From the holdings of the Department of Special Collections , Princeton University Library, Princeton, NJ

These documents are either in the public domain, or can only be used for educational and research purposes (Fair use) as per U.S. Copyright law (text below). By accessing this file, all users agree that their use falls in the public domain or within fair use as defined by the copyrightlaw. We do not charge any permission or usage fees for the publication of images of material inour collections, including those provided by Princeton University Library, via our website, catalog, or directly from staff.For more information regarding copyright, credit, and citation please review the information onthe Copyright, Credit and Citation Guidelines page of our website.
https://library.princeton.edu/special-collections/policies/copyright-credit-and-citation-guidelines. If you have any questions, including how to order higher-quality images from the collection, contact Special Collections Public Services via our Ask Us! Form. https://library.princeton.edu/special-collections/ask-us

---

Master's Theses, 2022-2026

Computer Science, 2023

2024

Towards Optimizing In-Context Learning and Ensuring Data Transparency in Large Language Models

Ajith, Anirudh

Princeton, NJ : Princeton University

---

http://arks.princeton.edu/ark:/88435/dsp01kk91fp94z

U.S. Copyright law: The copyright law of the United States (Title 17, §108, United States Code) governs the making of photocopies or other reproductions of copyrighted material. Under certain conditions specified in the law, libraries and archives are authorized to furnish a photocopy or other reproduction. One of these specified conditions is that the photocopy or reproduction is not to be used for any purpose other than private study, scholarship, or research. If a user makes a request for, or later uses, a photocopy or reproduction for purposes in excess of fair use that user may be liable for copyright infringement. This institution reserves the right to refuse to accept a copying order if, in its judgment, fulfillment of the order would involve a violation of copyright law. https://www.copyright.gov/title17/title17.pdf

Towards Optimizing In-Context Learning and Ensuring Data Transparency in Large Language Models

Anirudh Ajith

Advisor: Karthik Narasimhan

Reader: Danqi Chen

Submitted in partial fulfillment

of the requirements for the degree of

Master of Science in Engineering

Department of Computer Science

Princeton University

May 2024

I hereby declare that I am the sole author of this thesis.

I authorize Princeton University to lend this thesis to other institutions or individuals for the purpose of scholarly research.

I further authorize Princeton University to reproduce this thesis by photocopying or by other means, in total or in part, at the request of other institutions or individuals for the purpose of scholarly research.

_____

Anirudh Ajith

Abstract

Contemporary large language models (LLMs) are trained on enormous text corpora sourced from books and the internet. Although in-context learning (ICL) enables them to show strong performance on a variety of tasks, these models remain very sensitive to the precise details of this prompts. Instructions, a key component of these prompts have remained understudied in the literature thus far. In the first part of this thesis, we develop InstructEval – an evaluation suite for the systematic evaluation of instruction selection methods, and perform experiments to comprehensively evaluate 7 popular instruction selection methods. Our experiments reveal that curated manually-written instructions or simple instructions without any task-specific descriptions often elicit superior ICL performance overall than automatic instruction-induction methods, hence pointing to a lack of generalizability among the latter. Since the leakage of ICL benchmark test data into pretraining corpora can compromise benchmark evaluations, we are motivated to study the pretraining data detection problem in the second part of this thesis. Given a piece of text and black-box access to an LLM, we ask if we can determine whether the model was pretrained on the provided text. We introduce the WikiMIA benchmark to facilitate this study and propose a new detection method called Min-k% Prob that unlike prior work, does not require reference models, additional training or any knowledge about a model's pretraining data distribution. In addition to showing that Min-k% Prob outperforms baselines on WikiMIA, we also demonstrate its utility for detecting the leakage of benchmark data and copyrighted content.

Acknowledgements

I am very grateful to have had opportunities to work with Professors Karthik Narasimhan and Danqi Chen from the Princeton NLP group. I've learned a lot about research and the field in general not only from them, but also from their students.

I am extremely grateful to Ameet Deshpande, Tianyu Gao, Alex Wettig and Alexis Chevalier for being patient enough to mentor me and allow me to grow as a researcher over the last 2 years. But I am *especially* grateful to Mengzhou Xia for truly guiding and nurturing me with seemingly infinite compassion and patience as I've begun my research career at Princeton. In addition, I'd like to thank Chris Pan and Weijia Shi for being great project-mates. I'm sure that none of my accomplishments here would have come to pass if it weren't for every one of these people.

I would like to thank my friends Nathan, Sabhya, Rish, Anika, Creston, Emma, Prachi, Hussein, Kevin, Mihir, Kym, Andy and so on (you know who you are) for all the fun we've had. Oh, and also Nikhil.

I'm very grateful to both my parents for their constant moral support and encouragement to keep pushing and keep learning.

To my family, and to my friends

Contents

Abstract iii

Acknowledgements iv

List of Tables ix

List of Figures xii

1 Introduction **1**

1.1 Structure 3

2 Investigating Instruction Choice for In-Context Learning **4**

2.1 Introduction 4

2.2 Related Work 6

2.2.1 In-Context Learning and Existing Benchmarks 6

2.2.2 Automated Prompt Engineering Methods 7

2.2.3 Understanding in-context learning 7

2.3 Evaluation Suite 8

2.3.1 Prompt format 8

2.3.2 Metrics 8

2.3.3 Aggregating metrics across Models 10

2.3.4 Tasks 12

2.3.5 Models 13

2.4 Experimental setup 13

2.5 Results 15
2.5.1 Less sophisticated instruction selection methods tend to show higher accuracy 17
2.5.2 Ranges of variation of aggregated scores 17
2.5.3 Analysis 19
2.6 Discussion 20

3 Membership-inference Attacks for Black Box Large Language Models 22

3.1 Introduction 22
3.2 Problem: Membership Inference Attacks 25
3.3 Related Work 27
3.3.1 Membership inference attacks in NLP 27
3.3.2 Dataset contamination 27
3.4 Min-K% Prob: a novel MIA for blackbox LLMs 28
3.5 WikiMIA: A Dynamic Evaluation Benchmark 29
3.5.1 Construction 29
3.5.2 Benchmark setting. 29
3.6 Experiments 30
3.6.1 Datasets and Metrics 30
3.6.2 Baseline Detection Methods 31
3.6.3 Implementation Details 31
3.6.4 Results 32
3.6.5 Analysis 32
3.7 Detecting Pretraining on Copyrighted Material 33
3.7.1 Experimental Setup 34
3.7.2 Results 34
3.8 Detecting Downstream Dataset Contamination 35

3.8.1 Experimental Setup 36
3.8.2 Results 36
3.8.3 Analysis 37

4 Conclusion and Future Work **40**

A Supplementary material for Chapter 2 **42**
A.1 Implementation details 42
A.1.1 Evaluation 42
A.1.2 Instruction selection methods 43
A.2 Drawbacks of aggregation techniques used in previous work 44
A.3 Full results 46

B Supplementary material for Chapter 3 **47**
B.1 Additional Results 47
B.2 Details of WikiMIA 47

List of Tables

2.1 Tasks included in our evaluation suite, and the demonstrations templates we use for each task. 12

2.2 Model families and corresponding model scales included in our evaluation suite. 13

2.3 Instruction selection methods we evaluate 13

2.4 Mean relative gain values associated with zero-shot accuracy, and few-shot accuracy with unperturbed and perturbed test inputs. Only values that correspond to the same task and metric should be compared. Positive values represent above-average performance, and negative values represent below-average performance. The '# wins' column shows the number of tasks where a method achieved the highest aggregated performance. 16

2.5 Mean standard deviation of few-shot accuracy on varying selections and permutations of demonstrations respectively. The '# wins' column respresents the number of tasks where a method achieves best performance. 18

2.6 Mean relative gain values for zero-shot and few-shot accuracy computed separately over models with < 6 and ≥ 6 billion parameters, and averaged by task-type. We also tabulate the total range of variation of these values in each setting. 19

2.7 Mean relative gain values for zero-shot accuracy and few-shot accuracy computed separately over individual model families and averaged by task-type. Positive values represent above-average performance, and negative values represent below-average performance. We also tabulate the number of tasks where a method achieved highest aggregated performance in the '# wins' column under every model family. 20

3.1 AUC score for detecting pretraining examples from the given model on WIKIMIA for MIN-K% PROB and baselines. *Ori.* and *Para.* denote the original and paraphrase settings, respectively. **Bold** shows the best AUC within each column. 33

3.2 Top 20 copyrighted books in GPT-3's pretraining data. The listed contamination rate represents the percentage of text excerpts from each book identified in the pretraining data. 35

3.3 AUC scores for detecting contaminant downstream examples. **Bold** shows the best AUC score within each column. 36

3.4 AUC scores for detecting contaminant downstream examples using two different learning rates. Detection becomes easier when higher learning rates are used during training. **Bold** shows the best AUC score within each column. 39

A.1 Tasks included in our evaluation suite, and the demonstrations templates we use for each task. 43

A.2 Example instructions obtained using each method for the AG News task 43

A.3 Sample generic instructions 44

B.1 TPR@5%FPR score for detecting pretraining examples from the given model on WikiMIA for Min-k% Prob and baselines. *Ori.* and *Para.* denote the original and paraphrase settings, respectively. **Bold** shows the best score within each column. 47

B.2 TPR @ FPR=5% for detecting contaminant downstream examples using reference-based and reference-free methods. **Bold** shows the best reference-free TPR within each column. 48

B.3 Accuracy of the model finetuned in Section 3.8.1 on each non-contaminant and contaminant examples used for AUC computation for each downstream dataset. The difference in average classification accuracy of contaminant examples over that of non-contaminant examples is 0.04 at a learning rate of 1×10^{-5} and 0.11 at a learning rate of 1×10^{-4}. This indicates that memorization becomes a significantly more pronounced effect than generalization at larger learning rates. 48

B.4 Input template we use to prompt GPT-4 to obtain the similarity score. 49

List of Figures

2.1 InstructEval allows the assessment of instruction selection methods for ICL across a range of models and tasks along five metrics. 5

2.2 An example of a prompt following the template we use for IMDB. By 'prompt' we refer to the concatenation of the instruction, solved demonstrations and an unsolved test example. 8

2.3 We provide schematic diagrams that show prompts are modified to measure *perturbation accuracy*, *selectional sensitivity* and *permutational sensitivity*. We perturb the test input to measure perturbation accuracy, and demonstration selection and permutation respectively while measuring selectional and permutational sensitivity. 9

3.1 **Overview of Min-k% Prob**. To determine whether a text X is in the pretraining data of a LLM such as GPT, Min-k% Prob first gets the probability for each token in X, selects the k% tokens with minimum probabilities and calculates their average log likelihood. If the average log likelihood is high, the text is likely in the pretraining data. 24

3.2 As model size or text length increases, detection becomes easier. . . . 33

3.3 AUC scores for detecting the validation set of copyrighted books on GPT-3. 35

3.4 Distribution of detected contamination rate of 100 copyrighted books. 35

3.5 We show the effect of contamination rate (expressed as a percentage of the total number of pretraining tokens) and occurrence frequency on the ease of detection of data contaminants using MIN-K% PROB. . . 38

Chapter 1

Introduction

Transformer-based large language models have emerged as a significant development in the fields of natural language processing and artificial intelligence. These models, trained primarily to perform next-token prediction over large corpora of text scraped from webpages and books, show emergent abilities enabling strong (and even human-level) performance on a wide range of tasks.

These impressive abilities are typically elicited using "in-context learning" prompts that include instructions, annotated demonstrations and unsolved test examples. Recent work has shown that LLM performance remains sensitive to the precise details of the demonstrations, instructions and even formatting used in these prompts. While the optimal selection of demonstrations has been explored, instruction-choice remains understudied with existing analyses confined to narrow subsets of models and tasks, thus reducing the generalizability of their insights. In Chapter 2, we introduce the InstructEval suite for the systematic evaluation of instruction selection methods for ICL. We use InstructEval, which includes 13 open-sourced LLMs of varying scales from four model families and covers nine tasks across three categories to evaluate seven popular instruction selection methods over five metrics relevant to ICL. Our experiments in this chapter reveal that curated manually-written instructions or sim-

ple instructions without any task-specific descriptions often elicit superior ICL performance overall than automatic instruction-induction methods, hence pointing to a lack of generalizability among the latter.

Our ICL experiments in Chapter 2 are conducted under the assumption that the LLMs we use have never encountered the benchmark data we evaluate them on during the course of their pretraining. While this is not an uncommon assumption, prior work indicates that benchmark data does often contaminate pretraining data and that such contamination can compromise ICL benchmark evaluations. We are motivated by this possibility to explore membership inference attacks for blackbox LLMs in the Chapter 3 of this thesis.

As the scale of data used to train contemporary LLMs has grown into the trillions of tokens, model developers have also become increasingly reluctant to disclose their data sources. Simultaneously, there have been growing concerns that academic benchmarks used to evaluate LLMs may have been compromised by the leakage of test data into pretraining corpora. Concerns also exist that these LLMs may be trained on sensitive personal information or copyrighted content which could be problematic to generate post-deployment. In the second part of this thesis, we study the pretraining data detection problem: given a piece of text and black-box access to an LLM, can we determine if the model was trained on the provided text? We introduce the WikiMIA benchmark to facilitate this study and propose a new detection method called Min-K% Prob that unlike prior work, does not require reference models, additional training or any knowledge about a model's pretraining data distribution. In addition to showing that Min-K% Prob outperforms baselines on WikiMIA, we also demonstrate its utility for detecting the leakage of benchmark data and copyrighted content.

We have also released articles describing our work in Chapters 2 and 3 to the broader NLP research community in the form of conference publications [2, 71].

1.1 Structure

Following this introduction, this thesis is organized into 3 chapters.

- Chapter 2 details our exploration of the systematic evaluation of instruction selection methods for in-context learning in LLMs and presents relevant related work, ablations and discussion organized into appropriate sections and subsections.

- Chapter 3 describes the relevant work, motivations, experiments and case studies we conduct as a part of our exploration into the pretraining data detection problem for LLMs.

- Although each of chapters 2 and 3 presents a self-contained account of the material relevant to the problem they explore, we summarize our findings from both these sets of experiments and briefly discuss their implications for further research in Chapter 4.

Chapter 2

Investigating Instruction Choice for In-Context Learning

2.1 Introduction

One of the most effective insights in NLP research in recent years has been that large language models trained to perform next-token prediction show emergent in-context learning (ICL) abilities [10, 69, 100]. While the bulk of research interest has shifted away from task-specific models and towards creating "foundation models" to perform a variety of tasks using appropriately constructed prompts, the performance of ICL remains sensitive to the precise details of prompt construction. Prompt engineering remains critical for achieving optimal ICL performance [65, 103, 86].

