} \in \mathbb{R}^h$ and predicts:

$$\underset{y \in \mathcal{Y}}{\text{argmax}} \sum_{c \in \mathcal{C}} \mathbb{I}[c \in f(y)] \exp\left(\frac{\text{sim}(\mathbf{q}, \text{E}(c))}{\tau}\right),$$

where $\text{E}(c) \in \mathbf{R}^h$ is a vector corresponding to c, and τ is a hyperparameter.

NPM is given a query vector $\mathbf{q} \in \mathbb{R}^{2h}$ and predicts:

$$\underset{y \in \mathcal{Y}}{\text{argmax}} \sum_{c^* \in \mathcal{C}^*} \mathbb{I}[c^* \in f(y)] \exp\left(\frac{\text{sim}(\mathbf{q}, \text{E}(c^*))}{\tau}\right),$$

where $\text{E}(c^*) \in \mathbf{R}^{2h}$ is a vector corresponding to c^*. Note that this is essentially equivalent to

$$\underset{y \in \mathcal{Y}}{\text{argmax}} \sum_{c \in \mathcal{C}} \mathbb{I}[c \in f(y)] \exp\left(\frac{\text{sim}(\mathbf{q}^{\text{start}}, \text{E}(c))}{\tau} + \frac{\text{sim}(\mathbf{q}^{\text{end}}, \text{E}(c))}{\tau}\right).$$

Corpus name	Source	$\|\mathcal{C}\|$	Datasets used
En-Wiki+CCNews	Subset of En-Wiki 08/01/2019 and CCNews	126M	AGN, Yahoo, RTE
Subjectivity corpus	Raw IMDB	15M	Subj
Review corpus	Amazon and IMDB	62M	SST-2, MR, RT, CR, Amz
En-Wiki 2019	En-Wiki 08/01/2019	810M	All open-set tasks
En-Wiki 2022	En-Wiki 08/01/2022	858M	TempLAMA$^{22}_{19}$

Table B.2: Statistics of the retrieval corpus. $|\mathcal{C}|$ indicates the number of tokens in the corpus.

We use $\tau = 5.0$ for both NPM SINGLE and NPM.

B.1.3 Evaluation Details

Table B.1 reports statistics and templates on each downstream task, and Table B.2 reports statistics of the retrieval corpus used in experiments.

For closed-set tasks, we use templates and verbalizers provided by Shi et al. [2022] for most datasets, except two datasets. For RTE, we use the template from Artetxe et al. [2022]. For Subj, we write our own template, motivated by Zhong et al. [2022a] that found Subj is mainly about differentiating a review and a summary. For open-set tasks, we use templates provided by the original authors, except NQ and TQA for which we use the templates from GPT-3 [Brown et al., 2020a]. Due to limited computation resource, we subsample the data to include up to 3,000 examples, following the standard from prior work [Zhao et al., 2021; Shi et al., 2022]. For closed-set tasks, we use exactly the same set of data as Shi et al. [2022], and for open-set tasks, we use the same script to subsample the data. For LAMA T-REx and Google RE, we subsample up to 1,000 examples for each of 1, 2, 3 and 4+ grams. For the entity translation task, we subsample up to 1,000 examples per language.

The following is a more detailed description of open-set tasks used in Section 3.3.5.

LAMA [Petroni et al., 2019] is a factual probing benchmark that is designed to quantify the amount of factual knowledge in the model. It requires the model to predict the object given a subject-relation tuple in a cloze format. We use two versions of LAMA [Petroni et al., 2019]: (1) LAMA T-REx, derived from Elsahar et al. [2018b] and (2) LAMA Google-RE, derived from the Google-RE corpus.[2] For each version, we additionally consider the UHN (UnHelpfulNames) subset [Poerner et al., 2019]) where instances whose subject strongly hints the object by names (e.g., `Apple Watch` and `Apple`) are excluded. We also consider the hard subset of T-Rex from Zhong et al. [2021b].

Note that Petroni et al. [2019] only include triples whose object is one token based on BERT [Devlin et al., 2019]; however, with a different pretrained model like RoBERTa, entities could be multiple BPE tokens. Entities that are splitted into multiple BPE tokens are more rare entities.

[2] https://code.google.com/archive/p/relation-extraction-corpus

ISO Code	Language	$\lvert\mathcal{D}\rvert$	$\lvert\mathcal{D}_s\rvert$
zh	Chinese	3,199	1,000
ar	Arabic	2,013	1,000
el	Greek	1,618	1,000
iw	Hebrew	841	841
ru	Russian	758	758
jp	Japanese	471	471
hi	Hindi	427	427
ko	Korean	418	418
pl	Polish	177	177
tr	Turkish	150	150
cs	Czech	109	109
ta	Tamil	80	80
th	Thai	74	74
mn	Mongolian	64	64
ml	Malayalam	53	53
TOTAL		10,452	6,622

Table B.3: Statistics of the entity translation benchmark. Languages are sorted based on their availabilities.

KAMEL [Kalo and Fichtel, 2022] is another factual probing task as LAMA but with a few key differences to make it more general and broad: (1) it includes a broader coverage of triples, (2) it removes the constraint that the object is one token based on BERT, (3) it includes objects with literal values, and (4) it has a question answering format.

Natural Questions (NQ, Kwiatkowski et al. [2019]) and **TriviaQA** (TQA, Joshi et al. [2017]) are two welll-studied open-domain question answering datasets. We use the open-version of NQ [Lee et al., 2019] and TQA where the question is the only input and the model should use its knowledge to answer the question.

