

Evaluating and Understanding Declarative and Procedural Knowledge in Language Models

Rahul Nadkarni

A dissertation
submitted in partial fulfillment of the
requirements for the degree of

Doctor of Philosophy

University of Washington
2026

Reading Committee:

Noah A. Smith, Chair

Hannaneh Hajishirzi

Pang Wei Koh

Program Authorized to Offer Degree:
Computer Science and Engineering

© Copyright 2026

Rahul Nadkarni

University of Washington

Abstract

Evaluating and Understanding Declarative and Procedural
Knowledge in Language Models

Rahul Nadkarni

Chair of the Supervisory Committee:

Noah A. Smith

Computer Science and Engineering

Advancements in language model development have led to models with a broad range of useful capabilities, including those related to knowledge. These knowledge capabilities can be conceptualized as improvements in *declarative knowledge* (“knowing that”, being able to recall and use factual information) and *procedural knowledge* (“knowing how”, the ability to perform tasks or skills). While language models have improved substantially along both axes, many open questions remain regarding the extent of models’ declarative knowledge in various domains, the predictability of their procedural knowledge capabilities, and how each of these arises from the underlying datasets that models are trained on. In this thesis, we explore work in each of these directions, expanding our understanding of both declarative and procedural knowledge in language models. We begin with a study of declarative knowledge in language models via the task of scientific knowledge base completion, finding that language models effectively complement classical text-free methods on this task and are particularly valuable when applied to unseen scientific entities and concepts. Our work proceeds to investigate language models’ procedural knowledge capabilities by attempting to predict task performance from instructions, discovering that this remains a challenging task across various data- and model-related experimental settings. We then analyze how knowledge is related to language models’ training data by proposing an experimental recipe for performing interventional analyses on training, demonstrating the utility of this recipe by applying it to study various model behaviors. Finally, we conclude with thoughts on potential future directions on applying language models’ declarative knowledge for scientific discovery, modernizing the experimental study of predicting models’ procedural knowledge, and adapting our proposed experimental methodology to better understand how different stages of training each contribute to models’ knowledge capabilities.

Acknowledgements

First and foremost, I would like to thank my advisor, Noah A. Smith. I joined Noah's group under unusual circumstances, having already spent several years as a Ph.D. student doing research in an entirely different subject area. Nevertheless, Noah was gracious enough to accept me into his group and believe in my ability to make a contribution, and has been an incredible advisor and mentor throughout the years since then. I have grown tremendously as a researcher through the insights and feedback from our many meetings together, and the work presented in this dissertation was heavily shaped and improved by those conversations. I'm also grateful for Noah's patience during times when my research efforts were less productive or the path forward on certain projects was less clear, never failing to give helpful advice and encouragement even in those moments. From Noah I learned the importance of scientific rigor and a healthy amount of skepticism, both critical practices in this age of fast-paced research progress. My Ph.D. would not have been possible without Noah's invaluable guidance.

I would also like to thank my other committee members: Hannaneh Hajishirzi, Pang Wei Koh, and Gina-Anne Levow. I was fortunate enough to get a chance to collaborate with Hanna on my first project, which benefited tremendously from her suggestions and insights. The thoughtful questions and feedback on this work from Hanna, Pang Wei, and Gina really added value to this dissertation and helped me refine my presentation of the work.

I'd like to thank my many collaborators across the work presented in this dissertation: Dave Wadden, Iz Beltagy, Tom Hope, Yizhong Wang, Yanai Elazar, and Hila Gonen. This work would not have been possible without their valuable contributions and suggestions, and it's been a great pleasure to work with such an intelligent and collaborative group of researchers. I'd also like to acknowledge and thank collaborators on work that did not make it into this dissertation: Suchin Gururangan, Margaret Li, Rajarshi Das, Hanwen Xu, and Sheng Wang. Though our work together did not always yield the promising results we hoped for, I learned a lot from all of these talented researchers and really enjoyed working on challenging and exciting problems with them. I'd like to thank the many members of the ARK research group as well, who I've learned so much from on a wide range of fascinating research topics, and whose feedback always helped improve my work when I got the chance to discuss or present it.

I'd like to thank Nick Foti, for being my first research collaborator in grad school and teaching me a lot about how to carefully design and implement experiments, for always having the patience to go through all the technical details in my work, and for offering helpful perspectives on how to navigate the grad school journey from his own experiences. I would also like to thank Adrian KC Lee, for his research mentorship early in my Ph.D. and for his valuable thoughts on both my work as well as the decision of whether or not to continue in the program when I was faced with that unexpected choice.

Special thanks to both Elise Dorough and Joe Eckert, for promptly and clearly answering questions and helping navigate all administrative tasks, making it easy for me and many other grad students to not let those requirements distract us from our research. Thanks as well to Anna Karlin, for advice on dealing with changing advisors and research areas late in my Ph.D. that helped ease the stress of that process.

I would like to thank the many friends in the Allen School who have also been on their own Ph.D. journey alongside me, whose companionship along the way made the challenges less stressful and the good times even more enjoyable. A special thanks to Maarten Sap, for being an amazing roommate for so many years; for keeping our apartment lively and interesting with many friendly guests, delightful music, and random social gatherings; and for always taking the opportunity to reconnect when he's back in town with the added bonus of free food from Ai2. Special thanks as well to Chris Xie and Max Forbes – our CRM hangouts have been a highlight of my time in Seattle, both because of the opportunity to commiserate about the struggles of grad school life as well as the many unrelated but equally entertaining conversation topics (and in no small part because we rarely failed to get dessert). Being unsure that you would get to attend my defense as I was able to do for each of yours, I'm very grateful that you both managed to find the time to witness that culmination of this long journey (and in person no less!).

I would also like to thank the many friends outside of the research community who have been with me throughout my Ph.D., helping me stay grounded by reminding me of the life and relationships I have beyond grad school. Here I must give a special thanks to Tim Tsao, for being an amazing friend and source of joy and comfort for the entire duration of my Ph.D. You've been a part of this journey since the beginning, from picking me up from the airport as soon as I first arrived in Seattle to our many trips to visit each other. Our late night chats and movie watch parties have been integral to lifting my spirits and maintaining my sanity through this long and challenging endeavor, even when the conversations got way too intellectual. It's been an absolute pleasure being friends with you for so many years, and I look forward to many more years of friendship to come.

Last but certainly not least, I would like to thank my family: Ayee, Baba, and Rohan. Your unwavering patience and encouragement has helped me throughout my Ph.D., and you never ceased to believe in what I could accomplish even when I myself could not. I'm grateful beyond words for your support, and this dissertation is dedicated to you.

DEDICATION

To my family

Contents

1	Introduction	1
2	Scientific Language Models for Biomedical Knowledge Base Completion: An Empirical Study	6
2.1	Introduction	6
2.2	Task and Methods	8
2.2.1	KG Completion Task	8
2.2.2	Methods	9
2.2.3	Integrating KGE and LM: Model Averaging vs. Routing	10
2.2.4	Inductive KG Completion	10
2.3	Experimental Setup	11
2.3.1	Datasets	11
2.3.2	Pretrained LMs and KGE Integration	12
2.3.3	Evaluation	12
2.4	Experimental Results	13
2.4.1	Transductive Link Prediction Results	13
2.4.2	Model Averaging and Routing	15
2.4.3	Inductive KG Completion	16
2.5	Recent Work	17
2.6	Conclusion and Discussion	18
3	Third-Party Language Model Performance Prediction from Instruction	20
3.1	Introduction	20
3.2	Related Work	21
3.2.1	Instruction Tuning	21
3.2.2	Predicting Model Behavior	22
3.3	Methods	23

3.3.1	Instruction-Tuned Models (IMs)	24
3.3.2	Evaluation Metrics	24
3.3.3	Performance Predictors (PP)	25
3.4	Results	25
3.4.1	Performance Prediction is Challenging	25
3.4.2	Effect of Various Factors on Predictability	27
3.4.3	Predicting Loss	30
3.5	Discussion	31
3.6	Recent Work	32
3.7	Conclusion	32
4	Rewriting History: A Recipe for Interventional Analyses to Study Data Effects on Model Behavior	34
4.1	Introduction	34
4.2	Overview of Recipe	37
4.2.1	Components	37
4.2.2	Designing an Experiment	37
4.2.3	Stage 1: Selecting Items and a Data Batch	39
4.2.4	Stage 2: Identifying Relevant Documents	39
4.2.5	Stage 3: Intervening	40
4.3	Experimental Setup	42
4.3.1	Model	42
4.3.2	Evaluation	42
4.3.3	Selecting Items	43
4.3.4	Intervening	43
4.4	Impact of Term Cooccurrences on Relation Fact Learning	44
4.4.1	Experiment Setup	44
4.4.2	Result: Removing Cooccurrences and Occurrences Suppresses Learning	45
4.4.3	Result: Introducing Cooccurrences Promotes Learning	47
4.5	Impact of Pronoun-Profession Cooccurrence on Gender Bias	47
4.5.1	Experiment Setup	47
4.5.2	Result: Swapping Gendered Pronouns Mitigates Gender Bias	48
4.6	Impact of Retrieved Documents on Complex Knowledge Queries	49
4.6.1	Experiment Setup	49
4.6.2	Result: Removing Relevant Documents Suppresses Learning	50

4.6.3	Result: Adding Relevant Documents Promotes Learning	51
4.7	Determining Document Relevance	52
4.8	Related Work	54
4.9	Recent Work	54
4.10	Discussion and Conclusions	55
5	Conclusion	57
5.1	Summary	57
5.2	Future Work	58
A	Appendices for Chapter 2	78
A.1	Dataset Construction	78
A.1.1	Sources	78
A.1.2	Transductive Splits	78
A.1.3	Inductive Splits	79
A.1.4	Negative Validation/Test Triples	79
A.2	Training	80
A.2.1	Transductive Setting	80
A.2.2	Inductive Setting	84
A.3	Additional Results	85
A.3.1	Transductive Setting, Individual Models	85
A.3.2	Transductive Setting, Integrated Models	86
A.3.3	Inductive Setting	86
B	Appendices for Chapter 3	88
B.1	Finetuning Details	88
B.2	Details on BIG-bench	89
C	Appendices for Chapter 4	91
C.1	Swapping Documents Between Batches	91

List of Figures

2.1	Illustration of main methods used for biomedical knowledge graph completion	8
2.2	Fraction of test set examples where each model performs better.	14
2.3	Test set MRR for all pairs of KG completion models using an MLP router. The best combination of a KGE model and KG-PubMedBERT always performs better than the best pair of KGE models, and for RepoDB and Hetionet all pairs involving KG-PubMedBERT outperform all KGE-only pairs.	15
3.1	A diagram illustrating our complete analysis pipeline. We begin with a pretrained LM that is instruction-tuned using training tasks from chosen instruction data , resulting in an instruction-tuned model (IM) . The IM is evaluated using the test tasks of the instruction data (not necessarily from the same dataset as the training tasks) and a choice of evaluation metric . Each pair of test task instruction (x) and evaluation performance metric value (y) is used to construct the performance data, which itself is split into train, validation, and test sets. The train and validation sets are used to train <i>another</i> (“third party”) pretrained LM to predict the performance of the IM as a regression model, resulting in the performance predictor (PP) . Finally, the PP is evaluated on the test set of the performance data to determine how well it can predict the performance of the IM on unseen tasks. The sections of the diagram highlighted in blue indicate the components of the pipeline that we vary to determine their effect on performance prediction: the size of the IM , the choice of instruction data , the choice of evaluation metric , and the size and type of PP model.	23
3.2	Predicted vs. true metric value when using RoBERTa-large to map from task instruction to performance – either ROUGE-L (top row) or Exact Match (bottom row) – for various instruction-following models (columns).	26
3.3	Predicted vs. true loss value when using RoBERTa-large to map from task instruction to loss for various instruction-following models.	30

4.1	Diagram illustrating our proposed interventional analysis recipe, with stages including selecting items, matching documents to items, and intervening on matched documents.	35
4.2	Results for replacing matching documents and re-training checkpoints to <i>suppress learning</i> of selected items from ParaRel. Colored bars show accuracy on the subset of target items for each intervention condition, including retraining on the original data batch (blue), training on a modified batch that replaces cooccurrences (orange), and training on a modified batch that (more aggressively) replaces occurrences (green). We also show results on control items ($\times$; i.e., all items other than the target group). For all results, we perform 5 repeated training runs and show the mean and one standard deviation across runs.	46
4.3	Results for replacing matching documents and retraining to <i>promote learning</i> . Replacing documents not containing a cooccurrence with documents that do improves performance in all cases.	47
4.4	Results for swapping the gender of pronouns in documents that mention stereotyped professions to mitigate gender bias. Swapping pronouns is an effective intervention for mitigating bias across model sizes and stereotyped profession subsets (male and female).	49
4.5	Percentage of documents labeled relevant to their matching item by GPT-5 , for every dataset, prompt, and matching method.	53
4.6	Percentage of documents labeled relevant to their matching item by manual annotation , for every dataset, prompt, and matching method.	53
A.1	Test set MRR for the best KGE model compared to KG-PubMedBERT broken down by relation type for Hetionet and MSI.	85
A.2	Feature importances for GBDT router for a selection of most important features. Ranking scores output by each model tend to be the most important, with other graph- and text-based features also contributing.	86
C.1	Toy example that shows how token IDs in a batch of tokens are altered when replacing a target document with a source document. Note how the token IDs to the right of the target document are shifted to the right to accommodate the longer source document.	91
C.2	Number of items learned per pretraining step, for each model size and dataset.	94
C.3	Varying the method of matching documents to items results in different levels of performance reduction on ParaRel items. Selecting relevant documents using BM25 and DPR scores both lead to similar or less reduction in performance compared to selecting based on term cooccurrence alone, when removing the same number of documents per item or in total.	95

C.4	Modifying documents by replacing them with ones expressing the same ParaRel relation type but without having the term cooccurrence has a similar effect to replacing with random unrelated documents, indicating the importance of the specific entity pair being mentioned.	95
-----	--	----

List of Tables

2.1	Statistics for our datasets and a sample of general domain benchmarks.	11
2.2	KG completion results. All values are in the range [0, 100], higher is better. Underlined values denote the best result within a model category (KGE, LM, two models with router, two models with input-dependent weighted average, three models with router), while bold values denote the best result for each dataset.	13
2.3	Examples from Hetionet where one model ranks the shown positive pair considerably higher than the other. LMs often perform better when there is semantic relatedness between head and tail text, but can be outperformed by a KGE model when head/tail entity text is missing or unrelated. Entity descriptions cut to fit.	14
2.4	Inductive KG completion results. NN-ComplEx refers to the version of ComplEx with unseen entity embeddings replaced using an LM to find the 1-nearest neighbor, either with PubMedBERT frozen or fine-tuned for KG completion (KG-PubMedBERT).	17
3.1	Test set RMSE of mean baseline and various PP models finetuned to predict performance from task instruction for various IMs (columns). Subscript shows standard deviation across 10 splits of the SuperNI test tasks.	26
3.2	Test set RMSE of predicting Exact Match given task instruction, for various sizes of LLaMA models instruction-tuned on ShareGPT. Model performance in ROUGE-L on the SuperNI test set is given in parentheses next to the name of each model.	28
3.3	Test set RMSE of using RoBERTa-large to predict Exact Match given task instruction, using either just SuperNI instructions or SuperNI and BIG-bench instructions for training the PP model.	28
3.4	Test set RMSE of using RoBERTa-large to predict Exact Match given task instruction, using SuperNI tasks with an instruction-only prompt or the instruction with 2 positive demonstrations.	29
3.5	Test set RMSE of predicting cross-entropy loss given task instruction, for various LLaMA-based instruction-following models.	30

4.1	Results on <i>suppressing learning</i> for MCQA datasets. Results show the mean and standard deviation (subscript) across 3 training runs. $\Delta\text{acc.}$ is the change in accuracy as compared to the mean accuracy of the retrained model.	51
4.2	Results on <i>promoting learning</i> for MCQA datasets. Results show the mean and standard deviation (subscript) across 3 training runs. $\Delta\text{acc.}$ is the change in accuracy as compared to the mean accuracy of the retrained model.	51
A.1	Links and sources of entity names and descriptions for each dataset.	79
A.2	Complete list of features used by router classifiers.	84
A.3	Effect of descriptions on KG-PubMedBERT test set MRR.	85
A.4	Test set MRR for the best pair of a KGE model and KG-PubMedBERT for different methods of model integration.	86
A.5	Samples of unseen entities and their nearest neighbors found by KG-PubMedBERT and PubMedBERT, for test set examples in the Hetionet inductive split where the PubMedBERT neighbor performs better than the KG-PubMedBERT neighbor (first three) and vice versa (last three). Each LM offers a larger improvement per example when its nearest neighbor is more semantically related to the unseen entity.	87
B.1	Experiment settings for finetuning RoBERTa performance predictor models.	88
B.2	Experiment settings for finetuning LLaMA performance predictor models.	89
B.3	List of BIG-bench tasks included when training performance predictors on additional tasks.	90
C.1	Prompt templates used for document relevance annotation with GPT-5.	92
C.2	A sample of items from MMLU with example documents matched by BM25 score that were given “yes” and “no” annotations by GPT-5, for each of the three annotation prompts.	93
C.3	All relations we used from ParaRel along with number of items, query template, and an example item for each.	94
C.4	Additional details on MMLU, OpenBookQA, and SciQ including subsets used, number of items per dataset, and an example item.	95
C.5	Detailed information on the ParaRel interventions for each intervention type (suppressing learning by removing cooccurrences, suppressing learning by removing occurrences, and promoting learning by adding cooccurrences), model size, and set of pretraining steps. Values include the number of target items ($ \mathcal{E}^{\text{target}} $), percentage of documents changed in the data batch (% docs), percentage of tokens changed in the data batch (% tokens), and the Jensen-Shannon divergence between the token distributions of the original and modified data batches (JSD, takes values between 0 and 1).	96

C.6	Detailed information on the WinoBias interventions for each intervention type (reducing pro-male bias, reducing pro-female bias), model size, and set of pretraining steps. Values include the number of target items ($ \mathcal{E}^{\text{target}} $), percentage of documents changed in the data batch (% docs), percentage of tokens changed in the data batch (% tokens), and the Jensen-Shannon divergence between the token distributions of the original and modified data batches (JSD, takes values between 0 and 1).	96
C.7	Detailed information on the MCQA dataset interventions for each dataset, model size, set of pretraining steps, and item-document matching method. Values include the number of target items ($ \mathcal{E}^{\text{target}} $), percentage of documents changed in the data batch (% docs), percentage of tokens changed in the data batch (% tokens), and the Jensen-Shannon divergence between the token distributions of the original and modified data batches (JSD, takes values between 0 and 1).	97

Chapter 1

Introduction

Modern natural language technologies have undergone dramatic improvements at a rapid pace, powered largely by advancements in neural network language models (LMs) (Bengio et al., 2003). Driven by factors such as a novel neural network architecture (Vaswani et al., 2017), training on massive text corpora derived from the Internet (Raffel et al., 2020; Penedo et al., 2024; Soldaini et al., 2024; Li et al., 2024a), and scaling compute to develop increasingly larger models (Narayanan et al., 2021; Rae et al., 2022; Chowdhery et al., 2023; Shazeer et al., 2017), LM-based systems have become capable of performing a wide variety of useful tasks. More recently, new techniques have been applied to train LMs as general purpose digital assistants via a natural language “chatbot” interface (Ouyang et al., 2022; Lambert et al., 2025), leading to their widespread adoption far beyond the academic setting in which these models were traditionally developed and studied (OpenAI, 2022; Anthropic, 2023; Google, 2023). This growth in the usage of LM-based systems makes it even more crucial to understand their capabilities and limitations.

Among the many traits of LMs that have been expanded are their *knowledge* capabilities, which underpin a variety of useful tasks. LMs that store factual knowledge in their parameters and generate it when prompted can be useful for question answering (Rajpurkar et al., 2016; Roberts et al., 2020), fact verification (Thorne et al., 2018), knowledge base completion (Yao et al., 2019), and many other important information-centric tasks (Petroni et al., 2021). More recently, LMs have displayed knowledge capabilities related to performing a broad range of skills beyond simply generating or evaluating real-world facts, enabling behavior such as mathematical reasoning (Cobbe et al., 2021; Hendrycks et al., 2021b) and code generation (Austin et al., 2021; Chen et al., 2021), or more generally following arbitrary human-written instructions (Ouyang et al., 2022). This dichotomy can be described as advancements in *declarative* vs. *procedural* knowledge, a way of characterizing types of knowledge that originated in philosophy (Ryle, 1945) and was later used to describe knowledge and memory in cognitive science (ten Berge and van Hezewijk, 1999). While we note that this connection to the study of knowledge in living organisms does

not necessarily justify treating LMs as “intelligent” in the same way, the declarative/procedural split is a useful framework for describing and studying the capabilities we see in LMs today.

Generally, *declarative knowledge* can be described as the quality of “knowing that”, or being able to store and recall factual information that describes a property of the world. Early work demonstrating that LMs had developed declarative knowledge capabilities explored the idea of using LMs as knowledge bases with no additional training (Petroni et al., 2019), a capability which was shown to improve with model scale (Roberts et al., 2020) to the extent that it could sometimes even surpass the performance of models specifically trained to answer factual questions (Brown et al., 2020). This capability was also explored in other domains (Sung et al., 2021; Meng et al., 2022) and across multiple languages (Jiang et al., 2020; Kassner et al., 2021), further highlighting its usefulness. At the same time, issues with this approach have emerged, with evidence of models using shortcuts to complete factual knowledge (Poerner et al., 2020; Cao et al., 2021) as well as inconsistencies in knowledge that can be derived from model parameters versus elicited through natural language queries (Gekhman et al., 2025). A core deficit in factual knowledge capabilities is the issue of hallucinations, i.e., models generating factually incorrect information, a problem which continues to be studied (Lin et al., 2022b; Ji et al., 2023).

Early efforts to assess declarative knowledge in LMs tested models on short-form queries of factual information, which they could sometimes answer correctly even without being explicitly trained to do so (Petroni et al., 2019; Roberts et al., 2020; Brown et al., 2020). This form of evaluation was not without its limitations, with cases where models perform well by guessing based on entity types (Cao et al., 2021) or spurious correlations (Poerner et al., 2020), among other shortcuts. The rapid improvement of models’ declarative knowledge capabilities spurred development of more challenging and specialized benchmarks (Hendrycks et al., 2021a; Rein et al., 2024), often applied in a multiple-choice question answering format (Robinson and Wingate, 2023) with standardized evaluation methodology (Gao et al., 2021; Biderman et al., 2024; Gu et al., 2025). This form of evaluation has its own shortcomings, with models exhibiting sensitivity to factors such as answer choice order or formatting (Sclar et al., 2024; Pezeshkpour and Hruschka, 2024; Alzahrani et al., 2024). Furthermore, datasets themselves can contain flaws such as mislabeled instances or ones that allow models to make educated guesses without possessing the required knowledge (Gema et al., 2025; Balepur et al., 2026). Despite their weaknesses, these benchmarks and the associated evaluation methodology are crucial for properly assessing LMs’ evolving declarative knowledge capabilities, provided that they are applied with care and an awareness of their limitations.

In contrast to declarative knowledge, *procedural knowledge* can be characterized as “knowing how”, the ability to perform a task or skill that may not rely on using or recalling particular factual information. Initial evidence of this capability arose when LMs were trained to perform tasks by following natural language instructions, demonstrating their ability to extend this behavior to tasks that were not explicitly

included during training (Mishra et al., 2022; Wang et al., 2022; Sanh et al., 2022; Wei et al., 2022; Ouyang et al., 2022). This ability in particular has been capitalized on to develop general purpose LM-based systems that have become commercially available, making LMs a much more pervasive part of modern society outside the research community (OpenAI, 2022; Anthropic, 2023; Google, 2023). Shortcomings exist here as well, with evidence that instruction-following models struggle with phenomena like negation (Jang et al., 2023b), ignore parts of the instruction they are given (Yin et al., 2023), perform well even with a misleading instruction (Webson and Pavlick, 2022), and generally struggle to follow instructions when given specific constraints that are programmatically verifiable (Zhou et al., 2023b; Pyatkin et al., 2025).

Early work evaluating procedural knowledge in LMs used large collections of existing natural language understanding tasks combined with metrics like Exact Match and ROUGE-L that compared each model output to a reference output (Wei et al., 2022; Sanh et al., 2022; Wang et al., 2022; Chung et al., 2024; Longpre et al., 2023). While such metrics are easy to compute, deterministic, and interpretable, they are insufficient for evaluating procedural knowledge across diverse tasks, especially those where LMs can produce open-ended generations and quality is either subjective or there is an indeterminate number of acceptable responses. Simple, deterministic evaluations are still possible in certain domains, such as unit tests for code generation (Chen et al., 2021; Jimenez et al., 2024) or programmatic verification of precise instruction-following (Zhou et al., 2023b; Pyatkin et al., 2025). For a wider range of tasks however, recent work has instead turned to more flexible evaluation methods such as large-scale collection of human preferences (Chiang et al., 2024a) or using LMs to evaluate model outputs via an “LLM-as-a-judge” setup (Li et al., 2023; Zheng et al., 2023). These approaches have limitations – human evaluation is not easily scalable and difficult to obtain for tasks requiring deep domain expertise, while LLM-as-a-judge can exhibit biases that hinder fair evaluation (Chen et al., 2024; Wang et al., 2024; Koo et al., 2024). While progress relies on applying each of these evaluation methods in the appropriate setting, robust and general evaluation of procedural knowledge in LMs remains challenging and is a subject of ongoing research.

In sum, current work demonstrates that while LMs’ knowledge-related abilities have grown rapidly, many limitations exist and there is much we still don’t know about their knowledge capabilities. In particular, several questions deserve further study to expand our understanding of knowledge in LMs:

- What is the extent of *declarative knowledge* in LMs? How well do they perform at tasks that require knowledge in specialized domains?
- Is it possible to determine the limits of *procedural knowledge* in LMs based on their past behavior? Can we predict performance on unseen tasks given performance on seen ones?
- What are the *mechanisms* that give rise to knowledge in LMs? How do properties of LMs’ training data lead to knowledge acquisition?

In this thesis, we present work that takes steps towards answering each of these questions, expanding our understanding of both declarative and procedural knowledge in LMs.

In Chapter 2, we study the extent of declarative knowledge in pretrained LMs by evaluating the performance of general and domain-specific LMs at the task of knowledge graph completion in the biomedical domain. We compare LMs to classical models that rely on the structure of the knowledge graph without making use of textual information about entities, highlighting the different strengths of the two approaches. We then make use of their complementary benefits by exploring various methods of ensembling the two classes of methods, demonstrating that this greatly improves performance, particularly when an LM is included in the ensemble. Finally, we show that LMs are particularly valuable when making predictions involving entities that do not have known connections to other entities, even compared to other baselines that can make use of textual information about these novel entities.

Chapter 3 focuses on procedural knowledge in LMs by training models to predict the performance of instruction-following LMs given a task instruction alone, attempting to generalize from observed behavior on instructions that the LMs have already been evaluated on. We find that performance prediction is incredibly challenging, with a wide range of methods failing to adequately predict performance on unseen instructions. Varying the size of the instruction-following LM, the size of the performance predictor, the number of training tasks, and even the quantitative evaluation metric being predicted all fail to improve performance. While results predicting language modeling loss show some promise, we generally find that while LM instruction-following behavior shows remarkable performance and ability to generalize to novel instructions, our study did not identify a clearly defined structure as a function of task instruction alone.

In Chapter 4 we perform a deeper analysis that expands our understanding of how LMs' pretraining data gives rise to their declarative knowledge capabilities. To do so, we present a general recipe for performing interventional analyses on pretraining, outlining considerations for identifying and modifying pretraining documents to perform a controlled intervention followed by repeating portions of pretraining to observe the effects on model behavior. We instantiate this recipe on tasks that evaluate factual knowledge recall as well as gender bias, demonstrating that our interventions can successfully promote or suppress the expected behavior for each task while noting that some level of performance still remains unexplained. Our experiments explore a range of methods of implementing each of the steps in our recipe, showing that our proposed framework can easily be applied to study a wide variety of model behaviors.

We conclude in Chapter 5 with a summary of the work presented here as well as suggestions for future work that could extend each research direction in order to further explore LMs' knowledge capabilities. The work in this thesis adds to a growing literature on understanding knowledge in LMs, while providing additional tools to extend this line of inquiry in future research.

Chapter 2

Scientific Language Models for Biomedical Knowledge Base Completion: An Empirical Study

We begin with a study of *declarative knowledge* in LMs, one of the first kinds of knowledge capabilities discovered in modern models. This chapter evaluates declarative knowledge present in LMs by adapting them for scientific knowledge base completion, filling in missing links between entities in the biomedical domain. Leveraging LMs’ declarative knowledge about connections between concepts in specialized domains has potentially high value, assisting domain experts in dealing with the vast and growing knowledge in their specialty. We compare LMs to language-free methods of performing this task, combine the two classes of methods to improve performance, analyze what factors make each approach perform well alone and how to best combine them, and conclude with the special case of using LMs to make predictions about entities with no known connections to existing entities.

This work was originally presented in [Nadkarni et al. \(2021\)](#).

2.1 Introduction

Understanding complex diseases such as cancer, HIV, and COVID-19 requires rich biological, chemical, and medical knowledge. This knowledge plays a vital role in the process of discovering therapies for these diseases — for example, identifying targets for drugs ([Lindsay, 2003](#)) requires knowing what genes or proteins are involved in a disease, and designing drugs requires predicting whether a drug molecule will interact with specific target proteins. In addition, to alleviate the great costs of designing new drugs, drug repositioning ([Luo et al., 2021](#)) involves identification of *existing* drugs that can be re-purposed for other

diseases. Due to the challenging combinatorial nature of these tasks, there is need for automation with machine learning techniques. Given the many links between biomedical entities, recent work (Bonner et al., 2022a,b) has highlighted the potential benefits of *knowledge graph* (KG) data representations, formulating the associated tasks as *KG completion* problems — predicting missing links between drugs and diseases, diseases and genes, and so forth.

The focus of KG completion work — in the general domain, as well as in biomedical applications — is on using graph structure to make predictions, such as with KG embedding (KGE) models and graph neural networks (Zitnik et al., 2018; Chang et al., 2020). In parallel, recent work in the general domain has explored the use of pretrained language models (LMs) as “soft” knowledge bases, holding factual knowledge latently encoded in their parameters (Petroni et al., 2019, 2020). An emerging direction for using this information for the task of KG completion involves fine-tuning LMs to predict relations between pairs of entities based on their textual descriptions (Yao et al., 2019; Kim et al., 2020; Wang et al., 2021; Daza et al., 2021). In the scientific domain, this raises the prospect of using LMs trained on millions of research papers to tap into the scientific knowledge that may be embedded in their parameters. While this text-based approach has been evaluated on general domain benchmarks derived from WordNet (Miller, 1995) and Freebase (Bollacker et al., 2008), to our knowledge it has not been applied to the task of scientific KG completion prior to this work.

In this chapter we perform an extensive study of LM-based KG completion in the biomedical domain, focusing on three datasets centered on drugs and diseases, two of which have not previously been used for the KG completion task. To enable exploration of LM-based models, we collect missing entity descriptions, obtaining them for over 35k entities across all datasets. We evaluate a range of KGE models and domain-specific scientific LMs pretrained on different biomedical corpora (Beltagy et al., 2019; Lee et al., 2020; Alsentzer et al., 2019; Gu et al., 2021). We conduct analyses of predictions made by both types of models and find them to have complementary strengths, echoing similar observations made in recent work in the general domain (Wang et al., 2021) and motivating integration of both text and graph modalities. Unlike previous work, we train a router that selects for each input instance which type of model is likely to do better, finding it to often outperform average-based ensembles. Integration of text and graph modalities provides substantial relative improvements of 13–36% in mean reciprocal rank (MRR), and routing across multiple LM-based models further boosts results. Finally, we demonstrate the utility of LM-based models when applied to entities unseen during training, an important scenario in the rapidly evolving scientific domain. Our hope is that this work will encourage further research into using scientific LMs for biomedical KG completion, tapping into knowledge embedded in these models for making relational inferences between complex scientific concepts.