In practice, ICL typically involves prompting a language model using a concatenation of a task-specific *instruction*, a short sequence of annotated in-context examples known as *demonstrations*, and a *test example* (Figure 2.2). Much of the research interest surrounding in-context learning has focused on understanding the optimal selection, ordering of demonstrations, and label-space choices [40, 77, 66, 84, 43, 89, 64]. However, *instruction choice* remains a relatively underexplored aspect of prompt en-

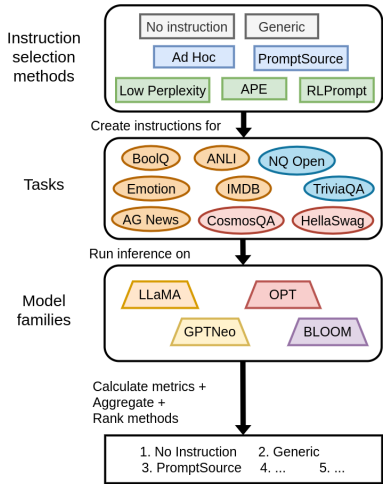


Figure 2.1: InstructEval allows the assessment of instruction selection methods for ICL across a range of models and tasks along five metrics.

gineering despite its established significance [54] on downstream performance.

Even among recent works exploring automatic instruction selection [28, 24, 19, 104], the use of different evaluation protocols makes the comparison of their relative performances difficult. Existing studies typically limit their analyses to specific models or tasks; for example, [104] focus on a single model, and while [19] consider multiple model scales, they all belong to a single model family. Moreover, evaluations often span disparate task selections with minimal overlap and are primarily dominated by classification tasks, neglecting other task types like multiple-choice QA or generation. Lastly, most previous works studying automatic instruction selection tend to emphasize zero-shot accuracy, overlooking other pertinent ICL metrics such as few-shot accuracy and robustness measures.

To address these issues, we build InstructEval, an evaluation suite for the comprehensive evaluation of instruction selection methods. The suite covers a diverse collection of 13 open-sourced autoregressive LLMs from four model families and nine tasks spanning three task types. Additionally, it also incorporates three accuracy metrics and two sensitivity metrics that are of interest to ICL. We perform evaluations of seven popular instruction selection methods including trivial instruction

baselines, manually curated instructions, and sophisticated automatic methods using our suite.

Overall, we find that the relative effectiveness of these approaches varies significantly across different models and task types. We discover that curated manually-written instructions and task-agnostic instructions can elicit better aggregated performance (over models) than automatically induced ones, highlighting the lack of generalizability of the latter. We also find that including instructions in few-shot prompts usually tends to hurt ICL performance at the model scales we consider. Our findings suggest that it may be optimal for ICL practitioners to omit instructions in few-shot settings and use curated manually-written instructions in zero-shot settings, rather than contemporary automatic induction techniques that require substantial computation and hyperparameter tuning to achieve competitive performance. We release the evaluation suite we develop to aid the systematic study of even more questions regarding prompt engineering that we do not explicitly address in our work.

2.2 Related Work

2.2.1 In-Context Learning and Existing Benchmarks

As language models have scaled, in-context learning has emerged as a popular paradigm and remains ubiquitous among several autoregressive LLM families [10, 81, 68, 8, 101]. Benchmarks like BigBench [75] and HELM [38] have been created for the holistic evaluation of these models. BigBench focuses on few-shot abilities of state-of-the-art large language models, while HELM extends its evaluation to consider metrics like robustness and bias. However, these benchmarks focus on evaluating and ranking *language models*, and do not address the systematic evaluation of *prompting methods*. Although contemporary work by [92] also aims to perform a similar systematic analysis of prompting methods, they focus on simple probability-based prompt selection while

we evaluate a broader range of methods including trivial instruction baselines, curated manually selected instructions, and sophisticated automated instruction selection.

2.2.2 Automated Prompt Engineering Methods

There has been interest in performing automated prompt-engineering for target downstream tasks within ICL. This has led to the exploration of various prompting methods, ranging from simple heuristics such as selecting instructions with the lowest perplexity [24], inducing instructions from large language models using a few annotated input-output pairs [104], to utilizing RL objectives to create discrete token sequences as prompts [19]. However, these works restrict their evaluation to small sets of models and tasks with little intersection, hindering their objective comparison.

2.2.3 Understanding in-context learning

There has been much recent work attempting to understand the mechanisms that drive in-context learning. Studies have found that the selection of demonstrations included in prompts significantly impacts few-shot accuracy across most tasks [41, 1, 91]. Works like [44] also show that altering the ordering of a fixed set of demonstrations can affect downstream accuracy. Prompts sensitive to demonstration permutation often exhibit lower accuracies [14], making them less reliable, particularly in low-resource domains.

Our work aims to bridge these gaps by systematically evaluating the efficacy of popular instruction selection approaches over a diverse set of tasks and models, facilitating objective comparison. We evaluate these methods not only on accuracy metrics, but also on sensitivity metrics to glean additional insights. We recognize that other facets of prompting not covered by instruction engineering exist [88, 93, 83], and defer these explorations to future work.



Figure 2.2: An example of a prompt following the template we use for IMDB. By 'prompt' we refer to the concatenation of the instruction, solved demonstrations and an unsolved test example.

2.3 Evaluation Suite

2.3.1 Prompt format

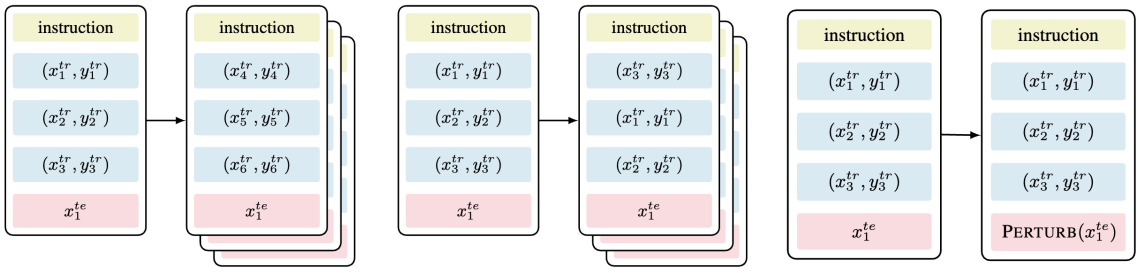
We define a 'prompt' as the full textual input provided to an LLM. Our evaluation suite supports the use of any number of demonstrations, arbitrary demonstration templates and the inclusion of custom strings anywhere within the prompt. Since the instructions used can be set to any arbitrary strings, users are free to use any external means to select instructions and have them evaluated by our suite.

For consistency, we conduct all experiments in this work using prompts that begin with an instruction, continue with a sequence of annotated training demonstrations, and conclude with an unsolved test example[1] (Figure 2.2). We express each example in a minimal, task-specific key-value format (Table A.1) that reflects task semantics.

2.3.2 Metrics

Accuracy metrics Accuracy is typically the primary metric of interest in ICL. While ICL is most commonly performed in few-shot settings where a handful of

[1]Instructions are omitted during 'Null instruction' evaluations. Demonstrations are omitted in zero-shot evaluations.



(a) Perturbation accuracy (b) Selectional sensitivity (c) Permutational sensitivity

Figure 2.3: We provide schematic diagrams that show prompts are modified to measure *perturbation accuracy*, *selectional sensitivity* and *permutational sensitivity*. We perturb the test input to measure perturbation accuracy, and demonstration selection and permutation respectively while measuring selectional and permutational sensitivity.

annotated demonstrations are included in the prompt, models are also prompted zero-shot without the use of such demonstrations. Since real-world scenarios can often contain grammatical errors and misspellings in the test input, it is desirable to find prompts robust to these perturbations. Hence, we measure *zero-shot accuracy*, *few-shot accuracy*, and *perturbation accuracy*[2] in our evaluations. Following [38], we measure perturbation accuracy by introducing random capitalization, spacing, contractions and common misspellings in the test input.

Sensitivity metrics Previous work has shown that the accuracy obtained using a prompt template can fluctuate significantly as a function of the set of demonstrations included in the prompt [40, 77, 66, 84] and the order they are presented in [44]. It may be desirable in practice to identify prompt templates and instructions that offer consistent performance regardless of the choice of demonstrations and their arrangement. Hence, we introduce *selectional sensitivity* and *permutational sensitivity* metrics to measure the sensitivity of chosen instructions respectively to selected demonstrations, and the order in which they are arranged. We quantify the sensitivity

[2]We choose to treat this as an accuracy metric rather than a sensitivity metric since it is not meaningful to measure sensitivity to such perturbations in situations where a prompt only elicits near random-chance task performance from a model.

of an instruction (given a model and task) using the standard deviation of accuracies obtained on varying the selection or permutation of the demonstrations used, each across 16 random choices.

2.3.3 Aggregating metrics across Models

Each instruction selection method being tested across N models and M datasets yields NM values per metric. Comparing these NM-dimensional vectors directly is complex. It can be challenging to reduce them to a single representative scalar. Simple approaches such as computing the mean of these NM values can prove inadequate since the resulting scores would tend to be heavily influenced by metric values that exhibit a high variance across different inspected methods.

We opt against using aggregation techniques used by previous works [38, 75] due to their drawbacks (Section A.2) and instead adopt 'mean relative gain' as a means to aggregate accuracy metrics across multiple models. We rely on simple averaging for sensitivity metrics, partly because we observe that these quantities do not show much variation across methods.

Accuracy metrics

Considering the range of models and datasets in our evaluation suite, we unsurprisingly observe substantial variation in accuracy magnitudes across model scales and tasks. However, we notice that the degree of variation in accuracy due to instruction choice is usually considerably smaller than the degree of variation due to model and task choice. To meaningfully compare and aggregate the relative performance of different instruction selection methods across models, we use a measure called *mean relative gain*. First, we define the *relative gain* for a value x from a population P as the percentage by which x exceeds the mean value of P:

$$\text{r-gain}_P(x) = 100 \times \frac{x - \mu_P}{\mu_P}$$

Consider a collection of models $\mathcal{M}$ and instructions $\mathcal{I}$ for a task t. Given a model m, we calculate the raw accuracy scores s_{tmi} for each instruction $i \in \mathcal{I}$. Taking this set S_{tm} to be the population, we compare the performances of the instructions against each other by computing their corresponding relative gains $r_{tmi} = \text{r-gain}_{S_{tm}}(s_{tmi})$. Each r_{tmi} represents the degree by which method i outperforms the average performance along the metric on task t for model m.

We now define the mean relative gain as

$$\overline{r}_{ti} = \frac{1}{|\mathcal{M}|} \sum_{m \in \mathcal{M}} r_{tmi}$$

These $\overline{r}_{ti}$ values, tabulated and analyzed in 2.5, capture not only the ordinal information about each method's performance on a given task but also provide an intuitive sense of the magnitude by which these methods outperform others. Specifically, if an induction method i has a mean relative gain $\overline{r}_{ti}$ on task t, this means that method i exceeds average performance (across $\mathcal{I}$) on task t by $\overline{r}_{ti}$ percent when averaged across models $\mathcal{M}$.

Sensitivity metrics

To aggregate the sensitivity of an instruction selection/induction method i over all models for a task t, we simply compute the average of the raw sensitivity scores (described in 2.3.2). Specifically, if σ_{tmi} is the raw sensitivity score obtained for model m and task t when using instruction i, then the aggregated sensitivity score $\overline{\sigma}_{ti}$ is given by

$$\overline{\sigma}_{ti} = \frac{1}{|\mathcal{M}|} \sum_{m \in \mathcal{M}} \sigma_{tmi}$$

We choose to avoid more sophisticated aggregation strategies like relative gain for sensitivity metrics since standard deviations are already secondary metrics, hence making it unintuitive to discuss the relative gain of the standard deviation obtained using a method over the average.

2.3.4 Tasks

Task Type	**Tasks**	**Demonstration template**
Classification (CLS)	AG News [102]	`News:` (text)\`nCategory:` (label)
	ANLI [61]	`Premise:` (premise)\`nHypothetisis:` (hypothesis)\`nRelation:` (label)
	BoolQ [16]	`Passage:` (passage)\`nQuestion:` (question) \`nAnswer:` (label)
	IMDB [45]	`Review:` (text)\`nSentiment:` (label)
	TweetEval Emotion [56]	`Tweet:` (text)\`nEmotion:` (label)
Multiple-choice (MCQ)	CosmosQA [29]	`Passage:` (context)\`nQuestion:` (question)\`nAnswer:` (answer)
	HellaSwag [97]	`Sentence:` (ctx)\`nAnswer:` (answer)
Generative QA (GQA)	NQ-Open [36]	`Question:` (question)\`nAnswer:` (answer)
	TriviaQA [34]	`Question:` (question)\`nAnswer:` (answer)

Table 2.1: Tasks included in our evaluation suite, and the demonstrations templates we use for each task.

While previous instruction induction [104, 19] work has tended to focus mostly on classification tasks, we include 9 tasks (Table 2.1) in our evaluation suite spanning classification (CLS), multiple-choice question-answering (MCQ) and generative question-answering (GQA) to assess the applicability of instruction selection and induction methods to other task-types as well. We concentrate on tasks that are challenging to contemporary language models, and yet are not so demanding that the performance of these models does not exceed random chance. We exclude certain generative tasks, like summarization, which are challenging to assess objectively. [3]

Model Family	Size
BLOOM [68]	1.1B, 1.7B, 3B, 7.1B
GPT Neo [8, 6]	1.3B, 2.7B, 20B
LLaMA [81]	7B, 13B
OPT [101]	1.3B, 2.7B, 6.7B, 13B

Table 2.2: Model families and corresponding model scales included in our evaluation suite.

Method	Task-specific	Automatic induction
Null instruction	✗	✗
Generic instruction	✗	✗
PromptSource [3]	✓	✗
Ad hoc	✓	✗
Low Perplexity [24]	✓	✓
APE [104]	✓	✓
RLPrompt [19]	✓	✓

Table 2.3: Instruction selection methods we evaluate

2.3.5 Models

We include a diverse range of 13 autoregressive LLMs (2.2) from 4 model families of sizes ranging from 1.1 billion to 20 billion parameters in our evaluation suite. We choose contemporary models that span different architectures and training paradigms which are known to show good ICL performance. This diversity bolsters the generalizability of insights obtained using our evaluation suite while mitigating potential bias towards any specific model family. Moreover, we select open-source models which are large enough to show non-trivial ICL performance while still being small enough to run on reasonable consumer hardware to ensure the practical significance of our findings.

2.4 Experimental setup

We perform experiments evaluating 3 families of instruction selection methods (listed in 2.3).

[3]Standard summarization metrics correlate poorly with human preferences [38, 25].

Task-agnostic instructions In practical ICL settings, it is straightforward to use instructions that contain no task-specific information.

- **Null instruction:** We assess the impact of omitting instructions from the prompt. This amounts to constructing prompts that consist of demonstrations and a test example in few-shot, and only an unanswered test-example in zero-shot settings.
- **Generic instructions:** We assess the impact of using generic task-agnostic instructions such as `Complete the following task:`. These instructions require minimal effort to write since they do not demand knowledge of the task. We list the set of generic instructions we evaluate in Table A.3.

Manual task-specific instructions We evaluate manually-written task-specific instructions that ICL practitioners may use in practice.

- **PromptSource:** PromptSource [3] is a public collection of manually-curated prompt templates pertaining to 170+ datasets which are often used off-the-shelf for ICL and are generally considered high-quality.
- **Ad hoc:** ICL practitioners often create task-specific instructions ad hoc, based on the semantics of the given task. We simulate this mode of instruction selection by asking ChatGPT to generate several paraphrases of task-specific seed instructions we obtain from PromptSource and randomly sampling from the generated set.