TempLAMA$^{22}_{19}$ is a task that requires probing knowledge with temporal updates. The task is first introduced by Dhingra et al. [2022] and Jang et al. [2022a]; however, we could not use either of existing data as their time split do not match our training. We therefore create the data by using a script provided by Dhingra et al. [2022] but using the 2019 and the 2022 dumps. We take Wikipedia triples whose relations are available for a template from either Petroni et al. [2019] or Dhingra et al. [2022]. We then include triples whose object entities differ between the 2019 dump and the 2022 dump (due to the entity being updated), or only appear in the 2022 dump (due to the subject or the relation being added) to the *changed* set. Otherwise, triples are included in the *unchanged* set. We additionally find that many triples are overly difficult because the fact is extremely niche and not really known. We thus filter the data to only include facts that appear in Wikipedia. Specifically, we include triples if the subject has a corresponding Wikipedia page and the object entity appears in that Wikipedia page.

Model	#Params	$\mathcal{C}$	T-REx			Google RE		KML	TQA	NQ
			All	UHN	Hard	All	UHN			
Baselines (encoder-decoder)										
T5	2.2x		13.3	5.5	10.7	1.1	0.4	1.6	4.2	0.5
T5 3B	8.5x		12.1	8.2	11.5	2.1	0.7	3.6	9.0	2.0
BM25 + T5	2.2x	✓	22.2	20.3	22.4	16.4	16.6	13.9	31.4	5.2
BM25 + T5 3B	8.5x	✓	21.6	19.0	21.8	18.5	15.5	**16.2**	39.6	10.8
Baselines (decoder-only)										
OPT 2.7B	7.6x		9.8	6.7	8.3	0.0	0.0	1.6	9.9	2.1
GPT-3 2.7B	7.6x		4.4	2.6	3.8	0.0	0.0	2.1	5.2	1.1
OPT 6.7B	19x		11.6	9.9	10.7	0.6	0.3	3.2	20.9	4.2
GPT-3 6.7B	19x		8.1	5.0	6.7	0.0	0.0	2.1	12.4	3.1
OPT 13B	37x		15.0	12.7	12.7	0.3	0.3	2.5	22.5	4.2
GPT-3 13B	37x		16.4	13.7	15.5	0.8	0.4	2.2	25.5	5.2
GPT-3 175B	500x		25.7	24.1	24.7	1.1	1.0	6.5	**49.0**	**11.4**
BM25 + OPT 2.7B	7.6x	✓	14.8	14.1	13.8	4.4	3.7	11.3	28.5	8.3
BM25 + GPT-3 2.7B	7.6x	✓	3.5	3.4	3.6	0.1	0.1	5.2	14.5	6.1
BM25 + OPT 6.7B	19x	✓	14.8	14.3	14.9	4.1	3.3	8.2	29.9	10.7
BM25 + GPT-3 6.7B	19x	✓	14.9	15.3	15.1	4.4	3.5	7.0	21.1	8.8
BM25 + OPT 13B	37x	✓	18.9	19.1	19.3	3.8	3.1	10.6	34.0	10.7
BM25 + GPT-3 13B	37x	✓	22.2	22.7	22.4	11.8	11.2	8.9	32.4	11.2
BM25 + GPT-3 175B	500x	✓	32.0	**31.6**	31.3	11.4	11.9	12.2	44.9	6.4
Ours (encoder-only, nonparametric)										
NPM	1.0x	✓	**34.5**	29.0	**32.1**	**27.9**	**23.0**	15.6	32.2	10.8

Table B.4: Results on open-set tasks (raw numbers in Figure 3.7). *# Params* indicates the relative number of model parameters compared to RoBERTa large (354M), and $\mathcal{C}$ indicates whether a text corpus is used. For LAMA (T-REx and Google RE), the macro-averaged EM over 1, 2, 3 and 4+ grams are reported. All models are zero-shot. NPM significantly outperforms larger parameters models, either with and without a retrieval-and-generate approach that uses BM25.

Entity translation requires translating an entity from English to other languages that are not Latin based. While this is mainly to evaluate if the model can generate rare or unseen characters that are not in English, the entity translation task itself is a vital and challenging task in real applications such as machine translation [Babych and Hartley, 2003; Yan et al., 2018] and cross-lingual question answering [Clark et al., 2020a; Asai et al., 2021a]. It is often beyond a series of simple translations of each word, or spelling out its pronunciation [Moore, 2003; Hassan et al., 2007; Sun et al., 2017]. For instance, the Korean translation of *Banpo Bridge* in Figure 3.3 (반포대교) is not the concatenation of the translations of *Banpo* and *Bridge* (반포 다리).

We first identify a list of 15 non-Latin languages: Arabic (ar), Czech (cs), Greek (el), Hindi (hi), Hebrew (iw), Japanese (jp), Korean (ko), Malayalam (ml), Mongolian (mn), Polish (pl), Russian (ru), Tamil (ta), Thai (th), Turkish (tr), and Chinese (zh). We then implement heuristics to identify entities and their translations from English Wikipedia. Specifically, we parse the first paragraph of each Wikipedia article

Model	#Params	AGN		SST-2	
		0-shot	4-shot	0-shot	4-shot
Baselines (Parametric)					
RoBERTa	x1.0	71.3	-	84.5	-
GPT-3 2.7B [Zhao et al., 2021]	x7.6	44.7	43.3	57.2	59.1
+ CC [Zhao et al., 2021]	x7.6	63.2	71.1	71.4	79.9
GPT-3 2.7B [Holtzman et al., 2021]	x7.6	69.0	-	53.8	88.1
+ PMI [Holtzman et al., 2021]	x7.6	67.9	-	72.3	87.7
GPT-3 6.7B [Holtzman et al., 2021]	x19	64.2	-	54.5	92.9
+ PMI [Holtzman et al., 2021]	x19	57.4	-	80.0	79.8
GPT-3 13B [Holtzman et al., 2021]	x37	69.8	-	69.0	85.4
+ PMI [Holtzman et al., 2021]	x37	70.3	-	81.0	86.9
GPT-3 175B [Zhao et al., 2021]	x500	43.9	61.0	71.6	93.6
+ CC [Zhao et al., 2021]	x500	73.9	85.9	75.8	94.3
GPT-3 175B [Holtzman et al., 2021]	x500	75.4	-	63.6	89.9
+ PMI [Holtzman et al., 2021]	x500	74.7	-	71.4	95.5
Ours (Nonparametric)					
NPM SINGLE	x1.0	74.2	-	86.8	-
NPM	x1.0	74.5	-	87.2	-