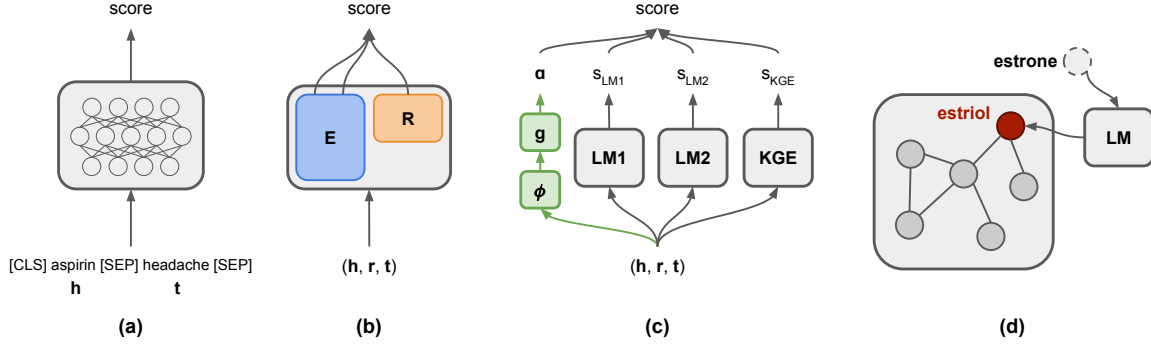


Figure 2.1: Illustration of the main methods we apply for biomedical KG completion: (a) LM fine-tuning; (b) KGE models; (c) an approach that combines both; and (d) using an LM to impute missing entities in a KGE model.

2.2 Task and Methods

We begin by presenting the KG completion task and the approaches we employ for predicting missing links in biomedical KGs, including our model integration and inductive KG completion methodologies. An overview of our approaches is illustrated in Figure 3.1.

2.2.1 KG Completion Task

Formally, a KG consists of entities $\mathcal{E}$, relations $\mathcal{R}$, and triples $\mathcal{T}$ representing *facts*. Each triple $(h, r, t) \in \mathcal{T}$ consists of head and tail entities $h, t \in \mathcal{E}$ and a relation $r \in \mathcal{R}$. An entity can be one of many types, with the type of an entity e denoted as $T(e)$. In our setting, each entity is also associated with some text, denoted as $\text{text}(e)$ for $e \in \mathcal{E}$. This text can be an entity name, description, or both; we use the entity’s name concatenated with its description when available, or just the name otherwise. For example, the fact $(\text{aspirin}, \text{treats}, \text{headache})$ might be an (h, r, t) triple found in a biomedical KG that relates drugs and diseases, with the head and tail entities having types $T(h) = \text{drug}$ and $T(t) = \text{disease}$.

The task of *KG completion* or *link prediction* involves receiving a triple $(h, r, ?)$ (where $?$ can replace either the head or tail entity) and scoring all candidate triples $\{(h, r, t') \mid t' \in \mathcal{S}\}$ such that the correct entity that replaces $?$ has the highest score. For the example listed above, a well-performing model that receives the incomplete triple $(\text{aspirin}, \text{treats}, ?)$ should rank the tail entity *headache* higher than an incorrect one such as *diabetes*. $\mathcal{S}$ can be the entire set of entities (i.e., $\mathcal{S} = \mathcal{E}$) or some fixed subset. In the *transductive* setting, the set of facts $\mathcal{T}$ is split into a training set $\mathcal{T}_{\text{train}}$ and a test set $\mathcal{T}_{\text{test}}$ such that all positive triples in the test set contain entities seen during training. In contrast, for *inductive* KG completion the triples in the test set may contain entities not seen during training (see Section 2.2.4).

2.2.2 Methods

Ranking-based KG completion. Each KG completion model in our experiments learns a function f that computes a ranking score $s = f(x)$ for a given triple $x = (h, r, t)$. Models are trained to assign a high ranking score to correct positive triples from the set of known facts $\mathcal{T}$ and a low ranking score to triples that are likely to be incorrect. To do so, we use the max-margin loss function $\mathcal{L}_{\text{rank}}(x) = \frac{1}{N} \sum_{i=1}^N \max(0, \lambda - f(x) + f(x'_i))$, where λ is a margin hyperparameter, $x \in \mathcal{T}$ is a known positive triple in the KG, and x'_i is a negative triple constructed by randomly corrupting either the head or tail entity of x with an entity of the same type. We generate N random negatives for each positive, with the value of N being a hyperparameter that may vary depending on the model and dataset.

KG embedding (KGE) models. For each entity $e \in \mathcal{E}$ and each relation $r \in \mathcal{R}$, KG embedding (KGE) models learn a vector representation $E(e) \in \mathbb{R}^m$ and $R(r) \in \mathbb{R}^n$. For a given triple (h, r, t) , each model computes the ranking score $f(h, r, t)$ as a simple function of these embeddings (Figure 3.1b). We include a variety of different KGE models in our experiments, including TransE (Bordes et al., 2013), DistMult (Yang et al., 2015), ComplEx (Trouillon et al., 2016), and RotatE (Sun et al., 2019).

LM-based models. KGE methods do not capture the rich information available from textual descriptions of nodes. To address this limitation, previous KG completion approaches have incorporated textual representations (Toutanova et al., 2015; Wang and Li, 2016), most recently with approaches such as KG-BERT (Yao et al., 2019) that fine-tune the BERT language model (LM) (Devlin et al., 2019) for the task of KG completion. Our focus in this work is on LMs pretrained on corpora of biomedical documents (e.g., PubMedBERT (Gu et al., 2021); see Appendix A.2.1 for full details). To score a triple using an LM, we use a cross-encoder approach (Yao et al., 2019; Kim et al., 2020) (Fig. 3.1a), where we encode the text of the head and tail entities together along with the appropriate special tokens. Specifically, a triple (h, r, t) is encoded as $v = \text{LM}([\text{CLS}] \text{ text}(h) [\text{SEP}] \text{ text}(t) [\text{SEP}])$, where v is the contextualized representation of the [CLS] token at the last layer.¹ We then apply an additional linear layer with a single output dimension to v to compute the ranking score for the triple ($f(x) = W_{\text{rank}}v \in \mathbb{R}$), and train the LM with the same max-margin loss. Recent work on applying BERT for KG completion on general domain benchmarks has shown that multi-task training improves performance (Wang et al., 2021; Kim et al., 2020). We use the approach of Kim et al. (2020) and incorporate two additional losses for each LM: a binary triple classification loss to identify if a triple is positive or negative, and a multi-class relation classification loss.²

¹We experiment with encoding the relation text as well, but find that this did not improve performance.

²See details in Appendix A.2. Wang et al. (2021) omit the relation classification loss and use a bi-encoder; we find that both of these modifications reduce performance in our setting.

2.2.3 Integrating KGE and LM: Model Averaging vs. Routing

Previous work using text for KG completion on general domain benchmarks has demonstrated the benefit of combining KGE and text-based models (Xie et al., 2016; Wang et al., 2021). We study integration of graph-based and text-based methods (Figure 3.1c), exploring whether learning to route input instances adaptively to a *single* model can improve performance over previous approaches that compute a weighted average of ranking scores (Wang et al., 2021). We also explore the more general setup of combining more than two models.

More formally, for a given triple $x = (h, r, t)$, let $\phi(x)$ be its feature vector. We can learn a function $g(\phi(x))$ that outputs a set of weights $\alpha = [\alpha_1, \dots, \alpha_k]$, $\sum_i \alpha_i = 1$, $\alpha_i > 0 \forall i$. These weights can be used to perform a weighted average of the ranking scores $\{s_1, \dots, s_k\}$ for a set of k models we wish to combine, such that the final ranking score is $s = \sum_i \alpha_i s_i$. We use a variety of graph-, triple-, and text-based features to construct the feature vector $\phi(x)$ such as node degree, entity and relation type, string edit distance between head and tail entity names, and overlap in graph neighbors of head and tail nodes. We explore these features further in Section 2.4.1, and provide a full list in Appendix A.2.1 (Table A.2).

For the function $g(\cdot)$, we experiment with an **input-dependent weighted average** that outputs arbitrary weights α and a **router** that outputs a constrained α such that $\alpha_i = 1$ for some i and $\alpha_j = 0, \forall j \neq i$ (i.e., α is a one-hot vector).³ In practice, we implement the router as a classifier which selects a single KG completion model for each example by training it to predict which model will perform better.⁴ For the input-dependent weighted average we train a multilayer perceptron (MLP) using the max-margin ranking loss. We train all models on the validation set and evaluate on the test set for each dataset. When performing ranking evaluation, we use the features $\phi(x)$ of each positive example to compute the weights α , then apply the same weights to all negative examples ranked against that positive example.

2.2.4 Inductive KG Completion

KGE models are limited to the *transductive* setting where all entities seen during evaluation have appeared during training. *Inductive* KG completion is important in the biomedical domain, where we may want to make predictions on novel entities such as emerging biomedical concepts or drugs/proteins mentioned in the literature that are missing from existing KGs. Due to their ability to form compositional representations from entity text, LMs are well-suited to this setting. In addition to using LMs fine-tuned for KG completion, we try a simple technique using LMs to “fill in” missing KGE embeddings without explicitly using the LM for prediction (Fig. 3.1d). Given a set of entities $\mathcal{E}$ for which a KGE model has

³We also try a global weighted average with a single set of weights; see Appendix A.2.1 for details.

⁴We explore a range of methods for the router’s classifier, with the best found to be gradient boosted decision trees (GBDT) and multilayer perceptrons (MLP).

Dataset	#Entities	#Rel	#Positive Edges			Avg. Desc. Length
			Train	Dev.	Test	
RepoDB	2,748	1	5,342	667	668	49.54
Hetionet (our subset)	12,733	4	124,544	15,567	15,568	44.65
MSI	29,959	6	387,724	48,465	48,465	45.13
WN18RR	40,943	11	86,835	3,034	3,134	14.26
FB15k-237	14,541	237	272,115	17,535	20,466	139.32

Table 2.1: Statistics for our datasets and a sample of general domain benchmarks.

trained embeddings and a set of unknown entities $\mathcal{U}$, for each $e \in \mathcal{E} \cup \mathcal{U}$ we encode its text using an LM to form $v_e = \text{LM}([\text{CLS}] \text{ text}(e) [\text{SEP}])$, $\forall e \in \mathcal{E} \cup \mathcal{U}$, where v_e is the [CLS] token representation at the last layer. We use the cosine similarity between embeddings to replace each unseen entity’s embedding with the closest trained embedding as $E(u) = E(\arg\max_{e \in \mathcal{E}} \cos\text{-sim}(v_e, v_u))$ where e is of the same type as u , i.e., $T(e) = T(u)$.

2.3 Experimental Setup

2.3.1 Datasets

We use three datasets in the biomedical domain that cover a range of sizes comparable to existing general domain benchmarks, each pooled from a broad range of biomedical sources. Our datasets include **RepoDB** (Brown and Patel, 2017), a collection of drug-disease pairs intended for drug repositioning research; **MSI** (multiscale interactome; (Ruiz et al., 2021)), a network of diseases, proteins, genes, drug targets, and biological functions; and **Hetionet** (Himmelstein and Baranzini, 2015), a heterogeneous biomedical knowledge graph which following Alshahrani et al. (2021) we restrict to interactions involving drugs, diseases, symptoms, genes, and side effects.⁵ Statistics for all datasets and a sample of popular general domain benchmark KGs can be found in Table 2.1.

While Hetionet has previously been explored for the task of KG completion as link prediction using KGE models (though not LMs) (Alshahrani et al., 2021; Bonner et al., 2022b), to our knowledge neither RepoDB nor MSI have been represented as KGs and used for evaluating KG completion models prior to this work, despite the potential benefits of this representation (Bonner et al., 2022a) especially in conjunction with textual information. In order to apply LMs to each dataset, we scrape entity names

⁵More information on each dataset is available in Appendix A.1.1.

(when not provided by the original dataset) as well as descriptions from the original online sources used to construct each KG (see Table A.1 in the appendix).

We construct an 80%/10%/10% training/development/test transductive split for each KG by removing edges from the complete graph while ensuring that all nodes remain in the training graph. We also construct inductive splits, where each positive triple in the test set has one or both entities unseen during training. Details on how these splits were performed can be found in Appendix A.1.2 for the transductive split and Appendix A.1.3 for the inductive split.

2.3.2 Pretrained LMs and KGE Integration

We experiment with several LMs pretrained on biomedical corpora (see Table 2.2 and Appendix A.2.1). For each LM that has been fine-tuned for KG completion, we add the prefix “KG-” (e.g., KG-PubMedBERT) to differentiate it from the base LM. We use the umbrella term “model integration” for both model averaging and routing, unless stated otherwise.

Model integration. We explore integration of all pairs of KGE models as well as each KGE model paired with KG-PubMedBERT. This allows us to compare the effect of integrating pairs of KG completion models in general with integrating graph- and text-based approaches. For all pairs of models, we use the router-based and input-dependent weighted average methods. We also explore combinations of multiple KGE models and LMs, where we start with the best pair of KG-PubMedBERT and a KGE model based on the validation set and add either KG-BioBERT or the best-performing KGE model (or the second-best, if the best KGE model is in the best pair with KG-PubMedBERT).

2.3.3 Evaluation

At test time, each positive triple is ranked against a set of negatives constructed by replacing either the head or tail entity by a fixed set of entities of the same type. When constructing the edge split for each of the three datasets, we generate a fixed set of negatives for every positive triple in the validation and test sets, each corresponding to replacing the head or tail entity with an entity of the same type and filtering out negatives that appear as positive triples in either the training, validation, or test set (exact details in Appendix A.1.4). For each positive triple, we use its rank to compute the mean reciprocal rank (MRR), Hits@3 (H@3), and Hits@10 (H@10) metrics.

		RepoDB			Hetionet			MSI		
		MRR	H@3	H@10	MRR	H@3	H@10	MRR	H@3	H@10
KGE	ComplEx	<u>62.3</u>	<u>71.1</u>	<u>85.6</u>	45.9	53.6	77.8	<u>40.3</u>	<u>44.3</u>	<u>57.5</u>
	DistMult	62.0	70.4	85.2	46.0	53.5	77.8	29.6	34.1	53.6
	RotatE	58.8	65.9	79.8	<u>50.6</u>	<u>58.2</u>	79.3	32.4	35.3	49.8
	TransE	60.0	68.6	81.1	50.2	58.0	<u>79.8</u>	32.7	36.5	53.8
LM (fine-tuned)	RoBERTa	51.7	60.3	82.3	46.4	53.6	76.9	30.1	33.3	50.6
	SciBERT	59.7	67.6	88.5	50.3	57.1	79.1	34.2	37.9	55.0
	BioBERT	58.2	65.8	86.8	50.3	57.5	79.4	33.4	37.1	54.8
	Bio+ClinicalBERT	55.7	64.0	84.1	43.6	49.1	72.6	32.6	36.1	53.5
	PubMedBERT-abs	<u>60.8</u>	<u>70.7</u>	<u>89.5</u>	50.8	58.0	80.0	<u>34.3</u>	<u>38.0</u>	<u>55.3</u>
	PubMedBERT-full	59.9	69.3	88.8	<u>51.7</u>	<u>58.7</u>	<u>80.8</u>	34.2	37.7	55.1
Two models (router)	Best pair of KGE	62.2	70.4	83.7	56.1	65.5	85.4	45.2	50.6	66.2
	Best KGE + LM	<u>70.6</u>	<u>80.3</u>	<u>94.3</u>	<u>59.7</u>	<u>68.6</u>	<u>87.2</u>	<u>48.5</u>	<u>54.4</u>	<u>70.1</u>
Two models (input-dep. avg.)	Best pair of KGE	65.2	74.3	87.6	65.3	75.3	90.2	39.8	<u>44.9</u>	<u>62.0</u>
	Best KGE + LM	<u>65.9</u>	<u>74.4</u>	<u>91.5</u>	70.3	78.7	92.2	<u>40.6</u>	44.6	61.2
Three models (router)	2 KGE + 1 LM	72.7	81.6	95.2	<u>62.6</u>	71.7	89.4	50.9	57.1	73.2
	1 KGE + 2 LM	72.1	82.5	95.7	62.1	<u>71.9</u>	<u>89.5</u>	51.2	57.0	73.0

Table 2.2: KG completion results. All values are in the range [0, 100], higher is better. Underlined values denote the best result within a model category (KGE, LM, two models with router, two models with input-dependent weighted average, three models with router), while bold values denote the best result for each dataset.

2.4 Experimental Results

2.4.1 Transductive Link Prediction Results

We report performance on the link prediction task across all datasets and models in Table 2.2, with full training details for the transductive setting in Appendix A.2.1. While LMs perform competitively with KGE models and even outperform some, they generally do not match the best KGE model on RepoDB and MSI. This echoes results in the general domain for link prediction on subsets of WordNet and Freebase (Yao et al., 2019; Wang et al., 2021). On all datasets and metrics, the best-performing LM is KG-PubMedBERT, which aligns with results for natural language understanding tasks over biomedical text (Gu et al., 2021). The biomedical LMs also generally outperform KG-RoBERTa, illustrating the benefit of in-domain pretraining even in the KG completion setting.

Relation	RotatE better	KG-PubMedBERT better
Disease <i>presents</i> Symptom	Disease: mediastinal cancer ; a cancer in the mediastinum. Symptom: hoarseness ; a deep or rough quality of voice.	Disease: stomach cancer ; a gastrointestinal cancer in the stomach. Symptom: weight loss ; decrease in existing body weight.
Compound <i>treats</i> Disease	Compound: methylprednisolone ; a prednisolone derivative glucocorticoid with higher potency. Disease: allergic rhinitis ; a rhinitis that is an allergic inflammation and irritation of the nasal airways.	Compound: altretamine ; an alkylating agent proposed as an antineoplastic. Disease: ovarian cancer ; a female reproductive organ cancer that is located in the ovary.
Compound <i>causes</i> Side Effect	Compound: cefactor ; semi-synthetic, broad-spectrum anti-biotic derivative of cephalexin. Side Effect: tubulointerstitial nephritis ; <i>no description</i>	Compound: perflutren ; a diagnostic medication to improve contrast in echocardiograms. Side Effect: palpitations ; irregular and/or forceful beating of the heart.

Table 2.3: Examples from Hetionet where one model ranks the shown positive pair considerably higher than the other. LMs often perform better when there is semantic relatedness between head and tail text, but can be outperformed by a KGE model when head/tail entity text is missing or unrelated. Entity descriptions cut to fit.

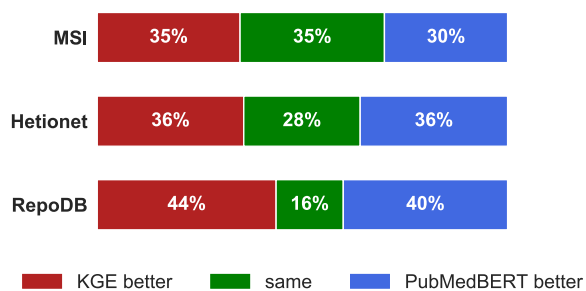


Figure 2.2: Fraction of test set examples where each model performs better.

Comparing model errors. By examining a selected set of examples in Table 2.3, we can observe cases where information in text provides LMs an advantage and where a lack of context favors KGE models. KG-PubMedBERT is able to make connections between biomedical concepts – like the fact that a disease that affects the *stomach* might cause *weight loss* – and align related concepts expressed with different terminology – like connecting *anti-neoplastic* with *cancer* (a type of *neoplasm*), or recognizing that an *echocardiogram* is a technique for imaging the *heart*. In contrast, RotatE offers an advantage when the descriptions do not immediately connect the two terms (*mediastinal cancer*, *hoarseness*), where a description may be too technical or generic to be informative (*methylprednisolone*, *allergic rhinitis*), or where no description is available

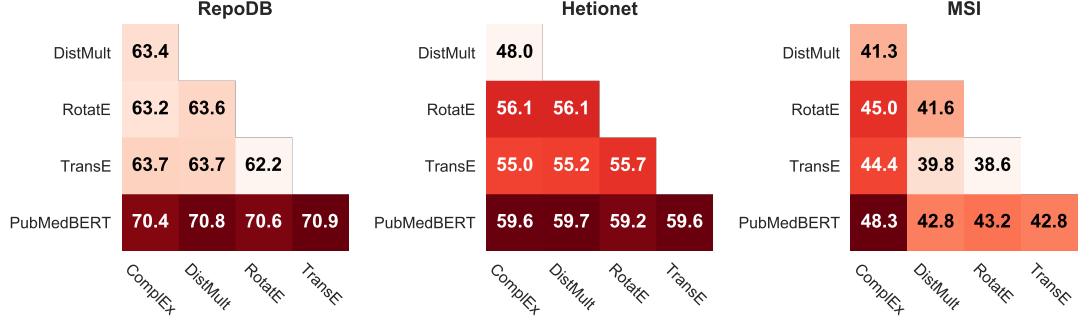


Figure 2.3: Test set MRR for all pairs of KG completion models using an MLP router. The best combination of a KGE model and KG-PubMedBERT always performs better than the best pair of KGE models, and for RepoDB and Hetionet all pairs involving KG-PubMedBERT outperform all KGE-only pairs.

(*cefactor*, *tubulointerstitial nephritis*).⁶ Furthermore, Fig. 2.2 shows that KG-PubMedBERT outperforms the best KGE model on a substantial fraction of the test set examples for each dataset.⁷ These observations motivate an approach that leverages the strengths of both types of models by identifying examples where each model might do better, which leads to our results for model integration.

2.4.2 Model Averaging and Routing

Integrating pairs of models. Table 2.2 shows that combining each class of models boosts results by a large relative improvement of 13–36% in MRR across datasets. Moreover, the best-performing combination always includes a KGE model and KG-PubMedBERT rather than two KGE models (Fig. 2.3), showing the unique benefit of using LMs to augment models relying on KG structure alone.

Averaging vs. routing. We also compare the router and input-dependent weighted average approaches of integrating a pair of models in Table 2.2, with the router-based approach outperforming the weighted average for the best KGE + LM pair on RepoDB and MSI. This presents routing as a promising alternative for integrating KGE and LM models. Since the gradient boosted decision trees (GBDT) router achieves the best validation set performance in most cases across classifiers and integration methods, we use this method for combinations of more than two models, such as multiple LMs with a single KGE model.

Integrating additional models. The bottom of Table 2.2 shows results for three-model combinations. Adding a third model improves performance compared to the best pair of models, though whether the best third model is an LM or KGE model varies across datasets and metrics. Although there are diminishing

⁶Table A.3 in the appendix shows the drop in performance when one or both entities are missing descriptions.

⁷See MRR breakdown by relation type in Fig. A.1 in the appendix.

returns to including a third model, the three-model combinations provide the best performance for RepoDB and MSI.

Interpreting model routing. We compute average feature gain for all datasets, using a GBDT router implemented with XGBoost (Chen and Guestrin, 2016) (see Fig. A.2 in the appendix). We find that the most salient features are the ranking scores output by each model, which is intuitive as these scores reflect each model’s confidence. Graph features like node degree and PageRank also factor into the classifier’s predictions, as well as textual features such as entity text length and edit distance between entity names. General concepts such as *Hypertensive disease* and *Infection of skin and/or subcutaneous tissue* are central nodes for which we observe KGE models to often do better. KGE models also tend to do better on entities with short, non-descriptive names (e.g., *P2RY14*), especially when no descriptions are available. Generally, these patterns are not clear-cut, and non-linear or interaction effects likely exist. It remains an interesting challenge to gain deeper understanding into the strengths and weaknesses of LM-based and graph-based models.

2.4.3 Inductive KG Completion

For our inductive KG completion experiments⁸, we use ComplEx as the KGE model and KG-PubMedBERT as our LM-based model, and compare the performance of each method to ComplEx with entity embeddings imputed using the method described in Section 2.2.4. We use either the untrained PubMedBERT or the fine-tuned KG-PubMedBERT as the LM for retrieving nearest-neighbor (NN) entities (see examples in Table A.5 in the appendix). We also compare to DKRL (Xie et al., 2016), which constructs entity representations from text using a CNN encoder and uses the TransE scoring function. We use PubMedBERT’s token embeddings as input to DKRL and train with the same multi-task loss. While other methods for inductive KG completion exist, such as those based on graph neural networks (Schlichtkrull et al., 2018; Vashishth et al., 2020; Bhowmik and de Melo, 2020), they require the unseen entity to have *known connections* to entities that were seen during training in order to propagate information needed to construct the new embedding. In our inductive experiments, we consider the more challenging setup where every test set triple has at least one entity with no known connections to entities seen during training, such that graph neural network-based methods cannot be applied. This models the phenomenon of rapidly emerging concepts in the biomedical domain, where a novel drug or protein may be newly studied and discussed in the scientific literature without having been integrated into existing knowledge bases.

As seen in Table 2.4, ComplEx unsurprisingly performs poorly as it attempts link prediction with random embeddings for unseen entities. DKRL does substantially better, with KG-PubMedBERT further

⁸Full experimental details in Appendix A.2.2.

	RepoDB			Hetionet			MSI		
	MRR	H@3	H@10	MRR	H@3	H@10	MRR	H@3	H@10
DKRL	15.6	15.9	28.2	17.8	18.5	31.9	13.3	14.1	22.4
KG-PubMedBERT	38.8	43.4	67.5	21.6	22.3	42.8	20.2	21.7	32.2
ComplEx	0.8	0.4	1.6	3.6	0.7	2.8	0.5	0.1	0.4
NN-ComplEx, frozen LM	20.1	22.3	31.2	18.1	18.4	32.8	15.8	16.9	23.4
NN-ComplEx, fine-tuned LM	26.9	30.3	39.4	13.9	12.9	25.5	14.6	15.4	21.4

Table 2.4: Inductive KG completion results. NN-ComplEx refers to the version of ComplEx with unseen entity embeddings replaced using an LM to find the 1-nearest neighbor, either with PubMedBERT frozen or fine-tuned for KG completion (KG-PubMedBERT).

increasing MRR with a relative improvement of 21% (Hetionet) to over 2x (RepoDB). Our strategy for replacing ComplEx embeddings for unseen entities performs comparably to or better than DKRL in most cases, with untrained PubMedBERT encodings generally superior to using KG-PubMedBERT’s encodings. In either case, this simple strategy for replacing the untrained entity embeddings of a KGE model shows the ability of an LM to augment a structure-based method for KG completion that is typically only used in the transductive setting, even without using the LM to compute ranking scores.

2.5 Recent Work

Subsequent research has referenced the findings of our work in various ways, whether by using our datasets directly to evaluate novel approaches for knowledge graph completion (Safavi et al., 2022) or by recognizing our highlighting of the importance of combining structural and textual forms of knowledge for building better biomedical knowledge graphs (Chandak et al., 2023) or domain-specific models (Balabin et al., 2022). Separately, other work has produced datasets specifically for evaluating the biomedical knowledge capabilities of language models, recognizing their utility as sources of information in this domain (Sung et al., 2021; Meng et al., 2022). As LMs’ knowledge capabilities have advanced, the question of whether or not LMs can function as knowledge bases has been revisited, with mixed results (He et al., 2025; Sun et al., 2024). One particular shortcoming is LMs’ difficulty dealing with long-tail knowledge (Mallen et al., 2023; Kandpal et al., 2023), which is especially relevant in the biomedical domain where connections to rare, understudied drugs or diseases may yield transformative insights. While the declarative knowledge capabilities of LMs have advanced considerably, much work remains in addressing such limitations of LM-based systems used as knowledge repositories.

2.6 Conclusion and Discussion

We perform the first empirical study of scientific language models (LMs) applied to biomedical knowledge graph (KG) completion. We evaluate *domain-specific* biomedical LMs, fine-tuning them to predict missing links in KGs that we construct by enriching biomedical datasets with textual entity descriptions. We find that LMs and more standard KG embedding models have complementary strengths, and propose a routing approach that integrates the two by assigning each input example to either type of model to boost performance. Finally, we demonstrate the utility of LMs in the inductive setting with entities not seen during training, an important scenario in the scientific domain with many emerging concepts.

Our work raises several directions for further study. For instance, several structural differences exist between general-domain and biomedical text that would be interesting to explore in more depth and leverage more explicitly to improve KG completion performance. For example, entities with uninformative technical names – such as protein names that are combinations of numbers and letters (e.g., P2RY14) – appear very often in scientific KGs, and are likely related to the benefit of adding descriptions (Table A.3, appendix). The surface forms of entity mentions in the biomedical literature on which the LMs were pretrained tend to be diverse with many aliases, while entities such as cities or people in the general domain often show less variety in their surface forms used in practice. This could potentially be challenging when trying to tap into the latent knowledge LMs hold on specific entities as part of the KG completion task, and likely requires LMs to disambiguate these surface forms to perform the task well. General-domain LMs are also trained on corpora such as Wikipedia which has “centralized” pages with comprehensive information about entities, while in the scientific literature information on entities such as drugs or genes is scattered across the many papers that form the training corpora for the LMs.

Previous work (Wang et al., 2021) has also observed that combining graph- and LM-based models improves KG completion results. We provide further analyses into this phenomenon based on textual and graph properties, but a deeper understanding of the strengths and weaknesses of each modality is needed. Interpreting neural models is generally a challenging problem; further work in our setting could help reveal the latent scientific knowledge embedded in language models. Importantly, our results point to the potential for designing new models that capitalize on both graph and text modalities, perhaps by injecting structured knowledge into LMs (Peters et al., 2019) or with entity-centric pretraining (Zemlyanskiy et al., 2021). Finally, our findings provide a promising direction for biomedical knowledge completion tasks, and for literature-based scientific discovery (Swanson, 1986; Gopalakrishnan et al., 2019).

Chapter 3

Third-Party Language Model Performance Prediction from Instruction

In the previous chapter we focused on declarative knowledge, finding that LMs demonstrate great potential in expressing domain-specific knowledge in a useful way. Not long after the discovery that pretrained LMs possess this kind of knowledge, subsequent work discovered that LMs could be trained to follow task instructions and adapt to unseen tasks, demonstrating *procedural knowledge* capabilities as well. However, early work on procedural knowledge in LMs was missing a more comprehensive understanding of how training LMs on certain tasks leads to improved performance on unseen ones, and whether or not this phenomenon is predictable. We explore exactly this question in this chapter, finding that predicting unseen task performance from tasks that the LM has already been evaluated on is remarkably challenging, across many variations of the experimental setup. The results in this chapter underscore how much we still have left to learn about procedural knowledge capabilities in LMs.

This work was originally presented in [Nadkarni et al. \(2024\)](#).

3.1 Introduction

Despite much-discussed advances in the capabilities of language model (LM)-based systems that follow instructions ([Mishra et al., 2022](#); [Sanh et al., 2022](#); [Wei et al., 2022](#); [Ouyang et al., 2022](#); [OpenAI, 2022, 2023](#)), the research community lacks an understanding of the limits of these capabilities. Ideally, purveyors of a technological product would clearly explain to users its limitations.¹ At present, the best a user can

¹While documentation such as that of [OpenAI \(2023\)](#) breaks down performance by factors such as task categories, languages, or benchmarks (each benchmark being a collection of many tasks), there is little transparency at any finer granularity (e.g., instruction-level) and no publicly-available tool or mechanism to give users a sense of performance for tasks that do not fit neatly into the documented categories.

do is explore: try out a prompt and see whether the LM can correctly complete the task. We find this state of affairs concerning, because the cost of such tests will fall on users. Without coordination and information-sharing, different users will make the same explorations and incur unnecessary costs while potentially relying on systems for tasks which they are incapable of performing adequately.²

In this chapter, we take a step toward empowering users of LM-based systems by proposing a **third party** approach to predicting model performance at the task level. Consider a user with a particular task in mind. Our proposed task performance predictor takes as input the same prompt the user intends for the LM, and *without querying the LM itself*, offers an estimate of the model’s performance on the task. We instantiate such models by regressing quantitative model performance metrics on natural language task instructions. If successful, such predictors could help users decide among commercial systems, or even opt out of delegating a task to an LM at all.

Our experiments examine how well existing instruction-tuned LMs’ performance can be predicted as a function of model size, choice of evaluation metric, amount of training data, and other factors. We find overall that the task is challenging, with the various factors we explore providing little improvement to predictability. Our results underscore how much progress still needs to be made in designing instruction-following natural language systems whose performance can be accurately predicted and made known for the sake of transparency and user safety.