Automatically synthesized task-specific instructions We evaluate 3 popular automated instruction selection and induction methods that are representative of previous work.

- **Low Perplexity:** [24] find that the perplexity a model associates with an instruction is negatively correlated with its ICL performance when using that instruction. We use the SPELL algorithm proposed by [24] to select the least perplexity instructions (for each model) from a large pool of ChatGPT paraphrased instructions.

- **APE**: [104] is an automatic few-shot method for inducing instructions by prompting a language model to describe the given task, and refining the set of generated prompts using accuracy on a small held-out validation set. While [104] limit their evaluation to GPT-3 [10] and InstructGPT [63], we assess APE's applicability to a significantly larger set of models and tasks.

- **RLPrompt** [19] is a reinforcement-learning-based approach for few-shot prompt induction. While the original authors only evaluate their method using GPT-2 on a few classification tasks, we expand this assessment to many more models and tasks. Notably, we assess the extensibility of RLPrompt to MCQ tasks, but do not test RLPrompt performance on GQA tasks since the algorithm is not directly applicable to generation tasks.

2.5 Results

We tabulate the mean relative gain values over accuracy metrics in Table 2.4, and the mean standard deviations corresponding to selectional and permutational sensitivity metrics in Table 2.5.

Method	CLS					MCQ		GQA		# wins
	AG News	ANLI	BoolQ	IMDB	Emotion	HellaSwag	CosmosQA	TriviaQA	NQ-Open	
Zero-shot accuracy (mean relative gain) ↑										
Null Instruction	2.26	1.07	**2.48**	−3.52	−5.30	**2.54**	**5.94**	−3.02	−25.67	3
Generic Instruction	3.55	−0.39	0.03	1.69	2.39	−0.13	−1.67	−1.46	−5.99	0
PromptSource	**5.81**	**1.38**	−0.65	**4.34**	**5.13**	−1.54	−3.42	**17.08**	**22.15**	6
Ad hoc	−0.33	0.21	0.55	1.41	0.66	−0.27	−2.46	−1.97	2.31	0
Low Perplexity	−0.59	1.22	0.56	0.84	−4.07	−1.38	−2.18	−5.99	2.81	0
APE	−15.63	−3.86	−1.07	−1.77	−0.26	−1.06	0.00	−4.64	4.39	0
RLPrompt	4.92	0.37	−1.89	−2.99	1.46	1.85	3.79	–	–	0
Few-shot accuracy (mean relative gain) ↑										
Null Instruction	4.09	−0.22	**0.87**	−0.80	**5.89**	0.17	**1.33**	**0.45**	−0.02	4
Generic Instruction	**5.16**	−0.20	−0.10	0.45	4.84	0.04	−0.18	0.11	0.11	1
PromptSource	0.83	0.14	−0.79	0.39	−4.39	−0.06	−0.94	−0.36	**0.61**	1
Ad hoc	2.18	−0.10	−0.05	**0.60**	−5.63	−0.21	−0.59	0.09	−0.49	1
Low Perplexity	−1.96	**0.31**	−0.40	0.20	−6.79	−0.23	−0.61	−0.06	−0.02	1
APE	−15.43	0.10	0.06	−0.69	1.17	0.02	0.17	−0.24	−0.19	0
RLPrompt	5.13	−0.02	0.40	−0.14	4.90	**0.27**	0.81	–	–	1
Few-shot perturbation accuracy (mean relative gain) ↑										
Null Instruction	4.09	−0.08	0.11	−0.27	**5.98**	0.11	**1.10**	**0.81**	**1.28**	4
Generic Instruction	**5.15**	−0.18	−0.16	**0.56**	4.23	−0.02	−0.02	0.08	0.10	2
PromptSource	1.14	0.27	−0.02	0.33	−3.92	0.06	−0.53	−0.65	0.04	0
Ad hoc	1.68	0.51	−0.34	0.37	−5.87	−0.08	−0.63	−0.28	−0.61	0
Low Perplexity	−2.39	**0.68**	−0.12	−0.20	−6.61	−0.09	−0.66	−0.03	−0.78	1
APE	−14.32	−1.20	**0.28**	−0.82	1.26	−0.13	0.21	0.06	−0.03	1
RLPrompt	4.65	−0.01	0.24	0.03	4.94	**0.15**	0.53	–	–	1

Table 2.4: Mean relative gain values associated with zero-shot accuracy, and few-shot accuracy with unperturbed and perturbed test inputs. Only values that correspond to the same task and metric should be compared. Positive values represent above-average performance, and negative values represent below-average performance. The '# wins' column shows the number of tasks where a method achieved the highest aggregated performance.

2.5.1 Less sophisticated instruction selection methods tend to show higher accuracy

We find that **task-agnostic instructions dominate in few-shot settings** with Null instructions and Generic instructions achieving the highest aggregated performance in 5/9 tasks for few-shot accuracy and 6/9 tasks for perturbation accuracy. Although both these methods show above-average performance in few-shot settings, Null instructions tend to perform better among the two.

Although PromptSource instructions only show an average performance in few-shot settings, their **manually curated task-specific instructions prove most effective in zero-shot settings**, achieving the highest aggregated performance in 6/9 tasks and usually achieving markedly higher mean relative gain values than even the runner-up method for the task. This is especially true of GQA tasks where PromptSource instructions outperform the average by >17%.

Automatic task-specific instructions are usually outperformed by simple baselines. They fail to achieve the best zero-shot performance on any task we consider. While they do sometimes perform competitively with simpler baselines in the few-shot setting, emerging as the best-performing instructions in 2/9 tasks, this behavior is inconsistent. Low Perplexity instructions and APE instructions seldom show above-average performance in either setting while RLPrompt instructions show above-average performance in 5/7 tasks in both settings. They are still usually outperformed by instructions obtained through simpler means such as Null and PromptSource instructions.

2.5.2 Ranges of variation of aggregated scores

We notice that instructions have a more significant impact in zero-shot settings as compared to few-shot settings. For most tasks, we find that the highest mean rel-

ative gain values achieved in the zero-shot setting are markedly greater than those in the few-shot setting. Accordingly, the minimum values for each task are also relatively lower in zero-shot settings. This finding suggests that instructions play a significant role in informing models of semantics in zero-shot settings whereas in few-shot settings, most of a model's understanding of task-semantics comes from the demonstrations.

The degree of variation in accuracy due to instruction choice varies considerably across tasks. AG News and Emotion show the highest variability in few-shot performance while GQA tasks show the most variability in zero-shot settings.

	CLS					MCQ		GQA		# wins
Method	**AG News**	**ANLI**	**BoolQ**	**IMDB**	**Emotion**	**HellaSwag**	**CosmosQA**	**TriviaQA**	**NQ-Open**	
					Selectional sensitivity (mean standard deviation) ↓					
Null Instruction	**6.69**	2.45	4.73	**5.28**	6.97	2.46	**8.10**	2.59	2.28	3
Generic Instruction	6.87	2.50	4.76	5.40	6.97	2.48	8.16	2.61	2.26	0
PromptSource	6.73	2.26	4.85	5.37	6.43	2.43	8.26	**2.59**	2.28	1
Ad hoc	6.95	2.41	**4.62**	5.38	6.34	2.42	8.20	2.65	2.37	1
Low Perplexity	7.07	**2.17**	4.69	5.64	**6.25**	2.42	8.27	2.59	2.30	2
APE	7.44	2.98	4.63	5.70	6.67	2.43	8.16	2.65	**2.21**	1
RLPrompt	6.76	2.35	4.79	5.50	6.96	**2.36**	8.16	–	–	1
					Permutational sensitivity (mean standard deviation) ↓					
Null Instruction	6.02	**1.99**	3.82	**4.56**	5.34	1.12	1.87	1.48	1.24	2
Generic Instruction	**6.01**	2.19	3.89	4.56	5.49	1.15	1.68	**1.33**	1.22	2
PromptSource	6.06	2.15	3.61	4.69	4.30	**1.07**	1.67	1.47	**1.17**	2
Ad hoc	6.10	2.37	3.77	4.61	4.37	1.11	1.66	1.41	1.23	0
Low Perplexity	6.13	2.24	**3.50**	4.61	**4.29**	1.13	1.69	1.46	1.27	2
APE	6.14	2.36	3.69	4.84	5.08	1.10	1.78	1.41	1.21	0
RLPrompt	6.26	2.06	3.82	4.89	5.64	1.08	**1.65**	–	–	1

Table 2.5: Mean standard deviation of few-shot accuracy on varying selections and permutations of demonstrations respectively. The '# wins' column respresents the number of tasks where a method achieves best performance.

Table 2.5 shows that selectional and permutational sensitivities vary dramatically across tasks even though they are roughly consistent across all methods for a given task. This implies that all the methods we evaluate are comparable in sensitivity, which is unsurprising since none of them explicitly optimize for it. We also find that most methods show comparable, but usually lower permutational sensitivity than selectional sensitivity.

Method	< 6B parameters			≥ 6B parameters		
	CLS	**MCQ**	**GQA**	**CLS**	**MCQ**	**GQA**
Zero-shot accuracy (mean relative gain) ↑						
Null Instruction	−2.89	1.71	−15.86	2.07	**7.19**	−12.58
Generic Instruction	1.71	0.69	−0.64	1.16	−2.76	−7.33
PromptSource	**2.77**	−2.18	**25.03**	**3.70**	−2.83	**13.30**
Ad hoc	1.87	−0.94	4.56	−1.11	−1.86	−4.95
Low Perplexity	−2.35	−1.09	−8.24	1.85	−2.58	6.17
APE	−3.13	−0.54	−4.85	−6.14	−0.51	5.39
RLPrompt	2.01	**2.37**	–	−1.54	3.34	–
Variation Range	5.90	4.55	40.89	9.84	10.02	25.88
Few-shot accuracy (mean relative gain) ↑						
Null Instruction	2.63	**0.75**	**0.89**	1.20	**0.76**	−0.57
Generic Instruction	**3.09**	−0.10	−0.15	0.80	−0.03	0.41
PromptSource	−1.18	−0.58	−0.20	−0.28	−0.41	**0.51**
Ad hoc	−0.55	−0.45	0.04	−0.65	−0.35	−0.47
Low Perplexity	−2.57	−0.48	−0.30	−0.75	−0.35	0.26
APE	−4.10	0.13	−0.28	−1.62	0.06	−0.13
RLPrompt	2.69	0.73	–	**1.31**	0.32	–
Variation Range	7.19	1.33	1.19	2.93	1.17	1.08

Table 2.6: Mean relative gain values for zero-shot and few-shot accuracy computed separately over models with < 6 and ≥ 6 billion parameters, and averaged by task-type. We also tabulate the total range of variation of these values in each setting.

2.5.3 Analysis

We tabulate the mean relative gain values for zero-shot and few-shot accuracies computed separately for "small" models with < 6 billion parameters and "large" models with ≥ 6 billion parameters in Table 2.6. For ease of comparison, we average the mean relative gain values thus obtained by task-type. Although the observations that PromptSource and task-agnostic instructions tend to perform the best across zero- and few-shot settings persist across model scales, we find that the ranges of variation in the few-shot mean relative gain values for large models are consistently smaller than those for small models for every task-type. This suggests that large models are able to grasp task semantics from demonstrations (when provided) while small models are more sensitive to the instruction used.

We also tabulate the mean relative gain values for zero-shot and few-shot accuracies computed separately for each model family in Table 2.7, to understand the effect that model family has on instruction performance. Although the trends we discuss in Section 2.5.1 regarding task-agnostic instructions and PromptSource instructions respectively tending to dominate few-shot and zero-shot settings persist, the instruction

Method	BLOOM				GPT Neo				LLaMA				OPT			
	CLS	MCQ	GQA	# wins	CLS	MCQ	GQA	# wins	CLS	MCQ	GQA	# wins	CLS	MCQ	GQA	# wins
Zero-shot accuracy (mean relative gain) ↑																
Null Instruction	−1.40	**3.60**	−11.93	3	−1.80	**1.34**	−10.46	1	4.02	**9.25**	−9.73	3	−1.22	**4.54**	−22.07	4
Generic Instruction	**5.03**	−1.27	−0.72	1	−0.35	−0.20	−1.86	1	−2.73	−2.09	−13.25	0	1.33	−0.47	−3.47	0
PromptSource	2.03	−2.89	**14.22**	2	1.61	−0.69	**35.75**	2	**10.01**	−3.89	7.82	2	**2.16**	−2.70	**18.70**	4
Ad hoc	0.45	−1.15	2.93	0	**2.23**	0.56	0.08	2	−3.70	−2.94	−2.56	0	1.35	−2.23	−1.26	0
Low Perplexity	−3.76	−2.19	3.94	1	−2.85	0.45	−20.87	1	4.97	−4.87	5.10	2	2.09	−1.50	4.32	1
APE	−6.10	0.75	−8.44	0	0.14	−2.00	−2.87	1	−7.01	−0.09	**12.62**	2	−5.18	−0.92	3.78	0
RLPrompt	3.76	3.15	–	2	1.02	0.54	–	1	−5.57	4.64	–	0	−0.53	3.28	–	0
Few-shot accuracy (mean relative gain) ↑																
Null Instruction	3.04	**1.11**	**0.85**	4	**1.31**	**0.33**	−0.11	2	**1.40**	0.79	−0.35	4	1.67	**0.70**	0.11	1
Generic Instruction	**3.64**	−0.24	−0.44	2	1.27	0.31	0.04	2	0.24	−0.11	0.23	2	1.89	−0.17	**0.64**	3
PromptSource	−1.21	−0.83	0.19	1	−0.44	−0.17	−0.08	2	−0.30	−0.65	0.18	0	−0.79	−0.34	0.20	1
Ad hoc	−0.35	−0.57	0.41	1	−1.02	−0.12	−0.59	0	−1.26	−0.62	**0.31**	0	−0.20	−0.33	−0.77	0
Low Perplexity	−3.00	−0.58	−0.23	0	−1.04	−0.15	−0.15	0	−0.94	−0.59	0.25	1	−1.37	−0.38	0.07	1
APE	−5.26	0.29	−0.78	0	−1.14	−0.29	**0.85**	2	0.29	0.38	−0.62	1	−3.64	0.05	−0.24	0
RLPrompt	3.14	0.82	–	1	1.06	0.10	–	1	0.57	**0.80**	–	1	**2.45**	0.47	–	3

Table 2.7: Mean relative gain values for zero-shot accuracy and few-shot accuracy computed separately over individual model families and averaged by task-type. Positive values represent above-average performance, and negative values represent below-average performance. We also tabulate the number of tasks where a method achieved highest aggregated performance in the '# wins' column under every model family.

selection method that emerges the best-performing alternative often changes on varying the choice of model family and task-type. For instance, the automatic instruction induction methods APE and RLPrompt do show above-average performance for certain model families and task-types, but this behavior does not consistently extend to other families and types. This indicates a lack of generalizability in these methods.