Table B.5: Comparison to GPT-3 on AG News and SST-2. *# Params* indicates the relative number of model parameters compared to RoBERTa large (354M). All GPT-3 numbers are taken from previous work. k-shot indicates that the model performs in-context learning with k labeled examples with no gradient updates. We report on SST-2 and AGN, because they are all datasets shared between our paper and previous papers that report GPT-3 results [Zhao et al., 2021; Holtzman et al., 2021]. Our zero-shot models outperform 500x larger zero-shot GPT-3 and 7.6x larger 4-shot GPT-3, but lag behind 4-shot GPT-3 that is 19x or larger.

and pair the found translation with a topic entity of the article. For instance, a Korean translation of `Banpo Bridge` is found from the first sentence of `https://en.wikipedia.org/wiki/Banpo_Bridge`. Per-language statistics are reported in Table B.3.

B.1.4 Additional Results

Full results on knowledge tasks. Table B.4 reports full results on five knowledge tasks. See Figure 3.7 for an illustration, and Section 3.3.5 for discussion.

Comparison to few-shot GPT-3. Table B.5 compares zero-shot NPM SINGLE and NPM with zero- and four-shot GPT-3. Our zero-shot models outperform 500x larger zero-shot GPT-3 and 7.6x larger 4-shot GPT-3, but lag behind 4-shot GPT-3 that is 19x or larger. We think future work can explore extending our models to a few-shot setup.

Entity translation given an oracle passage. We evaluate models on the entity translation task where an oracle passage—a passage that is guaranteed to contain the translation information—is provided to the model. Baselines prepend oracle passages to the input, as it does with the retrieve-and-generate approach. NPM uses oracle passages to restrict the search space.

Model	#Params	#L	ar	cs	el	hi	iw	jp	ko	ml	mn	pl	ru	ta	th	tr	zh	AVG
Baselines, English-only																		
T5	2.2x		0.0	0.0	0.0	0.0	0.1	0.2	0.0	0.0	0.0	1.1	0.9	0.0	0.0	0.0	0.0	0.2
T5 3B	8.5x		0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	5.6	1.3	0.0	0.0	0.0	0.0	0.5
OPT 6.7B	19x		0.0	0.0	0.3	0.0	0.0	0.0	3.1	0.0	0.0	0.0	2.9	0.0	0.0	0.0	0.0	0.4
OPT 13B	37x		1.5	0.0	1.2	0.7	0.0	0.0	1.4	0.0	0.0	1.1	7.4	0.0	0.0	1.3	0.1	1.0
BM25 + T5	2.2x		0.0	5.5	0.3	0.2	0.5	0.0	0.2	1.9	0.0	6.8	0.8	1.2	0.0	11.3	0.0	1.9
BM25 + T5 3B	8.5x		0.0	12.8	0.1	0.7	0.2	0.8	0.0	0.0	1.6	28.8	1.7	0.0	0.0	20.0	0.0	4.4
BM25 + OPT 6.7B	19x		26.4	**54.1**	15.5	11.2	11.8	14.4	19.6	5.7	3.1	**47.5**	52.5	6.2	12.2	32.0	22.7	22.3
BM25 + OPT 13B	37x		17.3	51.4	24.9	15.5	27.8	12.3	22.0	11.3	7.8	45.8	48.2	8.8	18.9	**34.0**	23.3	24.6
Ours, English-only																		
NPM	1.0x		**51.9**	33.0	**60.9**	**63.2**	**63.7**	**59.0**	**60.5**	**50.9**	**46.9**	33.3	**61.2**	**51.2**	**60.8**	32.7	**56.9**	**52.4**
References, Multilingual																		
mT5	3.4x	101	0.3	1.8	1.5	0.0	0.4	1.9	0.7	0.0	0.0	1.1	4.6	2.5	1.4	3.3	0.7	1.3
mT5 XL	11x	101	4.4	3.7	4.9	6.8	0.7	2.3	4.1	1.9	4.7	5.6	8.0	5.0	0.0	6.7	2.8	4.1
BLOOM 3B	8.5x	46	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.3	0.0
BLOOM 7.1B	20x	46	0.0	0.9	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.5	0.1
BM25 + mT5	3.4x	101	12.4	22.9	21.6	9.8	12.5	28.9	19.1	11.3	18.8	15.8	16.0	17.5	28.4	16.7	33.4	19.0
BM25 + mT5 XL	11x	101	**64.4**	**64.2**	54.3	**65.6**	62.7	55.4	**69.4**	43.4	**62.5**	52.0	53.7	37.5	50.0	**48.7**	**65.0**	**56.6**
BM25 + BLOOM 3B	8.5x	46	24.2	25.7	1.7	13.3	15.1	18.5	17.9	5.7	6.2	21.5	11.1	10.0	27.0	18.0	44.5	17.4
BM25 + BLOOM 7.1B	20x	46	19.0	49.5	11.4	20.8	8.1	30.1	25.4	5.7	6.2	**54.2**	29.0	6.2	37.8	33.3	53.7	26.0

Table B.6: Results on the entity translation task. *#L* indicates the number of languages multilingual models are trained on. **Bold** and **Bold** indicate the best among monolingual models and the best including multilingual models, respectively. NPM significantly outperforms all existing monolingual models, and approaches or outperforms larger multilingual models.