3.2 Related Work

3.2.1 Instruction Tuning

Our work focuses on analyzing the behavior of models that have been trained to follow task instructions. This includes models trained on human-generated instructions and instances (Mishra et al., 2022; Sanh et al., 2022; Wei et al., 2022; Ouyang et al., 2022; Wang et al., 2022) as well as model-generated data (Wang et al., 2023b; Honovich et al., 2023; Taori et al., 2023; Chiang et al., 2023). We primarily use models trained by Wang et al. (2023a) on a variety of instruction-following datasets and initialized with the publicly available LLaMA family of LMs (Touvron et al., 2023). We also explore using the closed models GPT-3.5 (OpenAI, 2022) and GPT-4 (OpenAI, 2023). For more details on our choice of instruction-tuned models, please refer to §3.3.1.

²A second issue, not addressed here, is that users may not realize that they need to check system output for correctness, and may simply assume that any confident answer from a model can be trusted. We suspect that this problem will worsen as users explore more use cases not anticipated by the builders of such systems and therefore unaddressed by so-called refusal training.

3.2.2 Predicting Model Behavior

Initial work on performance prediction involved training simpler models to predict the performance of larger models as a function of various features, such as model family, model size, task, language, and training procedure (Xia et al., 2020; Ye et al., 2021, 2023). The primary motivation of that work was to address computational and data constraints – training separate performance prediction models could alleviate issues where the computational cost of finetuning on all datasets was prohibitive or where scarce data in a particular language or domain prevented finetuning altogether. While much of that work predicted performance at the task level as in our study, its motivations were different and its methods were often implemented by predicting from hand-crafted features related to properties of each dataset (e.g., model parameter count or language features) rather than the text prompt input to the model itself.

More recent efforts to better understand model behavior have analyzed the ability to predict whether or not a model will perform well on a given input, either as determined by the LM itself or a separately-trained model. Kadavath et al. (2022) analyze models to determine whether they can identify examples for which they can generate the correct response, either by prompting for a “True/False” label with an instance and one or more generations from the model itself or by finetuning the model as a binary classifier using a dataset of inputs and correctness labels from previous model outputs. Yao and Koller (2024) similarly finetune models as binary classifiers to predict whether another model’s generated response is correct for tagging, parsing, and semantic parsing tasks. Other work explores models’ ability to generate calibrated uncertainty about their responses either through logits, multiple generations, or verbalized expressions of uncertainty, applied to domains such as solving arithmetic problems (Lin et al., 2022a) or question answering (QA; Si et al., 2023; Zhou et al., 2023c; Cheng et al., 2023). Models’ ability to verbalize their uncertainty has also been explored for models trained with reinforcement learning with human feedback (RLHF), again using QA datasets (Tian et al., 2023). Notably, all of these efforts examine being able to predict model performance at the *instance level*, whereas we are primarily interested in predicting performance at the *task level* given a task instruction.

The most similar work to ours in spirit is that of Fu et al. (2023), which also attempts to predict how well a model will perform on a given dataset by training a separate predictor. There are a number of key differences from our efforts: their work focuses on the in-context learning setting using purely pretrained models rather than models specifically trained to follow instructions; they assume access to a set of unlabeled examples for each dataset, rather than just an instruction; their analysis requires access to model internals by using model output logits to construct a “confidence profile” feature vector for each dataset; and they restrict their analysis to a small set of question answering datasets. A modern trend has seen public releases of models whose internal states are inaccessible, but that can be prompted with a wide variety of user-defined instructions. This situation challenges many of the assumptions made by Fu

et al. (2023) and motivates our own work. However, their explorations inspire our efforts, and we attempt to extend their analysis while focusing on task instructions and models trained to follow them.

3.3 Methods

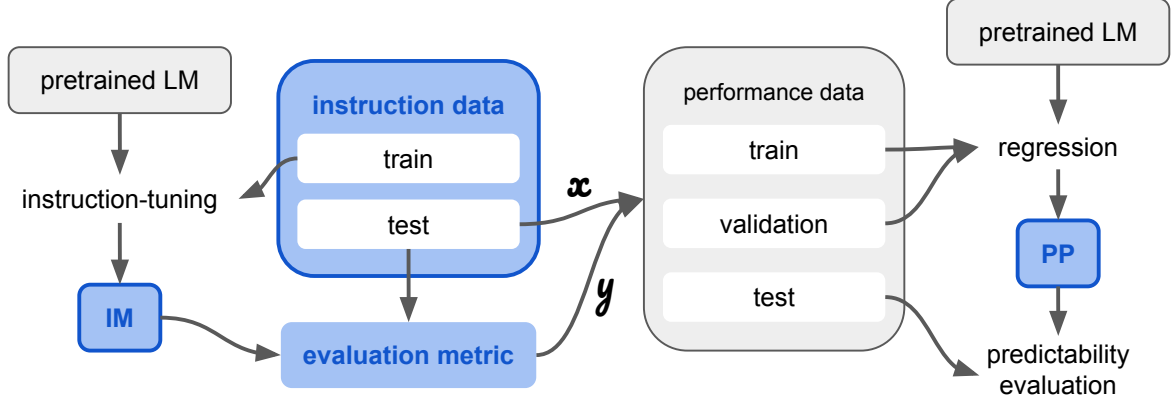


Figure 3.1: A diagram illustrating our complete analysis pipeline. We begin with a pretrained LM that is instruction-tuned using training tasks from chosen **instruction data**, resulting in an **instruction-tuned model (IM)**. The **IM** is evaluated using the test tasks of the **instruction data** (not necessarily from the same dataset as the training tasks) and a choice of **evaluation metric**. Each pair of test task instruction (x) and evaluation performance metric value (y) is used to construct the performance data, which itself is split into train, validation, and test sets. The train and validation sets are used to train *another* (“third party”) pretrained LM to predict the performance of the **IM** as a regression model, resulting in the **performance predictor (PP)**. Finally, the **PP** is evaluated on the test set of the performance data to determine how well it can predict the performance of the **IM** on unseen tasks. The sections of the diagram highlighted in blue indicate the components of the pipeline that we vary to determine their effect on performance prediction: the size of the **IM**, the choice of **instruction data**, the choice of **evaluation metric**, and the size and type of **PP** model.

We begin by describing our complete analysis pipeline, also illustrated as a diagram in Figure 3.1. Each of our experiments involves two finetuned LMs: one that is trained to follow instructions to perform tasks, which we term the *instruction-tuned model* (IM), and another that is trained to map from an instruction to some measure of the IM’s performance on that task, which we call the *performance predictor* (PP). The IMs considered here are drawn from past work. To tune a PP, we evaluate a single IM on instructions it was not exposed to during its training (including instruction-tuning). The resulting pairs – each an instruction x paired with the model’s performance y – are divided into training, validation, and test sets for the PP. We train the PP to predict the IM’s performance y on instructions x unseen to the IM, and

evaluate those predictions on a different set of instructions unseen to either the IM or the PP. We explore a variety of choices for IMs, PPs, and evaluation metrics, which are detailed below.

3.3.1 Instruction-Tuned Models (IMs)

Our experiments use a range of pretrained LMs that have been finetuned to follow instructions. Part of our goal is to assess the effect of IM size (i.e., parameter count) and choice of instruction-tuning dataset on how well the model’s performance can be predicted. To this end, we primarily use LLaMA models (Touvron et al., 2023) of various sizes finetuned on a range of instruction-tuning datasets (Wang et al., 2023a). The LLaMA family of models is among the best performing open models, and they have already been used to build various instruction-tuned models (Taori et al., 2023; Chiang et al., 2023; Wang et al., 2023a). All models were trained by maximizing the per-token likelihood of the gold output given an instruction and possibly an input, with no additional training procedures such as RLHF (Christiano et al., 2017). While adapting our analysis to models trained using RLHF would be an interesting direction of future work, at the time of this work such models and the data to train them were limited relative to models trained with supervised finetuning alone, and we aim to perform an initial analysis on the least-complicated systems that are still able to follow instructions. For the sake of completeness, we additionally include the closed, API-based models GPT-3.5 (gpt-3.5-turbo) and GPT-4, as our goal is to implement a pipeline that is applicable even when access is restricted to model inputs and outputs.

3.3.2 Evaluation Metrics

For each instruction-tuned model, we perform inference on a separate evaluation set of instructions in order to generate a dataset of the model’s behavior (y) on unseen instructions (x) for training performance prediction models. For each instruction and output pair, we calculate a quantitative evaluation metric that compares the model-generated output to the gold output, in most cases averaging this metric across multiple instances for each task. We treat this final metric as the instruction-tuned model’s performance for that instruction. Following Wang et al. (2022), we explore two quantitative metrics used when evaluating instruction-tuned models on a broad range of tasks: ROUGE-L (Lin, 2004) and Exact Match score. Again, since these metrics can be computed by comparing a model-generated output to a gold output, we can apply them when we only have access to model generations. Additionally, we briefly compare to using model loss as the metric to predict in §3.4.3, to explore what we can achieve with performance prediction when additional information (in this case, the LM’s output distribution) is available.

3.3.3 Performance Predictors (PP)

Once we’ve evaluated an IM on unseen instructions, we use the performance data ($\langle x, y \rangle$ pairs) to build models that predict the IM’s performance, and evaluate those predictions on the test subset of the IM performance data. We primarily finetune RoBERTa (both base and large sizes; [Liu et al., 2019](#)) as the PP, motivated by the goal of having a lightweight separate model that can predict performance without incurring the inference costs of a much larger model. RoBERTa models are trained as regression models by adding a linear layer to the [CLS] token at the output layer and training to minimize mean-squared error between the predicted and true evaluation metric for each instruction. In some cases, we also train the base LLaMA model used to build the IM by similarly adding a linear layer to the EOS token at the output layer and adding and updating LoRA adapters ([Hu et al., 2022](#)) rather than updating all model weights. This is done in an effort to establish an “upper bound” of performance prediction that can be achieved while incurring the prohibitive computational cost of using a much larger model. We additionally include a simple baseline of predicting the mean metric value across all training instructions, as a “lower bound” to establish whether training a separate predictor offers any benefit at all.

Unless otherwise specified, PPs are trained and evaluated on the results of evaluating IMs on the Super-NaturalInstructions (SuperNI) test set tasks ([Wang et al., 2022](#)). For each experiment, we perform 10 random 80%/10%/10% train/validation/test splits of these tasks and report mean and standard deviation of the performance predictor’s root mean squared error (RMSE) in predicting the true evaluation metric from the task instruction. Validation data is used to tune hyperparameters for the PPs, namely batch size and learning rate; for full experiment details, please refer to [Appendix B.1](#).

3.4 Results

3.4.1 Performance Prediction is Challenging

[Table 3.1](#) shows our main results of predicting performance on SuperNI test set instructions, with mean and standard deviation (subscript) RMSE values across all train-test splits of the SuperNI test tasks. We explore using both Exact Match and ROUGE-L as the target metrics to be predicted for each task, with base and large RoBERTa models as the PP models as well as LLaMA-13B (upper bound) and the simple mean baseline (lower bound). For the IMs, we use the 13B-parameter versions of the Alpaca, Vicuna³, and Tulu models, as well as GPT-3.5 (gpt-3.5-turbo) and GPT-4.

Results demonstrate that performance prediction from task instruction is incredibly difficult, with RMSE values generally around 20 or higher (for metrics in the 0–100 range) across all experimental

³We use the 13B-parameter version of LLaMA finetuned on the ShareGPT dataset by [Wang et al. \(2023a\)](#) as an attempt to replicate the training of Vicuna.

IM:		Alpaca-13B	Vicuna-13B	Tulu-13B	GPT-3.5	GPT-4
avg. performance (ROUGE-L):		45.9	44.6	61.7	53.6	63.5
Exact Match	mean	25.7 _{3.4}	26.4 _{3.2}	32.4 _{2.3}	30.8 _{3.9}	36.4 _{3.8}
	RoBERTa-base	26.2 _{4.3}	26.1 _{4.9}	33.6 _{3.3}	32.5 _{5.4}	38.2 _{4.1}
	RoBERTa-large	27.4 _{5.4}	26.9 _{5.4}	34.4 _{4.9}	33.0 _{5.7}	37.2 _{4.5}
	LLaMA-13B	19.1 _{5.3}	18.9 _{5.8}	21.4 _{3.8}	21.7 _{5.6}	22.4 _{8.7}
ROUGE-L	mean	21.3 _{3.5}	20.4 _{3.3}	21.4 _{1.8}	22.4 _{2.9}	24.2 _{4.2}
	RoBERTa-base	22.5 _{4.1}	21.2 _{3.6}	22.1 _{3.8}	23.4 _{3.4}	25.6 _{4.0}
	RoBERTa-large	22.6 _{4.2}	21.2 _{3.6}	22.0 _{3.7}	23.5 _{3.4}	25.7 _{3.9}
	LLaMA-13B	22.2 _{4.7}	21.2 _{3.8}	21.5 _{3.4}	21.6 _{3.5}	22.1 _{5.1}

Table 3.1: Test set RMSE of mean baseline and various PP models finetuned to predict performance from task instruction for various IMs (columns). Subscript shows standard deviation across 10 splits of the SuperNI test tasks.

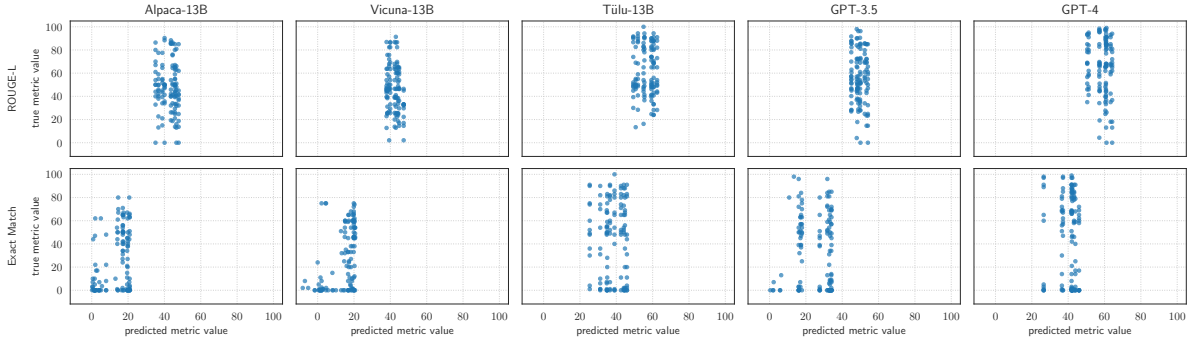


Figure 3.2: Predicted vs. true metric value when using RoBERTa-large to map from task instruction to performance – either ROUGE-L (top row) or Exact Match (bottom row) – for various instruction-following models (columns).

conditions. Both RoBERTa-base and RoBERTa-large perform comparably to the simple mean baseline, likely indicating that there is little learnable signal within the set of instruction-metric pairs. The lack of a meaningful difference in performance between RoBERTa-base and RoBERTa-large suggests that size of the PP model also makes little difference in performance prediction at the RoBERTa scale. In general, ROUGE-L is more predictable than Exact Match, with consistently lower RMSE values across choices of PPs and IMs. The values below the name of each model indicating performance in ROUGE-L on the SuperNI test tasks show that models which tend to perform better on SuperNI are *less* predictable, exhibiting the worst RMSE values in general.

The results for LLaMA-13B as PP are more promising, with RMSE values that outperform the RoBERTa models as well as the mean baseline when predicting the Exact Match score across all IMs. However, results are still poor, with RMSE values near 20 for both evaluation metrics and all IMs, so even a much larger PP with a more modern base model is unable to predict performance values that are near ground truth. It is additionally worth noting that finetuning LLaMA-13B is a much more prohibitive method of performance prediction, as it involves using a model with around $36\times$ the number of parameters of RoBERTa-large and incurring comparable inference costs to many of the IMs themselves. As such, we treat the LLaMA-13B performance prediction results as demonstration of a hopeful upper bound while simultaneously underscoring the large room for improvement in developing better-performing and lightweight PP models.

Figure 3.2 shows a more detailed view of a subset of the same results, with scatter plots of the predicted versus true metric value (ROUGE-L or Exact Match) when using RoBERTa-large as the PP for both metrics and all IMs. Results are shown for all train-test splits of the SuperNI test tasks with 12 held-out instructions per random split (10% of the SuperNI test tasks), resulting in a total of 120 predicted and true performance values for each combination of evaluation metric and IM. This qualitative view of the predictions highlights the fact that RoBERTa-large generally learns to predict roughly the same mean performance value across all tasks within each train-test split of the SuperNI test tasks. These results explain the similarity in performance to the simple mean baseline and demonstrate that the PP model does not learn a meaningful association between instruction and performance.

3.4.2 Effect of Various Factors on Predictability

We perform more detailed analyses by altering various factors that may affect the behavior of a PP when used to predict the performance of an IM. These factors include the size of the IM, the number of tasks used to train the PP, and the choice of prompt (instruction-only or instruction + 2 positive demonstrations).

Size of Instruction-tuned Model

We examine the effect of IM size by predicting the performance of various-sized LLaMA models instruction-tuned by Wang et al. (2023a) on the ShareGPT dataset, ranging in scale from 7B to 65B parameters. Results are shown in Table 3.2, focusing on the mean baseline and RoBERTa-large PP model as well as the Exact Match metric (results for RoBERTa-base and ROUGE-L are similar). The RMSE values indicate that performance prediction worsens for increasing model size, while model performance on the SuperNI test set improves (ROUGE-L values in parentheses next to each model name). However, the mean baseline also exhibits increasing RMSE values with model scale. This likely suggests that larger, better-performing models are less predictable not because they exhibit more dissimilar behavior on tasks

	PP: mean	RoBERTa-large
LLaMA-7B (35.8)	21.3 _{3.4}	20.7 _{4.6}
LLaMA-13B (44.6)	26.4 _{3.2}	27.0 _{5.4}
LLaMA-30B (44.2)	27.5 _{2.8}	27.9 _{5.3}
LLaMA-65B (48.9)	30.0 _{3.3}	31.7 _{5.3}

Table 3.2: Test set RMSE of predicting Exact Match given task instruction, for various sizes of LLaMA models instruction-tuned on ShareGPT. Model performance in ROUGE-L on the SuperNI test set is given in parentheses next to the name of each model.

with similar instructions, but because their performance metric values cover a broader range. In any case, we can conclude that scale of the IM alone does not improve performance prediction.

Number of Training Tasks

PP training:	SuperNI	+ BIG-bench
Alpaca-13B	27.4 _{5.4}	25.4 _{6.2}
Vicuna-13B	26.9 _{5.4}	26.1 _{5.5}
Tulu-13B	34.4 _{4.9}	34.6 _{3.9}
GPT-3.5	33.0 _{5.7}	32.3 _{6.7}
GPT-4	37.2 _{4.5}	37.7 _{4.5}

Table 3.3: Test set RMSE of using RoBERTa-large to predict Exact Match given task instruction, using either just SuperNI instructions or SuperNI and BIG-bench instructions for training the PP model.

We examine the effect of increasing the number of tasks / instructions used to train the PP model by including tasks from BIG-bench (BIG-bench authors, 2023). Since BIG-bench tasks do not come with instructions, we manually annotate tasks either by converting the task description (if provided) into a SuperNI-style declarative instruction or writing an instruction from scratch when necessary. We additionally filter out a number of tasks, leaving a total of 156 tasks to use as additional training tasks (a full list of included tasks can be found in Appendix B.2). We perform the same splits of the SuperNI test tasks as before, only adding the BIG-bench tasks to the training split in each case to increase the number of instructions used to train the PP model in an effort to improve its performance.

The results can be found in Table 3.3. While the inclusion of BIG-bench task instructions represents a nearly $2.5\times$ increase in the number of training instructions for the PP model, the RMSE values demonstrate

that the performance difference is negligible. There are a number of plausible explanations for why this is the case – in general, BIG-bench tasks are substantially different from SuperNI tasks, including those which deviate from standard language understanding such as decoding encrypted text, performing arithmetic operations, and generating chess moves. The domain shift between the two datasets may explain why the inclusion of the additional instructions did not provide enough meaningful signal for the PP models.

Prompt Format

format:	instruction	+ 2 demonstrations
Alpaca-13B	27.4 _{5.4}	27.3 _{3.7}
Vicuna-13B	26.9 _{5.4}	29.0 _{2.6}
Tulu-13B	34.4 _{4.9}	33.6 _{2.4}
GPT-3.5	33.0 _{5.7}	31.6 _{3.3}
GPT-4	37.2 _{4.5}	37.8 _{3.8}

Table 3.4: Test set RMSE of using RoBERTa-large to predict Exact Match given task instruction, using SuperNI tasks with an instruction-only prompt or the instruction with 2 positive demonstrations.

We primarily focus on instruction-tuned models evaluated in a zero-shot manner with only an instruction and instance input, as this has become standard practice. However, previous work exploring the effect of the prompt format has shown improvements in IM performance when including additional information, such as positive demonstrations of the task (Wang et al., 2022). We explore whether this improvement in performance also leads to an improvement in predictability by also evaluating all IMs with a prompt format that uses a task instruction with two positive demonstrations, using the positive examples provided with each SuperNI task. The RMSE values resulting from training RoBERTa-large to predict the Exact Match score in this setting, as well as the original instruction-only RMSE values, can be seen in Table 3.4. While the inclusion of demonstrations improves the performance of the IMs themselves by roughly 2–6 points in Exact Match score, this does not lead to models being more predictable as there are no meaningful differences in values between prompt formats for any model.

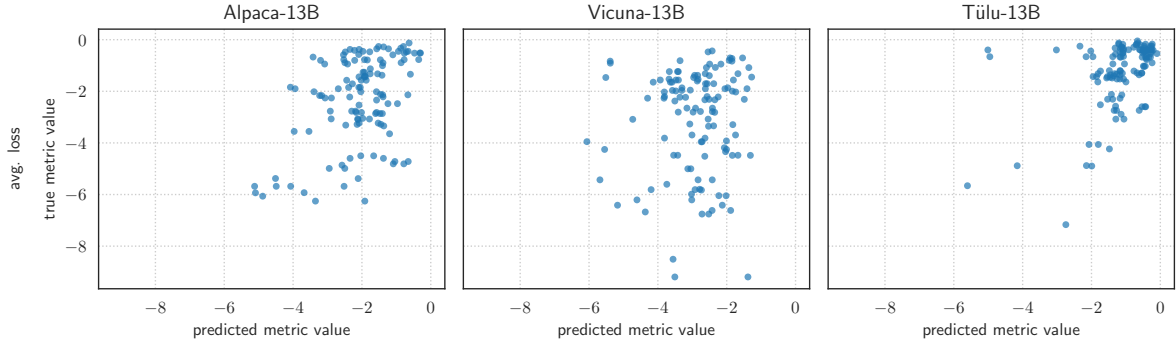


Figure 3.3: Predicted vs. true loss value when using RoBERTa-large to map from task instruction to loss for various instruction-following models.

3.4.3 Predicting Loss

	PP:	mean	RoBERTa-large
Alpaca-13B		1.70 _{0.27}	1.51 _{0.43}
Vicuna-13B		1.97 _{0.46}	2.10 _{0.53}
Tulu-13B		1.25 _{0.32}	1.10 _{0.36}

Table 3.5: Test set RMSE of predicting cross-entropy loss given task instruction, for various LLaMA-based instruction-following models.

Our core analysis above relied on the use of automated evaluation metrics that can be formulated as functions applied to a pair of generated and gold outputs. This assumes that we do not have access to model internals or outputs other than generated text, a practical assumption given the increasing use of modern systems with limited access such as through API requests. However, while such metrics may correlate with other variables such as accuracy on classification tasks (Wang et al., 2022), they come with their own limitations. For instance, for tasks that require more creative or free-form generations, it is possible for the model output and gold label to be semantically equivalent and equally valid responses to the instruction while differing in surface form such that metrics like Exact Match and ROUGE-L are inappropriate (Holtzman et al., 2021). To address this, we perform an additional experiment training PP models to predict the model’s *loss* (on the gold output) instead, avoiding token-based comparison between two pieces of text entirely. We do this by evaluating the average per-token cross-entropy loss of the gold label given a prompt (task instruction and input) for each instance in each dataset, averaging across instances to get a single average loss value per task / instruction. We then follow the same regression training procedure to build PP models that predict this value.

Scatter plots similar to Figure 3.2 can be seen in Figure 3.3, and the quantitative RMSE values corresponding to these results are in Table 3.5. As this analysis requires access to model output distributions to compute the loss, we are limited to the LLaMA-based open models. The plots show the predictions made by RoBERTa-large finetuned to predict average per-task loss, using the same 10 splits of the SuperNI test tasks as in previous results. Qualitatively, these results appear to show greater correlation between the true and predicted values as compared to the metrics considered earlier, indicating that there may be more of a learnable signal for predicting loss. However, the mean RMSE values show that the finetuned RoBERTa-large model still does not outperform the simple mean baseline on average, indicating that performance prediction remains challenging even with access to a quantitative metric not based on token-level comparison between generated and gold outputs. Taken together, our results motivate further work in identifying quantitative task-level metrics that can be accurately predicted, and perhaps in designing IMs themselves to be more predictable.

3.5 Discussion

Our experiments varied a number of factors that could impact the predictability of an instruction-tuned model’s performance on unseen instructions. We broadly summarize our findings below:

Performance prediction remains incredibly challenging regardless of setup. RMSE values remain at 20 or higher for metrics in the range of 0–100, indicating that PP models fail to predict values that are even somewhat close to true performance. This remains true across a variety of open and closed instruction-following models, for multiple automated evaluation metrics. There is also little performance difference between RoBERTa PP models of different scales, and none of them outperform a simple mean baseline. One optimistic result occurs when using LLaMA-13B as the PP model, but performance is still relatively poor and this comes at the cost of scaling up the PP model to the size of the IM.

Increasing instruction-tuned model scale, increasing number of training tasks, and adding demonstrations to prompts all fail to improve performance prediction. The behavior of larger models does not seem to be any more predictable, nor is the behavior of models with access to additional information in the prompt. The typical strategy of increasing the amount of training data available for the finetuned PP model to better learn a prediction signal is also insufficient.

Predicting cross-entropy loss does not improve performance. Despite avoiding the issues inherent in using metrics based on token-level comparison between generated and gold outputs, training PP models to predict loss still does not lead to better results than the simple mean baseline.

3.6 Recent Work

Subsequent studies have also tackled the performance prediction problem, with varying setups and degrees of success. [Sam et al. \(2025\)](#) predict performance from continuous vectors constructed by querying a model with multiple follow-up questions and collecting the answer probabilities (either directly via logits or estimated via sampling) as features, though their predictions are at the instance-level rather than task-level and they do not attempt to transfer predictors to unseen tasks. Very recently, [Park et al. \(2026\)](#) revisit the performance prediction problem using detailed task descriptions, configurations, and inference settings as inputs, mining task information from arXiv papers while taking great care to ensure that predictors are being assessed fairly based on their knowledge cutoff date. They find that only GPT-5 ([OpenAI, 2025](#)) meaningfully outperforms the trivial test set mean baseline, although still showing considerable prediction error and high variance in performance across tasks.

A related but distinct approach that has gained substantial traction is *routing* – training a classifier which, given some input text, selects one LM from a set of possible models that it predicts will perform the best for that input. Routers can be trained in various ways, such as using data and evaluation results from collections of benchmarks ([Shnitzer et al., 2024](#)), large-scale crowd-sourced human preference data ([Ong et al., 2025](#); [Chiang et al., 2024b](#)), or reward models trained from such preference datasets ([Feng et al., 2025](#)). Notably, these approaches are typically applied at the instance- rather than task-level, and do not allow for estimating the confidence of a single model as could be done with our performance prediction setup. Nevertheless, routing methods have become effective enough to be commercially viable ([Martineau, 2024](#); [Raman and Lopp, 2025](#); [Singh et al., 2025](#)), highlighting the real world utility of developing methods to understand how models might behave on particular inputs without directly incurring their inference costs.

3.7 Conclusion

Recent natural language processing systems seem to be able to perform arbitrary tasks given an instruction. Yet we are still not able to understand or explain to users the limitations of these systems such as by reliably predicting their success or failure on new, previously unseen instructions. We take a first step toward this goal by training a separate predictor model to map from a task instruction to the quantified performance of a given instruction-tuned model on that task. Our results show that performance prediction is challenging, with numerous factors like choice of evaluation metric, predictor model size, instruction-following model size, number of training tasks, and prompt format all showing negligible effect on the predictability of instruction-tuned model behavior. Much work remains to be done in designing systems whose limitations can be well-predicted and revealed transparently and freely to their users.

Chapter 4

Rewriting History: A Recipe for Interventional Analyses to Study Data Effects on Model Behavior

In previous chapters we focused on assessing and understanding LMs’ declarative and procedural knowledge capabilities, but without explicitly drawing a connection to the source of these capabilities: their training data. In this chapter, we present an approach for better understanding how LMs’ training data gives rise to knowledge capabilities by way of a structured recipe for performing interventional analyses on training. We demonstrate the utility of our recipe by applying it to study multiple ways of evaluating declarative knowledge as well as gender bias, and perform additional analysis on the data we intervene on to further understand its relevance to observed behavior. We believe clearly outlined research methodology such as the work discussed in this chapter is crucial to better characterize the relationship between data and knowledge in LMs.

This work was originally presented in [Nadkarni et al. \(2025\)](#).

4.1 Introduction

When training LMs, the choice of pretraining data is generally believed to have a profound effect on what the learned models can do and how well they can do it. Numerous prior works have curated large pretraining datasets and studied the effect of pretraining data composition on downstream tasks ([Li et al., 2024b](#); [Longpre et al., 2024](#)), with some explicitly predicting downstream task performance ([Grattafiori et al., 2024](#); [Bhagia et al., 2025](#); [Magnusson et al., 2025](#)). The primary motivation here has generally been

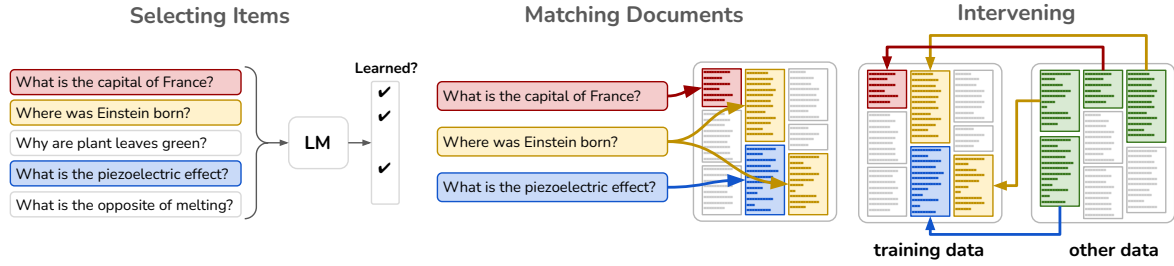


Figure 4.1: Diagram illustrating our proposed interventional analysis recipe, with stages including selecting items, matching documents to items, and intervening on matched documents.

performance improvement, not necessarily understanding model behavior, while mainly operating on the scale of entire pretraining datasets.

An equally important body of work relates pretraining data to model behavior to understand *how* data affects performance without necessarily optimizing for it, retraining over modified data to do so. This includes various data-behavior relationships, such as how relative frequencies of verb tenses change subject-verb agreement performance (Wei et al., 2021) or how pronouns affect gender bias (Biderman et al., 2023). Some recent work focuses on the specific behavior of factual knowledge acquisition, retraining models to understand how data gives rise to knowledge learning. This work operates across a range of scales, from pretraining from scratch on fictional biographies (Allen-Zhu and Li, 2024, 2025; Zucchet et al., 2025), to continued pretraining of a base model on carefully-constructed documents expressing factual relations (Zhang et al., 2024), and even intervening on intermediate stages of pretraining by injecting synthetic data to study the effect of repetition and paraphrasing (Chang et al., 2024) or entropy across activations (Kim et al., 2025) on fact learning and forgetting.