2.6 Discussion

Our findings reveal that in practical in-context learning settings, simpler prompting methods, such as task-agnostic or expert manually written instructions, often outperform automatically synthesized ones at the model scales we consider. Task-agnostic methods show strong performance in few-shot settings, whereas expert manual instructions appear crucial for achieving good zero-shot accuracy. The superiority of these straightforward methods over automatically induced instructions, which are often not competitive even with simple baselines, suggests a lack of transferability and generalizability among automatic induction methods. The competitive performance

of automatic induction methods like APE and RLPrompt as reported by their authors implies either a limitation in their generalizability to a broader range of models and tasks, or the need for substantial hyperparameter tuning to get them to work well across models and tasks.

Our findings suggest that ICL practitioners may often be better off forgoing computationally expensive instruction induction or selection methods in favor of task-agnostic or manually written instructions, which seem to generalize better. Interestingly, we also find that methods that excel for one model and task do not necessarily also perform well for other tasks and models. Consequently, ICL practitioners may be forced to experiment with various instruction selection methods on a model- and task-specific basis in a manner reminiscent of hyperparameter tuning to find the best choice.

On the other hand, since few-shot ICL performance remains largely consistent regardless of the choice of instruction, practitioners could perhaps benefit from simply providing a few in-context demonstrations when available. The fact that null instructions tend to outperform all other methods in our study in few-shot settings suggests that it can be challenging to find instructions that reliably inform diverse models about task semantics. When models fail to grasp the semantics signaled by instructions, these may simply serve as a source of noise, hence impairing ICL performance.

Our findings underscore a broader issue regarding the inconsistent and often insufficient evaluation of instruction selection and induction techniques. We call for more comprehensive evaluations in this space and encourage the use of our evaluation suite to facilitate this process.

Chapter 3

Membership-inference Attacks for Black Box Large Language Models

3.1 Introduction

As the scale of language model (LM) training corpora has grown, model developers (e.g, GPT-4 [9] and LLaMA 2 [82]) have become reluctant to disclose the full composition or sources of their data. This lack of transparency poses critical challenges to scientific model evaluation and ethical deployment. Critical private information may be exposed during pretraining; previous work showed that LLMs generated excerpts from copyrighted books [13] and personal emails [57], potentially infringing upon the legal rights of original content creators and violating their privacy. Additionally, [67, 47, 58] showed that the pretraining corpus may inadvertently include benchmark evaluation data, making it difficult to assess the effectiveness of these models.

In the second part of this thesis, we study the pretraining data detection problem: given a piece of text and black-box access to an LLM with no knowledge of its pretraining data, can we determine if the model was pretrained on the text? We present a benchmark, WikiMIA, and an approach, Min-k% Prob, for pretraining data de-

tection. This problem is an instance of Membership Inference Attacks (MIAs), which was initially proposed by [72]. Recent work has studied *fine-tuning* data detection [74, 70, 48] as an MIA problem. However, adopting these methods to detect the pertaining data of contemporary large LLMs presents two unique technical challenges: First, unlike fine-tuning which usually runs for multiple epochs, pretraining uses a much larger dataset but exposes each instance only once, significantly reducing the potential memorization required for successful MIAs [37, 35]. Besides, previous methods often rely on one or more reference models [11, 85] trained in the same manner as the target model (e.g., on the shadow data sampled from the same underlying pretraining data distribution) to achieve precise detection. This is not possible for large language models, as the training distribution is usually not available and training would be too expensive.

Our first step towards addressing these challenges is to establish a reliable benchmark. We introduce WikiMIA, a dynamic benchmark designed to periodically and automatically evaluate detection methods on any newly released pretrained LLMs. By leveraging the Wikipedia data timestamp and the model release date, we select old Wikipedia event data as our member data (i.e, *seen* data during pretraining) and recent Wikipedia event data (e.g., after 2023) as our non-member data (*unseen*). Our datasets thus exhibit three desirable properties: (1) **Accurate**: events that occur after LLM pretraining are guaranteed not to be present in the pretraining data. The temporal nature of events ensures that non-member data is indeed unseen and not mentioned in the pretraining data. (2) **General**: our benchmark is not confined to any specific model and can be applied to various models pretrained using Wikipedia (e.g., OPT, LLaMA, GPT-Neo) since Wikipedia is a commonly used pretraining data source. (3) **Dynamic**: we will continually update our benchmark by gathering newer non-member data (i.e., more recent events) from Wikipedia since our data construction pipeline is fully automated.

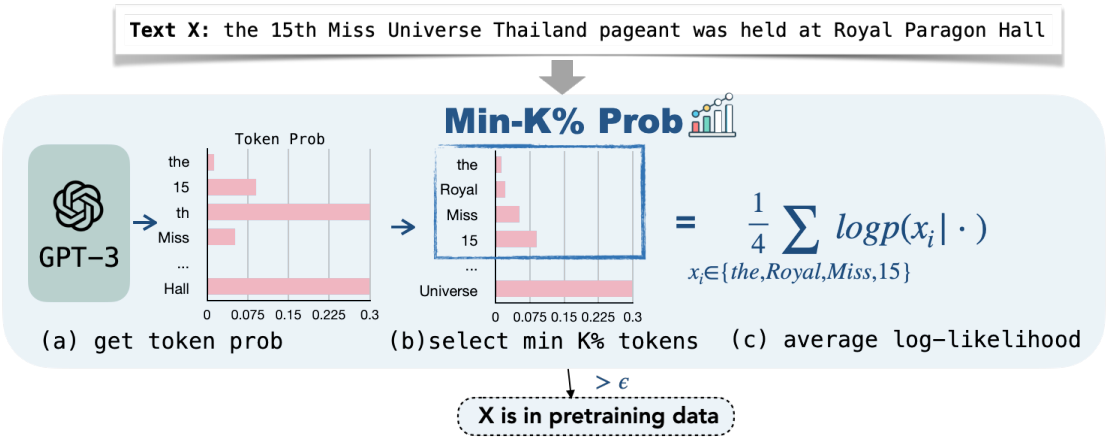


Figure 3.1: **Overview of Min-k% Prob**. To determine whether a text X is in the pretraining data of a LLM such as GPT, Min-k% Prob first gets the probability for each token in X, selects the k% tokens with minimum probabilities and calculates their average log likelihood. If the average log likelihood is high, the text is likely in the pretraining data.

MIA methods for finetuning [11, 85] usually calibrate the target model probabilities of an example using a shadow reference model that is trained on a similar data distribution. However, these approaches are impractical for pretraining data detection due to the black-box nature of pretraining data and its high computational cost. Therefore, we propose a reference-free MIA method Min-k% Prob. Our method is based on a simple hypothesis: an unseen example tends to contain a few outlier words with low probabilities, whereas a seen example is less likely to contain words with such low probabilities. Min-k% Prob computes the average probabilities of outlier tokens. Min-k% Prob can be applied without any knowledge about the pretrainig corpus or any additional training, departing from existing MIA methods, which rely on shadow reference models [50, 12]. Our experiments demonstrate that Min-k% Prob outperforms the existing strongest baseline by 7.4% in AUC score on WikiMIA. Further analysis suggests that the detection performance correlates positively with the *model size* and *detecting text length*.

To verify the applicability of our proposed method in real-world settings, we perform two case studies: copyrighted book detection (§3.7) and dataset contamination detection (§3.8). We find that Min-k% Prob significantly outperforms baseline

methods in both scenarios. From our experiments on copyrighted book detection, we see strong evidence that GPT-3 [1] is pretrained on copyrighted books from the Books3 dataset [22, 51]. Furthermore, our controlled study on dataset contamination detection sheds light on the impact of pretraining design choices on detection difficulty; we find detection becomes harder when training data sizes increase, and occurrence frequency of the detecting example and learning rates decreases.

3.2 Problem: Membership Inference Attacks

We follow the standard definition of the membership inference attack (MIA) by [72, 50]. Given a language model f_θ and its associated pretraining data $\mathcal{D} = \{z_i\}_{i\in[n]}$ sampled from an underlying distribution $\mathbb{D}$, the task objective is to learn a detector h that can infer the membership of an arbitrary data point x: $h(x, f_\theta) \to \{0, 1\}$. We follow the standard setup of MIA, assuming that the detector has access to the LM only as a black box, and can compute token probabilities for any data point x.

Challenge 1: Unavailability of the pretraining data distribution. Existing state-of-art MIA methods for data detection during finetuning [42, 85, 52] typically use reference models g_γ to compute the background difficulty of the data point and to calibrate the output probability of the target language model : $h(x, f_\theta, g_\gamma) \to \{0, 1\}$. Such reference models usually share the same model architecture as f_θ and are trained on shadow data $D_{\text{shadow}} \subset \mathbb{D}$ [11, 85], which are sampled from the same underlying distribution $\mathbb{D}$. These approaches assume that the detector can access (1) the distribution of the target model's training data, and (2) a sufficient number of samples from $\mathbb{D}$ to train a calibration model.

However, this assumption of accessing the distribution of pretraining training data is not realistic because such information is not always available (e.g., not released

[1] `text-davinci-003`.

by model developers [82, 62]). Even if access were possible, pretraining a reference model on it would be extremely computationally expensive given the incredible scale of pretraining data. In summary, the pretraining data detection problem aligns with the MIA definition but includes an assumption that the detector has no access to pretraining data distribution $\mathbb{D}$.

Challenge 2: Detection difficulty. Pretraining and finetuning differ significantly in the amount of data and compute used, as well as in optimization setups like training epochs and learning rate schedules. These factors significantly impact detection difficulty. One might intuitively deduce that detection becomes harder when dataset sizes increase, and the training epochs and learning rates decrease. We briefly describe some theoretical evidence that inform these intuitions in the following and show empirical results that support these hypotheses in §3.8.

To illustrate, given an example $z \in D$, we denote the model output as $f_\theta(z)$ Now, take another example y sampled from $\mathbb{D} \setminus D$ (not part of the pretraining data). Determining whether an example x was part of the training set becomes challenging if the outputs $f_\theta(z)$ and $f_\theta(y)$ are similar. The degree of similarity between $f_\theta(z)$ and $f_\theta(y)$ can be quantified using the total variation distance. According to previous research [27, 4], the bound on this total variation distance between $f_\theta(z)$ and $f_\theta(y)$ is directly proportional to the *occurrence frequency of the example* x, *learning rates*, and the *inverse of dataset size*, which implies the detection difficulty correlates with these factors as well.

3.3 Related Work

3.3.1 Membership inference attacks in NLP

Membership Inference Attacks (MIAs) aim to determine whether an arbitrary sample is part of a given model's training data [73, 94]. These attacks pose substantial privacy risks to individuals and often serve as a basis for more severe attacks, such as data reconstruction [12, 26, 18]. Due to its fundamental association with privacy risk, MIA has more recently found applications in quantifying privacy vulnerabilities within machine learning models and in verifying the accurate implementation of privacy-preserving mechanisms [33, 32, 96, 60, 30, 59, 76]. Initially applied to tabular and computer vision data, the concept of MIA has recently expanded into the realm of language-oriented tasks. However, this expansion has predominantly centered around finetuning data detection [74, 70, 48, 31, 53]. Our work focuses on the application of MIA to pretraining data detection, an area that has received limited attention in previous research efforts.

3.3.2 Dataset contamination

The dataset contamination issue in LMs has gained attention recently since benchmark evaluation is undermined if evaluation examples are accidentally seen during pre-training. [10], [87], and [20] consider an example contaminated if there is a 13-gram collision between the training data and evaluation example. [15] further improves this by deeming an example contaminated if 70% of its 8-grams appear in the training data. [82] builds on these methods by extending the framework to tokenized inputs and judging a token to be contaminated if it appears in any token n-gram longer than 10 tokens. However, their methods require access to retraining corpora, which is largely unavailable for recent model releases. Other approaches try to detect contamination without access to pretraining corpora. [67] simply prompts

ChatGPT to generate examples from a dataset by providing the dataset's name and split. They found that the models generate verbatim instances from NLP datasets. [23] extends this framework to extract more memorized instances by incorporating partial instance content into the prompt. Similarly, [90] demonstrates the ability to extract memorized snippets from Wikipedia via prompting. While these methods study contamination in closed-sourced models, they cannot determine contamination on an instance level. [49] argues that model-developers should release training data membership testing tools accompanying their LLMs to remedy this. However, this is not yet widely practiced.

3.4 Min-K% Prob: a novel MIA for blackbox LLMs

We introduce a pretraining data detection method Min-k% Prob that leverages minimum token probabilities of a text for detection. Min-k% Prob is based on the hypothesis that a non-member example is more likely to include a few outlier words with high negative log-likelihood (or low probability), while a member example is less likely to include words with high negative log-likelihood.

Consider a sequence of tokens in a sentence, denoted as $x = x_1, x_2, ..., x_N$, the log-likelihood of a token, x_i, given its preceding tokens is calculated as $\log p(x_i|x_1, ..., x_{i-1})$. We then select the $k\%$ of tokens from x with the minimum token probability to form a set, Min-K%(x), and compute the average log-likelihood of the tokens in this set:

$$\text{Min-k\% Prob}(x) = \frac{1}{E} \sum_{x_i \in \text{Min-K\%}(x)} \log p(x_i|x_1, ..., x_{i-1}). \tag{3.1}$$

where E is the size of the Min-K%(x) set. We can detect if a piece of text was included in pretraining data simply by thresholding this Min-k% Prob result.

3.5 WikiMIA: A Dynamic Evaluation Benchmark

We construct our benchmark by using events added to Wikipedia after specific dates, treating them as non-member data since they are guaranteed not to be present in the pretraining data, which is the key idea behind our benchmark.

3.5.1 Construction

We collect recent event pages from Wikipedia.

1. We set January 1, 2023 as the cutoff date, considering events occurring post-2023 as recent events (non-member data). We used the Wikipedia API to automatically retrieve articles and applied two filtering criteria: (1) the articles must belong to the event category, and (2) the page must be created post 2023.

2. For member data, we collected articles created before 2017 because many pre-trained models, e.g., LLaMA, GPT-NeoX and OPT, were released after 2017 and incorporate Wikipedia dumps into their pretraining data.

3. Additionally, we filtered out Wikipedia pages lacking meaningful text, such as pages titled "Timeline of ..." or "List of ...". Given the limited number of events post-2023, we ultimately collected 394 recent events as our non-member data, and we randomly selected 394 events from pre-2016 Wikipedia pages as our member data.

The data construction pipeline is automated, allowing for the curation of new non-member data for future cutoff dates.

3.5.2 Benchmark setting.

In practice, LM users may need to detect texts that are paraphrased and edited, as well.

Previous studies employing MIA have exclusively focused on detecting examples that exactly match the data used during pretraining. It remains an open question whether MIA methods can be employed to identify paraphrased examples that convey the same meaning as the original. In addition to the verbatim setting (*original*), we therefore introduce a *paraphrase setting* we leverage ChatGPT[2] to paraphrase the examples and subsequently assess if the MIA metric can effectively identify semantically equivalent examples.

Moreover, previous MIA evaluations usually mix different-length data in evaluation and report a single performance metric. However, our results reveal that data length significantly impacts the difficulty of detection. Intuitively, shorter sentences are harder to detect. Consequently, different data length buckets may lead to varying rankings of MIA methods. To investigate this further, we propose a *different-length setting*: we truncate the Wikipedia event data into different lengths—32, 64, 128, 256—and separately report the MIA methods' performance for each length segment. We describe the desirable properties in Appendix B.2.