Model	#Params	#L	ar	cs	el	hi	iw	jp	ko	ml	mn	pl	ru	ta	th	tr	zh	AVG
Baselines, English-only																		
T5	2.2x		0.0	13.8	0.7	0.9	0.6	1.1	0.5	3.8	1.6	15.8	1.3	7.5	0.0	16.7	0.4	4.0
T5 3B	8.5x		0.2	21.1	1.0	0.7	0.7	2.3	1.2	3.8	4.7	37.3	2.9	8.8	1.4	30.7	0.4	7.3
OPT 6.7B	19x		24.4	56.9	22.9	15.5	19.7	19.1	32.5	24.5	3.1	**56.5**	60.9	22.5	23.0	46.0	30.2	30.5
OPT 13B	37x		20.7	**62.4**	22.7	15.7	30.9	17.6	36.1	18.9	15.6	**56.5**	52.2	22.5	35.1	**48.7**	40.0	33.0
Ours, English-only																		
NPM	1.0x		**70.3**	44.0	**76.8**	**74.0**	**82.4**	**71.3**	**73.2**	**58.5**	**59.4**	45.2	**71.5**	**68.8**	**66.2**	45.3	**74.5**	**65.4**
References, Multilingual																		
mT5	3.4x	101	19.4	25.7	30.8	19.0	20.6	33.8	28.2	28.3	40.6	18.6	23.1	30.0	29.7	26.7	37.4	27.5
mT5 XL	11x	101	**83.2**	**76.1**	69.6	**81.5**	77.4	68.2	**85.2**	49.1	**67.2**	**65.5**	62.7	51.2	**68.9**	**64.0**	**79.0**	**69.9**
BLOOM 3B	8.5x	46	51.2	27.5	3.1	30.2	34.1	34.0	30.9	11.3	7.8	28.2	23.0	17.5	37.8	22.0	70.1	28.6
BLOOM 7.1B	20x	46	29.6	43.1	12.0	27.6	12.2	32.5	30.9	9.4	15.6	59.3	38.1	13.8	43.2	32.0	65.5	31.0

Table B.7: Results on the entity translation task given an oracle passage. *#L* indicates the number of languages multilingual models are trained on. **Bold** and **Bold** indicate the best excluding multilingual models and the best including multilingual models, respectively.

Table B.7 reports results. While performance overall increases compared to when the oracle passage is not provided, the overall comparison between models does not change from Table B.6: (1) all monolingual models significantly suffer, except for a couple of languages that are derived from Latin; (2) NPM significantly outperforms all monolingual models; (3) NPM even outperforms 3.4x larger mT5 and 20x larger BLOOM, and approaches 11x larger mT5.

Chapter C

Appendix: Responsible Language Models

C.1 Details for Section 4.4

Details on the parametric component SILO. Table C.1 reports the hyperparameters for the parametric component of SILO. We keep these hyperparameters fixed for all parametric models that we report in this paper. We follow the model architecture of LLaMa [Touvron et al., 2023], and we use the GPT-NeoX-20B tokenizer [Black et al., 2022], with 50432 BPE types. During training, we use 2,048 token sequences that are packed across document boundaries, and we pre-pend a beginning-of-text token to every document. We use weight decay of 0.1, the Adam optimizer with $\beta_2 = 0.95$, 2,000 steps of warmup, with a cosine learning rate scheduler. We train for multiple epochs in each dataset, tracking validation perplexity every 10B tokens, and perform early stopping. We train our PD, PDSW and PDSWBY models for 60B, 250B, and 350B tokens in total, respectively.

Model	#L	#H	d_{model}	LR	Batch
1.3B	24	16	2048	1e-3	2.6M

Table C.1: Basic hyperparameters for the parametric component of SILO.

Details on the nonparametric component of SILO. For kNN-LM, we use IndexIVFPQ which quantizes vectors into 64-bytes and clusters them into 4,096 centroids, learned from 1 million sampled vectors, following Khandelwal et al. [2020]. Instead of recomputing the exact L2 distance using the original embeddings, we use the L2 distance beteen quantized vectors returned by the FAISS index (ablations in Appendix C.2.2). Since their scale is not preserved, we use $\frac{d(\mathbf{x}_q, \mathbf{y}_q)}{\tau}$ as a proxy of $d(\mathbf{x}, \mathbf{y})$, where $\mathbf{x}_q$ and $\mathbf{y}_q$ are vectors quantized from $\mathbf{x}$ and $\mathbf{y}$. Hyperparameters, including k, λ, and τ, are chosen based on the validation data in a domain-specific manner.

Table C.2 reports the datastore statistics for both RIC-LM and kNN-LM, as well as hyperparameter values for kNN-LM (λ, k, τ). Due to the resource constraints, the datastore size is capped to up to 10% of the PILE training data (and to 1024.0M tokens in the case of kNN-LM), but future work can investigate further scaling the datastore.

Data	RIC-LM		kNN-LM			
	# tokens	# blocks	# tokens	λ	k	τ
Github	3084.3M	6.0M	1024.0M	0.2	128	10.0
NIH ExPorter	72.2M	0.1M	72.2M	0.3	32,768	20.0
Wikipedia	1177.9M	2.3M	1024.0M	0.3	4,096	20.0
CC News	382.2M	0.7M	382.2M	0.7	4,096	20.0
Books3	1424.7M	2.8M	1024.0M	0.2	4,096	25.0
Enron Emails	45.0M	0.1M	45.0M	0.5	4,096	1.0
Amazon	1214.3M	2.4M	1024.0M	0.5	32,768	20.0
MIMIC-III	519.5M	1.0M	519.5M	0.7	1,024	15.0

Table C.2: Datastore statistics as well as hyperparameter values for kNN-LM. Underline indicates exact nearest neighbor search (instead of approximate) was performed for kNN-LM because the datastore is small enough. Hyperparameters are chosen based on the validation data of each domain.