While the aforementioned works all modify pretraining data to study model behavior, each takes a specific approach that does not easily transfer to other hypotheses. Seeking to draw connections across these works and formalize their methodologies, in this chapter we propose a general recipe for *interventional analyses* to study the relationship between pretraining data and model behavior at a fine-grained level through *retraining* on modified data. Our proposed recipe (illustrated in Fig. 4.1) consists of three stages: (1) selecting which items in a benchmark to intervene on; (2) matching pretraining documents to those items; and (3) modifying those documents before repeating that portion of pretraining and measuring the effects. Our recipe organizes and comprehensively articulates the different considerations one must take into account when performing such an intervention, contextualized within past work. Our goal is to provide a recipe for practitioners to use when testing hypotheses about how specific properties of pretraining documents give rise to model behaviors, allowing for flexibility in the scale of the intervention (how much data is changed) and what portion of the training process is studied.

We demonstrate the utility of our recipe through case studies on *factual knowledge acquisition* and *gender bias*. LMs have been established as sources of factual information that can be queried with natural language (Petroni et al., 2019; Roberts et al., 2020), but how pretraining data yields knowledge capabilities is still under study. Similarly, LMs are known to exhibit various forms of bias (Navigli et al., 2023) including gender bias (Bordia and Bowman, 2019; Nadeem et al., 2021; Van Der Wal et al., 2022) that can be linked to their training data, yet direct interventions that tie bias to data are limited (Biderman et al., 2023). We perform three case studies to explore these phenomena: the first relates term cooccurrence to fact triple completion (§4.4), the second links pronoun-profession frequency and gender bias in coreference resolution (§4.5), and the third uses information retrieval methods to study more complex knowledge queries (§4.6). These studies aim to improve our understanding of phenomena like knowledge acquisition and gender bias through intervening on and retraining over pretraining data, complementing previous observational studies while also showcasing our recipe’s utility for different data sources and hypotheses.

For our first analysis, we operationalize knowledge acquisition through ⟨subject, relation, object⟩ fact triplets such as ⟨“France”, “capital city”, “Paris”⟩, considering a fact as having been learned when a model can identify the object given the subject and relation, i.e., correctly select “Paris” from a list of candidates when prompted with “The capital city of France is”. We use our recipe to study the hypothesis that fact learning is related to subject-object *cooccurrence* in pretraining documents. Cooccurrence has long been used as a source of distant supervision for aligning facts to text (Mintz et al., 2009; Augenstein et al., 2016; Elsayhar et al., 2018) and has a well-established relationship to LM behavior through *observational* analyses on the scale of entire pretraining datasets and fully-pretrained models (Razeghi et al., 2022; Kandpal et al., 2023; Elazar et al., 2022; Kang and Choi, 2023; Zhang et al., 2024; Mallen et al., 2023; Merullo et al., 2025; Wang et al., 2025a; Verma et al., 2025; Seshadri et al., 2024). The second study applies the recipe to better understand how pretraining data leads to gender bias in a coreference resolution task, where models exhibit better performance when certain gendered pronouns are paired with their stereotyped professions (Zhao et al., 2018; Rudinger et al., 2018). We again explore the effect of term cooccurrence, by swapping the gender of pronouns in documents that mention target professions to mitigate bias (Biderman et al., 2023). By analyzing another kind of model behavior, this study illustrates the generality of our recipe. We then apply our recipe to multiple-choice question answering (MCQA) benchmarks used to evaluate LM factual knowledge (MMLU, Hendrycks et al., 2021a; OpenBookQA, Mihaylov et al., 2018; SciQ, Welbl et al., 2017), exploring another way of expressing and querying for factual knowledge that requires moving beyond cooccurrence and further serves to demonstrate the flexibility of our recipe. Finally, we perform an analysis that annotates the training documents we intervene on with their level of relevance to the knowledge that we attempt to alter in the LM, discovering that our methods of matching documents vary in their effectiveness at identifying truly relevant training data. Overall, our results advance the study

of how pretraining documents give rise to model behavior through direct intervention, complementing post-hoc observational analysis.

4.2 Overview of Recipe

4.2.1 Components

We begin with the components of our interventional analysis recipe for studying the relationship between pretraining¹ data and model behavior:

- A pretrained LM $\mathcal{M}$. We assume access to a set of *checkpoints* $\mathcal{M} = \{\mathcal{M}_1, \dots, \mathcal{M}_T\}$, where each checkpoint $\mathcal{M}_s$ consists of model weights at intermediate step s during pretraining and $\mathcal{M}_T = \mathcal{M}$ is the final pretrained model. Since we wish to retrain from intermediate checkpoints, we also assume access to the optimizer state at each checkpoint.
- The pretraining dataset $\mathcal{D} = \{d_1, \dots, d_N\}$ used to train $\mathcal{M}$, where each d_i is a sequence of tokens that form a document.² We assume the dataset can be partitioned into T *data batches* $\mathcal{D}_1, \dots, \mathcal{D}_T$, such that $\mathcal{D}_s$ is the ordered sequence of tokens seen between checkpoints $\mathcal{M}_{s-1}$ and $\mathcal{M}_s$ during pretraining.
- A task or skill of interest represented by an evaluation dataset $\mathcal{E} = \{(x_1, y_1), \dots, (x_n, y_n)\}$ of *test items*, where each item (x_i, y_i) is an input and its corresponding correct output.
- An *evaluation metric* $\mathcal{F}(\mathcal{M}_s, (x_i, y_i))$ that applies a checkpoint $\mathcal{M}_s$ to an item (x_i, y_i) to quantify model behavior on that item (e.g., correctness at predicting y_i given x_i , log-probability of y_i given x_i , etc.).

Given $\mathcal{D}$ and a full specification of the LM training procedure, we can replicate pretraining from a model checkpoint $\mathcal{M}_s$ to $\mathcal{M}_{s+k}$ for any two checkpoints indexed s and $s+k$, by retraining over the ordered data batches $\mathcal{D}_{s+1}, \dots, \mathcal{D}_{s+k}$. We can also make targeted interventions on those data batches and re-run pretraining to generate a counterfactual model checkpoint $\mathcal{M}_{s+k}^{\text{int}}$, which we propose doing in our interventional analysis recipe.

4.2.2 Designing an Experiment

Suppose we are interested in how pretraining data affects the behavior of our model $\mathcal{M}$ at some timestep t , as measured by applying our metric $\mathcal{F}$ to $\mathcal{M}_t$ on some target subset of test items $\mathcal{E}^{\text{target}} \subseteq \mathcal{E}$. We have

¹We focus only on pretraining in this work, but a similar recipe could be applied to other stages of training as well.

²Note that each d_i does not have to be a complete document from the original text data, as documents can be split and concatenated together to fit into the maximum sequence length window when training the LM.

some hypothesis about how the relationship between documents in $\mathcal{D}_t$ ³ and items in $\mathcal{E}^{\text{target}}$ cause this behavior, and we wish to intervene on $\mathcal{D}_t$ to test our hypothesis. Our interventional analysis recipe can be broken down into three stages:

- Selecting a subset $\mathcal{E}^{\text{target}} \subseteq \mathcal{E}$ of target evaluation items on which our LM exhibits the behavior at timestep t , and corresponding data batch $\mathcal{D}_t$ to intervene on (§4.2.3);
- Identifying a set of relevant documents in $\mathcal{D}_t$ by matching to target items from $\mathcal{E}^{\text{target}}$ (§4.2.4);
- Intervening by modifying relevant documents in $\mathcal{D}_t$ to construct intervention data batch $\mathcal{D}_t^{\text{int}}$ and retraining over it starting from $\mathcal{M}_{t-1}$ to yield intervention checkpoint $\mathcal{M}_t^{\text{int}}$ (§4.2.5).

Finally, we compare $\mathcal{M}_t$ and $\mathcal{M}_t^{\text{int}}$ by applying $\mathcal{F}$ to measure their behavior on $\mathcal{E}^{\text{target}}$, determining if the intervention confirms our hypothesis.

Since we are performing an interventional analysis, our recipe instantiates *causal inference*, relating properties of pretraining data to model behavior. Foundational texts on causality (Pearl, 2009) and its application in NLP (Feder et al., 2022) define a causal relationship between some property of the data (e.g., term cooccurrence) and some model behavior (e.g., task accuracy) through the *causal effect* of that property, i.e., a measurable change in behavior after modifying the data through a controlled intervention, where the magnitude of the change reflects the strength of the causal relationship. Note that this definition (1) does not require that the modified property of the data is the *sole* cause of the behavior, and (2) does not depend on the behavior being altered in some *complete* way (e.g., accuracy dropping to zero). We do not expect or claim that our interventions target the sole causal factor in each case study, and in fact our results show that they do not. We present our recipe as a methodological tool, enabling future work that explores other potential causes of the kinds of model behavior that we study.

There are technical decisions to be made at each stage, and these will depend on the specific model behaviors one wishes to study, the hypotheses, and the computational budget. We go into detail for each of these stages below, with examples of how they can be implemented. Note that we do not provide an exhaustive overview of how to implement each stage of the analysis or anticipate all possible interventions. We believe creative choices will be essential in future studies, and offer a starting point of various options and their tradeoffs. Exploration of other ways of implementing each stage will lead to new instantiations of this recipe that study a wider range of questions.

³Here we consider only data seen since the previous model checkpoint, but in practice the partitioning of $\mathcal{D}$ into data batches can be defined in such a way that $\mathcal{D}_t$ spans multiple checkpoints.

4.2.3 Stage 1: Selecting Items and a Data Batch

Given an hypothesis about how data influences some model behavior of interest, we need to first select items from our evaluation dataset $\mathcal{E}$ on which our model $\mathcal{M}$ exhibits that behavior. For example, if we are interested in factual knowledge acquisition, we might select questions that our model answers correctly, whereas if we are interested in forgetting behavior we might select items that the model answers incorrectly at a given checkpoint after being correct for some number of previous checkpoints.

Formally, we wish to select a *target subset* $\mathcal{E}^{\text{target}}$ of items from our evaluation dataset $\mathcal{E}$ (i.e., $\mathcal{E}^{\text{target}} \subseteq \mathcal{E}$). Consider a function of each item, all model checkpoints, and a particular timestep of the form $f_{\text{sel}}((x_i, y_i), \mathcal{M}, s) \in \{0, 1\}$, which is 1 when our model exhibits the behavior of interest on item (x_i, y_i) at step s , and 0 otherwise (note that this is a function of all model checkpoints $\mathcal{M}$ since we may be interested in behavior that involves multiple checkpoints, such as the forgetting example above). Then we define the target subset $\mathcal{E}_s^{\text{target}} \subseteq \mathcal{E}$ of items at a given step s as

$$\mathcal{E}_s^{\text{target}} = \{(x_i, y_i) \in \mathcal{E} \mid f_{\text{sel}}((x_i, y_i), \mathcal{M}, s) = 1\}$$

Since we have different target subsets $\mathcal{E}_s^{\text{target}}$ for each step s , we need to choose a step to intervene on. This can be motivated by different considerations – the stage of pretraining one is interested in, properties of items in $\mathcal{E}_s^{\text{target}}$, etc. One goal may be for $\mathcal{E}^{\text{target}}$ to contain as many items as possible, as a larger sample size improves our ability to measure the true effect of our intervention (Card et al., 2020). This may motivate choosing a target subset and corresponding timestep t as the subset $\mathcal{E}_s^{\text{target}}$ of the largest size

$$t = \underset{s \in [1, \dots, T]}{\operatorname{argmax}} \quad |\mathcal{E}_s^{\text{target}}| \quad \mathcal{E}^{\text{target}} = \mathcal{E}_t^{\text{target}}$$

There are also many ways to design $f_{\text{sel}}(\cdot)$ to select items, and this choice should be guided by the behavior under study. For example, Chang et al. (2024) examine changes in log-probability of target spans in text probes to study factual knowledge acquisition, and Porada et al. (2022) do the same for commonsense knowledge triples. Selection could also be based on intermediate layer activations: Kim et al. (2025) track the entropy of activations in feedforward layers and relate it to fact learning and forgetting.

Existing work typically uses synthetic data or all items in a dataset rather than just ones on which the model exhibits a certain behavior, but we propose selecting a target subset since not all items in a benchmark may be relevant to the hypothesis (e.g., for fact learning, not all items may be learned at a given checkpoint). We believe properly selecting a target set of items is an important consideration when performing these kinds of analyses, and so we include it in our recipe for completeness.

4.2.4 Stage 2: Identifying Relevant Documents

Given a target subset of items $\mathcal{E}^{\text{target}}$ on which our model exhibits the behavior of interest and intervention timestep t , we next identify a subset of pretraining documents within the data batch $\mathcal{D}_t$ that we hypothesize

to affect the behavior on the target items. The matching process should also be motivated by the original hypothesis, i.e., what relationship between documents and items is believed to be driving model behavior.

In practice, it may be hard to identify documents that we believe contribute to model behavior on our target set, so we approximate with some heuristic for matching documents to items. For example, if our hypothesis is that pretraining documents which contain multiple terms that also appear in an item lead to the model making a correct prediction on that item, we should match the item to documents with some lexical overlap (e.g., term cooccurrence). Suppose we define a matching function $g_{\text{match}}(d, (x_i, y_i)) \in \{0, 1\}$, which has value 1 if document d matches item (x_i, y_i) , and 0 otherwise.⁴ We can then define a function that matches a document to our entire target subset of items $\mathcal{E}^{\text{target}}$ as

$$f_{\text{match}}(d, \mathcal{E}^{\text{target}}) = \bigvee_{(x_i, y_i) \in \mathcal{E}^{\text{target}}} g_{\text{match}}(d, (x_i, y_i)),$$

i.e., we consider a document d to be matching if it matches *any* item in our target group $\mathcal{E}^{\text{target}}$. We will use this function to identify documents that we wish to modify to perform our intervention.

There are several considerations when choosing a method of matching documents to items, including tradeoffs in how effective different methods are at identifying target documents. While term cooccurrence can be used as an effective proxy for expression of a relation between entities (Elsahar et al., 2018), spurious matches are possible (e.g., documents where “Paris” and “France” cooccur without expressing the relation “The capital city of France is Paris”). Similarly, Yauney et al. (2023) find that item-document similarity measured using n-gram distributions or pretrained model embeddings does not correlate with performance across a range of tasks, though without performing any interventions. Our recipe provides a way of testing the hypotheses that motivate using each of these methods to identify relevant documents.

Selecting a data batch to intervene on and even defining how much data constitutes a “data batch” in one’s own analysis also requires careful consideration. While frequent checkpoints can lead to data batches with several million documents and enable a wide range of possible matching methods, applying our recipe to large batches or entire pretraining datasets with billions of documents would render all but the simplest of methods (e.g., searching for terms) computationally intractable. Thus, potential computational constraints may determine the scale of the intervention.

4.2.5 Stage 3: Intervening

Our final stage performs the intervention by modifying the documents in $\mathcal{D}_t$ that match items in $\mathcal{E}^{\text{target}}$ and retraining over the modified $\mathcal{D}_t$ to observe the effects. Our intervention should also be motivated by the hypothesized relationship between documents in $\mathcal{D}_t$ and items in $\mathcal{E}^{\text{target}}$ such that it alters (e.g., removes)

⁴This formulation can be general – if we instead have a method that assigns a real-valued similarity score to each document-item pair quantifying their relatedness, we can threshold this score to produce an indicator function $g_{\text{match}}(\cdot)$ as described.

this relationship. As an example, if we believe that a model correctly answers “What is the capital of France?” with “Paris” because it was trained on documents where “Paris” and “France” cooccur, it is sufficient to intervene on documents in any way that removes this cooccurrence. This can be as small an intervention as editing a few tokens, or as large as replacing the entire document with one in which “Paris” and “France” do not cooccur.

We define an intervention function $g_{\text{int}}(\cdot)$ that takes in a matched document, the set of target items, and the remaining pretraining data and returns a modified document $d' = g_{\text{int}}(d, \mathcal{E}^{\text{target}}, \mathcal{D})$. We include $\mathcal{E}^{\text{target}}$ and $\mathcal{D}$ as arguments to the intervention function since we may want to make modifications that make use of properties of items (e.g., salient terms that occur in x_i or y_i) or replace d with other documents in $\mathcal{D}$. Since we only want to intervene on documents which match items in our target set $\mathcal{E}^{\text{target}}$, we define a function

$$f_{\text{int}}(d) = \begin{cases} g_{\text{int}}(d, \mathcal{E}^{\text{target}}, \mathcal{D}) & \text{if } f_{\text{match}}(d, \mathcal{E}^{\text{target}}) = 1 \\ d & \text{if } f_{\text{match}}(d, \mathcal{E}^{\text{target}}) = 0 \end{cases}$$

Then we construct our intervention data batch $\mathcal{D}_t^{\text{int}}$ from our original data batch $\mathcal{D}_t$ as $\mathcal{D}_t^{\text{int}} = \{f_{\text{int}}(d) \mid d \in \mathcal{D}_t\}$. Finally, we re-run pretraining over the intervention data. We initialize with model checkpoint $\mathcal{M}_{t-1}$ (and its optimizer state) and train over $\mathcal{D}_t^{\text{int}}$ instead of $\mathcal{D}_t$, resulting in an intervention checkpoint $\mathcal{M}_t^{\text{int}}$. We then apply our evaluation metric $\mathcal{F}$ to items in $\mathcal{E}^{\text{target}}$ using both $\mathcal{M}_t$ and $\mathcal{M}_t^{\text{int}}$ separately to compare their behavior and determine if the effect of our intervention is consistent with our hypothesis. The implementation depends on the evaluation metric. For instance, we can compare $\mathcal{F}(\mathcal{M}_t, (x_i, y_i))$ to $\mathcal{F}(\mathcal{M}_t^{\text{int}}, (x_i, y_i))$ for each $(x_i, y_i) \in \mathcal{E}^{\text{target}}$ (e.g., change in log-probability of y_i given x_i), or we can aggregate $\mathcal{F}(\mathcal{M}_t, (x_i, y_i))$ over all $(x_i, y_i) \in \mathcal{E}^{\text{target}}$ (and similarly for $\mathcal{M}_t^{\text{int}}$) and compare the aggregated values (e.g., accuracy on $\mathcal{E}^{\text{target}}$).

The intervention function $g_{\text{int}}(\cdot)$ can vary in the degree to which it alters the pretraining data, depending on the hypothesis being studied. [Biderman et al. \(2023\)](#) perform a token-level intervention that swaps masculine and feminine pronouns late in pretraining to observe the effects on gender bias, while [Wei et al. \(2021\)](#) add and remove small subsets of documents to see how verb frequencies affect subject-verb agreement performance. At the more extreme end, [Kandpal et al. \(2023\)](#) remove all documents containing cooccurrences of terms from a subset of factual questions, ablating 30% of the pretraining dataset in total before training a counterfactual model from scratch.

Different interventions come with their own pros and cons: token-level interventions enable studying more precise data-behavior relationships but may be too small to have an effect, while removing large sets of documents is more likely to alter behavior but introduces confounding factors like altered token frequencies, domain shift, etc. All of these considerations should be taken into account when designing one’s own document intervention procedure.

In summary, instantiating our recipe requires choosing an item selection function f_{sel} , a document matching function g_{match} , and an intervention function g_{int} .

4.3 Experimental Setup

We demonstrate our recipe through case studies where we intervene on pretraining documents to study their effect on knowledge acquisition. Our first study relates term cooccurrences to relational fact learning (§4.4). Our second study also considers term cooccurrences, but relates them to a different phenomenon by studying gender bias in a coreference resolution task (§4.5). In our final study, we experiment with multiple-choice question answering (MCQA) datasets MMLU, OpenBookQA, and SciQ to study knowledge acquisition while moving beyond the term cooccurrence setting (§4.6). We conclude with an analysis of how relevant documents actually are to the items that they are matched with, across different matching methods and datasets (§4.7). We use a similar method of selecting items and general intervention procedure across studies, with specific variations discussed below and in the respective sections for each study. We use the same models and evaluation setup for all experiments.

4.3.1 Model

We use the OLMo models (Groeneveld et al., 2024; Walsh et al., 2025) for our experiments, as they include all the necessary components for our recipe as listed in §4.2.1.⁵ Specifically, we use OLMo 2 1B and 7B with associated pretraining data, OLMo 2 Mix 1124 (Walsh et al., 2025). For the 7B model we use the second and third checkpoints (at 2,000 and 3,000 steps, respectively). We create additional checkpoints at steps 2,150 and 3,150 (retraining over the original ordered data starting from steps 2,000 and 3,000, respectively) to perform interventions over smaller data batches. For the 1B model, we retrain the model from scratch for 10,000 steps on the original ordered data batches, checkpointing every 500 steps to yield more granular checkpoints.

4.3.2 Evaluation

Key components of our analysis require evaluating checkpoints on the items in each dataset, and we use OLMES to perform these evaluations (Gu et al., 2025). The OLMES framework outlines a variety of choices for prompting the model and ranking answer choices to standardize evaluations for MCQA tasks, which fits all of our datasets.

Following OLMES notation, we use the cloze/completion task formulation (CF) with a QA prompt as input (e.g., “Question: The capital city of France is \nAnswer:”). We use this format for all datasets

⁵The availability of such resources is essential for performing the kinds of analyses we describe, underscoring the importance of fully open LMs for studying pretraining.

and experiments. We rank LM probabilities of candidate answer choices with different normalizations (detailed in each section) based on which gives the best performance. We label items as correct if the correct response is ranked highest and calculate accuracy as our evaluation metric $\mathcal{F}$. We use zero-shot prompting for all datasets.

4.3.3 Selecting Items

For our factual knowledge studies, we perform two kinds of interventions for a particular data batch: (1) replacing documents that match items learned at a particular checkpoint with unrelated documents in order to *suppress learning*; and (2) replacing unrelated documents with documents that match items learned at a later checkpoint to *promote learning* earlier. To do so, we identify the first checkpoint at which certain facts (represented by items in our evaluation dataset) are “definitively learned”. We define a checkpoint at which a fact is first “definitively learned” as the one where its item is first answered correctly (being incorrect at all previous checkpoints) and remains correct until a certain future checkpoint, specifically checkpoint 20 (10K steps) for OLMo 2 1B and checkpoint 30 (30K steps) for OLMo 2 7B in our experiments. Our definition enforces correctness for some future checkpoints to reduce the risk of targeting items labeled correctly by chance.

For our gender bias study, we perform a single type of intervention that aims to mitigate bias exhibited by the model, defined as a difference in task performance that favors one gender over another. In this setting, we simply select target items as those which exhibit bias at a particular checkpoint.

The early stages of pretraining are where the OLMo models exhibit the most improvement in performance for all datasets. We also find that the early checkpoints are where most items are learned (by our definition), with at least 80% of learned items being labeled correctly by step 4,000 for the 1B model and step 3,000 for the 7B model (see Appendix Fig. C.2). For items learned at a particular checkpoint, we identify the subset that have some number of matching documents in the data batch immediately preceding that checkpoint and use those as our target subset $\mathcal{E}^{\text{target}}$.

4.3.4 Intervening

For our factual knowledge experiments, we identify a pair of consecutive model checkpoints $\mathcal{M}_t$ and $\mathcal{M}_{t+1}$ where we either want to *promote learning* of items learned at $\mathcal{M}_{t+1}$ (but not yet learned at $\mathcal{M}_t$) or *suppress learning* of items learned at $\mathcal{M}_t$. In either setting, we intervene on the data batch $\mathcal{D}_t$ just prior to $\mathcal{M}_t$, with details on each intervention described separately for each experiment. In general, we attempt to *suppress learning* by replacing matching documents in $\mathcal{D}_t$ with unrelated ones from $\mathcal{D}_{t+1}$, and we *promote learning* by moving matching documents from $\mathcal{D}_{t+1}$ into $\mathcal{D}_t$ (replacing unrelated documents to do so). Specific details on this document-swapping operation can be found in Appendix §C.1.

For our gender bias experiments, we similarly identify a pair of consecutive model checkpoints $\mathcal{M}_t$ and $\mathcal{M}_{t+1}$ where we want intervene on the data batch between them to modify some behavior observed at checkpoint $\mathcal{M}_{t+1}$. Rather than swapping whole documents, our intervention swaps the gender of pronouns in documents to alter term frequencies in a way that mitigates the observed bias (more details can be found in §4.5).

After constructing our intervention data batch $\mathcal{D}_t^{\text{int}}$, we take $\mathcal{M}_{t-1}$ and continue training up to the next checkpoint using the same optimizer state, learning rate, and learning rate schedule as the original OLMo 2 pretraining run at that point. Other than the intervened documents, the model will see the same documents in the same order as the original model that was trained from $\mathcal{M}_{t-1}$ to $\mathcal{M}_t$, allowing us to construct an intervention checkpoint $\mathcal{M}_t^{\text{int}}$ that exhibits the effects of only our intervention on the original data batch $\mathcal{D}_t$.

Interestingly, we observe different results even with repeated runs of the exact same portion of pretraining (i.e., the same initial state and ordered data batches). We attribute this to irreducible noise arising from distributing training across GPUs. We repeat training runs over each intervention data batch as well as the original data batch to take this variation into account, reporting the mean and standard deviation across runs for each condition.

4.4 Impact of Term Cooccurrences on Relation Fact Learning

In the first study, we test the hypothesis that the presence of term cooccurrences – for example, the cooccurrence of “Paris” and “France” given the factual relation “Paris is the capital of France” – in the data batch immediately prior to a fact being learned affects whether it is learned at all. Existing work has studied the relationship between term cooccurrence and factual knowledge acquisition either through observational data (Razeghi et al., 2022; Kandpal et al., 2023; Elazar et al., 2022; Kang and Choi, 2023; Zhang et al., 2024) or by pretraining from scratch on synthetic data (Kassner et al., 2020) or modified pretraining datasets (Wei et al., 2021; Kandpal et al., 2023). In contrast to past work, which primarily analyzes the *final* pretrained model, we study the intermediate checkpoints where these facts are *first* learned, by making specific, targeted interventions on the data.

4.4.1 Experiment Setup

Dataset We use ParaRel⁶ as our evaluation dataset $\mathcal{E}$ (Elazar et al., 2021), which consists of WikiData facts formatted as a relation between two entities – a subject and an object – allowing us to use subject-object cooccurrence as a heuristic to identify matching documents. Each relation is also associated with

⁶We use a version of ParaRel that was modified for autoregressive language modeling (CoAStAL NLP Group, 2024).

various templates that express it in natural language, and we use a single template per relation when querying our LM. We use the 12 relations where both models show better than chance performance at the intervention checkpoint, with full details on those relations in Appendix Table C.3. When evaluating, we use no length normalization when ranking candidate answer choices, after comparing to all other OLMES normalization strategies and finding that no normalization gave the best performance for this dataset.

Counting (co)occurrences We count occurrences of entities from ParaRel items mentioned in a batch of pretraining data using WIMBD⁷ (Elazar et al., 2024). This allows us to identify the documents entities occur in and their starting and ending character location within those documents. For items where the strings of the subject and object have partial overlap (e.g., “Microsoft Windows is a product of Microsoft”), we only consider cooccurrences where the character windows of the two terms do not overlap. We note that cooccurrence is a heuristic which does not guarantee that each matched document is truly relevant.

Intervening We attempt to suppress and promote learning of items originally learned at two different pretraining steps each for the 1B and 7B OLMo 2 models, retraining over steps 1.5–2K and 2.5–3K for the 1B model and steps 2–2.15K and 3–3.15K for the 7B model. In each case, we attempt to suppress learning of items originally learned at the end of each data batch, and promote learning of items learned at the end of the next batch. To suppress learning, we use both cooccurrence of subject and object entities as well as occurrence of either entity to identify related documents. The replacement of occurrences is a more aggressive intervention to determine a lower bound on performance change. This results in replacing 0.5–4.7% of documents for cooccurrences and 17.3–43.7% for occurrences, depending on the model size and data batch. For promoting learning, we only use cooccurrences to identify relevant documents to move into the previous batch, replacing 1.3–9.9% of documents for any given model size and pretraining step.

In all cases, we initialize from the OLMo 2 checkpoint at the initial step and continue pretraining for 500 (1B) or 150 (7B) steps on the modified data batch, then evaluate the counterfactual checkpoints on our target subset for which we are attempting to suppress or promote learning. We perform 5 repeated training runs for each result. Additional details on each intervention can be found in Appendix Table C.5.

4.4.2 Result: Removing Cooccurrences and Occurrences Suppresses Learning

Fig. 4.2 shows that performance for learned items is reduced when cooccurrences are removed, and reduced even further when removing occurrences. This trend holds for multiple checkpoints and across model scales. We also compare all results to a *majority baseline* that selects the most commonly-occurring object entity for each relation in the ParaRel training set.

⁷Specifically, we use the `wimbd search` command for fast exact string matching.

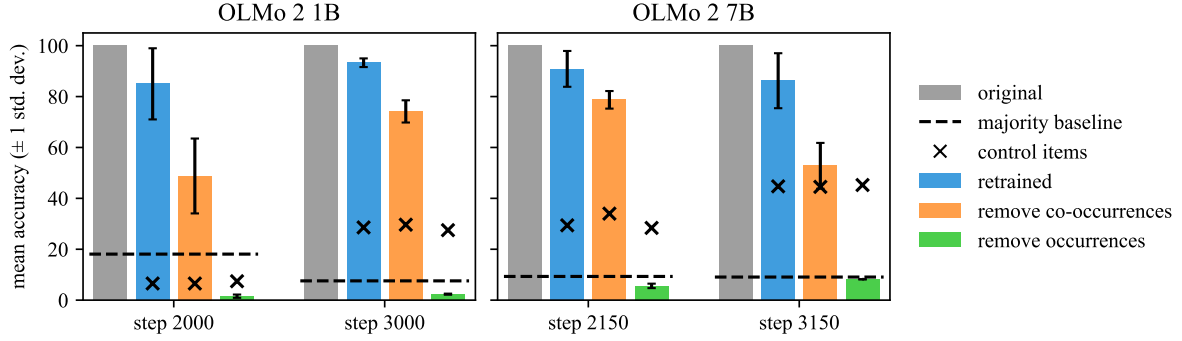


Figure 4.2: Results for replacing matching documents and re-training checkpoints to *suppress learning* of selected items from ParaRel. Colored bars show accuracy on the subset of target items for each intervention condition, including retraining on the original data batch (blue), training on a modified batch that replaces cooccurrences (orange), and training on a modified batch that (more aggressively) replaces occurrences (green). We also show results on control items (x; i.e., all items other than the target group). For all results, we perform 5 repeated training runs and show the mean and one standard deviation across runs.

Notably, removing all cooccurrences yields higher performance than the majority baseline, with a 30.7–66.5% absolute difference in accuracy depending on the model size and pretraining step. If an LM’s transition from answering an item incorrectly at one checkpoint to correctly at the next was entirely due to cooccurrences in the data seen between those two checkpoints, we would expect removing those cooccurrences to drop performance to the majority baseline. As this is not the case, we conclude that while cooccurrences in the most recent data batch encourage learning, they are not responsible for all of it. The remaining difference could be explained by a number of factors – cooccurrences from previous data batches, multi-hop cross-document chains of relations linking the two entities, cooccurrences of different terms with similar model representations, etc. – all of which would be interesting future interventions for further understanding this phenomenon.

In contrast, the more drastic intervention of removing occurrences is sufficient to completely suppress learning for all steps and model scales, although this modifies a much larger fraction of each data batch. This suggests that learning is in fact caused by the documents with entity mentions, although not just by documents that contain cooccurrences. Testing other hypotheses could determine whether the same reduction in performance could be obtained by a smaller intervention.

Finally, we also show the effect of our intervention on a *control group* of all items other than those we target for suppressing learning. Change in control group performance is minimal, indicating that our intervention successfully targets items for which we aim to alter performance. This strengthens the claim that the documents we intervene on have a direct relationship with the target evaluation items, as opposed to having a more general effect on task performance.

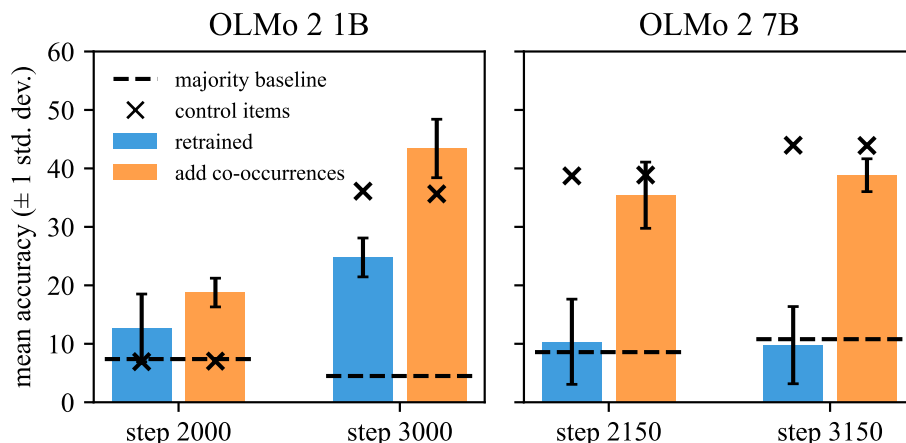


Figure 4.3: Results for replacing matching documents and retraining to *promote learning*. Replacing documents not containing a cooccurrence with documents that do improves performance in all cases.