3.6 Experiments

We evaluate the performance of Min-k% Prob and baseline detection methods against LMs such as LLaMA [80], GPT-Neo [7], and Pythia [5] on WikiMIA.

3.6.1 Datasets and Metrics

Our experiments use WikiMIA of different lengths (32, 64, 128, 256), *original* and *paraphrase* settings. Following [11, 52], we evaluate the effectiveness of a detection method using the True Positive Rate (TPR) and its False Positive Rate (FPR). We plot the ROC curve to measure the trade-off between the TPR and FPR and report

[2]OpenAI. `https://chat.openai.com/chat`

the AUC score (the area under ROC curve) and TPR at low FPRs (TPR@5%FPR) as our metrics.

3.6.2 Baseline Detection Methods

We take existing reference-based and reference-free MIA methods as our baseline methods and evaluate their performance on WikiMIA. These methods only consider sentence-level probability.

Specifically, we use the *LOSS Attack* method [95], which predicts the membership of an example based on the loss of the target model when fed the example as input. In the context of LMs, this loss corresponds to perplexity of the example (***PPL***). Another method we consider is the neighborhood attack [50], which leverages probability curvature to detect membership (***Neighbor***). This approach is identical to the DetectGPT [55] method recently proposed for classifying machine-generated vs. human-written text. Finally, we compare with membership inference methods proposed in [12], including comparing the example perplexity to zlib compression entropy (***Zlib***), to the lowercased example perplexity (***Lowercase***) and to example perplexity under a smaller model pretrained on the same data (***Smaller Ref***). For the smaller reference model setting, we employ LLaMA-7B as the smaller model for LLaMA-65B and LLaMA-30B, GPT-Neo-125M for GPT-NeoX-20B, OPT-350M for OPT-66B and Pythia-70M for Pythia-2.8B.

3.6.3 Implementation Details

The key hyperparameter of Min-k% Prob is the percentage of tokens with the highest negative log-likelihood we select to form the *top-k*% set. We performed a small sweep over 10, 20, 30, 40, 50 on a held-out validation set using the LLAMA-60B model and found that $k = 20$ works best. We use this value for all experiments without further tuning. As we report the AUC score as our metric, we don't need to

determine the threshold ϵ.

3.6.4 Results

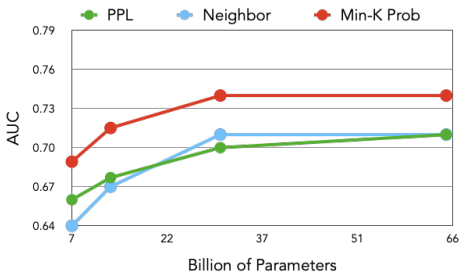
We compare MIN-K% PROB and baseline methods in Table 3.1. Our experiments show that MIN-K% PROB consistently outperforms all baseline methods across diverse target language models, both in original and paraphrase settings. MIN-K% PROB achieves an AUC score of 0.72 on average, marking a 7.4% improvement over the best baseline method (i.e., PPL). Among the baselines, the simple LOSS Attack (PPL) outperforms the others. This demonstrates the effectiveness and generalizability of MIN-K% PROB in detecting pretraining data from various LMs. Further results such as TPR@5%FPR can be found in Appendix A, which shows a trend similar to Table 3.1.

3.6.5 Analysis

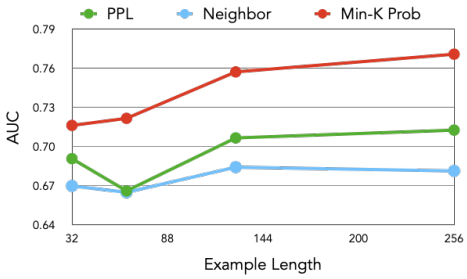
We further delve into the factors influencing detection difficulty, focusing on two aspects: (1) the size of the target model, and (2) the length of the text.

Model size. We evaluate the performance of reference-free methods on detecting pretraining 128-length texts from different-sized LLaMA models (7, 13, 30, 65B). Figure 3.2a demonstrates a noticeable trend: the AUC score of the methods rises with increasing model size. This is likely because larger models have more parameters and thus are more likely to memorize the pretraining data.

Length of text. In another experiment, we evaluate the detection method performance on examples of varying lengths in the original setting. As shown in Figure 3.2b, the AUC score of different methods increases as text length increases, likely because longer texts contain more information memorized by the target model, making them



(a) AUC score vs. model size



(b) AUC score vs. text length

Figure 3.2: As model size or text length increases, detection becomes easier.

more distinguishable from the unseen texts.

	Pythia-2.8B		**NeoX-20B**		**LLaMA-30B**		**LLaMA-65B**		**OPT-66B**		
Method	*Ori.*	*Para.*	*Ori.*	*Para.*	*Ori.*	*Para.*	*Ori.*	*Para.*	*Ori.*	*Para.*	**Avg.**
Neighbor	0.61	0.59	0.68	0.58	0.71	0.62	0.71	0.69	0.65	0.62	0.65
PPL	0.61	0.61	0.70	0.70	0.70	0.70	0.71	0.72	0.66	0.64	0.67
Zlib	0.65	0.54	0.72	0.62	0.72	0.64	0.72	0.66	0.67	0.57	0.65
Lowercase	0.59	0.60	0.68	0.67	0.59	0.54	0.63	0.60	0.59	0.58	0.61
Smaller Ref	0.60	0.58	0.68	0.65	0.72	0.64	0.74	0.70	0.67	0.64	0.66
Min-k% Prob	**0.67**	**0.66**	**0.76**	**0.74**	**0.74**	**0.73**	**0.74**	**0.74**	**0.71**	**0.69**	**0.72**

Table 3.1: AUC score for detecting pretraining examples from the given model on WikiMIA for Min-k% Prob and baselines. *Ori.* and *Para.* denote the original and paraphrase settings, respectively. **Bold** shows the best AUC within each column.

In the following two sections, we apply Min-k% Prob to real-world scenarios to detect copyrighted books and contaminated downstream tasks within LLMs.

3.7 Detecting Pretraining on Copyrighted Material

Min-k% Prob can also detect potential copyright infringement in training data, as we show in this section. Specifically, we use Min-k% Prob to detect excerpts from copyrighted books in the Books3 subset of the Pile dataset [22] that may have been included in the GPT-3[3] training data.

[3]text-davinci-003

3.7.1 Experimental Setup

Validation data to determine detection threshold. We construct a validation set using 50 books known to be memorized by ChatGPT, likely indicating their presence in its training data [13], as positive examples. For negative examples, we collected 50 new books with first editions in 2023 that could not have been in the training data. From each book, we randomly extract 100 snippets of 512 words, creating a balanced validation set of 10,000 examples. We determine the optimal classification threshold with Min-k% Prob by maximizing detection accuracy on this set.

Test data and metrics. We randomly select 100 books from the Books3 corpus that are known to contain copyrighted contents [51]. From each book, we extract 100 random 512-word snippets, creating a test set of 10,000 excerpts. We apply the threshold to decide if these books snippets have been trained with GPT-3. We then report the percentage of these snippets in each book (i.e., contamination rate) that are identified as being part of the pre-training data.

3.7.2 Results

Figure 3.3 shows Min-k% Prob achieves an AUC of 0.88, outperforming baselines in detecting copyrighted books. We apply the optimal threshold of Min-k% Prob to the test set of 10,000 snippets from 100 books from Books3. Table 3.2 represents the top 20 books with the highest predicted contamination rates. Figure 3.4 reveals nearly 90% of the books have an alarming contamination rate over 50%.

Method	Book
Neighbor	0.75
PPL	0.84
Zlib	0.81
Lowercase	0.80
Min-k% Prob	**0.88**

Figure 3.3: AUC scores for detecting the validation set of copyrighted books on GPT-3.

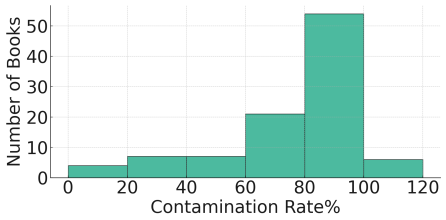


Figure 3.4: Distribution of detected contamination rate of 100 copyrighted books.

Contamination %	Book Title	Author	Year
100	The Violin of Auschwitz	Maria Àngels Anglada	2010
100	North American Stadiums	Grady Chambers	2018
100	White Chappell Scarlet Tracings	Iain Sinclair	1987
100	Lost and Found	Alan Dean	2001
100	A Different City	Tanith Lee	2015
100	Our Lady of the Forest	David Guterson	2003
100	The Expelled	Mois Benarroch	2013
99	Blood Cursed	Archer Alex	2013
99	Genesis Code: A Thriller of the Near Future	Jamie Metzl	2014
99	The Sleepwalker's Guide to Dancing	Mira Jacob	2014
99	The Harlan Ellison Hornbook	Harlan Ellison	1990
99	The Book of Freedom	Paul Selig	2018
99	Three Strong Women	Marie NDiaye	2009
99	The Leadership Mind Switch: Rethinking How We Lead in the New World of Work	D. A. Benton, Kylie Wright-Ford	2017
99	Gold	Chris Cleave	2012
99	The Tower	Simon Clark	2005
98	Amazon	Bruce Parry	2009
98	Ain't It Time We Said Goodbye: The Rolling Stones on the Road to Exile	Robert Greenfield	2014
98	Page One	David Folkenflik	2011
98	Road of Bones: The Siege of Kohima 1944	Fergal Keane	2010

Table 3.2: Top 20 copyrighted books in GPT-3's pretraining data. The listed contamination rate represents the percentage of text excerpts from each book identified in the pretraining data.

3.8 Detecting Downstream Dataset Contamination

Assessing the leakage of downstream task data into pretraining corpora is an important issue, but it is challenging to address given the lack of access to pretraining datasets. In this section, we investigate the possibility of using Min-k% Prob to detect information leakage and perform ablation studies to understand how various training factors impact detection difficulty. Specifically, we continually pretrain the 7B parameter LLaMA model [80] on pretraining data that have been purposefully contaminated with examples from the downstream task.

3.8.1 Experimental Setup

To simulate downstream task contamination that could occur in real-world settings, we create contaminated pretraining data by inserting examples from downstream tasks into a pretraining corpus. Specifically, we sample text from the RedPajama corpus [79] and insert formatted examples from the downstream datasets BoolQ [17], IMDB [46], Truthful QA [39], and Commonsense QA [78] in contiguous segments at random positions in the uncontaminated text. We insert 200 (positive) examples from each of these datasets into the pretraining data while also isolating a set of 200 (negative) examples from each dataset that are known to be absent from the contaminated corpus. This creates a contaminated pretraining dataset containing 27 million tokens with 0.1% drawn from downstream datasets.

We evaluate the effectiveness of Min-k% Prob at detecting leaked benchmark examples by computing AUC scores over these 400 examples on a LLaMA 7B model finetuned for one epoch on our contaminated pretraining data at a constant learning rate of 1e-4.

3.8.2 Results

We present the main attack results in Table 3.3. We find that Min-k% Prob outperforms all baselines. We report TPR@5%FPR in Table B.2 in Appendix A, where Min-k% Prob shows 12.2% improvement over the best baseline.

Method	BoolQ	Commonsense QA	IMDB	Truthful QA	Avg.
Neighbor	0.68	0.56	0.80	0.59	0.66
Zlib	0.76	0.63	0.71	0.63	0.68
Lowercase	0.74	0.61	0.79	0.56	0.68
PPL	0.89	0.78	0.97	0.71	0.84
Min-k% Prob	**0.91**	**0.80**	**0.98**	**0.74**	**0.86**

Table 3.3: AUC scores for detecting contaminant downstream examples. **Bold** shows the best AUC score within each column.

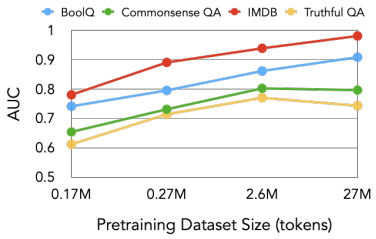
3.8.3 Analysis

The simulation with contaminated datasets allows us to perform ablation studies to empirically analyze the effects of *dataset size*, *frequency of data occurrence*, and *learning rate* on detection difficulty, as theorized in Section 3.2. The empirical results largely align with and validate the theoretical framework proposed. In summary, we find that detection becomes more challenging as data occurrence and learning rate decreases, and the effect of dataset size on detection difficulty depends on whether the contaminants are outliers relative to the distribution of the pretraining data.

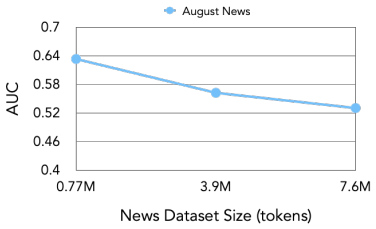
Pretraining dataset size. We construct contaminated datasets of 0.17M, 0.27M, 2.6M and 26M tokens by mixing fixed downstream examples (200 examples per downstream task) with varying amounts of RedPajama data, mimicking real-world pretraining. Despite the theory suggesting greater difficulty with more pretraining data, Figure 3.5a shows AUC scores counterintuitively increase with pre-training dataset size. This aligns with findings that LMs better memorize tail outliers [21, 98]. With more RedPajama tokens in the constructed dataset, downstream examples become more significant outliers. We hypothesize that their enhanced memorization likely enables easier detection with perplexity-based metrics.

To verify the our hypothesis, we construct control data where contaminants are not outliers. We sample Real Time Data News August 2023[4], containing post-2023 news absent from LLaMA pre-training. We create three synthetic corpora by concatenating 1000, 5000 and 10000 examples from this corpus, hence creating corpora of sizes 0.77M, 3.9M and 7.6M tokens respecitvely. In each setting, we consider 100 of these examples to be contaminant (positive) examples and set aside another set of 100 examples from News August 2023 (negative). Figure 3.5b shows AUC scores decrease as the dataset size increases.

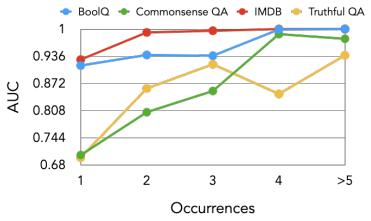
[4] `https://huggingface.co/datasets/RealTimeData/News_August_2023`



(a) Outlier contaminants, e.g., downstream examples, become easier to detect as dataset size increases.



(b) In-distribution contaminants, e.g., news articles, are harder to detect as dataset size increases.



(c) Contaminants that occur more frequently in the dataset are easier to detect.

Figure 3.5: We show the effect of contamination rate (expressed as a percentage of the total number of pretraining tokens) and occurrence frequency on the ease of detection of data contaminants using Min-k% Prob.

Detection of outlier contaminants like downstream examples gets easier as data size increases, since models effectively memorize long-tail samples. However, detecting general in-distribution samples from the pretraining data distribution gets harder with more data, following theoretical expectations.

Data occurrence. To study the relationship between detection difficulty and data occurrence, we construct a contaminated pretraining corpus by inserting multiple copies of each downstream data point into a pre-training corpus, where the occurrence of each example follows a Poisson distribution. We measure the relationship between the frequency of the example in the pretraining data and its AUC scores. Figure 3.5c shows that AUC scores positively correlates with the occurrence of examples.