C.2 Additional Experimental Results in Section 4.5

Table C.3 reports perplexity of the parametric LMs on the validation data that is analogous to Table 4.3. Table C.4 reports perplexity of both parametric and nonparametric LMs on the validation data that is analogous to Table 4.4. Findings based on the validation data and on the test data are largely consistent.

C.2.1 Ablations: Parametric Component (Section 4.5.1)

Effect of upsampling low-resource data. Since OPEN LICENSE CORPUS has an extremely skewed distribution of domains, we upsample less-representative domains during training. Table C.5 (left) compares the models trained on PDSW with and without domain upweighting. In-domain datasets that are not upweighted, e.g., FreeLaw, see slight degration in performance. On out-of-doain datasets, there is no significant differences, although the model with upsampling is marginally better (19.6 vs. 19.7 when averaged over 9 out-of-domain datasets). We note that we did not tune the upweighting ratio nor explore alternative upweighting approaches [Xie et al., 2023] due to resource constraints, and leave them for future work.

59B tokens of source code significantly help. When using SW, a substantial 59.1% of the training data is actual source code. To determine SW provides such large gains, we also run an ablation where we include SW data but exclude all of the actual source code, i.e., we only include Hacker News, Ubuntu IRC, Deepmind Math, and AMPS on top of the PD data. This leaves models trained on 99.6B tokens for OLC (PDSW) and

Eval data	PD	PDSW	PDSWBY	Pythia
FreeLaw	5.3	5.7	6.5	5.6
Gutenberg	14.6	11.9	13.4	12.7
HackerNews	36.6	12.1	13.2	12.5
Github	13.3	2.6	2.7	2.4
NIH ExPorter	28.6	19.3	15.1	11.2
PhilPapers	55.2	24.2	16.5	14.3
Wikipedia	27.9	19.7	11.1	9.0
CC News	30.8	21.3	19.3	10.9
BookCorpus2	25.2	19.2	20.2	12.8
Books3	25.9	18.7	18.1	12.4
OpenWebText2	38.1	21.2	18.8	11.5
Enron Emails	19.9	14.3	14.5	7.6
Amazon	81.9	34.7	37.0	22.8
MIMIC-III	18.2	16.4	13.6	11.5
Average	30.1	17.2	15.7	11.2

Table C.3: Perplexity on the parametric LMs trained on PD, PDSW, and PDSWBY, as well as Pythia 1.4B, a model trained with similar amounts of compute but on non-permissive data. We use ■, ■, and ■ to indicate text that is in-domain, out-of-domain, or out-of-domain but has relevant data in-domain data (e.g., non-permissive Github code versus our permissive training code). Reported on the validation data; see Table 4.3 for results on the test data.

40.7B for OLC (PDSW) excluding source code. Table C.5 (right) report results on a subset of the validation domains. Including source code provide significant benefits for certain test datasets, e.g., nearly a 20 point improvement in perplexity on PhilPapers, likely because it significantly increases the size of the training data.

C.2.2 Ablations: Adding the Nonparametric Component (Section 4.5.2)

Impact of adding a nonparametric component across varying LMs. We compare parametric-only LM, RIC-LM and kNN-LM over four different LMs: PD, PDSW and PDSWBY variants of SILO as well as Pythia. Figure C.1 reports their results on three evaluation datasets: Wikipedia, NIH ExPorter and Enron Emails. Findings from Section 4.5.2 hold across all models: both RIC-LM and kNN-LM are consistently better than the parametric-only LM, and kNN-LM overall achieves the best performance. For instance, kNN-LM allows the model to be comparable to or outperform the one-level relaxed variant, e.g., a PD-based kNN-LM is comparable to a PDSW-based parametric LM, and PDSW-based kNN-LM is comparable to a PDSWBY-based parametric LM.

Effect of scaling the datastore in-domain and out-of-domain. §4.5.2 shows that performance of both kNN-LM and RIC-LM rapidly improves as the datastore size grows, and kNN-LM improves more rapidly than RIC-LM does. This evaluation is mainly done with SILO where the test domains are out-of-domain. Does this trend hold when the test domains are in-domain? To answer this question, we examine effect of scaling the datastore with Pythia 1.4B, where all of our test datasets can be considered in-domain.

Eval data	PDSW			Pythia
	Prm-only	kNN-LM	RIC-LM	Prm-only
Github	2.6	2.4	2.4	2.4
NIH ExPorter	19.3	14.9	18.5	11.2
Wikipedia	19.7	14.1	18.9	9.0
CC News	21.3	7.1	14.8	10.9
Books3	18.8	17.3	18.5	12.5
Enron Emails	14.3	6.7	11.1	7.6
Amazon	34.7	26.2	33.7	22.8
MIMIC-III	16.3	7.2	14.1	11.5
Average	18.4	12.0	16.5	11.0

Table C.4: Perplexity of parametric LMs (Prm-only), kNN-LM and RIC-LM; ■ indicates in-domain; ■ indicates out-of-domain; ■ indicates out-of-domain but has relevant data in-domain. Reported on the validaiton data; see Table 4.4 for results on the test data.