4.4.3 Result: Introducing Cooccurrences Promotes Learning

Fig. 4.3 shows that introducing cooccurrences promotes learning, increasing performance on the target items above their original performance and well above the majority baseline. However, these items were selected because they were all labeled correctly (i.e., “learned”) at the very next checkpoint, and our intervention fails to attain perfect accuracy. As in the previous section, this finding suggests that documents beyond the ones that contain entity cooccurrences in this data batch (or possibly a prior one) also contribute to the model’s acquisition of factual knowledge for these items. As in the learning-suppression setting, our intervention has minimal impact on control group items.

We perform supplementary experiments varying our method of matching documents to items (Appendix Fig. C.3) and our method of intervening on matched documents (Appendix Fig. C.4). These experiments show a similar pattern to what is reported above.

4.5 Impact of Pronoun-Profession Cooccurrence on Gender Bias

Our recipe is intended to be general enough to analyze a broad range of phenomena. To test generality, we perform experiments altering cooccurrence of gendered pronouns and professions to affect gender bias behavior.

4.5.1 Experiment Setup

Dataset We use the WinoBias coreference resolution dataset (Zhao et al., 2018), which contains 783 sentence pairs, where each sentence mentions two professions and a gendered pronoun (e.g., “*The developer argued with the designer because she did not like the design.*”) and sentences in each pair

differ only in the pronoun’s gender. For evaluation, we prompt the model with a sentence and a question asking which profession the pronoun refers to, and select the answer by ranking each profession by its log-probability. We find that all OLMES length normalization strategies perform comparably, so we use no length normalization.

Quantifying bias We measure *gender bias* as the performance discrepancy when using a male versus a female pronoun with a profession. Specifically, we compute the accuracy difference between “pro”-stereotype items (e.g., a majority-male profession with a male pronoun; majority determined using U.S. Bureau of Labor Statistics data) and “anti”-stereotype items.

Counting cooccurrences Similar to ParaRel, we use regular expressions to match mentions of professions and pronouns. We do not enforce that pronouns and professions refer to the same entity, merely that they occur in the same document.

Selecting items Selected items are sentence pairs where the model is correct for the “pro”-stereotype profession-pronoun sentence and is incorrect for the “anti”-stereotype one, showing bias in favor of the “pro”-stereotype gender-profession pair. We perform separate experiments for target items where the “pro”-stereotype is male (i.e., majority-male professions) and those where it is female, as we observe bias favoring each gender depending on the profession.

Intervening We intervene on documents that contain a profession for which the model shows bias by changing all “pro”-stereotype pronouns to “anti”-stereotype ones. This gives two separate interventions for different professions, swapping all pronouns to either female or male. To preserve grammaticality, we resolve the ambiguous case of possessive pronouns (e.g., “his book” → “her book” vs. “the book was his” → “the book was hers”) by using the spaCy dependency parser (Honnibal et al., 2020) to identify the dependency label of each pronoun. We intervene between steps 5.5–6K for OLMo 2 1B and steps 2–2.15K for OLMo 2 7B, and perform 3 repeated runs. Additional details on each intervention can be found in Appendix Table C.6.

4.5.2 Result: Swapping Gendered Pronouns Mitigates Gender Bias

Results of our interventions are shown in Fig. 4.4. Swapping the gender of pronouns in documents mentioning stereotyped professions mitigates gender bias across both model sizes and both profession subsets (male- and female-stereotyped), as reflected by a decrease in the accuracy difference between “pro”- and “anti”-stereotyped sentences. The 1B model exhibits greater overall bias and a more pronounced reduction under the intervention. These results corroborate Biderman et al. (2023), showing that this

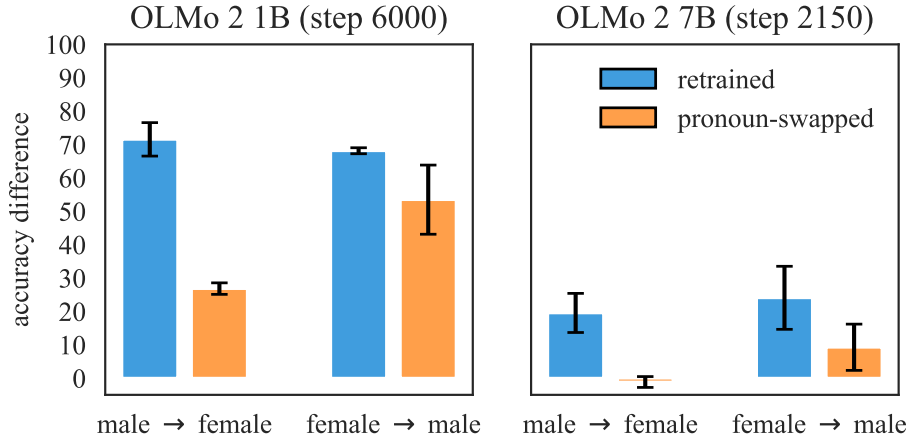


Figure 4.4: Results for swapping the gender of pronouns in documents that mention stereotyped professions to mitigate gender bias. Swapping pronouns is an effective intervention for mitigating bias across model sizes and stereotyped profession subsets (male and female).

intervention is effective for a newer model family and at intermediate stages of pretraining. They also complement our findings on term cooccurrence and knowledge acquisition, and demonstrate that our recipe applies to a broad range of data-behavior relationships.

4.6 Impact of Retrieved Documents on Complex Knowledge Queries

In §4.4, we studied the relationship between pretraining data and factual knowledge for facts that can be easily expressed as a relation between a pair of entities. While this format provides a useful way of studying knowledge acquisition via term cooccurrence, this is not always the way knowledge is expressed. Modern MCQA datasets test LM knowledge through complex questions paired with several possible answers, with questions and answers sometimes containing references to multiple entities or expressing abstract concepts involving no salient entities at all. Since term cooccurrence cannot be easily used to identify relevant documents for these items, we instead use methods from information retrieval (IR). As with our experiments on ParaRel, we believe that information in pretraining documents is contributing to knowledge learning for our MCQA datasets, and we hypothesize that IR methods are an effective way of identifying documents that contribute to knowledge acquisition.

4.6.1 Experiment Setup

Datasets We use common datasets for evaluating or comparing models (Groeneveld et al., 2024; Walsh et al., 2025; Gu et al., 2025): MMLU (Hendrycks et al., 2021a), OpenBookQA (Mihaylov et al., 2018), and SciQ (Welbl et al., 2017). OpenBookQA and SciQ consist of exam-style science questions, while

MMLU contains questions from 57 subjects. For all datasets, each item includes a question with 4 multiple-choice answer options which we rank according to the cloze formulation. We follow OLMES when performing length normalization for these datasets, using PMI normalization for OpenBookQA and character normalization for MMLU and SciQ (full details in Appendix Table C.4).

Retrieval-based matching Since we often cannot easily identify terms from items in these datasets to search for in pretraining documents, we instead use IR methods to match items to documents. Specifically, we use the keyword-based method BM25 (Roberts et al., 2020) and the embedding-based method Dense Passage Retriever (DPR; Karpukhin et al., 2020). BM25 is a popular method that scores a document’s relevance to a search query based on relative word frequencies,⁸ while DPR uses trained models to compute item and document embeddings and scores a document’s relevance to an item using embedding similarity.⁹ We include both the question and the correct answer in the search queries for BM25 and to construct query embeddings using DPR.

Intervening For our MCQA datasets, we also conduct interventions to both *suppress learning* and *promote learning*. When using BM25 and DPR to identify matching documents, we use the fact that each method assigns a real value similarity score to each document-item pair to construct our intervention. Specifically, we replace the top- k highest-scoring documents from $\mathcal{D}_t$ with the lowest-scoring documents from $\mathcal{D}_{t+1}$, while matching token counts. We match documents greedily, iterating over relevant documents in order of greatest to least number of matching items and finding a document to replace each one. In practice, this results in a lower average relevance score across replaced documents.

We intervene at steps 2K–2.5K for OpenBookQA and SciQ and steps 1K–1.5K for MMLU at the 1B scale, and steps 2K–2.15K for all datasets at the 7B scale; these steps were chosen because they correspond to stages when many items were learned. In all cases, we replace $k = 1\text{K}$ documents per item,¹⁰ which results in replacing 9.3–11.1% of documents for MMLU, 3.5–6.6% of documents for OpenBookQA, and 10.5–21.3% of documents for SciQ, depending on the method and model scale. We perform 3 repeated training runs for each setting. Additional details on each intervention can be found in Appendix Table C.7.

4.6.2 Result: Removing Relevant Documents Suppresses Learning

Results for suppressing learning are in Table 4.1, with all interventions leading to a decrease in average accuracy. Using DPR scores seems to have less of an effect than using BM25 scores across datasets and model scales, even when matching the number of replaced documents per item. Similar to ParaRel, the

⁸We use the bm25s Python package.

⁹We use the multiset DPR encoders from Facebook (Wolf et al., 2020a).

¹⁰Exploratory experiments suggested that increasing to 2K documents per item changed the results very little.

Model	Intervention	MMLU		OpenBookQA		SciQ	
		acc.	Δ acc.	acc.	Δ acc.	acc.	Δ acc.
OLMo 2 1B	retrained	89.8 _{0.8}	–	88.7 _{2.0}	–	89.8 _{1.4}	–
	replace high-scoring BM25 docs	76.0 _{0.6}	–13.8	70.2 _{3.0}	–18.5	48.3 _{2.3}	–41.5
	replace high-scoring DPR docs	86.0 _{2.8}	–3.8	78.0 _{2.0}	–10.7	59.4 _{5.7}	–30.4
	majority baseline	26.9	–62.9	27.6	–61.1	25.0	–64.8
OLMo 2 7B	retrained	84.8 _{3.7}	–	78.0 _{4.9}	–	80.6 _{2.5}	–
	replace high-scoring BM25 docs	46.8 _{2.1}	–38.0	65.3 _{4.1}	–12.7	50.6 _{2.4}	–30.0
	replace high-scoring DPR docs	57.1 _{6.5}	–27.7	70.0 _{1.6}	–8.0	55.0 _{1.6}	–25.6
	majority baseline	26.9	–57.9	27.6	–50.4	25.0	–55.6

Table 4.1: Results on *suppressing learning* for MCQA datasets. Results show the mean and standard deviation (subscript) across 3 training runs. Δ acc. is the change in accuracy as compared to the mean accuracy of the retrained model.

Model	Intervention	MMLU		OpenBookQA		SciQ	
		acc.	Δ acc.	acc.	Δ acc.	acc.	Δ acc.
	retrained	21.3 _{0.0}	–	15.2 _{5.4}	–	20.4 _{3.3}	–
	replace high-scoring BM25 docs	29.1 _{4.8}	+7.8	22.7 _{0.0}	+7.5	47.1 _{0.7}	+26.7
	replace high-scoring DPR docs	22.0 _{1.3}	+0.7	13.6 _{2.3}	–1.6	34.3 _{7.2}	+13.9
	majority baseline	26.9	+5.6	27.6	+12.4	25.0	+4.6
	retrained	13.6 _{3.6}	–	14.9 _{3.0}	–	11.9 _{3.7}	–
	replace high-scoring BM25 docs	53.7 _{4.6}	+40.1	34.5 _{7.0}	+19.6	59.4 _{2.0}	+47.5
	replace high-scoring DPR docs	31.9 _{2.6}	+18.3	25.3 _{5.2}	+10.4	49.6 _{1.1}	+37.7
	majority baseline	26.9	+13.3	27.6	+12.7	25.0	+13.1

Table 4.2: Results on *promoting learning* for MCQA datasets. Results show the mean and standard deviation (subscript) across 3 training runs. Δ acc. is the change in accuracy as compared to the mean accuracy of the retrained model.

performance after our intervention is still well above the baseline for all datasets, suggesting that the documents identified by the IR methods are still not entirely responsible for knowledge learning.

4.6.3 Result: Adding Relevant Documents Promotes Learning

Table 4.2 shows results for *promoting learning*, where similarly to suppressing learning nearly all of our interventions cause a change in average accuracy. The change in performance is again larger when

using BM25 scores rather than DPR scores across all datasets and model scales. In this setting, there is a consistently larger absolute increase in accuracy at the 7B model scale for all datasets, a pattern which was not present when suppressing learning. Given existing results showing that performance improves more for larger models with methods like instruction fine-tuning (Chung et al., 2024) or parameter-efficient techniques like prompt-tuning (Lester et al., 2021), this effect may be the consequence of larger models being more amenable to performance improvement.

4.7 Determining Document Relevance

While cooccurrence, BM25, and DPR are all reasonable methods of matching documents to items, ultimately they are approximations of our primary objective: identifying documents that express the information of a particular item. We assess the relevance of the documents matched by each of these methods by using GPT-5 (OpenAI, 2025) as a judge to annotate a subset of documents, focusing on our factual knowledge experiments. While using GPT-5 to perform the actual document matching over an entire data batch containing millions of documents is not scalable, using GPT-5 to annotate a subset of documents allows us to analyze the performance of our other more efficient matching methods. We also perform manual annotation on a smaller subset to compare to the GPT-5 annotations.

Setup For each combination of dataset (ParaRel, MMLU, OpenBookQA, and SciQ) and matching method (cooccurrence (ParaRel only), BM25, and DPR) across our factual knowledge experiments, we select the top-20 highest-scoring documents for each intervention group item from our OLMo 2 1B experiments to form a subset for annotation.¹¹ For each item-document pair, we provide GPT-5¹² with the question, correct answer, and document along with one of three prompts to elicit “yes”/“no” annotation of varying levels of document relevance. We use “correct answer”, “educated guess”, and “related topic” prompts, which are designed to determine whether the document contains sufficient information to correctly answer the question, contains enough information to make an educated guess about the answer (while not containing the answer), or is topically related to the question despite not containing enough information to answer it, respectively. The specific format of each prompt can be found in Appendix Table C.1. We set the verbosity to “low” and the reasoning level to “minimal” when querying the API.

Results A summary of the GPT-5 annotation results can be found in Fig. 4.5, where we plot the percentage of item-document pairs for which GPT-5 responded “yes” when asked to judge document relevance given each prompt. Overall, 9.8–61.3% of documents are determined to be relevant according

¹¹Since cooccurrence does not provide a score for each document, we instead use a random sample of 20 matching documents.

¹²gpt-5 in the OpenAI API.

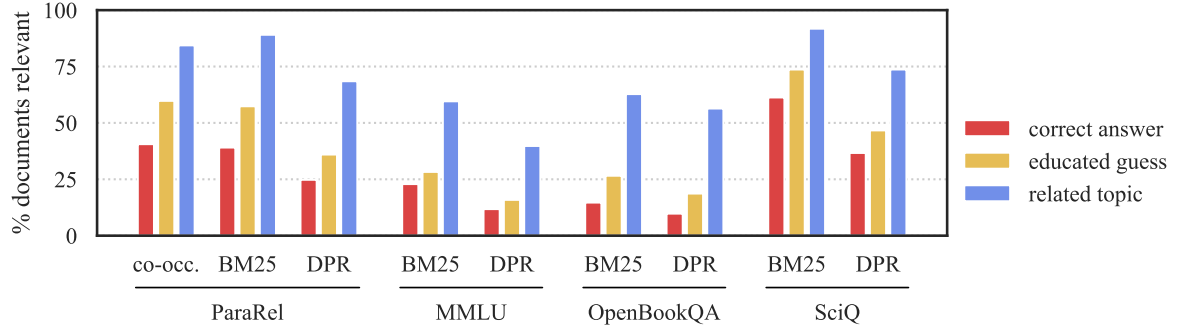


Figure 4.5: Percentage of documents labeled relevant to their matching item by **GPT-5**, for every dataset, prompt, and matching method.

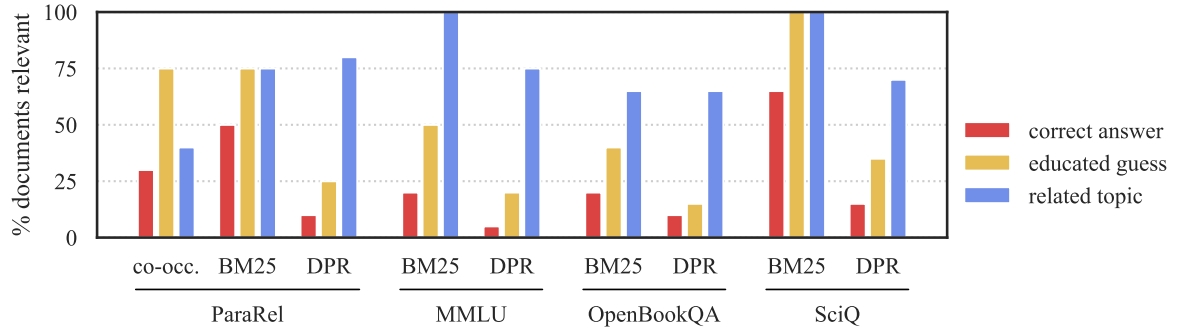


Figure 4.6: Percentage of documents labeled relevant to their matching item by **manual annotation**, for every dataset, prompt, and matching method.

to the “correct answer” prompt, 15.9–59.6% according to the “educated guess” prompt, and 39.8–91.8% according to the “relevant topic” prompt, with percentages varying considerably across datasets. This suggests that while our matching methods can identify documents that are generally relevant to a question, they are less precise when it comes to identifying documents that directly entail the answer and could provide a model with enough information during pretraining to make a correct prediction. Appendix Table C.2 shows example items from MMLU with documents matched by BM25 score that were annotated by GPT-5 as being relevant or not, for each of the three annotation prompts.

For ParaRel, cooccurrence performs comparably to BM25 at identifying relevant documents across prompts, likely because each item has a simple format with the subject and object entity names being the most relevant keywords. DPR performs worse than BM25 across all datasets at identifying relevant documents. These results align with our interventional analysis findings, where using BM25 scores to select documents led to both greater promotion and suppression of learning (depending on the experiment) than using DPR scores.

Human validation To manually validate the GPT-5 annotations, we further select a random subset of 20 item-document pairs for each {dataset, matching method} pair which were then annotated by one of the authors following the same prompts. The corresponding manual annotation results can be found in Fig. 4.6. Generally, the trends are similar to the GPT-5 annotations, although the exact percentages are different. We assess human/GPT-5 annotation agreement by computing Cohen’s κ which gives a value of 0.54 overall, indicating a moderate level of agreement.

4.8 Related Work

The most closely related work was highlighted earlier; here we note additional strands of interest.

Training data attribution Our work is related to quantifying the impact of a training example on a model’s prediction (Hammoudeh and Lowd, 2022). Among the most popular methods for doing so are gradient-based techniques such as influence functions (Koh and Liang, 2017), which have been applied to LM fine-tuning data (Han et al., 2020) as well as pretraining data (Grosse et al., 2023), with the latter requiring approximations due to high computational cost. These approaches have their own limitations, as they sometimes perform comparably to cheaper alternatives like BM25 at identifying relevant documents (Akyurek et al., 2022) and may assign high influence to documents which do not express the information (Chang et al., 2025). While often used to explain a model’s predictions rather than test hypotheses about how data causes behavior, influence functions could be integrated into our recipe for identifying target documents to intervene on.

Continued training In contrast to our method of intervening at intermediate steps, many other works continue training from the final checkpoint on select data to study behavior or improve LMs. Razeghi et al. (2023) continue pretraining on documents selected via embedding similarity or regular expression matching to improve math reasoning. Embedding or gradient similarity has also been used to select data for continued training to adapt LMs to domains and tasks (Gururangan et al., 2020), boost performance on certain tasks (Han and Tsvetkov, 2022), improve in-context learning ability (Han et al., 2023), or improve instruction-following (Iverson et al., 2023a; Xia et al., 2024). Notably, the last set of approaches focus on boosting performance rather than understanding LM behavior.

4.9 Recent Work

While the work presented in this chapter is very current at the time of writing, we highlight a few strands of relevant contemporaneous work here. Gottesman et al. (2025) have released a suite of various-sized

LMS pretrained on a version of Wikipedia with entities carefully annotated, to promote studying how pretraining data affects knowledge acquisition. Though they suggest the idea of performing interventional analyses, they do not do so in their work, although their artifacts provide a perfect testbed for the application of our recipe. [Bordt and Pawelczyk \(2026\)](#) perform experiments retraining OLMo 2 1B from scratch on a modified subset of its pretraining data to simultaneously replicate multiple past experiments, including those that study knowledge acquisition. Each experiment they perform can be considered an instantiation of our proposed recipe for studying a particular model behavior. Their results demonstrate that retraining-based analyses can be used to study multiple hypotheses simultaneously with minimal interference between interventions, a promising sign for the computational feasibility of such experiments.

4.10 Discussion and Conclusions

We have introduced a flexible recipe for interventional studies of the effects of pretraining data on LM behavior, formalizing each stage to enable systematic exploration. Our recipe highlights the importance of targeted interventions built around specific test items and studying changes at intermediate steps, not merely the final model. Our results show that simple interventions that add or remove documents relevant to some fact or behavior (either using term cooccurrence or information retrieval) have a consistent effect on promoting and suppressing learning or altering model bias. These findings support the position that exposure to a textual expression of a fact in training helps a model learn to correctly report it at test time. However, adding relevant documents does not guarantee learning, and removing them does not always prevent learning. More study is required to fully characterize how language models acquire knowledge, and we believe the methodological tools given by our recipe will aid those future efforts.

Chapter 5

Conclusion

5.1 Summary

As modern LM-based systems become more capable and widespread, it is increasingly important to better understand the limitations of their declarative and procedural knowledge capabilities. This thesis presents several streams of work that provide a clearer picture of these types of knowledge in LMs along with methodological tools for continuing this thread of research in the future.

We began in Chapter 2 by evaluating LMs’ declarative knowledge capabilities in the scientific domain via the task of knowledge graph completion, a potential indicator of LMs’ utility for scientific research and discovery. Our findings demonstrated that LMs’ are competitive with commonly-used non-text baselines for this task, with additional performance gains arising from combining the two classes of methods. We also discover that LMs are particularly useful when making predictions about entities with no existing connections in the knowledge graph, a critical capability in the biomedical domain given its rapidly growing literature with many novel drugs and rare diseases.

In Chapter 3, we studied procedural knowledge in instruction-following LMs by training other models to predict their performance on unseen instructions given known behavior on instructions they had previously been evaluated on. We conducted ablations that vary factors such as model family, model scale, prompt format, number of training instructions, and target evaluation metric to extensively study this task. Our findings indicate that performance prediction is exceptionally challenging, with our best setup only achieving slightly better performance than simple baselines, suggesting that further work is needed in either exploring novel strategies for predicting performance or building instruction-following LMs with inherently more predictable behavior.

Our final contribution in Chapter 4 proposes new research methodology for studying LM knowledge, outlining an interventional analysis recipe for connecting properties of training data to model behavior. We demonstrate the utility of this recipe through case studies on factual knowledge and gender bias,

finding that we can alter both behaviors in predictable ways with carefully constructed interventions on training documents. At the same time, we find that performance gaps remain, which suggests that our interventions do not entirely explain which documents are responsible for knowledge acquisition and indicate that further work is needed to fully characterize the data-behavior relationship in LMs.

5.2 Future Work

While the work presented in this thesis provides additional clarity on knowledge capabilities of LMs, each strand of work could lead to a number of exciting future directions. We discuss several potential directions here, with connections to relevant existing work.

Language Models for Scientific Discovery While our experiments in Chapter 2 showcased how LMs trained on the scientific literature can effectively predict connections between biomedical entities, our experimental setup did not explicitly guarantee predictions of *novel* knowledge. This is in part due to the fact that the domain-specific LMs we used had been recently developed at the time of our 2021 study, and their knowledge cutoffs were close enough to the facts being predicted that it is difficult to confirm true extrapolation to new knowledge. More recent work has integrated LMs into the scientific research pipeline in various ways, including hypothesis generation, experimental design, and data analysis (Majumder et al., 2025; Zhang et al., 2025). This includes work assessing models’ ability to make projections about novel knowledge, with varying results depending on the experimental setup (Qi et al., 2023; Wang et al., 2025b; Guo et al., 2025; Si et al., 2025). Notably, existing work either relies on the LMs’ parametric knowledge alone or augments the model with tools such as search engines (Press et al., 2023) or retrieval from a corpus (Lewis et al., 2020), but does not typically make use of structured representations of knowledge like knowledge graphs or models trained on them. Our work on scientific knowledge graph completion could be extended to test LMs’ potential for true scientific discovery via the same task, ensuring fair evaluation by accounting for factors such as the LM’s knowledge cutoff. Extensions could also explore how knowledge graph augmentation assists in this functionality, particularly if structured knowledge representations provide any additional benefit to newer, more capable LMs that are typically trained on and augmented with unstructured text.

Novel Experimental Design for Performance Prediction Our setup for predicting LM performance from task instruction made use of datasets, models, and metrics that were current at the time the work was done in early 2024, but with each having their own shortcomings. Metrics like ROUGE-L and Exact Match that we used to evaluate task performance are automatic, deterministic, and transparent, but likely too simple to properly assess complex instruction-following behavior. Other approaches for scoring model

outputs such as reward models (Ouyang et al., 2022) or LLM-as-a-judge (Zheng et al., 2023) have now become more widely used, and they may provide a signal that makes performance prediction a more learnable task. Additionally, our organization of instances into tasks along with their instructions was based on existing large-scale, human-developed instruction-following datasets (Wang et al., 2022; Wei et al., 2022; Longpre et al., 2023). Various studies of NLP tasks have shown that their relatedness and transferability is not necessarily intuitive based on the way they have been manually organized (Vu et al., 2020; Kim et al., 2023; Jang et al., 2023a). Additionally, human-written instructions may not be the best representations of a task or most predictive of an LM’s performance, since models can struggle to follow those instructions (Jang et al., 2023b; Zhou et al., 2023b; Pyatkin et al., 2025). This motivates exploring more automated ways of organizing examples into tasks, perhaps by using LMs to annotate instances with their relevant skills and grouping them into tasks based on skill similarity (Zeng et al., 2025). This could be combined with automated methods for writing and improving task instructions, to provide inputs that boost the performance of both instruction-following LMs and their performance predictors (Xu et al., 2024; Agarwal et al., 2025).

Interventional Analyses on LM Post-training While our experiments in Chapter 4 only applied our proposed interventional analysis recipe to pretraining, our recipe was designed to be generally applicable in studying how any training data relates to a behavior of interest. An intriguing future direction could apply it to studying procedural knowledge acquisition through interventions on the LM post-training process that yields instruction-following behavior. Such an approach could disentangle which downstream model behaviors are the result of examples seen during the supervised fine-tuning (Wang et al., 2022; Wei et al., 2022; Sanh et al., 2022; Wang et al., 2023a) versus reinforcement learning (Stiennon et al., 2020; Ouyang et al., 2022; Ivison et al., 2023b; Lambert et al., 2025) stages of post-training to determine their comparative contributions. This could contribute to a growing body of work studying to what extent post-training endows LMs with novel capabilities rather than surfacing skills learned during pretraining (Zhou et al., 2023a; Yue et al., 2026; Shao et al., 2025), and would motivate the use of alternative matching methods such as those used to select data for continued training to boost performance (Xia et al., 2024; Ivison et al., 2025). On a more practical note, performing an interventional analysis on post-training is dramatically more computationally feasible than it is for pretraining, allowing for interventions on entire post-training datasets for more comprehensive control over the data-behavior relationship.

References

- Eshaan Agarwal, Raghav Magazine, Joykirat Singh, Vivek Dani, Tanuja Ganu, and Akshay Nambi. 2025. [PromptWizard: Optimizing Prompts via Task-Aware, Feedback-Driven Self-Evolution](#). In *Findings of the Association for Computational Linguistics: ACL 2025*, pages 19974–20003, Vienna, Austria. Association for Computational Linguistics.
- Ekin Akyurek, Tolga Bolukbasi, Frederick Liu, Binbin Xiong, Ian Tenney, Jacob Andreas, and Kelvin Guu. 2022. [Towards Tracing Knowledge in Language Models Back to the Training Data](#). In *Findings of the Association for Computational Linguistics: EMNLP 2022*, pages 2429–2446, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.
- Zeyuan Allen-Zhu and Yuanzhi Li. 2024. [Physics of Language Models: Part 3.1, Knowledge Storage and Extraction](#). In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pages 1067–1077. PMLR.
- Zeyuan Allen-Zhu and Yuanzhi Li. 2025. [Physics of Language Models: Part 3.2, Knowledge Manipulation](#). In *The Thirteenth International Conference on Learning Representations*.
- Emily Alsentzer, John R. Murphy, Willie Boag, Wei-Hung Weng, Di Jin, Tristan Naumann, and Matthew B. A. McDermott. 2019. Publicly Available Clinical BERT Embeddings. In *2nd Clinical Natural Language Processing Workshop*.
- Mona Alshahrani, Maha A. Thafar, and Magbubah Essack. 2021. Application and evaluation of knowledge graph embeddings in biomedical data. *PeerJ Computer Science*, 7.
- Norah Alzahrani, Hisham Alyahya, Yazeed Alnumay, Sultan AlRashed, Shaykhah Alsubaie, Yousef Almushayqih, Faisal Mirza, Nouf Alotaibi, Nora Al-Twairesh, Areeb Alowisheq, M Saiful Bari, and Haidar Khan. 2024. [When Benchmarks are Targets: Revealing the Sensitivity of Large Language Model Leaderboards](#). In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 13787–13805, Bangkok, Thailand. Association for Computational Linguistics.
- Anthropic. 2023. [Introducing Claude](https://www.anthropic.com/news/introducing-claude). <https://www.anthropic.com/news/introducing-claude>. Accessed: 2026-04-05.
- Isabelle Augenstein, Diana Maynard, and Fabio Ciravegna. 2016. [Distantly supervised Web relation extraction for knowledge base population](#). *Semantic Web*, 7(4):335–349.
- Jacob Austin, Augustus Odena, Maxwell Nye, Maarten Bosma, Henryk Michalewski, David Dohan, Ellen Jiang, Carrie Cai, Michael Terry, Quoc Le, and Charles Sutton. 2021. [Program Synthesis with Large Language Models](#). *Preprint*, arXiv:2108.07732.
- Helena Balabin, Charles Tapley Hoyt, Colin Birkenbihl, Benjamin M Gyori, John Bachman, Alpha Tom Kodamullil, Paul G Plöger, Martin Hofmann-Apitius, and Daniel Domingo-Fernández. 2022. [STonKGs: a sophisticated transformer trained on biomedical text and knowledge graphs](#). *Bioinformatics*, 38(6):1648–1656.
- Nishant Balepur, Bhavya Rajasekaran, Jane Oh, Michael Xie, Atrey Desai, Vipul Gupta, Steven James Moore, Eunsol Choi, Rachel Rudinger, and Jordan Lee Boyd-Graber. 2026. [BenchMarker: An Education-Inspired Toolkit for Highlighting Flaws in Multiple-Choice Benchmarks](#). *Preprint*, arXiv:2602.06221.
- Iz Beltagy, Kyle Lo, and Arman Cohan. 2019. SciBERT: A Pretrained Language Model for Scientific Text. In *EMNLP*.