Learning rate. We also study the effect of varying the learning rates used during pretraining on the detection statistics of the contaminant examples (see Table 3.4). We find that raising the learning rate from 10^{-5} to 10^{-4} increases AUC scores significantly in all the downstream tasks, implying that higher learning rates cause models to memorize their pretraining data more strongly. A more in-depth analysis in Table B.3 in Appendix A demonstrates that a higher learning rate leads to more memorization rather than generalization for these downstream tasks.

Learning rate	BoolQ	Commonsense QA	IMDB	LSAT QA	Truthful QA
1×10^{-5}	0.64	0.59	0.76	0.72	0.56
1×10^{-4}	**0.91**	**0.80**	**0.98**	**0.82**	**0.74**

Table 3.4: AUC scores for detecting contaminant downstream examples using two different learning rates. Detection becomes easier when higher learning rates are used during training. **Bold** shows the best AUC score within each column.

Chapter 4

Conclusion and Future Work

In Chapter 2 of this thesis, we conduct the broadest attempt to our knowledge, to systematically study the generalizability of popular instruction selection and induction methods for ICL in LLMs. We find that simpler approaches such as using task-agnostic instructions, expert manual instructions, or even omitting instructions entirely tend to show good performance more consistently when evaluating across a wide variety of tasks and models. Our work indicates the need for more systematic and consistent evaluations in the instruction induction space. To facilitate such analyses, we release the InstructEval suite which provides coverage over 13 diverse autoregressive LLMs and 9 tasks spanning classification, multiple-choice QA, and generative QA.

In Chapter 3, we present a pre-training data detection dataset WikiMIA and a new approach Min-k% Prob. Our approach uses the intuition that trained data tends to contain fewer outlier tokens with very low probabilities compared to other baselines. Additionally, we verify the effectiveness of our approach in real-world setting, we perform two case studiies: detecting dataset contamination and published book detection. For dataset contamination, we observe empirical results aligning with theoretical predictions about how detection difficulty changes with dataset size,

example frequency, and learning rate. Most strikingly, our book detection experiments provide strong evidence that GPT-3 models may have been trained on copyrighted books.

An exciting direction that future work could take that continues are research into instruction selection methods would be to test if our findings hold even for instruction-tuned models and models of much larger scales than the ones we evaluate. Future work could also continue our work on the pretraining data detection problem by evaluating novel detection proposals using WikiMIA. It may be interesting to explore if Min-k% Prob performance can be improved by dynamically choosing the k hyperparameter as a function of the input sequence. We hope our methodology and findings will help inform future research into optimizing in-context learning and ensuring data transparency in contemporary and future generations of large language models.

Appendix A

Supplementary material for Chapter 2

A.1 Implementation details

A.1.1 Evaluation

We ameliorate the effect of statistical noise by rerunning each instruction selection/induction method we study using 5 random seeds independently for every task (and for every model, where applicable) and report results for each instruction selection/induction method by averaging the aggregated scores associated with all 5 instructions.

We use $K = 6$ demonstrations randomly sampled from the task's training set for every experiment we perform in the few-shot setting.

To maintain consistency, we perform all our experiments using fixed task-specific prompt templates. Each prompt begins with the instruction being tested, and continues into a sequence annotated demonstrations and a test example, each of which follow the templates listed in Table A.1.

Tasks	Demonstration template
AG News [102]	`News:` (text)\`nCategory:` (label)
ANLI [61]	`Premise:` (premise)\`nHypothetisis:` (hypothesis)\`nRelation:` (label)
BoolQ [16]	`Passage:` (passage)\`nQuestion:` (question) \`nAnswer:` (label)
IMDB [45]	`Review:` (text)\`nSentiment:` (label)
TweetEval Emotion [56]	`Tweet:` (text)\`nEmotion:` (label)
CosmosQA [29]	`Passage:` (context)\`nQuestion:` (question)\`nAnswer:` (answer)
HellaSwag [97]	`Sentence:` (ctx)\`nAnswer:` (answer)
NQ-Open [36]	`Question:` (question)\`nAnswer:` (answer)
TriviaQA [34]	`Question:` (question)\`nAnswer:` (answer)

Table A.1: Tasks included in our evaluation suite, and the demonstrations templates we use for each task.

Method	Example instruction
Null instruction	(empty string)
Generic instruction	`Solve the following task:`
PromptSource [3]	`What label best describes this news article?`
Ad hoc	`Which newspaper section is most likely to feature this news article?`
Low Perplexity [24]	`Which part of a newspaper do you think this article belongs to? World News, Sports, Business or Science and Technology?`
APE [104]	`classify each input into one of the following categories: World, U.S., Business, Sci/Tech, or Sports.`
RLPrompt [19]	`Tools undergradCam firmwareCam`

Table A.2: Example instructions obtained using each method for the AG News task

A.1.2 Instruction selection methods

PromptSource We sample and evaluate a random subset of instructions from those included in the public PromptSource repository for each task in our evaluation suite.

Ad hoc We obtain the set of ad hoc instructions we evaluate for a task by tasking ChatGPT with generating 40 paraphrases of instructions for the task that we obtain from PromptSource. We then select a random sample of instructions from this 40-instruction pool and perform evaluations using each sampled instruction.

Low Perplexity For each task, we rerank a pool of ChatGPT paraphrases of PromptSource instructions using the SPELL algorithm described by [24]. When prompting a specific model, we choose the instruction with the lowest perplexity as measured by that model.

Generic Instructions
Solve the following task:
Find the answer below:
Complete the problem.
Find the best solution to the question below:
Complete the question below:

Table A.3: Sample generic instructions

APE We use the official repository released by [104] to generate instructions for each of the tasks we consider. To remain consistent with the original methodology, we use the OpenAI DaVinci to induce and evaluate instructions during the induction phase. We opt to use the simpler version of the methodology proposed by the authors since they report that the computationally intensive Monte-Carlo search strategy only provides marginal improvements in accuracy.

RLPrompt We use the public repository released by [19] to induce instructions for the RLPrompt baseline in our evaluations. Although the original work only performs evaluations over classification datasets with a fixed label-space, we augment the codebase to allow instruction induction for MCQ tasks as well by formulating these as cloze-style completion tasks. We create instructions for all tasks using the default settings of hyperparameters included with the codebase.

Task-agnostic We completely omit instructions from the prompt when evaluating null instructions. We list the set of generic instructions we evaluate in Table A.3.

We include examples of the instructions we obtain for each method in Table A.2.

A.2 Drawbacks of aggregation techniques used in previous work

Some previous works like the HELM [38] benchmark also face similar challenges when attempting to compare high-dimensional vectors – each representing a model evalu-

ated over a variety of tasks – against each other. HELM resorts to scoring models using head-to-head win rates. The win rate associated with a model indicates the fraction of head-to-head comparisons between the given model and all other models, across all scenarios, where the given model performs better along a specific metric. A notable disadvantage of this scoring technique is that it obscures the magnitude of variation in the metric associated with each test model and only conveys ordinal information about the relative performances of each model. This characteristic of head-to-head win rates makes them unsuitable for spotting broad trends across families of prompting methods.

In other works like BIG-bench [75], raw metric scores representing task performance are normalized to vary from a range of 0-100 such that a normalized score of 0 corresponds to poor performance, while a normalized score of 100 corresponds to excellent performance on the task. This is done in an attempt to be able to compare the performance of a model across a variety of tasks of varying difficulty such that the normalization proves more forgiving on difficult tasks. While this score does capture cardinal information associated with the underlying variable, it relies on the knowledge of human experts to determine raw score thresholds that constitute poor or excellent performance along a given metric. To apply such a normalization scheme in our case, one would need access to a large array of such threshold scores corresponding to each model scale, task, and metric we consider. Obtaining such threshold scores across all our settings is challenging given the number of tests we perform and the variety of metrics we consider. Hence, this type of normalization proves infeasible in our case.

A.3 Full results

We make our entire array of unaggregated evalutation results available along with the InstructEval codebase at `https://github.com/princeton-nlp/InstructEval`.

Appendix B

Supplementary material for Chapter 3

B.1 Additional Results

	Pythia-2.8B		**NeoX-20B**		**LLaMA-30B**		**LLaMA-65B**		**OPT-66B**		
Method	*Ori.*	*Para.*	*Ori.*	*Para.*	*Ori.*	*Para.*	*Ori.*	*Para.*	*Ori.*	*Para.*	**Avg.**
Neighbor	10.2	16.2	15.2	19.3	20.1	17.2	17.2	20.0	17.3	18.8	17.2
PPL	9.4	18.0	17.3	24.9	23.7	18.7	16.5	23.0	20.9	20.1	19.3
Zlib	18.7	18.7	20.3	22.1	18.0	20.9	23.0	23.0	21.6	20.1	20.6
Lowercase	10.8	7.2	12.9	12.2	10.1	6.5	14.4	12.2	14.4	8.6	10.9
Smaller Ref	10.1	10.1	15.8	10.1	10.8	11.5	15.8	21.6	15.8	10.1	13.2
Min-k% Prob	13.7	15.1	21.6	27.3	22.3	25.9	20.9	30.9	21.6	23.0	22.2

Table B.1: TPR@5%FPR score for detecting pretraining examples from the given model on WikiMIA for Min-k% Prob and baselines. *Ori.* and *Para.* denote the original and paraphrase settings, respectively. **Bold** shows the best score within each column.

B.2 Details of WikiMIA

Data properties. Our WikiMIA benchmark demonstrates several desirable properties that make it suitable for evaluating methods to detect data during pretraining

Method	BoolQ	Commonsense QA	IMDB	Truthful QA	Avg.
Neighbor	19	7	41	13	20
PPL	52	24	74	17	42
Zlib	18	9	19	7	13
Lowercase	24	3	26	14	17
Min-k% Prob	55	23	83	21	46

Table B.2: TPR @ FPR=5% for detecting contaminant downstream examples using reference-based and reference-free methods. **Bold** shows the best reference-free TPR within each column.

Learning rate	BoolQ	Commonsense QA	IMDB	LSAT QA	Truthful QA	Avg.
Non-contaminant examples						
1×10^{-5}	0.68	0.47	0.89	0.22	0.28	0.51
1×10^{-4}	0.69	0.48	0.90	0.24	0.33	0.53
Contaminant examples						
1×10^{-5}	0.71	0.49	0.92	0.26	0.38	0.55
1×10^{-4}	0.81	0.60	0.89	0.35	0.56	0.64

Table B.3: Accuracy of the model finetuned in Section 3.8.1 on each non-contaminant and contaminant examples used for AUC computation for each downstream dataset. The difference in average classification accuracy of contaminant examples over that of non-contaminant examples is 0.04 at a learning rate of 1×10^{-5} and 0.11 at a learning rate of 1×10^{-4}. This indicates that memorization becomes a significantly more pronounced effect than generalization at larger learning rates.

on any newly released models.

(1) *Accurate:* Since non-member data consists of events that occurred after the LM pretraining, there is a guarantee that this data was not present during pretraining, ensuring the accuracy of our dataset. We consider Wikipedia event data because of its time sensitivity. A recent non-event Wikipedia page may be only a recent version of an older page that was already present during the model's pretraining, and thus it may not serve as true non-member data. For example, a Wikipedia page created after 2023 about a historical figure or a well-known concept could contain substantial text already mentioned in the pretraining corpus.

(2) *General:* Our benchmark is designed to be widely applicable across different models pretrained on Wikipedia, a commonly used source of pretraining data. This includes models like OPT [99], LLaMA [80, 82], GPT-Neo [7], and Pythia [5], thereby

Table B.4: Input template we use to prompt GPT-4 to obtain the similarity score.

System	You are a helpful assistant in evaluating the similarity between two outputs generated by two different AI chatbots. Your goal is to rate the similarity between the two outputs based on a scale of 1 to 5, with 1 being highly dissimilar and 5 being highly similar.
User	Rate the similarity between Output (a) and Output (b) on a scale of 1 to 5, where 1 indicates high dissimilarity, and 5 indicates high similarity. Here are some rules of the evaluation: (1) Consider how closely Output (a) matches Output (b) in terms of content, context, and relevance. (2) Do not provide a rating outside the 1 to 5 scale, and avoid giving a rating of 3 (neutral) whenever possible. (3) Your judgment should be as objective as possible, without being influenced by any potential bias. You should answer 'Score: ', followed by an integer rating between 1 to 5, where 1 indicates high dissimilarity, and 5 indicates high similarity. You should then output 'Reason: ' and provide a short sentence to explain your rating. Output (a): Output (b):

ensuring the benchmark's generalizability across various models.

(3) *Dynamic:* Our benchmark will be continually updated by incorporating the latest non-member data, such as recent events from Wikipedia. This consistent renewal ensures that the benchmark's non-member data is always up-to-date and can be used to evaluate MIA for any newly introduced pretrained models.

Bibliography

[1] S. Agrawal, C. Zhou, M. Lewis, L. Zettlemoyer, and M. Ghazvininejad. In-context examples selection for machine translation, 2022.

[2] A. Ajith, C. Pan, M. Xia, A. Deshpande, and K. Narasimhan. Instructeval: Systematic evaluation of instruction selection methods, 2023.

[3] S. H. Bach, V. Sanh, Z.-X. Yong, A. Webson, C. Raffel, N. V. Nayak, A. Sharma, T. Kim, M. S. Bari, T. Fevry, Z. Alyafeai, M. Dey, A. Santilli, Z. Sun, S. Ben-David, C. Xu, G. Chhablani, H. Wang, J. A. Fries, M. S. Al-shaibani, S. Sharma, U. Thakker, K. Almubarak, X. Tang, X. Tang, M. T.-J. Jiang, and A. M. Rush. Promptsource: An integrated development environment and repository for natural language prompts, 2022.

[4] R. Bassily, V. Feldman, C. Guzmán, and K. Talwar. Stability of stochastic gradient descent on nonsmooth convex losses. *Advances in Neural Information Processing Systems*, 33:4381–4391, 2020.

[5] S. Biderman, H. Schoelkopf, Q. Anthony, H. Bradley, K. O'Brien, E. Hallahan, M. A. Khan, S. Purohit, U. S. Prashanth, E. Raff, A. Skowron, L. Sutawika, and O. van der Wal. Pythia: A suite for analyzing large language models across training and scaling, 2023.

[6] S. Black, S. Biderman, E. Hallahan, Q. Anthony, L. Gao, L. Golding, H. He, C. Leahy, K. McDonell, J. Phang, M. Pieler, U. S. Prashanth, S. Purohit,

L. Reynolds, J. Tow, B. Wang, and S. Weinbach. Gpt-neox-20b: An open-source autoregressive language model, 2022.

[7] S. Black, S. Biderman, E. Hallahan, Q. Anthony, L. Gao, L. Golding, H. He, C. Leahy, K. McDonell, J. Phang, M. Pieler, U. S. Prashanth, S. Purohit, L. Reynolds, J. Tow, B. Wang, and S. Weinbach. GPT-NeoX-20B: An open-source autoregressive language model. In *Proceedings of the ACL Workshop on Challenges & Perspectives in Creating Large Language Models*, 2022.

[8] S. Black, G. Leo, P. Wang, C. Leahy, and S. Biderman. GPT-Neo: Large Scale Autoregressive Language Modeling with Mesh-Tensorflow, Mar. 2021. If you use this software, please cite it using these metadata.