Data	PDSW w/o upsampling	PDSW w upsampling
FreeLaw	4.9	5.7
Github	2.4	2.6
NIH ExPorter	20.0	19.3
PhilPapers	23.9	24.2
Wikipedia	19.9	19.7
CC News	21.8	21.3
BookCorpus2	19.4	19.2
OpenWebText2	21.0	21.2
Enron Emails	13.5	14.3
Amazon	35.7	34.7

Data	PD	PDSW w/o code	PDSW
FreeLaw	5.3	5.7	5.7
Github	13.3	8.2	2.6
NIH ExPorter	28.6	26.2	19.3
PhilPapers	55.2	36.4	24.2
Wikipedia	27.9	26.5	19.7
CC News	30.8	28.8	21.3
BookCorpus2	25.2	23.8	19.2
OpenWebText2	38.1	31.7	21.2
Enron Emails	19.9	18.5	14.3
Amazon	81.9	46.1	34.7

Table C.5: (Left) Effect of re-weighting rare domains, comparing models trained on OLC (PDSW) with and without upsampling. **(Right)** Effect of SW data, with and without explicit source code—we train an LM with SW data but remove all of the actual source code (i.e., we leave Hacker News, Ubuntu IRC, Deepmind Math, and AMPS). Both tables report perplexity on the validation data.

Figure C.2 reports the results: Pythia on the left, SILO (PDSW) on the right. Results show that both Pythia and SILO see consistent improvements from kNN-LM and RIC-LM as the datastore gets larger, although the slope is larger with SILO than with Pythia. Again consistent to findings in §4.5.2, kNN-LM scales better than RIC-LM does, resulting in kNN-LM outperforming RIC-LM with a reasonably large datastore in most cases (with an exception of Pythia on Github, where RIC-LM outperforms kNN-LM with a reasonable size of a datastore).

Ablations on variants of RIC-LM. We explore four different variants of RIC-LM. (1) The **basic** is the method described in §4.4.2, which uses text blocks with a length of L each. At inference, it takes the top 1 text block from the datastore and feeds it to the LM, i.e., $P_{\text{LM}}(y|\hat{b}, x)$ where x is the input and $\hat{b}$ is the top 1 text block. (2) The **ensbl-*k*** ($k = 10$) variants is also based on text blocks with a length of L each. At

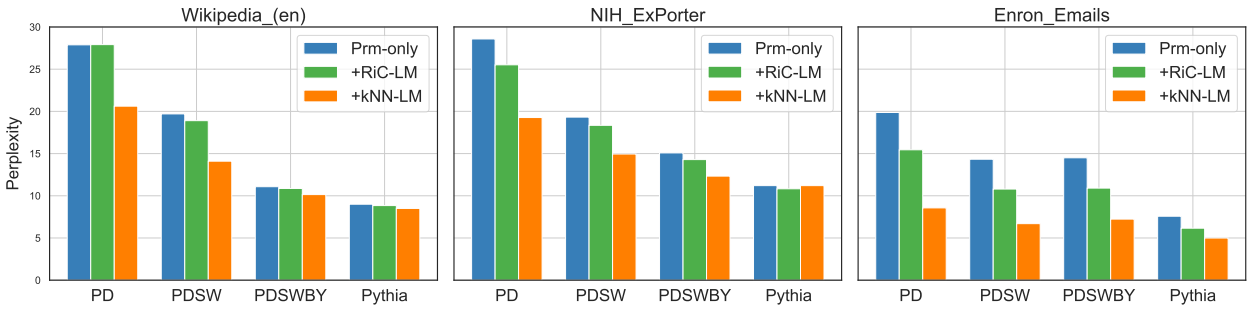

Figure C.1: Results on different variants of models: PD, PDSW and PDSWBY variants of SILO as well as Pythia. Adding a nonparametric component through either RIC-LM and kNN-LM helps and kNN-LM is overall better than RIC-LM, consistently across all models and evaluation datasets.

Data	PDSW				Pythia			
	basic	ensbl-10	concat-2	concat-next	basic	ensbl-10	concat-2	concat-next
CC News	14.8	13.5	17.0	18.8	8.2	7.9	9.2	9.9
Enron Emails	11.1	10.0	12.8	13.4	6.3	6.117	7.1	7.3

Table C.6: Ablations on different variants of RIC-LMs. Perplexity on the validation data reported. *ensbl-10* is 10x slower than other three methods.

inference, it takes the top k text blocks from the datastore, feeds each to the LM in parallel and aggregates the probability distributions, i.e., $\frac{1}{k}\sum_{1\leq i\leq k} P_{\text{LM}}(y|\hat{b}_i, x)$ where $\hat{b}_1 ... \hat{b}_k$ are the top k text blocks. This follows a method from Shi et al. [2024b]. (3) The **concat-k** ($k = 2$) variant uses text blocks with a length of $\frac{L}{k}$ each. At inference, it takes the top k text blocks from the datastore, concatenates them in a reverse order, and feeds it into the LM, e.g., $P_{\text{LM}}(y|\hat{b}_k, \cdots, \hat{b}_1, x)$ where $\hat{b}_1 ... \hat{b}_k$ are the top k text blocks. (4) The **concat-next** variant uses text blocks with a length of $\frac{L}{2}$ each. At inference, it takes the top 1 text block from the datastore, concatenates the text block and the subsequent text block in a datastore, and feeds it into the LM. This is based on the intuition that the continuation of the text block that is most similar to the query can be useful for the continuation of the query; Borgeaud et al. [2022] has explored a similar approach based on the same intuition. We use $L = 1024$ for all variants. The ensbl-k variant has run-time that is approximately k times of run-time of the basic, concat-k and concat-next.

Results are reported in Table C.6. The concat-2 and concat-next variants perform poorly, while the ensbl-10 outperforms the basic variant. However, we reached the conclusion that the significant run-time cost (i.e., 20x compared to a parametric LM) does not justify the improvements, and thus, we primarily use the basic variant for the remaining experiments. Future work may involve re-evaluating models using the ensbl-k approach or enhancing its run-time efficiency.