- Yoshua Bengio, Réjean Ducharme, Pascal Vincent, and Christian Janvin. 2003. A neural probabilistic language model. *J. Mach. Learn. Res.*, 3(null):1137–1155.
- Akshita Bhagia, Jiacheng Liu, Alexander Wettig, David Heineman, Oyvind Tafjord, Ananya Harsh Jha, Luca Soldaini, Noah A. Smith, Dirk Groeneveld, Pang Wei Koh, Jesse Dodge, and Hannaneh Hajishirzi. 2025. [Establishing Task Scaling Laws via Compute-Efficient Model Ladders](#). In *Second Conference on Language Modeling*.
- Rajarshi Bhowmik and Gerard de Melo. 2020. Explainable Link Prediction for Emerging Entities in Knowledge Graphs. In *SEMWEB*.
- Stella Biderman, Hailey Schoelkopf, Quentin Gregory Anthony, Herbie Bradley, Kyle O’Brien, Eric Hallahan, Mohammad Aflah Khan, Shivanshu Purohit, Usven Sai Prashanth, Edward Raff, Aviya Skowron, Lintang Sutawika, and Oskar Van Der Wal. 2023. [Pythia: A Suite for Analyzing Large Language Models Across Training and Scaling](#). In *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pages 2397–2430. PMLR.
- Stella Biderman, Hailey Schoelkopf, Lintang Sutawika, Leo Gao, Jonathan Tow, Baber Abbasi, Alham Fikri Aji, Pawan Sasanka Ammanamanchi, Sidney Black, Jordan Clive, Anthony DiPofi, Julien Etxaniz, Benjamin Fattori, Jessica Zosa Forde, Charles Foster, Jeffrey Hsu, Mimansa Jaiswal, Wilson Y. Lee, Haonan Li, and 11 others. 2024. [Lessons from the Trenches on Reproducible Evaluation of Language Models](#). *Preprint*, arXiv:2405.14782.
- BIG-bench authors. 2023. [Beyond the Imitation Game: Quantifying and extrapolating the capabilities of language models](#). *Transactions on Machine Learning Research*.
- Kurt Bollacker, Colin Evans, Praveen K. Paritosh, Tim Sturge, and Jamie Taylor. 2008. Freebase: A Collaboratively Created Graph Database For Structuring Human Knowledge. In *SIGMOD Conference*.
- Stephen Bonner, Ian P Barrett, Cheng Ye, Rowan Swiers, Ola Engkvist, Andreas Bender, Charles Tapley Hoyt, and William L Hamilton. 2022a. [A review of biomedical datasets relating to drug discovery: a knowledge graph perspective](#). *Briefings in Bioinformatics*, 23(6):bbac404.
- Stephen Bonner, Ian P. Barrett, Cheng Ye, Rowan Swiers, Ola Engkvist, Charles Tapley Hoyt, and William L. Hamilton. 2022b. [Understanding the performance of knowledge graph embeddings in drug discovery](#). *Artificial Intelligence in the Life Sciences*, 2:100036.
- Antoine Bordes, Nicolas Usunier, Alberto García-Durán, Jason Weston, and Oksana Yakhnenko. 2013. Translating Embeddings for Modeling Multi-relational Data. In *NIPS*.
- Shikha Bordia and Samuel R. Bowman. 2019. [Identifying and Reducing Gender Bias in Word-Level Language Models](#). In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Student Research Workshop*, pages 7–15, Minneapolis, Minnesota. Association for Computational Linguistics.
- Sebastian Bordt and Martin Pawelczyk. 2026. [Train Once, Answer All: Many Pretraining Experiments for the Cost of One](#). In *The Fourteenth International Conference on Learning Representations*.
- Adam S. Brown and Chirag J. Patel. 2017. [A standard database for drug repositioning](#). *Scientific Data*, 4.
- 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, and 12 others. 2020. [Language Models are Few-Shot Learners](#). In *Advances in Neural Information Processing Systems*, volume 33, pages 1877–1901. Curran Associates, Inc.

- Boxi Cao, Hongyu Lin, Xianpei Han, Le Sun, Lingyong Yan, Meng Liao, Tong Xue, and Jin Xu. 2021. [Knowledgeable or Educated Guess? Revisiting Language Models as Knowledge Bases](#). In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 1860–1874, Online. Association for Computational Linguistics.
- Dallas Card, Peter Henderson, Urvashi Khandelwal, Robin Jia, Kyle Mahowald, and Dan Jurafsky. 2020. [With Little Power Comes Great Responsibility](#). In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 9263–9274, Online. Association for Computational Linguistics.
- Payal Chandak, Kexin Huang, and Marinka Zitnik. 2023. [Building a knowledge graph to enable precision medicine](#). *Scientific Data*, 10(67):1–16.
- David Chang, Ivana Balažević, Carl Allen, Daniel Chawla, Cynthia Brandt, and Andrew Taylor. 2020. Benchmark and Best Practices for Biomedical Knowledge Graph Embeddings. In *19th SIGBioMed Workshop on Biomedical Language Processing*, pages 167–176.
- Hoyeon Chang, Jinho Park, Seonghyeon Ye, Sohee Yang, Youngkyung Seo, Du-Seong Chang, and Minjoon Seo. 2024. [How Do Large Language Models Acquire Factual Knowledge During Pretraining?](#) In *Advances in Neural Information Processing Systems*.
- Tyler A. Chang, Dheeraj Rajagopal, Tolga Bolukbasi, Lucas Dixon, and Ian Tenney. 2025. [Scalable Influence and Fact Tracing for Large Language Model Pretraining](#). In *The Thirteenth International Conference on Learning Representations*.
- Guiming Hardy Chen, Shunian Chen, Ziche Liu, Feng Jiang, and Benyou Wang. 2024. [Humans or LLMs as the Judge? A Study on Judgement Bias](#). In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 8301–8327, Miami, Florida, USA. 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, Alex Ray, Raul Puri, Gretchen Krueger, Michael Petrov, Heidy Khlaaf, Girish Sastry, Pamela Mishkin, Brooke Chan, Scott Gray, and 39 others. 2021. [Evaluating Large Language Models Trained on Code](#). *Preprint*, arXiv:2107.03374.
- Tianqi Chen and Carlos Guestrin. 2016. XGBoost: A Scalable Tree Boosting System. *KDD*.
- Silei Cheng, Zhe Gan, Zhengyuan Yang, Shuohang Wang, Jianfeng Wang, Jordan Boyd-Graber, and Lijuan Wang. 2023. [Prompting GPT-3 To Be Reliable](#). In *International Conference on Learning Representations*.
- Wei-Lin Chiang, Zhuohan Li, Zi Lin, Ying Sheng, Zhanghao Wu, Hao Zhang, Lianmin Zheng, Siyuan Zhuang, Yonghao Zhuang, Joseph E. Gonzalez, Ion Stoica, and Eric P. Xing. 2023. [Vicuna: An Open-Source Chatbot Impressing GPT-4 with 90%* ChatGPT Quality](#).
- Wei-Lin Chiang, Lianmin Zheng, Ying Sheng, Anastasios N. Angelopoulos, Tianle Li, Dacheng Li, Banghua Zhu, Hao Zhang, Michael I. Jordan, Joseph E. Gonzalez, and Ion Stoica. 2024a. Chatbot arena: an open platform for evaluating LLMs by human preference. In *Proceedings of the 41st International Conference on Machine Learning, ICML’24*. JMLR.org.
- Wei-Lin Chiang, Lianmin Zheng, Ying Sheng, Anastasios Nikolas Angelopoulos, Tianle Li, Dacheng Li, Banghua Zhu, Hao Zhang, Michael Jordan, Joseph E. Gonzalez, and Ion Stoica. 2024b. [Chatbot Arena: An Open Platform for Evaluating LLMs by Human Preference](#). In *Forty-first International Conference on Machine Learning*.

- Aakanksha Chowdhery, Sharan Narang, Jacob Devlin, Maarten Bosma, Gaurav Mishra, Adam Roberts, Paul Barham, Hyung Won Chung, Charles Sutton, Sebastian Gehrmann, Parker Schuh, Kensen Shi, Sashank Tsvyashchenko, Joshua Maynez, Abhishek Rao, Parker Barnes, Yi Tay, Noam Shazeer, Vinodkumar Prabhakaran, and 48 others. 2023. PaLM: scaling language modeling with pathways. *J. Mach. Learn. Res.*, 24(1).
- Paul Francis Christiano, Jan Leike, Tom B. Brown, Miljan Martic, Shane Legg, and Dario Amodei. 2017. [Deep Reinforcement Learning from Human Preferences](#). In *Conference on Neural Information Processing Systems*.
- Hyung Won Chung, Le Hou, Shayne Longpre, Barret Zoph, Yi Tay, William Fedus, Yunxuan Li, Xuezhi Wang, Mostafa Dehghani, Siddhartha Brahma, Albert Webson, Shixiang Shane Gu, Zhuyun Dai, Mirac Suzgun, Xinyun Chen, Aakanksha Chowdhery, Alex Castro-Ros, Marie Pellat, Kevin Robinson, and 16 others. 2024. [Scaling Instruction-Finetuned Language Models](#). *Journal of Machine Learning Research*, 25(70):1–53.
- CoAStAL NLP Group. 2024. [coastalcph/pararel_patterns](#). https://huggingface.co/datasets/coastalcph/pararel_patterns.
- Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, Christopher Hesse, and John Schulman. 2021. [Training Verifiers to Solve Math Word Problems](#). *Preprint*, arXiv:2110.14168.
- Daniel Daza, Michael Cochez, and Paul T. Groth. 2021. Inductive Entity Representations from Text via Link Prediction. In *WWW*.
- Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In *NAACL-HLT*.
- Yanai Elazar, Akshita Bhagia, Ian Helgi Magnusson, Abhilasha Ravichander, Dustin Schwenk, Alane Suhr, Evan Pete Walsh, Dirk Groeneveld, Luca Soldaini, Sameer Singh, Hannaneh Hajishirzi, Noah A. Smith, and Jesse Dodge. 2024. [What’s In My Big Data?](#) In *The Twelfth International Conference on Learning Representations*.
- Yanai Elazar, Nora Kassner, Shauli Ravfogel, Amir Feder, Abhilasha Ravichander, Marius Mosbach, Yonatan Belinkov, Hinrich Schütze, and Yoav Goldberg. 2022. [Measuring Causal Effects of Data Statistics on Language Model’s ‘Factual’ Predictions](#). *arXiv preprint arXiv:2207.14251v2*.
- Yanai Elazar, Nora Kassner, Shauli Ravfogel, Abhilasha Ravichander, Eduard Hovy, Hinrich Schütze, and Yoav Goldberg. 2021. [Measuring and Improving Consistency in Pretrained Language Models](#). *Transactions of the Association for Computational Linguistics*, 9:1012–1031.
- Hady Elsahar, Pavlos Vougiouklis, Arslan Remaci, Christophe Gravier, Jonathon Hare, Frederique Laforest, and Elena Simperl. 2018. [T-REx: A Large Scale Alignment of Natural Language with Knowledge Base Triples](#). In *Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC 2018)*, Miyazaki, Japan. European Language Resources Association (ELRA).
- Amir Feder, Katherine A. Keith, Emaad Manzoor, Reid Pryzant, Dhanya Sridhar, Zach Wood-Doughty, Jacob Eisenstein, Justin Grimmer, Roi Reichart, Margaret E. Roberts, Brandon M. Stewart, Victor Veitch, and Diyi Yang. 2022. [Causal Inference in Natural Language Processing: Estimation, Prediction, Interpretation and Beyond](#). *Transactions of the Association for Computational Linguistics*, 10.
- Aosong Feng, Balasubramaniam Srinivasan, Yun Zhou, Zhichao Xu, Kang Zhou, Sheng Guan, Yueyan Chen, Xian Wu, Ninad Kulkarni, Yi Zhang, Zhengyuan Shen, Dmitriy Besspalov, Soumya Smruti Mishra, Yifei Teng, Darren Yow-Bang Wang, Haibo Ding, and Lin Lee Cheong. 2025. [IPR: Intelligent Prompt Routing with User-Controlled Quality-Cost Trade-offs](#). In *Proceedings of the 2025 Conference*

- on *Empirical Methods in Natural Language Processing: Industry Track*, pages 2484–2498, Suzhou (China). Association for Computational Linguistics.
- Harvey Fu, Qinyuan Ye, Albert Xu, Xiang Ren, and Robin Jia. 2023. [Estimating Large Language Model Capabilities without Labeled Test Data](#). In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 9530–9546.
- Leo Gao, Jonathan Tow, Stella Biderman, Sid Black, Anthony DiPofi, Charles Foster, Laurence Golding, Jeffrey Hsu, Kyle McDonell, Niklas Muennighoff, Jason Phang, Laria Reynolds, Eric Tang, Anish Thite, Ben Wang, Kevin Wang, and Andy Zou. 2021. [A framework for few-shot language model evaluation](#).
- Zorik Gekhman, Eyal Ben-David, Hadas Orgad, Eran Ofek, Yonatan Belinkov, Idan Szpektor, Jonathan Herzig, and Roi Reichart. 2025. [Inside-Out: Hidden Factual Knowledge in LLMs](#). In *Second Conference on Language Modeling*.
- Aryo Pradipta Gema, Joshua Ong Jun Leang, Giwon Hong, Alessio Devoto, Alberto Carlo Maria Mancino, Rohit Saxena, Xuanli He, Yu Zhao, Xiaotang Du, Mohammad Reza Ghasemi Madani, Claire Barale, Robert McHardy, Joshua Harris, Jean Kaddour, Emile Van Krieken, and Pasquale Minervini. 2025. [Are We Done with MMLU?](#) In *Proceedings of the 2025 Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pages 5069–5096, Albuquerque, New Mexico. Association for Computational Linguistics.
- Google. 2023. [Introducing Gemini: our largest and most capable AI model](#). <https://blog.google/innovation-and-ai/technology/ai/google-gemini-ai>. Accessed: 2026-04-05.
- Vishrawas Gopalakrishnan, Kishlay Jha, Wei Jin, and Aidong Zhang. 2019. A survey on literature based discovery approaches in biomedical domain. *Journal of biomedical informatics*, 93:103141.
- Daniela Gottesman, Alon Gilae-Dotan, Ido Cohen, Yoav Gur-Arieh, Marius Mosbach, Ori Yoran, and Mor Geva. 2025. [LMEnt: A Suite for Analyzing Knowledge in Language Models from Pretraining Data to Representations](#). *Preprint*, arXiv:2509.03405.
- Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, Amy Yang, Angela Fan, Anirudh Goyal, Anthony Hartshorn, Aobo Yang, Archi Mitra, Archie Sravankumar, Artem Korenev, Arthur Hinsvark, and 542 others. 2024. [The Llama 3 Herd of Models](#). *Preprint*, arXiv:2407.21783v3.
- Dirk Groeneveld, Iz Beltagy, Evan Walsh, Akshita Bhagia, Rodney Kinney, Oyvind Tafjord, Ananya Jha, Hamish Ivison, Ian Magnusson, Yizhong Wang, Shane Arora, David Atkinson, Russell Authur, Khyathi Chandu, Arman Cohan, Jennifer Dumas, Yanai Elazar, Yuling Gu, Jack Hessel, and 24 others. 2024. [OLMo: Accelerating the Science of Language Models](#). In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 15789–15809, Bangkok, Thailand. Association for Computational Linguistics.
- Roger Grosse, Juhan Bae, Cem Anil, Nelson Elhage, Alex Tamkin, Amirhossein Tajdini, Benoit Steiner, Dustin Li, Esin Durmus, Ethan Perez, Evan Hubinger, Kamilė Lukošiuūtė, Karina Nguyen, Nicholas Joseph, Sam McCandlish, Jared Kaplan, and Samuel R. Bowman. 2023. [Studying Large Language Model Generalization with Influence Functions](#).
- Yu Gu, Robert Tinn, Hao Cheng, Michael Lucas, Naoto Usuyama, Xiaodong Liu, Tristan Naumann, Jianfeng Gao, and Hoifung Poon. 2021. [Domain-Specific Language Model Pretraining for Biomedical Natural Language Processing](#). *ACM Trans. Comput. Healthcare*, 3(1).

- Yuling Gu, Oyvind Tafjord, Bailey Kuehl, Dany Haddad, Jesse Dodge, and Hannaneh Hajishirzi. 2025. [OLMES: A Standard for Language Model Evaluations](#). In *Findings of the Association for Computational Linguistics: NAACL 2025*, pages 5005–5033, Albuquerque, New Mexico. Association for Computational Linguistics.
- Sikun Guo, Amir Hassan Shariatmadari, Guangzhi Xiong, Albert Huang, Myles Kim, Corey M. Williams, Stefan Bekiranov, and Aidong Zhang. 2025. [IdeaBench: Benchmarking Large Language Models for Research Idea Generation](#). In *Proceedings of the 31st ACM SIGKDD Conference on Knowledge Discovery and Data Mining V.2*, KDD '25, page 5888–5899, New York, NY, USA. Association for Computing Machinery.
- Suchin Gururangan, Ana Marasović, Swabha Swayamdipta, Kyle Lo, Iz Beltagy, Doug Downey, and Noah A. Smith. 2020. [Don't Stop Pretraining: Adapt Language Models to Domains and Tasks](#). In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 8342–8360, Online. Association for Computational Linguistics.
- Zayd Hammoudeh and Daniel Lowd. 2022. [Training Data Influence Analysis and Estimation: A Survey](#). *Mach. Learn.*, 113:2351–2403.
- 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 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 12660–12673, Toronto, Canada. Association for Computational Linguistics.
- Xiaochuang Han and Yulia Tsvetkov. 2022. [ORCA: Interpreting Prompted Language Models via Locating Supporting Data Evidence in the Ocean of Pretraining Data](#). *Preprint*, arXiv:2205.12600v1.
- Xiaochuang Han, Byron C. Wallace, and Yulia Tsvetkov. 2020. [Explaining Black Box Predictions and Unveiling Data Artifacts through Influence Functions](#). In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 5553–5563, Online. Association for Computational Linguistics.
- Qiyuan He, Yizhong Wang, Jianfei Yu, and Wenya Wang. 2025. [Language Models over Large-Scale Knowledge Base: on Capacity, Flexibility and Reasoning for New Facts](#). In *Proceedings of the 31st International Conference on Computational Linguistics*, pages 1736–1753, Abu Dhabi, UAE. Association for Computational Linguistics.
- Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. 2021a. [Measuring Massive Multitask Language Understanding](#). In *International Conference on Learning Representations*.
- Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. 2021b. [Measuring Mathematical Problem Solving With the MATH Dataset](#). In *Proceedings of the Neural Information Processing Systems Track on Datasets and Benchmarks*, volume 1.
- Daniel S. Himmelstein and Sergio E. Baranzini. 2015. Heterogeneous Network Edge Prediction: A Data Integration Approach to Prioritize Disease-Associated Genes. *PLoS Computational Biology*, 11.
- Ari Holtzman, Peter West, Vered Shwartz, Yejin Choi, and Luke Zettlemoyer. 2021. [Surface Form Competition: Why the Highest Probability Answer Isn't Always Right](#). In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 7038–7051.
- Matthew Honnibal, Ines Montani, Sofie Van Landeghem, and Adriane Boyd. 2020. [spaCy: Industrial-strength Natural Language Processing in Python](#).

- Or Honovich, Thomas Scialom, Omer Levy, and Timo Schick. 2023. [Unnatural Instructions: Tuning Language Models with \(Almost\) No Human Labor](#). In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 14409–14428.
- Edward J Hu, yelong shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. 2022. [LoRA: Low-Rank Adaptation of Large Language Models](#). In *International Conference on Learning Representations*.
- Hamish Ivison, Noah A. Smith, Hannaneh Hajishirzi, and Pradeep Dasigi. 2023a. [Data-Efficient Finetuning Using Cross-Task Nearest Neighbors](#). In *Findings of the Association for Computational Linguistics: ACL 2023*, pages 9036–9061, Toronto, Canada. Association for Computational Linguistics.
- Hamish Ivison, Yizhong Wang, Valentina Pyatkin, Nathan Lambert, Matthew Peters, Pradeep Dasigi, Joel Jang, David Wadden, Noah A. Smith, Iz Beltagy, and Hannaneh Hajishirzi. 2023b. [Camels in a Changing Climate: Enhancing LM Adaptation with Tulu 2](#). *Preprint*, arXiv:2311.10702.
- Hamish Ivison, Muru Zhang, Faeze Brahman, Pang Wei Koh, and Pradeep Dasigi. 2025. [Large-Scale Data Selection for Instruction Tuning](#). *Preprint*, arXiv:2503.01807.
- 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 40th International Conference on Machine Learning, ICML’23*. JMLR.org.
- Joel Jang, Seonghyeon Ye, and Minjoon Seo. 2023b. [Can Large Language Models Truly Understand Prompts? A Case Study with Negated Prompts](#). In *Proceedings of The 1st Transfer Learning for Natural Language Processing Workshop*, volume 203 of *Proceedings of Machine Learning Research*, pages 52–62. PMLR.
- Ziwei Ji, Nayeon Lee, Rita Frieske, Tiezheng Yu, Dan Su, Yan Xu, Etsuko Ishii, Ye Jin Bang, Andrea Madotto, and Pascale Fung. 2023. [Survey of Hallucination in Natural Language Generation](#). *ACM Comput. Surv.*, 55(12).
- Zhengbao Jiang, Antonios Anastasopoulos, Jun Araki, Haibo Ding, and Graham Neubig. 2020. [X-FACTR: Multilingual Factual Knowledge Retrieval from Pretrained Language Models](#). In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 5943–5959, Online. Association for Computational Linguistics.
- Carlos E Jimenez, John Yang, Alexander Wettig, Shunyu Yao, Kexin Pei, Ofir Press, and Karthik R Narasimhan. 2024. [SWE-bench: Can Language Models Resolve Real-world Github Issues?](#) In *The Twelfth International Conference on Learning Representations*.
- Saurav Kadavath, Tom Conerly, Amanda Askell, T. J. Henighan, Dawn Drain, Ethan Perez, Nicholas Schiefer, Zachary Dodds, Nova DasSarma, Eli Tran-Johnson, Scott Johnston, Sheer El-Showk, Andy Jones, Nelson Elhage, Tristan Hume, Anna Chen, Yuntao Bai, Sam Bowman, Stanislav Fort, and 17 others. 2022. [Language Models \(Mostly\) Know What They Know](#). *ArXiv*, abs/2207.05221.
- Nikhil Kandpal, Haikang Deng, Adam Roberts, Eric Wallace, and Colin Raffel. 2023. [Large Language Models Struggle to Learn Long-Tail Knowledge](#). In *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pages 15696–15707. PMLR.
- Cheongwoong Kang and Jaesik Choi. 2023. [Impact of Co-occurrence on Factual Knowledge of Large Language Models](#). In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 7721–7735, Singapore. Association for Computational Linguistics.

- Vladimir Karpukhin, Barlas Oguz, 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 the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 6769–6781, Online. Association for Computational Linguistics.
- Nora Kassner, Philipp Dufter, and Hinrich Schütze. 2021. [Multilingual LAMA: Investigating Knowledge in Multilingual Pretrained Language Models](#). In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume*, pages 3250–3258, Online. Association for Computational Linguistics.
- Nora Kassner, Benno Krojer, and Hinrich Schütze. 2020. [Are Pretrained Language Models Symbolic Reasoners over Knowledge?](#) In *Proceedings of the 24th Conference on Computational Natural Language Learning*, pages 552–564, Online. Association for Computational Linguistics.
- Bosung Kim, Taesuk Hong, Youngjoong Ko, and Jungyun Seo. 2020. Multi-Task Learning for Knowledge Graph Completion with Pre-trained Language Models. In *COLING*.
- Jiyeon Kim, Hyunji Lee, Hyowon Cho, Joel Jang, Hyeonbin Hwang, Seungpil Won, Youbin Ahn, Dohaeng Lee, and Minjoon Seo. 2025. [Knowledge Entropy Decay during Language Model Pretraining Hinders New Knowledge Acquisition](#). In *The Thirteenth International Conference on Learning Representations*.
- Joongwon Kim, Akari Asai, Gabriel Ilharco, and Hannaneh Hajishirzi. 2023. [TaskWeb: Selecting Better Source Tasks for Multi-task NLP](#). In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 11032–11052, Singapore. Association for Computational Linguistics.
- Diederik P. Kingma and Jimmy Ba. 2015. Adam: A Method for Stochastic Optimization. In *ICLR*.
- Pang Wei Koh and Percy Liang. 2017. [Understanding Black-box Predictions via Influence Functions](#). In *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pages 1885–1894. PMLR.
- Ryan Koo, Minhwa Lee, Vipul Raheja, Jong Inn Park, Zae Myung Kim, and Dongyeop Kang. 2024. [Benchmarking Cognitive Biases in Large Language Models as Evaluators](#). In *Findings of the Association for Computational Linguistics: ACL 2024*, pages 517–545, Bangkok, Thailand. Association for Computational Linguistics.
- Nathan Lambert, Jacob Morrison, Valentina Pyatkin, Shengyi Huang, Hamish Ivison, Faeze Brahman, Lester James Validad Miranda, Alisa Liu, Nouha Dziri, Xinxu Lyu, Yuling Gu, Saumya Malik, Victoria Graf, Jena D. Hwang, Jiangjiang Yang, Ronan Le Bras, Oyvind Tafjord, Christopher Wilhelm, Luca Soldaini, and 4 others. 2025. [Tulu 3: Pushing Frontiers in Open Language Model Post-Training](#). In *Second Conference on Language Modeling*.
- Jinhyuk Lee, Wonjin Yoon, Sungdong Kim, Donghyeon Kim, Sunkyu Kim, Chan Ho So, and Jaewoo Kang. 2020. BioBERT: a pre-trained biomedical language representation model for biomedical text mining. *Bioinformatics*, 36:1234 – 1240.
- Brian Lester, Rami Al-Rfou, and Noah Constant. 2021. [The Power of Scale for Parameter-Efficient Prompt Tuning](#). In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 3045–3059, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.
- Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, Sebastian Riedel, and Douwe Kiela. 2020. [Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks](#). In *Advances in Neural Information Processing Systems*, volume 33, pages 9459–9474. Curran Associates, Inc.

- Jeffrey Li, Alex Fang, Georgios Smyrnis, Maor Ivgi, Matt Jordan, Samir Yitzhak Gadre, Hritik Bansal, Etash Kumar Guha, Sedrick Keh, Kushal Arora, Saurabh Garg, Rui Xin, Niklas Muennighoff, Reinhard Heckel, Jean Mercat, Mayee F Chen, Suchin Gururangan, Mitchell Wortsman, Alon Albalak, and 40 others. 2024a. [DataComp-LM: In search of the next generation of training sets for language models](#). In *The Thirty-eight Conference on Neural Information Processing Systems Datasets and Benchmarks Track*.
- Jeffrey Li, Alex Fang, Georgios Smyrnis, Maor Ivgi, Matt Jordan, Samir Yitzhak Gadre, Hritik Bansal, Etash Kumar Guha, Sedrick Keh, Kushal Arora, Saurabh Garg, Rui Xin, Niklas Muennighoff, Reinhard Heckel, Jean Mercat, Mayee F Chen, Suchin Gururangan, Mitchell Wortsman, Alon Albalak, and 40 others. 2024b. [DataComp-LM: In search of the next generation of training sets for language models](#). In *The Thirty-eight Conference on Neural Information Processing Systems Datasets and Benchmarks Track*.
- Xuechen Li, Tianyi Zhang, Yann Dubois, Rohan Taori, Ishaan Gulrajani, Carlos Guestrin, Percy Liang, and Tatsunori B. Hashimoto. 2023. AlpacaEval: An Automatic Evaluator of Instruction-following Models. https://github.com/tatsu-lab/alpaca_eval.
- Chin-Yew Lin. 2004. [ROUGE: A Package for Automatic Evaluation of Summaries](#). In *Association for Computational Linguistics*.
- Stephanie Lin, Jacob Hilton, and Owain Evans. 2022a. [Teaching Models to Express Their Uncertainty in Words](#). *Transactions on Machine Learning Research*.
- Stephanie Lin, Jacob Hilton, and Owain Evans. 2022b. [TruthfulQA: Measuring How Models Mimic Human Falsehoods](#). In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 3214–3252, Dublin, Ireland. Association for Computational Linguistics.
- Mark A Lindsay. 2003. Target discovery. *Nature Reviews Drug Discovery*, 2(10):831–838.
- 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:1907.11692*.
- Shayne Longpre, Le Hou, Tu Vu, Albert Webson, Hyung Won Chung, Yi Tay, Denny Zhou, Quoc V. Le, Barret Zoph, Jason Wei, and Adam Roberts. 2023. The Flan Collection: Designing Data and Methods for Effective Instruction Tuning. In *Proceedings of the 40th International Conference on Machine Learning, ICML’23*. JMLR.org.
- Shayne Longpre, Gregory Yauney, Emily Reif, Katherine Lee, Adam Roberts, Barret Zoph, Denny Zhou, Jason Wei, Kevin Robinson, David Mimno, and Daphne Ippolito. 2024. [A Pretrainer’s Guide to Training Data: Measuring the Effects of Data Age, Domain Coverage, Quality, & Toxicity](#). In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pages 3245–3276, Mexico City, Mexico. Association for Computational Linguistics.
- Huimin Luo, Min Li, Mengyun Yang, Fang-Xiang Wu, Yaohang Li, and Jianxin Wang. 2021. Biomedical data and computational models for drug repositioning: a comprehensive review. *Briefings in bioinformatics*, 22(2):1604–1619.
- Ian Magnusson, Nguyen Tai, Ben Bogin, David Heineman, Jena D. Hwang, Luca Soldaini, Akshita Bhagia, Jiacheng Liu, Dirk Groeneveld, Oyvind Tafjord, Noah A. Smith, Pang Wei Koh, and Jesse Dodge. 2025. [DataDecide: How to Predict Best Pretraining Data with Small Experiments](#). In *Forty-second International Conference on Machine Learning*.

- Bodhisattwa Prasad Majumder, Harshit Surana, Dhruv Agarwal, Bhavana Dalvi Mishra, Abhijeetsingh Meena, Aryan Prakhara, Tirth Vora, Tushar Khot, Ashish Sabharwal, and Peter Clark. 2025. [DiscoveryBench: Towards Data-Driven Discovery with Large Language Models](#). In *The Thirteenth International Conference on Learning Representations*.
- Alex Mallen, Akari Asai, Victor Zhong, Rajarshi Das, Daniel Khashabi, and Hannaneh Hajishirzi. 2023. [When Not to Trust Language Models: Investigating Effectiveness of Parametric and Non-Parametric Memories](#). In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 9802–9822, Toronto, Canada. Association for Computational Linguistics.
- Kim Martineau. 2024. [LLM routing for quality, low-cost responses](#). IBM Research Blog.
- Zaiqiao Meng, Fangyu Liu, Ehsan Shareghi, Yixuan Su, Charlotte Collins, and Nigel Collier. 2022. [Rewire-then-Probe: A Contrastive Recipe for Probing Biomedical Knowledge of Pre-trained Language Models](#). In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 4798–4810, Dublin, Ireland. Association for Computational Linguistics.
- Jack Merullo, Noah A. Smith, Sarah Wiegreffe, and Yanai Elazar. 2025. On Linear Representations and Pretraining Data Frequency in Language Models. In *The Thirteenth International Conference on Learning Representations*.
- 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 *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 2381–2391, Brussels, Belgium. Association for Computational Linguistics.
- George A. Miller. 1995. WordNet: a lexical database for English. *Commun. ACM*, 38:39–41.
- Mike Mintz, Steven Bills, Rion Snow, and Daniel Jurafsky. 2009. [Distant supervision for relation extraction without labeled data](#). In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP*, pages 1003–1011, Suntec, Singapore. Association for Computational Linguistics.
- Swaroop Mishra, Daniel Khashabi, Chitta Baral, and Hannaneh Hajishirzi. 2022. [Cross-Task Generalization via Natural Language Crowdsourcing Instructions](#). In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*.
- Moin Nadeem, Anna Bethke, and Siva Reddy. 2021. [StereoSet: Measuring stereotypical bias in pretrained language models](#). In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 5356–5371, Online. Association for Computational Linguistics.
- Rahul Nadkarni, Yanai Elazar, Hila Gonen, and Noah A. Smith. 2025. [Rewriting History: A Recipe for Interventional Analyses to Study Data Effects on Model Behavior](#). *Transactions of the Association for Computational Linguistics*. (to appear).
- Rahul Nadkarni, David Wadden, Iz Beltagy, Noah A. Smith, Hannaneh Hajishirzi, and Tom Hope. 2021. [Scientific Language Models for Biomedical Knowledge Base Completion: An Empirical Study](#). In *Conference on Automated Knowledge Base Completion (AKBC)*.
- Rahul Nadkarni, Yizhong Wang, and Noah A. Smith. 2024. [Third-Party Language Model Performance Prediction from Instruction](#). *Preprint*, arXiv:2403.12413.