[9] T. Brown, B. Mann, N. Ryder, M. Subbiah, J. D. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell, S. Agarwal, A. Herbert-Voss, G. Krueger, T. Henighan, R. Child, A. Ramesh, D. Ziegler, J. Wu, C. Winter, C. Hesse, M. Chen, E. Sigler, M. Litwin, S. Gray, B. Chess, J. Clark, C. Berner, S. McCandlish, A. Radford, I. Sutskever, and D. Amodei. Language models are few-shot learners. In H. Larochelle, M. Ranzato, R. Hadsell, M. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems*, volume 33, pages 1877–1901. Curran Associates, Inc., 2020.

[10] T. Brown, B. Mann, N. Ryder, M. Subbiah, J. D. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell, et al. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901, 2020.

[11] N. Carlini, S. Chien, M. Nasr, S. Song, A. Terzis, and F. Tramer. Membership inference attacks from first principles. In *2022 IEEE Symposium on Security and Privacy (SP)*, pages 1897–1914. IEEE, 2022.

[12] N. Carlini, F. Tramer, E. Wallace, M. Jagielski, A. Herbert-Voss, K. Lee, A. Roberts, T. Brown, D. Song, U. Erlingsson, et al. Extracting training data from large language models. In *30th USENIX Security Symposium (USENIX Security 21)*, pages 2633–2650, 2021.

[13] K. K. Chang, M. Cramer, S. Soni, and D. Bamman. Speak, memory: An archaeology of books known to chatgpt/gpt-4. *arXiv preprint arXiv:2305.00118*, 2023.

[14] Y. Chen, C. Zhao, Z. Yu, K. McKeown, and H. He. On the relation between sensitivity and accuracy in in-context learning, 2023.

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

[16] C. Clark, K. Lee, M.-W. Chang, T. Kwiatkowski, M. Collins, and K. Toutanova. Boolq: Exploring the surprising difficulty of natural yes/no questions. In *NAACL*, 2019.

[17] C. Clark, K. Lee, M.-W. Chang, T. Kwiatkowski, M. Collins, and K. Toutanova. Boolq: Exploring the surprising difficulty of natural yes/no questions. In *NAACL*, 2019.

[18] R. Cummings, D. Desfontaines, D. Evans, R. Geambasu, M. Jagielski, Y. Huang, P. Kairouz, G. Kamath, S. Oh, O. Ohrimenko, et al. Challenges towards the next frontier in privacy. *arXiv preprint arXiv:2304.06929*, 2023.

[19] M. Deng, J. Wang, C.-P. Hsieh, Y. Wang, H. Guo, T. Shu, M. Song, E. Xing, and Z. Hu. RLPrompt: Optimizing discrete text prompts with reinforcement learning. In *Proceedings of the 2022 Conference on Empirical Methods in Natu-*

ral Language Processing, pages 3369–3391, Abu Dhabi, United Arab Emirates, Dec. 2022. Association for Computational Linguistics.

[20] N. Du, Y. Huang, A. M. Dai, S. Tong, D. Lepikhin, Y. Xu, M. Krikun, Y. Zhou, A. W. Yu, O. Firat, et al. Glam: Efficient scaling of language models with mixture-of-experts. In *International Conference on Machine Learning*, pages 5547–5569. PMLR, 2022.

[21] V. Feldman. Does learning require memorization? a short tale about a long tail. In *Proceedings of the 52nd Annual ACM SIGACT Symposium on Theory of Computing*, pages 954–959, 2020.

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

[23] S. Golchin and M. Surdeanu. Time travel in llms: Tracing data contamination in large language models. *arXiv preprint arXiv:2308.08493*, 2023.

[24] H. Gonen, S. Iyer, T. Blevins, N. A. Smith, and L. Zettlemoyer. Demystifying prompts in language models via perplexity estimation, 2022.

[25] T. Goyal, J. J. Li, and G. Durrett. News summarization and evaluation in the era of gpt-3, 2023.

[26] S. Gupta, Y. Huang, Z. Zhong, T. Gao, K. Li, and D. Chen. Recovering private text in federated learning of language models. *Advances in Neural Information Processing Systems*, 35:8130–8143, 2022.

[27] M. Hardt, B. Recht, and Y. Singer. Train faster, generalize better: Stability of stochastic gradient descent. In *International conference on machine learning*, pages 1225–1234. PMLR, 2016.

[28] O. Honovich, U. Shaham, S. R. Bowman, and O. Levy. Instruction induction: From few examples to natural language task descriptions. *arXiv preprint arXiv:2205.10782*, 2022.

[29] L. Huang, R. Le Bras, C. Bhagavatula, and Y. Choi. Cosmos QA: Machine reading comprehension with contextual commonsense reasoning. 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 2391–2401, Hong Kong, China, Nov. 2019. Association for Computational Linguistics.

[30] Y. Huang, C.-Y. Huang, X. Li, and K. Li. A dataset auditing method for collaboratively trained machine learning models. *IEEE Transactions on Medical Imaging*, 2022.

[31] A. Jagannatha, B. P. S. Rawat, and H. Yu. Membership inference attack susceptibility of clinical language models. *arXiv preprint arXiv:2104.08305*, 2021.

[32] M. Jagielski, J. Ullman, and A. Oprea. Auditing differentially private machine learning: How private is private sgd? *Advances in Neural Information Processing Systems*, 33:22205–22216, 2020.

[33] B. Jayaraman and D. Evans. Evaluating differentially private machine learning in practice. In *28th USENIX Security Symposium (USENIX Security 19)*, pages 1895–1912, 2019.

[34] M. Joshi, E. Choi, D. Weld, and L. Zettlemoyer. triviaqa: A Large Scale Distantly Supervised Challenge Dataset for Reading Comprehension. *arXiv e-prints*, page arXiv:1705.03551, 2017.

[35] N. Kandpal, E. Wallace, and C. Raffel. Deduplicating training data mitigates

privacy risks in language models. In *International Conference on Machine Learning*, pages 10697–10707. PMLR, 2022.

[36] T. Kwiatkowski, J. Palomaki, O. Redfield, M. Collins, A. Parikh, C. Alberti, D. Epstein, I. Polosukhin, J. Devlin, K. Lee, K. Toutanova, L. Jones, M. Kelcey, M.-W. Chang, A. M. Dai, J. Uszkoreit, Q. Le, and S. Petrov. Natural questions: A benchmark for question answering research. *Transactions of the Association for Computational Linguistics*, 7:453–466, 2019.

[37] K. Leino and M. Fredrikson. Stolen memories: Leveraging model memorization for calibrated {White-Box} membership inference. In *29th USENIX security symposium (USENIX Security 20)*, pages 1605–1622, 2020.

[38] P. Liang, R. Bommasani, T. Lee, D. Tsipras, D. Soylu, M. Yasunaga, Y. Zhang, D. Narayanan, Y. Wu, A. Kumar, B. Newman, B. Yuan, B. Yan, C. Zhang, C. Cosgrove, C. D. Manning, C. Ré, D. Acosta-Navas, D. A. Hudson, E. Zelikman, E. Durmus, F. Ladhak, F. Rong, H. Ren, H. Yao, J. Wang, K. Santhanam, L. Orr, L. Zheng, M. Yuksekgonul, M. Suzgun, N. Kim, N. Guha, N. Chatterji, O. Khattab, P. Henderson, Q. Huang, R. Chi, S. M. Xie, S. Santurkar, S. Ganguli, T. Hashimoto, T. Icard, T. Zhang, V. Chaudhary, W. Wang, X. Li, Y. Mai, Y. Zhang, and Y. Koreeda. Holistic evaluation of language models, 2022.

[39] S. Lin, J. Hilton, and O. Evans. Truthfulqa: Measuring how models mimic human falsehoods, 2021.

[40] J. Liu, D. Shen, Y. Zhang, B. Dolan, L. Carin, and W. Chen. What makes good in-context examples for gpt-3? In *Workshop on Knowledge Extraction and Integration for Deep Learning Architectures; Deep Learning Inside Out*, 2021.

[41] J. Liu, D. Shen, Y. Zhang, B. Dolan, L. Carin, and W. Chen. What makes good in-context examples for gpt-3?, 2021.

[42] Y. Long, V. Bindschaedler, L. Wang, D. Bu, X. Wang, H. Tang, C. A. Gunter, and K. Chen. Understanding membership inferences on well-generalized learning models. *arXiv preprint arXiv:1802.04889*, 2018.

[43] Y. Lu, M. Bartolo, A. Moore, S. Riedel, and P. Stenetorp. Fantastically ordered prompts and where to find them: Overcoming few-shot prompt order sensitivity. *arXiv preprint arXiv:2104.08786*, 2021.

[44] Y. Lu, M. Bartolo, A. Moore, S. Riedel, and P. Stenetorp. Fantastically ordered prompts and where to find them: Overcoming few-shot prompt order sensitivity. *CoRR*, abs/2104.08786, 2021.

[45] A. L. Maas, R. E. Daly, P. T. Pham, D. Huang, A. Y. Ng, and C. Potts. Learning word vectors for sentiment analysis. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 142–150, Portland, Oregon, USA, June 2011. Association for Computational Linguistics.

[46] A. L. Maas, R. E. Daly, P. T. Pham, D. Huang, A. Y. Ng, and C. Potts. Learning word vectors for sentiment analysis. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 142–150, Portland, Oregon, USA, June 2011. Association for Computational Linguistics.

[47] I. Magar and R. Schwartz. Data contamination: From memorization to exploitation. *ArXiv*, abs/2203.08242, 2022.

[48] S. Mahloujifar, H. A. Inan, M. Chase, E. Ghosh, and M. Hasegawa. Membership

inference on word embedding and beyond. *arXiv preprint arXiv:2106.11384*, 2021.

[49] M. Marone and B. Van Durme. Data portraits: Recording foundation model training data, 2023.

[50] J. Mattern, F. Mireshghallah, Z. Jin, B. Schoelkopf, M. Sachan, and T. Berg-Kirkpatrick. Membership inference attacks against language models via neighbourhood comparison. In *Findings of the Association for Computational Linguistics: ACL 2023*, pages 11330–11343, Toronto, Canada, July 2023. Association for Computational Linguistics.

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

[52] F. Mireshghallah, K. Goyal, A. Uniyal, T. Berg-Kirkpatrick, and R. Shokri. Quantifying privacy risks of masked language models using membership inference attacks. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 8332–8347, Abu Dhabi, United Arab Emirates, Dec. 2022. Association for Computational Linguistics.

[53] F. Mireshghallah, K. Goyal, A. Uniyal, T. Berg-Kirkpatrick, and R. Shokri. Quantifying privacy risks of masked language models using membership inference attacks. *arXiv preprint arXiv:2203.03929*, 2022.

[54] S. Mishra, D. Khashabi, C. Baral, and H. Hajishirzi. 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)*, pages 3470–3487, Dublin, Ireland, May 2022. Association for Computational Linguistics.

[55] E. Mitchell, Y. Lee, A. Khazatsky, C. D. Manning, and C. Finn. Detectgpt: Zero-shot machine-generated text detection using probability curvature, 2023.

[56] S. Mohammad, F. Bravo-Marquez, M. Salameh, and S. Kiritchenko. Semeval-2018 task 1: Affect in tweets. In *Proceedings of the 12th international workshop on semantic evaluation*, pages 1–17, 2018.

[57] M. Mozes, X. He, B. Kleinberg, and L. D. Griffin. Use of llms for illicit purposes: Threats, prevention measures, and vulnerabilities, 2023.

[58] A. Narayanan. Gpt-4 and professional benchmarks: the wrong answer to the wrong question, 2023.

[59] M. Nasr, J. Hayes, T. Steinke, B. Balle, F. Tramèr, M. Jagielski, N. Carlini, and A. Terzis. Tight auditing of differentially private machine learning. *arXiv preprint arXiv:2302.07956*, 2023.

[60] M. Nasr, S. Songi, A. Thakurta, N. Papernot, and N. Carlin. Adversary instantiation: Lower bounds for differentially private machine learning. In *2021 IEEE Symposium on security and privacy (SP)*, pages 866–882. IEEE, 2021.

[61] Y. Nie, A. Williams, E. Dinan, M. Bansal, J. Weston, and D. Kiela. Adversarial nli: A new benchmark for natural language understanding. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*. Association for Computational Linguistics, 2020.

[62] OpenAI. Gpt-4 technical report, 2023.

[63] L. Ouyang, J. Wu, X. Jiang, D. Almeida, C. L. Wainwright, P. Mishkin, C. Zhang, S. Agarwal, K. Slama, A. Ray, J. Schulman, J. Hilton, F. Kelton, L. Miller, M. Simens, A. Askell, P. Welinder, P. Christiano, J. Leike, and

R. Lowe. Training language models to follow instructions with human feedback, 2022.

[64] J. Pan, T. Gao, H. Chen, and D. Chen. What in-context learning "learns" in-context: Disentangling task recognition and task learning, 2023.

[65] E. Perez, D. Kiela, and K. Cho. True few-shot learning with language models. In A. Beygelzimer, Y. Dauphin, P. Liang, and J. W. Vaughan, editors, *Advances in Neural Information Processing Systems*, 2021.

[66] O. Rubin, J. Herzig, and J. Berant. Learning to retrieve prompts for in-context learning. In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 2655–2671, 2022.

[67] O. Sainz, J. Ander Campos, I. García-Ferrero, J. Etxaniz, and E. Agirre. Did chatgpt cheat on your test?, 2023.