Effect of different approximation methods for L2 distance. Prior work [Khandelwal et al., 2020] typically uses approximate nearest neighbor search to find the top k nearest neighbors, and then computes

Method	PPL	Disk use
Param-only	19.7	0.0
No approximation	16.4	1.0
Quantized (4x)	16.6	0.25
Quantized (8x)	16.6	0.125
Quantized (16x)	16.8	0.0625
IVFPQ approximation	16.8	0.0178

Table C.7: Ablations on approximation methods on the validation data of Wikipedia, using the LM trained on PDSW and the datastore consisting of 51.2 million tokens (5% of the datastore in the main experiments). Relative disk memory usage reported (considering *no approximation* as 1.0).

Method	PPL	# tokens/s
Param-only	19.7	1828.6
RIC-LM (51.2M)	19.3	812.7
RIC-LM (102.4M)	19.2	731.4
RIC-LM (204.8M)	19.1	588.5
RIC-LM (409.6M)	18.9	478.5
RIC-LM (1,178M)	18.9	419.7
kNN-LM (51.2M)	16.8	184.2
kNN-LM (102.4M)	16.3	112.0
kNN-LM (204.8M)	15.7	59.3
kNN-LM (409.6M)	15.0	31.8
kNN-LM (1,024M)	14.2	14.2
kNN-LM (102M, $p = 1$)	16.7	560.8
kNN-LM (1,024M, $p = 1$)	14.6	71.1
kNN-LM (1,024M, $p = 2$)	14.4	45.5
kNN-LM (1,024M, $p = 4$)	14.2	27.0

Table C.8: Comparison in runtime speed (# tokens per second) on the validation data of Wikipedia. p indicates the number of probe, one of the hyperparameters in fast nearest neighbor search ($p = 8$ in all experiments if not specified otherwise).

the exact L2 distance using the original vectors. However, this may be inefficient in disk memory usage and run-time speed, due to needing to store large, original vectors and access them on-disk. We thus explore a few alternatives: (1) quantizing the original vectors to compute the L2 distance (but less aggressively than quantization for the nearest neighbor search index, thus it provides different levels of approximations for search and for L2 distance), or (2) completely dropping the original vectors and relying on approximated L2 distance from the index with aggressive quantization.

Table C.7 shows that all approximation methods only marginally affect performance. For this reason, we use the most aggressive approximation that completely drops the original embeddings at the cost of about 0.5% lose in performance while using $< 2\%$ of the memory footprint. Future work may study more accurate and efficient approximation methods.

Runtime speed. Table C.8 presents the runtime speed of the parametric LM, RIC-LM, and kNN-LM on the Wikipedia validation set. Speed is reported in tokens per second with a batch size of 1 using a single NVIDIA RTX 6000 GPU.

The results show that the parametric LM is notably faster than both RIC-LM and kNN-LM, and RIC-LM is faster than kNN-LM. Speed is slower as the datastore gets larger (for both RIC-LM and kNN-LM) and the nearest neighbor search gets less accurate (for kNN-LM; indicated by the number of probe p). kNN-LM can

eventually match RIC-LM's speed while surpassing its performance by using a smaller datastore and less accurate search, i.e., when using 102M tokens with $p = 1$.

However, the machine used for benchmarking has a very slow IO speed, leading to an underestimation of both RIC-LM and kNN-LM's runtime speed, and the comparison can significantly vary based on the hardware. Either way, kNN-LM is currently substantially slower than a parametric LM, leaving room for potential future improvements.

Qualitative examples. Figure C.9 provides six qualitative examples on the top-1 context retrieved by SILO-based kNN-LM. The model is able to assign a high probability to the ground truth token by retrieving highly relevant context, e.g., given the context (hockey) and the first name of the player, being able to retrieve the last name of the player, given the context (a show and its host), being able to complete the quote. These examples also highlight that a nonparametric approach addresses specific legal risks that we have discussed earlier, e.g., it assigns per-token attribution for free, and can provide a copyright notice when part of copyrighted text is being used for the probability distribution.