- Deepak Narayanan, Mohammad Shoeybi, Jared Casper, Patrick LeGresley, Mostofa Patwary, Vijay Korthikanti, Dmitri Vainbrand, Prethvi Kashinkunti, Julie Bernauer, Bryan Catanzaro, Amar Phanishayee, and Matei Zaharia. 2021. [Efficient Large-Scale Language Model Training on GPU Clusters Using Megatron-LM](#). In *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*, SC '21, New York, NY, USA. Association for Computing Machinery.
- Roberto Navigli, Simone Conia, and Björn Ross. 2023. [Biases in Large Language Models: Origins, Inventory, and Discussion](#). *J. Data and Information Quality*, 15(2).
- Isaac Ong, Amjad Almahairi, Vincent Wu, Wei-Lin Chiang, Tianhao Wu, Joseph E. Gonzalez, M Waleed Kadous, and Ion Stoica. 2025. [RouteLLM: Learning to Route LLMs from Preference Data](#). In *The Thirteenth International Conference on Learning Representations*.
- OpenAI. 2022. [Introducing ChatGPT](#). <https://openai.com/blog/chatgpt>. Accessed: 2024-01-15.
- OpenAI. 2023. [GPT-4 Technical Report](#).
- OpenAI. 2025. [Introducing GPT-5](#). <https://openai.com/gpt-5/>.
- Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, John Schulman, Jacob Hilton, Fraser Kelton, Luke Miller, Maddie Simens, Amanda Askell, Peter Welinder, Paul F Christiano, Jan Leike, and Ryan Lowe. 2022. [Training language models to follow instructions with human feedback](#). In *Advances in Neural Information Processing Systems*, volume 35, pages 27730–27744. Curran Associates, Inc.
- Jungsoo Park, Ethan Mendes, Gabriel Stanovsky, and Alan Ritter. 2026. [Anticipatory Evaluation of Language Models](#). *Preprint*, arXiv:2509.20645.
- Judea Pearl. 2009. *Causality: Models, Reasoning, and Inference*, 2 edition. Cambridge University Press, Cambridge, UK.
- F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, G. Louppe, P. Prettenhofer, R. Weiss, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. 2011. Scikit-learn: Machine Learning in Python. *J. Mach. Learn. Res.*, 12:2825–2830.
- Guilherme Penedo, Hynek Kydlíček, Loubna Ben allal, Anton Lozhkov, Margaret Mitchell, Colin Raffel, Leandro Von Werra, and Thomas Wolf. 2024. [The FineWeb Datasets: Decanting the Web for the Finest Text Data at Scale](#). In *The Thirty-eight Conference on Neural Information Processing Systems Datasets and Benchmarks Track*.
- Matthew E. Peters, Mark Neumann, Robert Logan, Roy Schwartz, Vidur Joshi, Sameer Singh, and Noah A. Smith. 2019. Knowledge Enhanced Contextual Word Representations. In *EMNLP*.
- 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 *AKBC*.
- Fabio Petroni, Aleksandra Piktus, Angela Fan, Patrick Lewis, Majid Yazdani, Nicola De Cao, James Thorne, Yacine Jernite, Vladimir Karpukhin, Jean Maillard, Vassilis Plachouras, Tim Rocktäschel, and Sebastian Riedel. 2021. [KILT: a Benchmark for Knowledge Intensive Language Tasks](#). In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 2523–2544, Online. Association for Computational Linguistics.
- Fabio Petroni, Tim Rocktäschel, Sebastian Riedel, Patrick Lewis, Anton Bakhtin, Yuxiang Wu, and Alexander Miller. 2019. [Language Models as Knowledge Bases?](#) In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 2463–2473, Hong Kong, China. Association for Computational Linguistics.

- Pouya Pezeshkpour and Estevam Hruschka. 2024. [Large Language Models Sensitivity to The Order of Options in Multiple-Choice Questions](#). In *Findings of the Association for Computational Linguistics: NAACL 2024*, pages 2006–2017, Mexico City, Mexico. Association for Computational Linguistics.
- Nina Poerner, Ulli Waltinger, and Hinrich Schütze. 2020. [E-BERT: Efficient-Yet-Effective Entity Embeddings for BERT](#). In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 803–818, Online. Association for Computational Linguistics.
- Ian Porada, Alessandro Sordoni, and Jackie Cheung. 2022. [Does Pre-training Induce Systematic Inference? How Masked Language Models Acquire Commonsense Knowledge](#). In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 4550–4557, Seattle, United States. Association for Computational Linguistics.
- Ofir Press, Muru Zhang, Sewon Min, Ludwig Schmidt, Noah Smith, and Mike Lewis. 2023. [Measuring and Narrowing the Compositionality Gap in Language Models](#). In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 5687–5711, Singapore. Association for Computational Linguistics.
- Valentina Pyatkin, Saumya Malik, Victoria Graf, Hamish Ivison, Shengyi Huang, Pradeep Dasigi, Nathan Lambert, and Hannaneh Hajishirzi. 2025. [Generalizing Verifiable Instruction Following](#). In *The Thirty-ninth Annual Conference on Neural Information Processing Systems Datasets and Benchmarks Track*.
- Biqing Qi, Kaiyan Zhang, Haoxiang Li, Kai Tian, Sihang Zeng, Zhang-Ren Chen, and Bowen Zhou. 2023. [Large Language Models are Zero Shot Hypothesis Proposers](#). In *NeurIPS 2023 Workshop on Instruction Tuning and Instruction Following*.
- Jack W. Rae, Sebastian Borgeaud, Trevor Cai, Katie Millican, Jordan Hoffmann, Francis Song, John Aslanides, Sarah Henderson, Roman Ring, Susannah Young, Eliza Rutherford, Tom Hennigan, Jacob Menick, Albin Cassirer, Richard Powell, George van den Driessche, Lisa Anne Hendricks, Maribeth Rauh, Po-Sen Huang, and 61 others. 2022. [Scaling Language Models: Methods, Analysis Insights from Training Gopher](#). *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*, 21(140):1–67.
- Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. [SQuAD: 100,000+ Questions for Machine Comprehension of Text](#). In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 2383–2392, Austin, Texas. Association for Computational Linguistics.
- Arun Raman and Sean Lopp. 2025. [Deploying the NVIDIA AI Blueprint for Cost-Efficient LLM Routing](#). NVIDIA Technical Blog.
- Yasaman Razeghi, Hamish Ivison, Sameer Singh, and Yanai Elazar. 2023. [Backtracking Mathematical Reasoning of Language Models to the Pretraining Data](#). In *NeurIPS Workshop on Attributing Model Behavior at Scale*.
- Yasaman Razeghi, Robert L Logan IV, Matt Gardner, and Sameer Singh. 2022. [Impact of Pretraining Term Frequencies on Few-Shot Numerical Reasoning](#). In *Findings of the Association for Computational Linguistics: EMNLP 2022*, pages 840–854, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

- Nils Reimers and Iryna Gurevych. 2019. Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks. In *EMNLP*.
- David Rein, Betty Li Hou, Asa Cooper Stickland, Jackson Petty, Richard Yuanzhe Pang, Julien Dirani, Julian Michael, and Samuel R. Bowman. 2024. [GPQA: A Graduate-Level Google-Proof Q&A Benchmark](#). In *First Conference on Language Modeling*.
- Adam Roberts, Colin Raffel, and Noam Shazeer. 2020. [How Much Knowledge Can You Pack Into the Parameters of a Language Model?](#) In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 5418–5426, Online. Association for Computational Linguistics.
- Joshua Robinson and David Wingate. 2023. [Leveraging Large Language Models for Multiple Choice Question Answering](#). In *The Eleventh International Conference on Learning Representations*.
- Rachel Rudinger, Jason Naradowsky, Brian Leonard, and Benjamin Van Durme. 2018. [Gender Bias in Coreference Resolution](#). In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*, New Orleans, Louisiana. Association for Computational Linguistics.
- Camilo Ruiz, Marinka Zitnik, and Jure Leskovec. 2021. Identification of disease treatment mechanisms through the multiscale interactome. *Nature communications*.
- Gilbert Ryle. 1945. [Knowing How and Knowing That: The Presidential Address](#). *Proceedings of the Aristotelian Society*, 46:1–16.
- Tara Safavi, Doug Downey, and Tom Hope. 2022. [CascadER: Cross-Modal Cascading for Knowledge Graph Link Prediction](#). In *4th Conference on Automated Knowledge Base Construction*.
- Dylan Sam, Marc Anton Finzi, and J Zico Kolter. 2025. [Predicting the Performance of Black-box Language Models with Follow-up Queries](#). In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*.
- Victor Sanh, Albert Webson, Colin Raffel, Stephen Bach, Lintang Sutawika, Zaid Alyafeai, Antoine Chaffin, Arnaud Stiegler, Arun Raja, Manan Dey, M Saiful Bari, Canwen Xu, Urmish Thakker, Shanya Sharma Sharma, Eliza Szczechla, Taewoon Kim, Gunjan Chhablani, Nihal Nayak, Debajyoti Datta, and 21 others. 2022. [Multitask Prompted Training Enables Zero-Shot Task Generalization](#). In *International Conference on Learning Representations*.
- Michael Schlichtkrull, Thomas Kipf, Peter Bloem, Rianne van den Berg, Ivan Titov, and Max Welling. 2018. Modeling Relational Data with Graph Convolutional Networks. In *ESWC*.
- Melanie Sclar, Yejin Choi, Yulia Tsvetkov, and Alane Suhr. 2024. [Quantifying Language Models’ Sensitivity to Spurious Features in Prompt Design or: How I learned to start worrying about prompt formatting](#). In *The Twelfth International Conference on Learning Representations*.
- Preethi Seshadri, Sameer Singh, and Yanai Elazar. 2024. The Bias Amplification Paradox in Text-to-Image Generation. In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pages 6367–6384.
- Rulin Shao, Shuyue Stella Li, Rui Xin, Scott Geng, Yiping Wang, Sewoong Oh, Simon Shaolei Du, Nathan Lambert, Sewon Min, Ranjay Krishna, Yulia Tsvetkov, Hannaneh Hajishirzi, Pang Wei Koh, and Luke Zettlemoyer. 2025. [Spurious Rewards: Rethinking Training Signals in RLVR](#). *Preprint*, arXiv:2506.10947.

- Noam Shazeer, Azalia Mirhoseini, Krzysztof Maziarczyk, Andy Davis, Quoc Le, Geoffrey Hinton, and Jeff Dean. 2017. [Outrageously Large Neural Networks: The Sparsely-Gated Mixture-of-Experts Layer](#). In *International Conference on Learning Representations*.
- Tal Shnitzer, Anthony Ou, Mírian Silva, Kate Soule, Yuekai Sun, Justin Solomon, Neil Thompson, and Mikhail Yurochkin. 2024. [Large Language Model Routing with Benchmark Datasets](#). In *First Conference on Language Modeling*.
- Chenglei Si, Zhe Gan, Zhengyuan Yang, Shuohang Wang, Jianfeng Wang, Jordan Lee Boyd-Graber, and Lijuan Wang. 2023. [Prompting GPT-3 To Be Reliable](#). In *International Conference on Learning Representations*.
- Chenglei Si, Diyi Yang, and Tatsunori Hashimoto. 2025. [Can LLMs Generate Novel Research Ideas? A Large-Scale Human Study with 100+ NLP Researchers](#). In *The Thirteenth International Conference on Learning Representations*.
- Aaditya Singh, Adam Fry, Adam Perelman, Adam Tart, Adi Ganesh, Ahmed El-Kishky, Aidan McLaughlin, Aiden Low, AJ Ostrow, Akhila Ananthram, Akshay Nathan, Alan Luo, Alec Helyar, Aleksander Madry, Aleksandr Efremov, Aleksandra Spyra, Alex Baker-Whitcomb, Alex Beutel, Alex Karpenko, and 465 others. 2025. [OpenAI GPT-5 System Card](#). *Preprint*, arXiv:2601.03267.
- Luca Soldaini, Rodney Kinney, Akshita Bhagia, Dustin Schwenk, David Atkinson, Russell Authur, Ben Bogin, Khyathi Chandu, Jennifer Dumas, Yanai Elazar, Valentin Hofmann, Ananya Jha, Sachin Kumar, Li Lucy, Xinxu Lyu, Nathan Lambert, Ian Magnusson, Jacob Morrison, Niklas Muennighoff, and 17 others. 2024. [Dolma: an Open Corpus of Three Trillion Tokens for Language Model Pretraining Research](#). In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 15725–15788, Bangkok, Thailand. Association for Computational Linguistics.
- Nisan Stiennon, Long Ouyang, Jeffrey Wu, Daniel Ziegler, Ryan Lowe, Chelsea Voss, Alec Radford, Dario Amodei, and Paul F Christiano. 2020. [Learning to summarize from human feedback](#). In *Advances in Neural Information Processing Systems*, volume 33, pages 3008–3021. Curran Associates, Inc.
- Kai Sun, Yifan Xu, Hanwen Zha, Yue Liu, and Xin Luna Dong. 2024. [Head-to-Tail: How Knowledgeable are Large Language Models \(LLMs\)? A.K.A. Will LLMs Replace Knowledge Graphs?](#) In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pages 311–325, Mexico City, Mexico. Association for Computational Linguistics.
- Zhiqing Sun, Zhihong Deng, Jian-Yun Nie, and Jian Tang. 2019. RotatE: Knowledge Graph Embedding by Relational Rotation in Complex Space. In *ICLR*.
- Mujeen Sung, Jinhyuk Lee, Sean Yi, Minji Jeon, Sungdong Kim, and Jaewoo Kang. 2021. [Can Language Models be Biomedical Knowledge Bases?](#) In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 4723–4734, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.
- Don R. Swanson. 1986. Fish oil, Raynaud’s syndrome, and undiscovered public knowledge. *Perspectives in biology and medicine*, 30(1):7–18.
- Rohan Taori, Ishaan Gulrajani, Tianyi Zhang, Yann Dubois, Xuechen Li, Carlos Guestrin, Percy Liang, and Tatsunori B. Hashimoto. 2023. [Stanford Alpaca: An Instruction-following LLaMA model](#). https://github.com/tatsu-lab/stanford_alpaca.
- Timon ten Berge and Rene van Hezewijk. 1999. [Procedural and Declarative Knowledge: An Evolutionary Perspective](#). *Theory & Psychology*, 9(5):605–624.

- James Thorne, Andreas Vlachos, Christos Christodoulopoulos, and Arpit Mittal. 2018. [FEVER: a Large-scale Dataset for Fact Extraction and VERification](#). In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 809–819, New Orleans, Louisiana. Association for Computational Linguistics.
- Katherine Tian, Eric Mitchell, Allan Zhou, Archit Sharma, Rafael Rafailov, Huaxiu Yao, Chelsea Finn, and Christopher Manning. 2023. [Just Ask for Calibration: Strategies for Eliciting Calibrated Confidence Scores from Language Models Fine-Tuned with Human Feedback](#). In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 5433–5442.
- Kristina Toutanova, Danqi Chen, Patrick Pantel, Hoifung Poon, Pallavi Choudhury, and Michael Gamon. 2015. Representing Text for Joint Embedding of Text and Knowledge Bases. In *EMNLP*.
- 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*, abs/2302.13971.
- Théo Trouillon, Johannes Welbl, S. Riedel, Éric Gaussier, and Guillaume Bouchard. 2016. Complex Embeddings for Simple Link Prediction. In *ICML*.
- Oskar Van Der Wal, Jaap Jumelet, Katrin Schulz, and Willem Zuidema. 2022. [The Birth of Bias: A case study on the evolution of gender bias in an English language model](#). In *Proceedings of the 4th Workshop on Gender Bias in Natural Language Processing (GeBNLP)*, pages 75–75, Seattle, Washington. Association for Computational Linguistics.
- Shikhar Vashishth, Soumya Sanyal, Vikram Nitin, and Partha Talukdar. 2020. Composition-based Multi-Relational Graph Convolutional Networks. In *ICLR*.
- Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. [Attention is All you Need](#). In *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc.
- Sahil Verma, Royi Rassin, Arnav Mohanty Das, Gantavya Bhatt, Preethi Seshadri, Chirag Shah, Jeff Bilmes, Hannaneh Hajishirzi, and Yanai Elazar. 2025. [How Many Images Does It Take? Estimating Imitation Thresholds in Text-to-Image Models](#). *Transactions on Machine Learning Research*.
- Tu Vu, Tong Wang, Tsendsuren Munkhdalai, Alessandro Sordani, Adam Trischler, Andrew Mattarella-Micke, Subhransu Maji, and Mohit Iyyer. 2020. [Exploring and Predicting Transferability across NLP Tasks](#). In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 7882–7926, Online. Association for Computational Linguistics.
- Evan Pete Walsh, Luca Soldaini, Dirk Groeneveld, Kyle Lo, Shane Arora, Akshita Bhagia, Yuling Gu, Shengyi Huang, Matt Jordan, Nathan Lambert, Dustin Schwenk, Oyvind Tafjord, Taira Anderson, David Atkinson, Faeze Brahman, Christopher Clark, Pradeep Dasigi, Nouha Dziri, Allyson Ettinger, and 23 others. 2025. [2 OLMo 2 Furious \(COLM’s Version\)](#). In *Second Conference on Language Modeling*.
- Bo Wang, Tao Shen, Guodong Long, Tianyi Zhou, Ying Wang, and Yi Chang. 2021. Structure-Augmented Text Representation Learning for Efficient Knowledge Graph Completion. In *WWW*.
- Peiyi Wang, Lei Li, Liang Chen, Zefan Cai, Dawei Zhu, Binghuai Lin, Yunbo Cao, Lingpeng Kong, Qi Liu, Tianyu Liu, and Zhifang Sui. 2024. [Large Language Models are not Fair Evaluators](#). In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 9440–9450, Bangkok, Thailand. Association for Computational Linguistics.

- Xinyi Wang, Antonis Antoniadis, Yanai Elazar, Alfonso Amayuelas, Alon Albalak, Kexun Zhang, and William Yang Wang. 2025a. Generalization vs Memorization: Tracing Language Models’ Capabilities Back to Pretraining Data. In *The Thirteenth International Conference on Learning Representations*.
- Yike Wang, Shangbin Feng, Yulia Tsvetkov, and Hannaneh Hajishirzi. 2025b. ScienceMeter: Tracking Scientific Knowledge Updates in Language Models. *arXiv preprint arXiv:2505.24302*.
- Yizhong Wang, Hamish Ivison, Pradeep Dasigi, Jack Hessel, Tushar Khot, Khyathi Chandu, David Wadden, Kelsey MacMillan, Noah A. Smith, Iz Beltagy, and Hannaneh Hajishirzi. 2023a. [How Far Can Camels Go? Exploring the State of Instruction Tuning on Open Resources](#). In *Conference on Neural Information Processing Systems*.
- Yizhong Wang, Yeganeh Kordi, Swaroop Mishra, Alisa Liu, Noah A. Smith, Daniel Khashabi, and Hannaneh Hajishirzi. 2023b. [Self-Instruct: Aligning Language Models with Self-Generated Instructions](#). In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*.
- Yizhong Wang, Swaroop Mishra, Pegah Alipoormolabashi, Yeganeh Kordi, Amirreza Mirzaei, Anjana Arunkumar, Arjun Ashok, Arut Selvan Dhanasekaran, Atharva Naik, David Stap, Eshaan Pathak, Giannis Karamanolakis, Haizhi Gary Lai, Ishan Purohit, Ishani Mondal, Jacob Anderson, Kirby Kuznia, Krma Doshi, Maitreya Patel, and 21 others. 2022. [Super-NaturalInstructions: Generalization via Declarative Instructions on 1600+ NLP Tasks](#). In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*.
- Zhigang Wang and Juan-Zi Li. 2016. Text-Enhanced Representation Learning for Knowledge Graph. In *IJCAI*.
- Albert Webson and Ellie Pavlick. 2022. [Do Prompt-Based Models Really Understand the Meaning of Their Prompts?](#) In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 2300–2344, Seattle, United States. Association for Computational Linguistics.
- Jason Wei, Maarten Bosma, Vincent Zhao, Kelvin Guu, Adams Wei Yu, Brian Lester, Nan Du, Andrew M. Dai, and Quoc V Le. 2022. [Finetuned Language Models are Zero-Shot Learners](#). In *International Conference on Learning Representations*.
- Jason Wei, Dan Garrette, Tal Linzen, and Ellie Pavlick. 2021. [Frequency Effects on Syntactic Rule Learning in Transformers](#). In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 932–948, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.
- 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*, pages 94–106, Copenhagen, Denmark. Association for Computational Linguistics.
- Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Remi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, and 3 others. 2020a. [Transformers: State-of-the-Art Natural Language Processing](#). In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 38–45, Online. Association for Computational Linguistics.
- Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Remi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, and 3 others. 2020b.

- Transformers: State-of-the-Art Natural Language Processing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*.
- Mengzhou Xia, Antonios Anastasopoulos, Ruochen Xu, Yiming Yang, and Graham Neubig. 2020. [Predicting Performance for Natural Language Processing Tasks](#). In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*.
- Mengzhou Xia, Sadhika Malladi, Suchin Gururangan, Sanjeev Arora, and Danqi Chen. 2024. [LESS: Selecting Influential Data for Targeted Instruction Tuning](#). In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pages 54104–54132. PMLR.
- Ruobing Xie, Zhiyuan Liu, Jia Jia, Huanbo Luan, and Maosong Sun. 2016. Representation Learning of Knowledge Graphs with Entity Descriptions. In *AAAI*.
- Can Xu, Qingfeng Sun, Kai Zheng, Xiubo Geng, Pu Zhao, Jiazhan Feng, Chongyang Tao, Qingwei Lin, and Daxin Jiang. 2024. [WizardLM: Empowering Large Pre-Trained Language Models to Follow Complex Instructions](#). In *The Twelfth International Conference on Learning Representations*.
- Bishan Yang, Wen tau Yih, Xiadong He, Jianfeng Gao, and Li Deng. 2015. Embedding Entities and Relations for Learning and Inference in Knowledge Bases. In *ICLR*.
- Liang Yao, Chengsheng Mao, and Yuan Luo. 2019. KG-BERT: BERT for Knowledge Graph Completion. *arXiv:1909.03193*.
- Yuekun Yao and Alexander Koller. 2024. [Predicting generalization performance with correctness discriminators](#). In *Findings of the Association for Computational Linguistics: EMNLP 2024*, pages 11725–11739, Miami, Florida, USA. Association for Computational Linguistics.
- Gregory Yauney, Emily Reif, and David Mimno. 2023. [Data Similarity is Not Enough to Explain Language Model Performance](#). In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 11295–11304, Singapore. Association for Computational Linguistics.
- Qinyuan Ye, Harvey Fu, Xiang Ren, and Robin Jia. 2023. [How Predictable Are Large Language Model Capabilities? A Case Study on BIG-bench](#). In *Findings of the Association for Computational Linguistics: EMNLP 2023*.
- Zihuiwen Ye, Pengfei Liu, Jinlan Fu, and Graham Neubig. 2021. [Towards More Fine-grained and Reliable NLP Performance Prediction](#). In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume*.
- Fan Yin, Jesse Vig, Philippe Laban, Shafiq Joty, Caiming Xiong, and Chien-Sheng Wu. 2023. [Did You Read the Instructions? Rethinking the Effectiveness of Task Definitions in Instruction Learning](#). In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 3063–3079, Toronto, Canada. Association for Computational Linguistics.
- Yang Yue, Zhiqi Chen, Rui Lu, Andrew Zhao, Zhaokai Wang, Yang Yue, Shiji Song, and Gao Huang. 2026. [Does Reinforcement Learning Really Incentivize Reasoning Capacity in LLMs Beyond the Base Model?](#) In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*.
- Yury Zemlyanskiy, Sudeep Gandhe, Ruining He, Bhargav Kanagal, Anirudh Ravula, Juraj Gottweis, Fei Sha, and Ilya Eckstein. 2021. DOCENT: Learning Self-Supervised Entity Representations from Large Document Collections. In *EACL*.

- Zhiyuan Zeng, Yizhong Wang, Hannaneh Hajishirzi, and Pang Wei Koh. 2025. [EvalTree: Profiling Language Model Weaknesses via Hierarchical Capability Trees](#). In *Second Conference on Language Modeling*.
- Xiao Zhang, Miao Li, and Ji Wu. 2024. [Co-occurrence is not Factual Association in Language Models](#). In *Advances in Neural Information Processing Systems*, volume 37, pages 64889–64914. Curran Associates, Inc.
- Yanbo Zhang, Sumeer A. Khan, Adnan Mahmud, Huck Yang, Alexander Lavin, Michael Levin, Jeremy Frey, Jared Dunnmon, James Evans, Alan Bundy, Saso Dzeroski, Jesper Tegner, and Hector Zenil. 2025. [Exploring the role of large language models in the scientific method: from hypothesis to discovery](#). *npj Artificial Intelligence*, 1(14).
- Jieyu Zhao, Tianlu Wang, Mark Yatskar, Vicente Ordonez, and Kai-Wei Chang. 2018. [Gender Bias in Coreference Resolution: Evaluation and Debiasing Methods](#). In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*, New Orleans, Louisiana. Association for Computational Linguistics.
- Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric Xing, Hao Zhang, Joseph E. Gonzalez, and Ion Stoica. 2023. [Judging LLM-as-a-Judge with MT-Bench and Chatbot Arena](#). In *Thirty-seventh Conference on Neural Information Processing Systems Datasets and Benchmarks Track*.
- Chunting Zhou, Pengfei Liu, Puxin Xu, Srini Iyer, Jiao Sun, Yuning Mao, Xuezhe Ma, Avia Efrat, Ping Yu, LILI YU, Susan Zhang, Gargi Ghosh, Mike Lewis, Luke Zettlemoyer, and Omer Levy. 2023a. [LIMA: Less Is More for Alignment](#). In *Thirty-seventh Conference on Neural Information Processing Systems*.
- Jeffrey Zhou, Tianjian Lu, Swaroop Mishra, Siddhartha Brahma, Sujoy Basu, Yi Luan, Denny Zhou, and Le Hou. 2023b. [Instruction-Following Evaluation for Large Language Models](#). *Preprint*, arXiv:2311.07911.
- Kaitlyn Zhou, Dan Jurafsky, and Tatsunori Hashimoto. 2023c. [Navigating the Grey Area: How Expressions of Uncertainty and Overconfidence Affect Language Models](#). In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 5506–5524.
- Marinka Zitnik, Monica Agrawal, and Jure Leskovec. 2018. Modeling polypharmacy side effects with graph convolutional networks. *Bioinformatics*, 34(13):i457–i466.
- Nicolas Zucchet, Jorg Bornschein, Stephanie C.Y. Chan, Andrew Kyle Lampinen, Razvan Pascanu, and Soham De. 2025. [How do language models learn facts? Dynamics, curricula and hallucinations](#). In *Second Conference on Language Modeling*.

Appendix A

Appendices for Chapter 2

A.1 Dataset Construction

A.1.1 Sources

RepoDB Drugs in RepoDB have statuses including *approved*, *terminated*, *withdrawn*, and *suspended*. We restrict our KG to pairs in the *approved* category.

Hetionet was constructed using data from various publicly-available scientific repositories. Following [Alshahrani et al. \(2021\)](#), we restrict the KG to the *treats*, *presents*, *associates*, and *causes* relation types. This includes interactions between drugs and the diseases they treat, diseases and their symptoms, diseases and associated genes, and drugs and their side effects. We use this subset of the full Hetionet dataset to avoid scalability issues that arise when training large Transformer-based language models, inspired by benchmark datasets such as FB15K ([Bordes et al., 2013](#)), a subset of the Freebase knowledge base.

MSI includes diseases and the proteins they perturb, drug targets, and biological functions designed to discover drug-disease treatment pairs through the pathways that connect them via genes, proteins, and their functions. We include all entities and relation types in the dataset.

We collect each of the datasets from the links listed in Table [A.1](#). For missing entity names and all descriptions, we write scripts to scrape the information from the resources listed above using the entity identifiers provided by each of the datasets.

A.1.2 Transductive Splits

To construct transductive splits for each dataset, we begin with the complete graph, and repeat the following steps:

1. Randomly sample an edge from the graph.

Dataset	Link	Sources
RepoDB	http://apps.chiragjpgroup.org/repoDB/	DrugBank UMLS
Hetionet	https://github.com/hetio/hetionet	DrugBank Disease Ontology Entrez SIDER MeSH
MSI	https://github.com/snap-stanford/multiscale-interactome	DrugBank Gene Ontology Entrez UMLS

Table A.1: Links and sources of entity names and descriptions for each dataset.

2. If the degree of both nodes incident to the edge is greater than one, remove the edge.
3. Otherwise, replace the edge and continue.

The above steps are repeated until validation and test graphs have been constructed of the desired size while ensuring that no entities are removed from the training graph. We construct 80%/10%/10% training/validation/test splits of all datasets.

A.1.3 Inductive Splits

To construct inductive splits for each dataset, we follow the procedure outlined in the “Technical Details” section of the appendix of [Daza et al. \(2021\)](#). We similarly construct a 80%/10%/10% training/validation/test split of each dataset in the inductive setting.

A.1.4 Negative Validation/Test Triples

In order to perform a ranking-based evaluation for each dataset in both the transductive and inductive settings, we generate a set of negative triples to be ranked against each positive triple. To generate negative entities to replace both the head and tail entity of each validation and test positive, we follow the procedure below:

1. Begin with the set of all entities in the knowledge graph.
2. Remove all entities that do not have the same entity type as the entity to be ranked against in the positive triple.

3. Remove all entities that would result in a valid positive triple in either the training, validation, or test sets.
4. Randomly sample a fixed set of size m from the remaining set of entities.

We use a value of $m = 500$ for RepoDB and MSI, and a value of $m = 80$ for Hetionet (due to the constraints above, the minimum number of valid entities remaining across positive triples for Hetionet was 80). Using a fixed set of entities allows for fair comparison when assessing performance of subsets of the test set, such as when examining the effect of subsets where descriptions are present for neither, one, or both entities (Table A.3).

A.2 Training

A.2.1 Transductive Setting

For all individual models, we train the models on the training set of each dataset while periodically evaluating on the validation set. We save the model with the best validation set MRR, then use that model to evaluate on the test set. We also perform hyperparameter tuning for all models, and use validation set MRR to select the final set of hyperparameters for each model.

Knowledge Graph Embeddings

We use the max-margin ranking loss for all KGE methods. We use a batch size of 512 for all models. We train models for 10,000 steps (958 epochs) on RepoDB, 50,000 steps (205 epochs) on Hetionet, and 50,000 steps (66 epochs) on MSI. We evaluate on the validation set every 500 steps for RepoDB and 5,000 steps for Hetionet and MSI. We use the Adam optimizer for training. We perform a hyperparameter search over the following values:

- Embedding dimension: 500, 1000, 2000
- Margin for max-margin loss: 0.1, 1
- Learning rate: $1e-3$, $1e-4$
- Number of negative samples per positive: 128, 256
- Parameter for L3 regularization of embeddings: $1e-5$, $1e-6$

Language Models

Pretrained scientific LMs. We explore various pretrained LMs, with their initialization, vocabulary, and pretraining corpora described below. In particular, we study a range of LMs trained on different scientific and biomedical literature, and also on clinical notes.

- **BioBERT** (Lee et al., 2020) Initialized from BERT and using the same general domain vocabulary, with additional pretraining on the PubMed repository of scientific abstracts and full-text articles.
- **Bio+ClinicalBERT** (Alsentzer et al., 2019) Initialized from BioBERT with additional pretraining on the MIMIC-III corpus of clinical notes.
- **SciBERT** (Beltagy et al., 2019) Pretrained from scratch with a domain-specific vocabulary on a sample of the Semantic Scholar corpus, of which biomedical papers are a significant fraction but also papers from other scientific domains.
- **PubMedBERT** (Gu et al., 2021) Pretrained from scratch with a domain-specific vocabulary on PubMed. We apply two versions of PubMedBERT, one trained on PubMed abstracts alone (PubMedBERT-abstract) and the other on abstracts as well as full-text articles (PubMedBERT-fulltext).