[68] T. L. Scao, A. Fan, C. Akiki, E. Pavlick, S. Ilić, D. Hesslow, R. Castagné, A. S. Luccioni, F. Yvon, M. Gallé, J. Tow, A. M. Rush, S. Biderman, A. Webson, P. S. Ammanamanchi, T. Wang, B. Sagot, N. Muennighoff, A. V. del Moral, O. Ruwase, R. Bawden, S. Bekman, A. McMillan-Major, I. Beltagy, H. Nguyen, L. Saulnier, S. Tan, P. O. Suarez, V. Sanh, H. Laurençon, Y. Jernite, J. Launay, M. Mitchell, C. Raffel, A. Gokaslan, A. Simhi, A. Soroa, A. F. Aji, A. Alfassy, A. Rogers, A. K. Nitzav, C. Xu, C. Mou, C. Emezue, C. Klamm, C. Leong, D. van Strien, D. I. Adelani, D. Radev, E. G. Ponferrada, E. Levkovizh, E. Kim, E. B. Natan, F. De Toni, G. Dupont, G. Kruszewski, G. Pistilli, H. Elsahar, H. Benyamina, H. Tran, I. Yu, I. Abdulmumin, I. Johnson, I. Gonzalez-Dios, J. de la Rosa, J. Chim, J. Dodge, J. Zhu, J. Chang, J. Frohberg, J. Tobing, J. Bhattacharjee, K. Almubarak, K. Chen, K. Lo, L. Von Werra, L. We-

ber, L. Phan, L. B. allal, L. Tanguy, M. Dey, M. R. Muñoz, M. Masoud, M. Grandury, M. Šaško, M. Huang, M. Coavoux, M. Singh, M. T.-J. Jiang, M. C. Vu, M. A. Jauhar, M. Ghaleb, N. Subramani, N. Kassner, N. Khamis, O. Nguyen, O. Espejel, O. de Gibert, P. Villegas, P. Henderson, P. Colombo, P. Amuok, Q. Lhoest, R. Harliman, R. Bommasani, R. L. López, R. Ribeiro, S. Osei, S. Pyysalo, S. Nagel, S. Bose, S. H. Muhammad, S. Sharma, S. Longpre, S. Nikpoor, S. Silberberg, S. Pai, S. Zink, T. T. Torrent, T. Schick, T. Thrush, V. Danchev, V. Nikoulina, V. Laippala, V. Lepercq, V. Prabhu, Z. Alyafeai, Z. Talat, A. Raja, B. Heinzerling, C. Si, E. Salesky, S. J. Mielke, W. Y. Lee, A. Sharma, A. Santilli, A. Chaffin, A. Stiegler, D. Datta, E. Szczechla, G. Chhablani, H. Wang, H. Pandey, H. Strobelt, J. A. Fries, J. Rozen, L. Gao, L. Sutawika, M. S. Bari, M. S. Al-shaibani, M. Manica, N. Nayak, R. Teehan, S. Albanie, S. Shen, S. Ben-David, S. H. Bach, T. Kim, T. Bers, T. Fevry, T. Neeraj, U. Thakker, V. Raunak, X. Tang, Z.-X. Yong, Z. Sun, S. Brody, Y. Uri, H. Tojarieh, A. Roberts, H. W. Chung, J. Tae, J. Phang, O. Press, C. Li, D. Narayanan, H. Bourfoune, J. Casper, J. Rasley, M. Ryabinin, M. Mishra, M. Zhang, M. Shoeybi, M. Peyrounette, N. Patry, N. Tazi, O. Sanseviero, P. von Platen, P. Cornette, P. F. Lavallée, R. Lacroix, S. Rajbhandari, S. Gandhi, S. Smith, S. Requena, S. Patil, T. Dettmers, A. Baruwa, A. Singh, A. Cheveleva, A.-L. Ligozat, A. Subramonian, A. Névéol, C. Lovering, D. Garrette, D. Tunuguntla, E. Reiter, E. Taktasheva, E. Voloshina, E. Bogdanov, G. I. Winata, H. Schoelkopf, J.-C. Kalo, J. Novikova, J. Z. Forde, J. Clive, J. Kasai, K. Kawamura, L. Hazan, M. Carpuat, M. Clinciu, N. Kim, N. Cheng, O. Serikov, O. Antverg, O. van der Wal, R. Zhang, R. Zhang, S. Gehrmann, S. Pais, T. Shavrina, T. Scialom, T. Yun, T. Limisiewicz, V. Rieser, V. Protasov, V. Mikhailov, Y. Pruksachatkun, Y. Belinkov, Z. Bamberger, Z. Kasner, A. Rueda, A. Pestana, A. Feizpour, A. Khan, A. Faranak, A. Santos,

A. Hevia, A. Unldreaj, A. Aghagol, A. Abdollahi, A. Tammour, A. Haji-Hosseini, B. Behroozi, B. Ajibade, B. Saxena, C. M. Ferrandis, D. Contractor, D. Lansky, D. David, D. Kiela, D. A. Nguyen, E. Tan, E. Baylor, E. Ozoani, F. Mirza, F. Ononiwu, H. Rezanejad, H. Jones, I. Bhattacharya, I. Solaiman, I. Sedenko, I. Nejadgholi, J. Passmore, J. Seltzer, J. B. Sanz, K. Fort, L. Dutra, M. Samagaio, M. Elbadri, M. Mieskes, M. Gerchick, M. Akinlolu, M. McKenna, M. Qiu, M. Ghauri, M. Burynok, N. Abrar, N. Rajani, N. Elkott, N. Fahmy, O. Samuel, R. An, R. Kromann, R. Hao, S. Alizadeh, S. Shubber, S. Wang, S. Roy, S. Viguier, T. Le, T. Oyebade, T. Le, Y. Yang, Z. Nguyen, A. R. Kashyap, A. Palasciano, A. Callahan, A. Shukla, A. Miranda-Escalada, A. Singh, B. Beilharz, B. Wang, C. Brito, C. Zhou, C. Jain, C. Xu, C. Fourrier, D. L. Periñán, D. Molano, D. Yu, E. Manjavacas, F. Barth, F. Fuhrimann, G. Altay, G. Bayrak, G. Burns, H. U. Vrabec, I. Bello, I. Dash, J. Kang, J. Giorgi, J. Golde, J. D. Posada, K. R. Sivaraman, L. Bulchandani, L. Liu, L. Shinzato, M. H. de Bykhovetz, M. Takeuchi, M. Pàmies, M. A. Castillo, M. Nezhurina, M. Sänger, M. Samwald, M. Cullan, M. Weinberg, M. De Wolf, M. Mihaljcic, M. Liu, M. Freidank, M. Kang, N. Seelam, N. Dahlberg, N. M. Broad, N. Muellner, P. Fung, P. Haller, R. Chandrasekhar, R. Eisenberg, R. Martin, R. Canalli, R. Su, R. Su, S. Cahyawijaya, S. Garda, S. S. Deshmukh, S. Mishra, S. Kiblawi, S. Ott, S. Sang-aroonsiri, S. Kumar, S. Schweter, S. Bharati, T. Laud, T. Gigant, T. Kainuma, W. Kusa, Y. Labrak, Y. S. Bajaj, Y. Venkatraman, Y. Xu, Y. Xu, Y. Xu, Z. Tan, Z. Xie, Z. Ye, M. Bras, Y. Belkada, and T. Wolf. Bloom: A 176b-parameter open-access multilingual language model, 2022.

[69] T. L. Scao, A. Fan, C. Akiki, E.-J. Pavlick, S. Ili'c, D. Hesslow, R. Castagn'e, A. S. Luccioni, F. Yvon, M. Gallé, J. Tow, A. M. Rush, S. R. Biderman, A. Webson, P. S. Ammanamanchi, T. Wang, B. Sagot, N. Muennighoff, A. V. del

Moral, O. Ruwase, R. Bawden, S. Bekman, et al. Bloom: A 176b-parameter open-access multilingual language model. *ArXiv*, abs/2211.05100, 2022.

[70] V. Shejwalkar, H. A. Inan, A. Houmansadr, and R. Sim. Membership inference attacks against NLP classification models. In *NeurIPS 2021 Workshop Privacy in Machine Learning*, 2021.

[71] W. Shi, A. Ajith, M. Xia, Y. Huang, D. Liu, T. Blevins, D. Chen, and L. Zettlemoyer. Detecting pretraining data from large language models, 2024.

[72] R. Shokri, M. Stronati, C. Song, and V. Shmatikov. Membership inference attacks against machine learning models. In *2017 IEEE Symposium on Security and Privacy (SP)*, pages 3–18, 2016.

[73] R. Shokri, M. Stronati, C. Song, and V. Shmatikov. Membership inference attacks against machine learning models. In *2017 IEEE symposium on security and privacy (SP)*, pages 3–18. IEEE, 2017.

[74] C. Song and V. Shmatikov. Auditing data provenance in text-generation models. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 196–206, 2019.

[75] A. Srivastava, A. Rastogi, A. Rao, A. A. M. Shoeb, A. Abid, A. Fisch, A. R. Brown, A. Santoro, A. Gupta, A. Garriga-Alonso, et al. Beyond the imitation game: Quantifying and extrapolating the capabilities of language models. *arXiv preprint arXiv:2206.04615*, 2022.

[76] T. Steinke, M. Nasr, and M. Jagielski. Privacy auditing with one (1) training run. *arXiv preprint arXiv:2305.08846*, 2023.

[77] H. Su, J. Kasai, C. H. Wu, W. Shi, T. Wang, J. Xin, R. Zhang, M. Ostendorf,

L. Zettlemoyer, N. A. Smith, and T. Yu. Selective annotation makes language models better few-shot learners. *ArXiv*, abs/2209.01975, 2022.

[78] A. Talmor, J. Herzig, N. Lourie, and J. Berant. CommonsenseQA: A question answering challenge targeting commonsense knowledge. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4149–4158, Minneapolis, Minnesota, June 2019. Association for Computational Linguistics.

[79] TogetherCompute. Redpajama: An open source recipe to reproduce llama training dataset, 2023.

[80] H. Touvron, T. Lavril, G. Izacard, X. Martinet, M.-A. Lachaux, T. Lacroix, B. Rozière, N. Goyal, E. Hambro, F. Azhar, et al. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*, 2023.

[81] H. Touvron, T. Lavril, G. Izacard, X. Martinet, M.-A. Lachaux, T. Lacroix, B. Rozière, N. Goyal, E. Hambro, F. Azhar, A. Rodriguez, A. Joulin, E. Grave, and G. Lample. Llama: Open and efficient foundation language models, 2023.

[82] H. Touvron, L. Martin, K. Stone, P. Albert, A. Almahairi, Y. Babaei, N. Bashlykov, S. Batra, P. Bhargava, S. Bhosale, D. Bikel, L. Blecher, C. C. Ferrer, M. Chen, G. Cucurull, D. Esiobu, J. Fernandes, J. Fu, W. Fu, B. Fuller, C. Gao, V. Goswami, N. Goyal, A. Hartshorn, S. Hosseini, R. Hou, H. Inan, M. Kardas, V. Kerkez, M. Khabsa, I. Kloumann, A. Korenev, P. S. Koura, M.-A. Lachaux, T. Lavril, J. Lee, D. Liskovich, Y. Lu, Y. Mao, X. Martinet, T. Mihaylov, P. Mishra, I. Molybog, Y. Nie, A. Poulton, J. Reizenstein, R. Rungta, K. Saladi, A. Schelten, R. Silva, E. M. Smith, R. Subramanian, X. E. Tan, B. Tang, R. Taylor, A. Williams, J. X. Kuan, P. Xu, Z. Yan, I. Zarov, Y. Zhang, A. Fan,

M. Kambadur, S. Narang, A. Rodriguez, R. Stojnic, S. Edunov, and T. Scialom. Llama 2: Open foundation and fine-tuned chat models, 2023.

[83] X. Wang, J. Wei, D. Schuurmans, Q. Le, E. Chi, S. Narang, A. Chowdhery, and D. Zhou. Self-consistency improves chain of thought reasoning in language models, 2023.

[84] X. Wang, W. Zhu, and W. Y. Wang. Large language models are implicitly topic models: Explaining and finding good demonstrations for in-context learning. *ArXiv*, abs/2301.11916, 2023.

[85] L. Watson, C. Guo, G. Cormode, and A. Sablayrolles. On the importance of difficulty calibration in membership inference attacks. In *International Conference on Learning Representations*, 2022.

[86] A. Webson and E. Pavlick. 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, July 2022. Association for Computational Linguistics.

[87] J. Wei, M. Bosma, V. Zhao, K. Guu, A. W. Yu, B. Lester, N. Du, A. M. Dai, and Q. V. Le. Finetuned language models are zero-shot learners. In *International Conference on Learning Representations*, 2022.

[88] J. Wei, X. Wang, D. Schuurmans, M. Bosma, E. H. hsin Chi, F. Xia, Q. Le, and D. Zhou. Chain of thought prompting elicits reasoning in large language models. *ArXiv*, abs/2201.11903, 2022.

[89] J. W. Wei, J. Wei, Y. Tay, D. Tran, A. Webson, Y. Lu, X. Chen, H. Liu, D. Huang, D. Zhou, and T. Ma. Larger language models do in-context learning differently. *ArXiv*, abs/2303.03846, 2023.

[90] O. Weller, M. Marone, N. Weir, D. Lawrie, D. Khashabi, and B. V. Durme. "according to ..." prompting language models improves quoting from pre-training data, 2023.

[91] B. Xu, Q. Wang, Z. Mao, Y. Lyu, Q. She, and Y. Zhang. $k$nn prompting: Beyond-context learning with calibration-free nearest neighbor inference, 2023.

[92] S. Yang, J. Kim, J. Jang, S. Ye, H. Lee, and M. Seo. Improving probability-based prompt selection through unified evaluation and analysis, 2023.

[93] S. Yao, J. Zhao, D. Yu, N. Du, I. Shafran, K. Narasimhan, and Y. Cao. React: Synergizing reasoning and acting in language models, 2023.

[94] S. Yeom, I. Giacomelli, M. Fredrikson, and S. Jha. Privacy risk in machine learning: Analyzing the connection to overfitting. In *2018 IEEE 31st computer security foundations symposium (CSF)*, pages 268–282. IEEE, 2018.

[95] S. Yeom, I. Giacomelli, M. Fredrikson, and S. Jha. Privacy risk in machine learning: Analyzing the connection to overfitting. In *2018 IEEE 31st Computer Security Foundations Symposium (CSF)*, pages 268–282, 2018.

[96] S. Zanella-Béguelin, L. Wutschitz, S. Tople, V. Rühle, A. Paverd, O. Ohrimenko, B. Köpf, and M. Brockschmidt. Analyzing information leakage of updates to natural language models. In *Proceedings of the 2020 ACM SIGSAC conference on computer and communications security*, pages 363–375, 2020.

[97] R. Zellers, A. Holtzman, Y. Bisk, A. Farhadi, and Y. Choi. Hellaswag: Can a machine really finish your sentence? In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, 2019.

[98] C. Zhang, D. Ippolito, K. Lee, M. Jagielski, F. Tramèr, and N. Carlini.

Counterfactual memorization in neural language models. *arXiv preprint arXiv:2112.12938*, 2021.

[99] S. Zhang, S. Roller, N. Goyal, M. Artetxe, M. Chen, S. Chen, C. Dewan, M. Diab, X. Li, X. V. Lin, et al. Opt: Open pre-trained transformer language models. *arXiv preprint arXiv:2205.01068*, 2022.

[100] S. Zhang, S. Roller, N. Goyal, M. Artetxe, M. Chen, S. Chen, C. Dewan, M. Diab, X. Li, X. V. Lin, T. Mihaylov, M. Ott, S. Shleifer, K. Shuster, D. Simig, P. S. Koura, A. Sridhar, T. Wang, and L. Zettlemoyer. Opt: Open pre-trained transformer language models. *ArXiv*, abs/2205.01068, 2022.

[101] S. Zhang, S. Roller, N. Goyal, M. Artetxe, M. Chen, S. Chen, C. Dewan, M. Diab, X. Li, X. V. Lin, T. Mihaylov, M. Ott, S. Shleifer, K. Shuster, D. Simig, P. S. Koura, A. Sridhar, T. Wang, and L. Zettlemoyer. Opt: Open pre-trained transformer language models, 2022.

[102] X. Zhang, J. J. Zhao, and Y. LeCun. Character-level convolutional networks for text classification. In *NIPS*, 2015.

[103] T. Z. Zhao, E. Wallace, S. Feng, D. Klein, and S. Singh. Calibrate before use: Improving few-shot performance of language models, 2021.

[104] Y. Zhou, A. I. Muresanu, Z. Han, K. Paster, S. Pitis, H. Chan, and J. Ba. Large language models are human-level prompt engineers. *ArXiv*, abs/2211.01910, 2022.