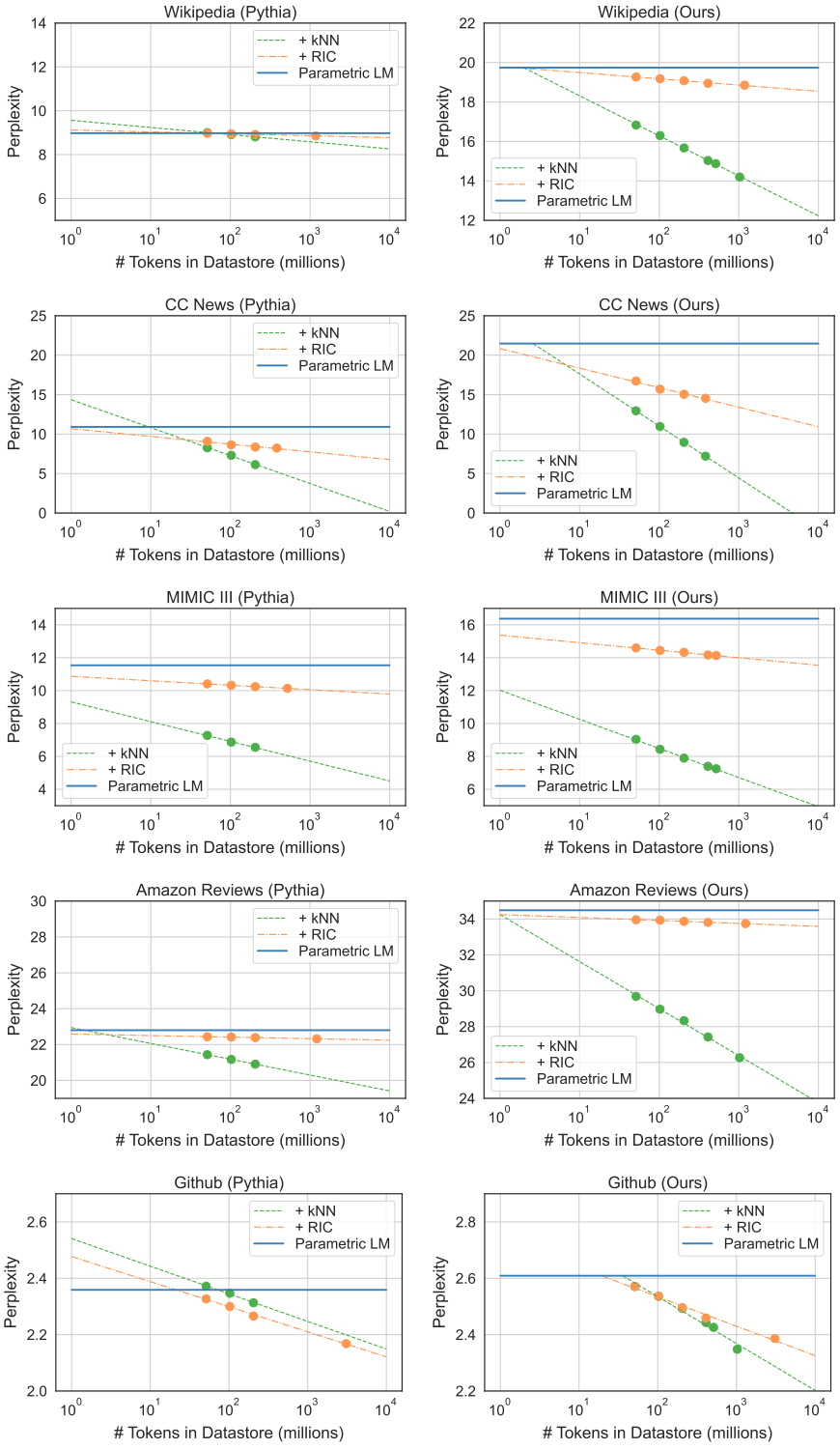

Figure C.2: Comparison between parametric LM, RIC-LM and kNN-LM on five domains, with Pythia (left) and SILO PDSW (right), respectively. Perplexity on random 128K tokens from the validation data reported.

Test Prefix
include '../lib/admin.defines.php';
include '../lib/admin.module.access.php';
include '../lib/admin.smarty.php';
if (! has_rights (
Test Continuation ACX_BILLING)) { Header . . .
Retrieved Prefix
(...)
* You should have received a copy of the GNU Affero General Public License
* along with this program. If not, see <http://www.gnu.org/licenses/>.
*
*
**/
if (! has_rights (
Retrieved Continuation ACX_ACCESS)) { Header ...

Test Prefix
0x5f #define S5K4AA_DEFAULT_BRIGHTNESS 0x10
/******************/
/* Kernel
Test Continuation module parameters */ extern int force_sensor; ...
Retrieved Prefix
* Copyright © 2011-2013 Jozsef Kadlecsik <kadlec@blackhold.kfki.hu>
*
* This program is free software; you can redistribute it and/or modify
* it under the terms of the GNU General Public License version 2 as
* published by the Free Software Foundation.
*/
/* Kernel
Retrieved Continuation module implementing an IP set type: . . .

Test Prefix . . . Mark or credit about hedge funds? Sara
Sara Shackleton
Enron North America Corp.
[Address]
[Phone number]
[Email address]
— Forwarded by Sara Shacleton/HOU/ECT on 01/2023/2022 05:41PM —
Tana
Test Continuation Jones 12/14/2000
Retrieved Prefix ... Food will be provided! Tana: Please feel free to extend the invitation to any Enron employees who may be interested in te presentation. 1st come, 1st serve. Thanks, Sylvia. — Forwarded by Sylvia Hu/Corp/Enron on 07/14/2000 03:17PM — Tana
Retrieved Continuation Jones@ECT. 07/13/2000

Test Prefix Ken Lay and Jeff Skilling were interviewed on CNNfn to discuss the succession of Jeff to CEO of Enron. (...) and then choose "Enron's Succession Plan.". The interview will be available every 15 minutes
Test Continuation through Friday, Dec. 15.
Retrieved Prefix Did you miss Jeff on CNBC "Street Signs" yesterday? Not to worry. (...) and then choose > "Skilling CNBC.". The interview will be available every ten minutes
Retrieved Continuation through > Wednesday, Dec. 6.

Test Prefix . . . The teams toured the city, explored west Edmonton mall and also got to take in an Oilers practice where they met German hockey star Leon
Test Continuation Draisaitl
Retrieved Prefix One minute and 19 seconds later, Cannor McDavid took a pass from Leon
Retrieved Continuation Draisaitl

Test Prefix ... Foley on RAW's run-time issues. Claiming that having the show run so late is one of the reasons why the final hour of RAW tends to struggle, Foley didn't end there. "No one else at 10:30pm is a
Test Continuation PG show. I won't say that across
Retrieved Prefix . . . way to the ring' podcast Foley cited RAW's duration and RG rating as hindrances to the show's popularity. Here's what he had to say: "Sometiems we try to look into the reasons why the third hour doesn't perform as well as the first two, and I'm like 'well that's because people go to bed! No one else at 10:30pm is a
Retrieved Continuation PG show. I won't say that across

Table C.9: Qualitative examples of retrieved context of SILO. **Red underlined text** indicates the next token that immediately follows the prefix. The first two are from Github; the next two are from Enron Emails; and the last two are from CC News.