We also use **RoBERTa** (Liu et al., 2019) – pretrained from scratch on the BookCorpus, English Wikipedia, CC-News, OpenWebText, and Stories datasets – as a strongly-performing general domain model for comparison. For all LMs, we follow Kim et al. (2020) and use the multi-task loss consisting of binary triple classification, multi-class relation classification, and max-margin ranking loss, with a margin of 1 for the max-margin loss. For triple classification, given the correct label $y \in \{0, 1\}$ (positive or negative triple) we apply a linear layer to the [CLS] token representation v to output the probability p of the triple being correct as $p = \sigma(W_{\text{triple}}v)$, and use the binary cross entropy loss $\mathcal{L}_{\text{triple}}(x) = -y \log(p) - (1 - y) \log(1 - p)$. For relation classification over R relation types, we apply a linear layer to v to calculate a probability distribution q over relation classes with $q = \text{softmax}(W_{\text{rel}}v)$, and use the cross entropy loss with one-hot vector $y \in \{0, 1\}^R$ as the correct relation label: $\mathcal{L}_{\text{rel}}(x) = -\sum_{i=1}^R y_i \log q_i$. The final loss is the equally-weighted sum of all three losses: $\mathcal{L}(x) = \mathcal{L}_{\text{rank}}(x) + \mathcal{L}_{\text{triple}}(x) + \mathcal{L}_{\text{rel}}(x)$.

We train for 40 epochs on RepoDB, and 10 epochs on Hetionet and MSI. We evaluate on the validation set every epoch for RepoDB, and three times per epoch for Hetionet and MSI. For RepoDB, Hetionet, and MSI we use 32, 16, and 8 negative samples per positive, respectively. We use the Adam optimizer for training. We perform a hyperparameter search over the following values:

- Batch size: 16, 32
- Learning rate: 1e-5, 3e-5, 5e-5

Integrated Models

Global weighted average. For the global weighted average, we compute ranking scores for positive and negative examples as the weighted average of ranking scores output by all KG completion models being integrated. Specifically, for a set of ranking scores $s_1, \dots, s_k$ output by k models for an example, we learn a set of weights $\alpha = [\alpha_1, \dots, \alpha_k]$ to compute the final ranking score as $s = \sum_{i=1}^k \alpha_i s_i$, where the same weight vector α is used for all examples. We search for each α_i over the grid $[0.05, 0.95]$ with steps of 0.05, ensuring that all α_i 's sum to 1. We choose values that maximize validation set MRR, then apply them to the test set.

Router. For the router-based method, we train a classifier to select a single model out of a set of KG completion models to use for computing ranking scores for a positive example and its associated negatives. The class to be predicted for a particular example corresponds to which model performs best on that example (i.e., gives the best rank), with an additional class for examples where all models perform the same. We explore a number of different classifiers, including logistic regression, decision tree, gradient boosted decision tree (GBDT), and multilayer perceptron (MLP), finding that GBDT and MLP classifiers perform the best. As input to the classifier, we use a diverse set of features computed from each positive example (listed in Table A.2) as well as each model's ranking score for the positive example. Classifiers are trained on the validation set and evaluated on the test set for each dataset. We additionally perform hyperparameter tuning over the following values for each classifier:

Logistic regression:

- Penalty: L1, L2
- Regularization parameter: 9 values evenly log-spaced between 1e-5 and 1e3

Decision tree:

- Max depth: 2, 4, 8
- Learning rate: 1e-1, 1e-2, 1e-3

GBDT:

- Number of boosting rounds: 100, 500, 1000
- Max depth: 2, 4, 8
- Learning rate: 1e-1, 1e-2, 1e-3

MLP:

- Number of hidden layers: 1, 2
- Hidden layer size: 128, 256
- Batch size: 64, 128, 256
- Learning rate: 1e-1, 1e-2, 1e-3

We perform five-fold cross-validation on the validation set and use validation set accuracy to choose the best set of hyperparameters for each classifier. We use Scikit-Learn (Pedregosa et al., 2011) to implement the logistic regression and MLP classifiers, and XGBoost (Chen and Guestrin, 2016) to implement the decision tree and GBDT classifiers, using default parameters other than the ones listed above.

Input-dependent weighted average. The input-dependent weighted average method of integrating KG completion models operates similarly to the global weighted average, except that the set of weights can vary for each positive example and are a function of its feature vector (the same set of weights is used for all negative examples used to rank against each positive example). We train an MLP to output a set of weights that are then used to compute a weighted average of ranking scores for a set of KG completion models. The MLP is trained on the validation set and evaluated on the test set for each dataset. We use the max-margin ranking loss with a margin of 1. In order to compare to the MLP trained as a router, we train the MLP using the Adam optimizer (Kingma and Ba, 2015) for 200 epochs with early stopping on the training loss and a patience of 10 epochs (the default settings for an MLP classifier in Scikit-Learn). We perform a hyperparameter search over the following values (matching the values for the MLP router where applicable):

- Number of hidden layers: 1, 2
- Hidden layer size: 128, 256
- Batch size: 64, 128, 256
- Learning rate: 1e-1, 1e-2, 1e-4
- Number of negatives (for max-margin loss): 16, 32

We select the best hyperparameters by MRR on a held-out portion of the validation set.

Features for integrated models. Both the router and input-dependent weighted average methods of model integration use a function to outputs weights based on a feature vector of an example. A complete list of the features used by each method can be found in Table A.2. We also use the ranking score for the positive example from each KG completion model being integrated as additional features.

entity type	length of text in chars.
relation type	presence of word “unknown” in name/desc.
head/tail node in-/out-degree	missing desc.
head/tail node PageRank	number/ratio of punctuation/numeric chars.
Adamic-Adar index of edge	tokens-to-words ratio of entity name/desc.
edit dist. between head/tail entity names	

Table A.2: Complete list of features used by router classifiers.

A.2.2 Inductive Setting

Knowledge Graph Embeddings and Language Models

For the KGE and LM models, we follow the same training procedure for the inductive splits as for the transductive splits. We perform hyperparameter tuning over the same grids of hyperparameters, periodically evaluate on the validation set and save the checkpoint with the best validation set MRR, and use the set of hyperparameters corresponding to the highest validation set MRR to evaluate on the test set.

DKRL

In addition to the KGE and LM-based methods, we also train DKRL (Xie et al., 2016) for inductive KG completion as another text-based baseline for comparison. DKRL uses a two-layer CNN encoder applied to the word or subword embeddings of an entity’s textual description to construct a fixed-length entity embedding. To score a triple, DKRL combines its entity embeddings constructed from text with a separately-learned relation embedding using the TransE (Bordes et al., 2013) scoring function. The original DKRL model uses a joint scoring function with structure-based and description-based components; we restrict to the description-based component as we are applying DKRL in the inductive setting. We use PubMedBERT subword embeddings at the input layer of the CNN encoder, encode entity names and descriptions, and apply the same multi-task loss as for the LM-based models. To apply the triple classification and relation classification losses, for head and tail entity embeddings $\mathbf{h}$ and $\mathbf{t}$, we apply a separate linear layer for each loss to the concatenated vector $[\mathbf{h}; \mathbf{t}; |\mathbf{h} - \mathbf{t}|]$, following previous work on models that use a bi-encoder to construct entity or sentence representations (Wang et al., 2021; Reimers and Gurevych, 2019). We use the same number of training epochs and number of negatives per positive for DKRL as for the LM-based methods on each dataset. We use a batch size of 64, and perform a hyperparameter search over the following values:

- Learning rate: 1e-3, 1e-4, 1e-5

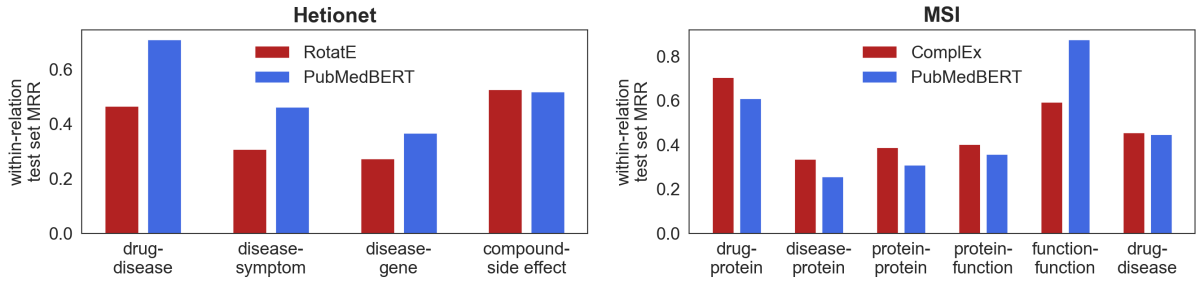


Figure A.1: Test set MRR for the best KGE model compared to KG-PubMedBERT broken down by relation type for Hetionet and MSI.

- Embedding dimension: 500, 1000, 2000
- Parameter for L2 regularization of embeddings: 0, 1e-3, 1e-2

A.3 Additional Results

A.3.1 Transductive Setting, Individual Models

Missing entity descriptions. Table A.3 shows test set MRR for KG-PubMedBERT on each dataset broken down by triples with either both, one, or neither entities having available descriptions. Across datasets, performance clearly degrades when fewer descriptions are available to provide context for the LM to generate a ranking score.

#entities with desc. in pair	MRR		
	RepoDB	Hetionet	MSI
None	N/A	25.6	25.1
One	59.5	43.6	25.4
Both	63.7	52.6	37.3

Table A.3: Effect of descriptions on KG-PubMedBERT test set MRR.

Relation-level performance. Figure A.1 shows test set MRR broken down by relation for the datasets with multiple relation types (Hetionet and MSI). KG-PubMedBERT performs better on all relation types except compound-side effect for Hetionet, and on the function-function relation for MSI.

	RepoDB	Hetionet	MSI
Global avg.	70.4	55.8	42.1
Input-dep. avg.	65.9	70.3	40.6
Router	70.6	59.7	48.5

Table A.4: Test set MRR for the best pair of a KGE model and KG-PubMedBERT for different methods of model integration.

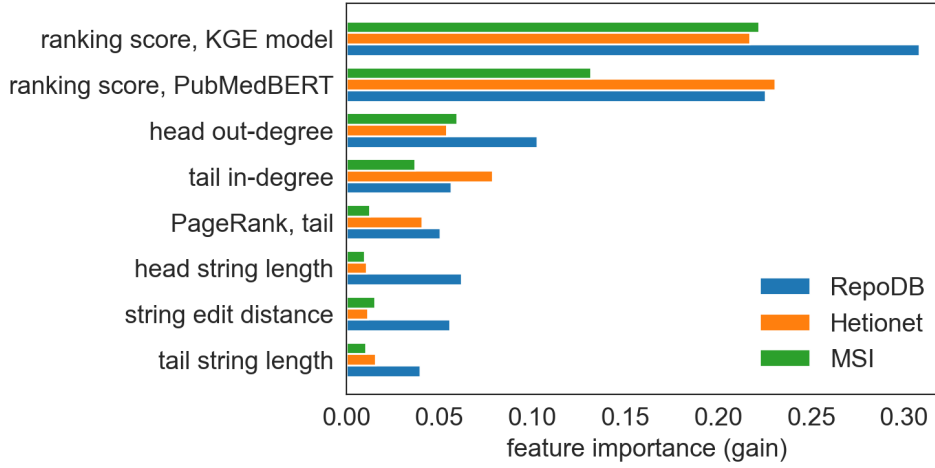


Figure A.2: Feature importances for GBDT router for a selection of most important features. Ranking scores output by each model tend to be the most important, with other graph- and text-based features also contributing.

A.3.2 Transductive Setting, Integrated Models

A.3.3 Inductive Setting

Imputation Model with Better Ranking	Unseen Entity	KG-PubMedBERT nearest neighbors	PubMedBERT nearest neighbors
PubMedBERT	eye redness	skin burning sensation, skin dis- comfort	conjunctivitis, throat sore
	ecchymosis	gas, thrombophlebitis	petechiae, macule
	estrone	vitamin a, methyltestosterone	estriol, calcitriol
KG-PubMedBERT	keratoconjunctivitis	conjunctivitis allergic, otitis externa	enteritis, parotitis
	malnutrition	dehydration, anaemia	meningism, wasting generalized
	congestive cardiomyopathy	diastolic dysfunction, cardiomyopathy	carcinoma breast, hypertrophic cardiomyopathy

Table A.5: Samples of unseen entities and their nearest neighbors found by KG-PubMedBERT and PubMedBERT, for test set examples in the Hetionet inductive split where the PubMedBERT neighbor performs better than the KG-PubMedBERT neighbor (first three) and vice versa (last three). Each LM offers a larger improvement per example when its nearest neighbor is more semantically related to the unseen entity.

Appendix B

Appendices for Chapter 3

B.1 Finetuning Details

We use the Huggingface transformers library to run all experiments (Wolf et al., 2020b). Details for finetuning RoBERTa-base and RoBERTa-large performance predictors can be found in Table B.1, while details for the LLaMA-13B performance predictors can be found in Table B.2. We perform a hyperparameter search over batch size and learning rate for all finetuning experiments. RoBERTa models are finetuned by updating all parameters, while LLaMA models are finetuned by adding and updating LoRA adapters (Hu et al., 2022). All performance predictors use an additional linear layer applied to either the [CLS] token (for RoBERTa) or the EOS token (for LLaMA) at the last layer to make the final prediction. All models were evaluated on the validation set of instructions at every epoch, with early stopping performed based on validation set RMSE.

Hyperparameter	Assignment
number of epochs	20
batch size	{4, 8, 16}
maximum learning rate	{1e-5, 5e-5, 1e-4, 5e-4}
optimizer	AdamW
epsilon	1e-8
betas	(0.9, 0.999)
learning rate schedule	constant
weight decay	0
warmup proportion	none
learning rate decay	none

Table B.1: Experiment settings for finetuning RoBERTa performance predictor models.

Hyperparameter	Assignment
number of epochs	20
batch size	{8, 16, 32}
maximum learning rate	{1e-5, 2e-5, 5e-5, 1e-4}
optimizer	AdamW
epsilon	1e-8
betas	(0.9, 0.999)
learning rate schedule	linear warmup
weight decay	0
warmup proportion	0.03
learning rate decay	linear
LoRA rank	256
LoRA alpha	256
LoRA dropout	0.05

Table B.2: Experiment settings for finetuning LLaMA performance predictor models.

B.2 Details on BIG-bench

The full list of tasks included from BIG-bench can be found in Table B.3.

Tasks

abstract_narrative_understanding, anachronisms, analogical_similarity, analytic_ entailment, arithmetic, ascii_word_recognition, authorship_verification, auto_categorization, auto_debugging, bbq_lite_json, bridging_anaphora_resolution_barqa, causal_judgment, cause_and_effect, checkmate_in_one, chess_state_tracking, chinese_remainder_theorem, cifar10_classification, code_line_description, codenames, color, common_morpheme, conceptual_combinations, conlang_translation, contextual_parametric_knowledge_conflicts, crash_blossom, crass_ai, cryobiology_spanish, cryptonite, cs_algorithms, dark_humor_detection, date_understanding, disambiguation_qa, discourse_marker_prediction, disfl_qa, dyck_languages, elementary_math_qa, emoji_movie, emojis_emotion_prediction, empirical_judgments, english_proverbs, english_russian_proverbs, entailed_polarity, entailed_polarity_hindi, epistemic_reasoning, evaluating_information_essentiality, fact_checker, fantasy_reasoning, few_shot_nlg, figure_of_speech_detection, formal_fallacies_syllogisms_negation, gem, gender_inclusive_sentences_german, general_knowledge, geometric_shapes, goal_step_wikihow, gre_reading_comprehension, hhh_alignment, hindi_question_answering, hindu_knowledge, hinglish_toxicity, human_organs_senses, hyperbaton, identify_math_theorems, identify_odd_metaphor, implicatures, implicit_relations, intent_recognition, international_phonetic_alphabet_nli, international_phonetic_alphabet_transliterate, intersect_geometry, irony_identification, kanji_ascii, kanada, key_value_maps, known_unknowns, language_games, language_identification, linguistics_puzzles, logic_grid_puzzle, logical_args, logical_fallacy_detection, logical_sequence, mathematical_induction, matrixshapes, metaphor_boolean, metaphor_understanding, minute_mysteries_qa, misconceptions, mnist_ascii, modified_arithmetic, moral_permissibility, movie_dialog_same_or_different, movie_recommendation, mult_data_wrangling, navigate, nonsense_words_grammar, novel_concepts, object_counting, odd_one_out, operators, paragraph_segmentation, parsinlu_qa, parsinlu_reading_comprehension, penguins_in_a_table, periodic_elements, persian_idioms, phrase_relatedness, physical_intuition, physics, physics_questions, play_dialog_same_or_different, polish_sequence_labeling, presuppositions_as_nli, qa_wikidata, question_selection, real_or_fake_text, reasoning_about_colored_objects, repeat_copy_logic, rephrase, riddle_sense, ruin_names, salient_translation_error_detection, scientific_press_release, semantic_parsing_in_context_sparc, semantic_parsing_spider, sentence_ambiguity, similarities_abstraction, simp_turing_concept, simple_ethical_questions, simple_text_editing, snarks, social_iqa, social_support, sports_understanding, strange_stories, strategyqa, sufficient_information, suicide_risk, swahili_english_proverbs, swedish_to_german_proverbs, symbol_interpretation, temporal_sequences, tense, timedial, topical_chat, tracking_shuffled_objects, understanding_fables, undo_permutation, unit_conversion, unit_interpretation, vitaminc_fact_verification, what_is_the_tao, which_wiki_edit, winowhy, word_sorting, word_unscrambling

Table B.3: List of BIG-bench tasks included when training performance predictors on additional tasks.

Appendix C

Appendices for Chapter 4

C.1 Swapping Documents Between Batches

All of our interventions involve taking documents from another data batch and using them to replace documents in the data batch that we wish to intervene on. We replace documents while matching total token counts when possible to minimize altering the data batch. For over 96% of documents, the document that replaces it has exactly the same number of tokens. In a few cases, the document being swapped in is too long by a few tokens, which requires shifting the remaining tokens in the sequence to the right and truncating at the end. Fig. C.1 shows a toy example of how this document-swapping operation is performed with token IDs across two batches of tokens.

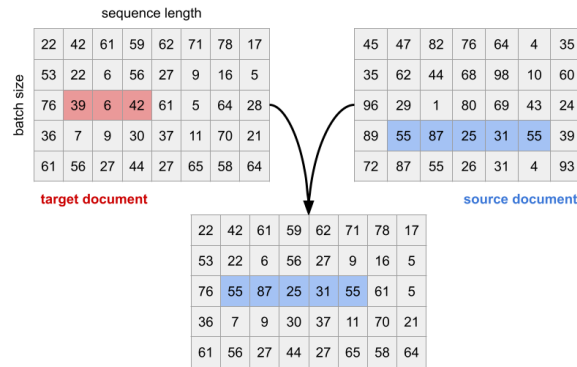


Figure C.1: Toy example that shows how token IDs in a batch of tokens are altered when replacing a target document with a source document. Note how the token IDs to the right of the target document are shifted to the right to accommodate the longer source document.

Prompt Name	Template
correct answer	You will be given a question or question stem and its correct answer, along with a document. Output “yes” if the document contains enough information to correctly answer the question, and “no” otherwise. Only output “yes” or “no”. Question: {question} Answer: {answer} Document: {document}
educated guess	You will be given a question or question stem and its correct answer, along with a document. Output “yes” if the document contains enough information to make an educated guess about the answer, and “no” otherwise. The document does not have to have enough information to be able to correctly answer the question. Only output “yes” or “no”. Question: {question} Answer: {answer} Document: {document}
related topic	You will be given a question or question stem and its correct answer, along with a document. Output “yes” if the document is topically related to the question and answer, and “no” otherwise. The document does not have to have enough information to be able to correctly answer the question. Only output “yes” or “no”. Question: {question} Answer: {answer} Document: {document}

Table C.1: Prompt templates used for document relevance annotation with GPT-5.

Item	Prompt	Document with “yes” annotation	Document with “no” annotation
Question: What do herpetologists study? Answer: Reptiles and amphibians	correct answer	What Classes Do I Need to Take in High School To Prepare for Getting a Zoology Degree? Zoologists study the impact of the environment on animal habitats... Some zoologists will specialize in a species, such as herpetologists, who study reptiles and snakes. Taking science courses while in high school will help you determine in which area of zoology you will specialize... This career may be physically demanding when spending long periods outdoors in rough conditions while researching.	Amphibians vs Reptiles Reptile? Or amphibian? Telling the difference. For most people, these animals are pretty much the same.... Definition Amphibian means “living double lives” on water and land. Reptile means “creeping stealthily under cover of darkness.”... Tetrafauna AquaSafe – The complete water conditioner.
Question: The use of ‘indentured labour’ in the nineteenth century involved: Answer: people being transported to the British colonies and forced to work for one employer under poor conditions	educated guess	History of Guyana Guyana history The recorded history of Guyana can be dated back to 1499, when Alonso de Ojeda’s first expedition arrived from Spain at the Essequibo River... Subsequently, in 1746, the Dutch authorities opened the area near the Demerara River to British immigrants because they were eager to attract more settlers... By 1786 the internal affairs of this Dutch colony were effectively under British control, though two-thirds of the plantation owners were still Dutch. Africans and Indians Africans were enslaved and transported to Guyana as slaves; however, East Indians came as indentured labourers who worked in order to provide for their families back home. By the 1660s, the enslaved	Banner Viadrina Keynote Lecture: Prof. Dr. Andreas Eckert Blurred Frontiers. “Unfree” and “Free” Labour since 1800 in a global perspective In the world of work relations and labour policies, the distinction between “free” and “unfree” labour has been crucial... This lecture critically discusses widespread assumptions which link “unfree” or forced labour to pre-industrial societies and instead argues that there was a close and complex connection between definitions and practices of unfree labor in Europe and in the colonies... The talk will present various historiographical debates related to the issue of unfree/free labour, but will also focus on what historical actors in different parts of the world themselves understood by unfree/free labour.
Question: When you squeeze an air-filled party balloon, you increase its density Answer: density	related topic	hydrogen is 0.09 kg/m ³ . Two balloons are inflated to the same size, one filled with air and the other filled with hydrogen. Which balloon experiences the greatest buoyant force? 6) A ten kilogram brick is lowered into a tub of water. A volume equal to two litres of water is displaced. If the density of water is 1g/cm ³ what is the reading of the scale when the brick is fully submerged? Does the brick lose mass when it is placed in water? Activity to measure the buoyant force. Put-Put boats and buoyancy	The Quartet Machine My instrument is called “The Quartet Machine.” It is a member of the string and woodwind families. It is made out of a wooden box, rope, string, beads, plastic bar, rubber bands, balloon, barrette, “squeakies,” and a pencil... The second is you can blow air into the balloon and pinch it near the opening and pull. The third is you can squeeze the “squeakies.” To change the pitch of this instrument you can do one of the following... Finally, to change the dynamics or volume of this instrument you can pluck the strings harder, squeeze the balloon harder or pinch the “squeakies” tighter.

Table C.2: A sample of items from MMLU with example documents matched by BM25 score that were given “yes” and “no” annotations by GPT-5, for each of the three annotation prompts.

Relation	# Items	Template	Example
P17	928	[X] is located in [Y]	Mezhdurechensky District is located in Russia
P27	966	[X] is a citizen of [Y]	Marc Lavoie is a citizen of Canada
P36	681	The capital city of [X] is [Y]	The capital city of Greece is Athens
P127	687	[X] is owned by [Y]	Colchester Community Stadium is owned by Colchester
P131	881	[X] can be found in [Y]	Ryerson University can be found in Ontario
P138	604	[X] is named after [Y]	Great Britain is named after Britannia
P176	961	[X] is a product of [Y]	Ferrari 365 is a product of Ferrari
P178	580	[X] is a product of [Y]	Windows Embedded Compact is a product of Microsoft
P276	877	[X] is located in [Y]	1996 Summer Olympics is located in Atlanta
P495	907	[X] was created in [Y]	Lancashire wrestling was created in England
P1376	216	[X] is the capital city of [Y]	Ottawa is the capital city of Canada
P1412	965	[X] communicated in [Y]	Lon Chaney communicated in English

Table C.3: All relations we used from ParaRel along with number of items, query template, and an example item for each.

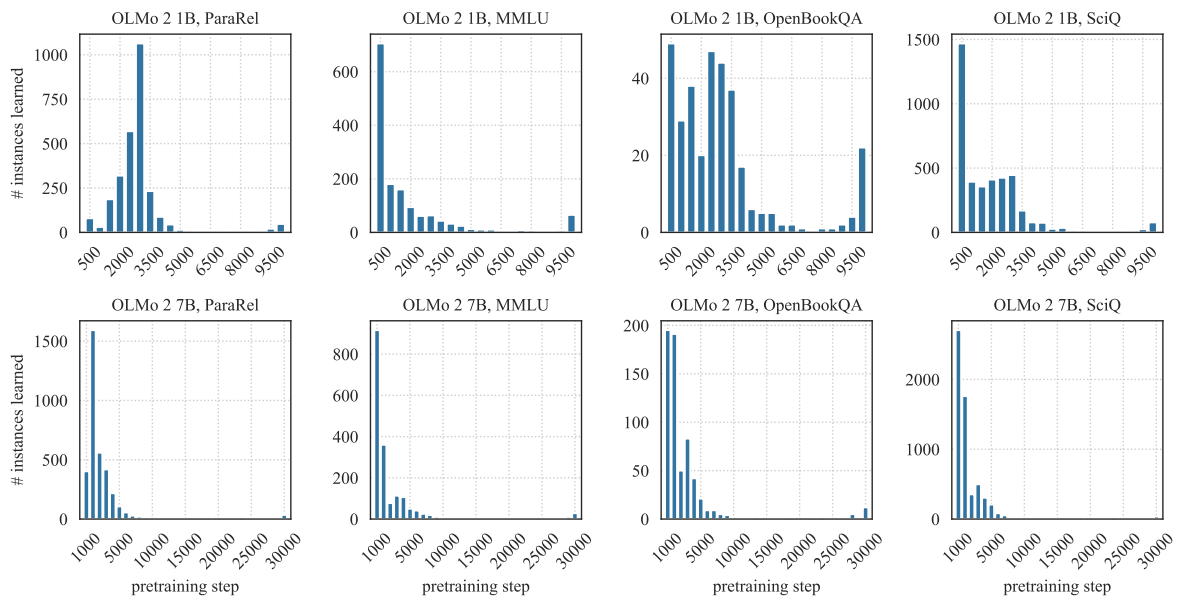


Figure C.2: Number of items learned per pretraining step, for each model size and dataset.

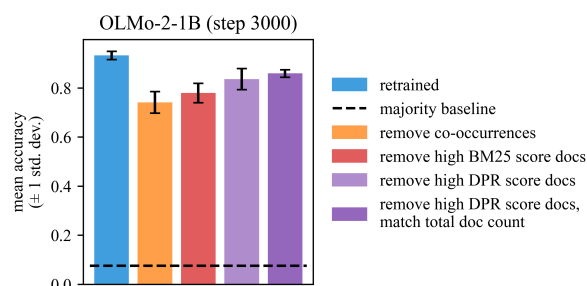


Figure C.3: Varying the method of matching documents to items results in different levels of performance reduction on ParaRel items. Selecting relevant documents using BM25 and DPR scores both lead to similar or less reduction in performance compared to selecting based on term cooccurrence alone, when removing the same number of documents per item or in total.

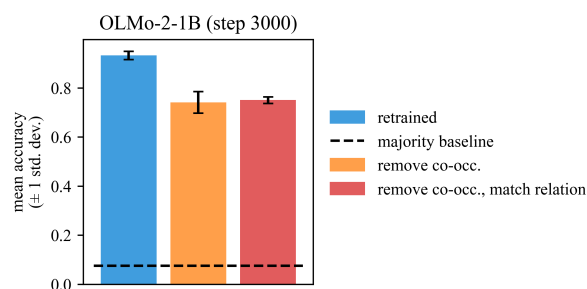


Figure C.4: Modifying documents by replacing them with ones expressing the same ParaRel relation type but without having the term cooccurrence has a similar effect to replacing with random unrelated documents, indicating the importance of the specific entity pair being mentioned.

Dataset	Subset(s)	# Items	Normalization	Example
MMLU	test	14,042	char	The Rhine River is an example of which type of boundary? (a) geometric (b) artificial (c) natural (d) relict
OpenBookQA	train, validation, test	5,957	PMI	What functions without contact between objects? (a) weight (b) friction (c) opposites attracting (d) pressure
SciQ	train, validation, test	13,679	char	In order to create food, what do photosynthetic protists use? (a) decayed matter (b) thermal energy (c) hydrocarbons (d) light energy

Table C.4: Additional details on MMLU, OpenBookQA, and SciQ including subsets used, number of items per dataset, and an example item.

intervention	model size	steps	$ \mathcal{E}^{\text{target}} $	% docs	% tokens	JSD
suppress learning (coocc.)	1B	1500–2000	121	2.9%	6.1%	0.011
	1B	2500–3000	367	4.7%	8.5%	0.017
	7B	2000–2150	272	2.2%	4.7%	0.010
	7B	3000–3150	62	0.5%	1.0%	0.003
suppress learning (occ.)	1B	1500–2000	116	24.0%	37.3%	0.052
	1B	2500–3000	362	35.0%	50.4%	0.085
	7B	2000–2150	272	43.7%	53.4%	0.096
	7B	3000–3150	62	17.3%	28.2%	0.047
promote learning (coocc.)	1B	1500–2000	162	3.2%	6.3%	0.011
	1B	2500–3000	88	1.3%	2.5%	0.005
	7B	2000–2150	371	9.9%	18.3%	0.031
	7B	3000–3150	86	1.7%	3.2%	0.008

Table C.5: Detailed information on the ParaRel interventions for each intervention type (suppressing learning by removing cooccurrences, suppressing learning by removing occurrences, and promoting learning by adding cooccurrences), model size, and set of pretraining steps. Values include the number of target items ($|\mathcal{E}^{\text{target}}|$), percentage of documents changed in the data batch (% docs), percentage of tokens changed in the data batch (% tokens), and the Jensen-Shannon divergence between the token distributions of the original and modified data batches (JSD, takes values between 0 and 1).

intervention	model size	steps	$ \mathcal{E}^{\text{target}} $	% docs	% tokens	JSD
reduce pro-male bias	1B	5500–6000	82	12.1%	0.6%	0.021
	7B	2000–2150	29	7.7%	0.4%	0.014
reduce pro-female bias	1B	5500–6000	91	7.1%	0.7%	0.010
	7B	2000–2150	18	4.5%	0.4%	0.006

Table C.6: Detailed information on the WinoBias interventions for each intervention type (reducing pro-male bias, reducing pro-female bias), model size, and set of pretraining steps. Values include the number of target items ($|\mathcal{E}^{\text{target}}|$), percentage of documents changed in the data batch (% docs), percentage of tokens changed in the data batch (% tokens), and the Jensen-Shannon divergence between the token distributions of the original and modified data batches (JSD, takes values between 0 and 1).

dataset	model size	steps	matching method	$ \mathcal{E}^{\text{target}} $	% docs	% tokens	JSD
SciQ	1B	2000–2500	BM25	424	15.4%	18.9%	0.038
			DPR		10.5%	10.3%	0.026
	7B	2000–2150	BM25	353	21.3%	26.0%	0.041
			DPR		14.4%	14.3%	0.032
OpenBookQA	1B	2000–2500	BM25	47	3.8%	5.1%	0.010
			DPR		3.5%	3.2%	0.008
	7B	2000–2150	BM25	50	6.6%	8.4%	0.013
			DPR		5.6%	5.1%	0.011
MMLU	1B	1000–1500	BM25	160	10.2%	17.9%	0.020
			DPR		11.1%	10.9%	0.014
	7B	2000–2150	BM25	77	10.0%	16.0%	0.020
			DPR		9.3%	9.7%	0.013

Table C.7: Detailed information on the MCQA dataset interventions for each dataset, model size, set of pretraining steps, and item-document matching method. Values include the number of target items ($|\mathcal{E}^{\text{target}}|$), percentage of documents changed in the data batch (% docs), percentage of tokens changed in the data batch (% tokens), and the Jensen-Shannon divergence between the token distributions of the original and modified data batches (JSD, takes values between 0 and 1).