Informatics

Identifying Data Contamination in LLMs for Mathematical Benchmarks

DIPLOMARBEIT

zur Erlangung des akademischen Grades

Diplom-Ingenieur

im Rahmen des Studiums

Data Science

eingereicht von

Tobias Gallus Mathis, BSc
Matrikelnummer 01621473

an der Fakultät für Informatik

der Technischen Universität Wien

Betreuung: Univ.Prof. Dipl.-Inf. Dr.rer.nat. Thomas Lukasiewicz,
Mitwirkung: Simon Frieder, MSc, MPhil

Wien, 1. Mai 2025

Tobias Gallus Mathis	Thomas Lukasiewicz

Informatics

Identifying Data Contamination in LLMs for Mathematical Benchmarks

DIPLOMA THESIS

submitted in partial fulfillment of the requirements for the degree of

Diplom-Ingenieur

in

Data Science

by

Tobias Gallus Mathis, BSc
Registration Number 01621473

to the Faculty of Informatics

at the TU Wien

Advisor: Univ.Prof. Dipl.-Inf. Dr.rer.nat. Thomas Lukasiewicz,
Assistance: Simon Frieder, MSc, MPhil

Vienna, May 1, 2025

Tobias Gallus Mathis	Thomas Lukasiewicz

Erklärung zur Verfassung der Arbeit

Tobias Gallus Mathis, BSc

Hiermit erkläre ich, dass ich diese Arbeit selbständig verfasst habe, dass ich die verwendeten Quellen und Hilfsmittel vollständig angegeben habe und dass ich die Stellen der Arbeit – einschließlich Tabellen, Karten und Abbildungen –, die anderen Werken oder dem Internet im Wortlaut oder dem Sinn nach entnommen sind, auf jeden Fall unter Angabe der Quelle als Entlehnung kenntlich gemacht habe.
Ich erkläre weiters, dass ich mich generativer KI-Tools lediglich als Hilfsmittel bedient habe und in der vorliegenden Arbeit mein gestalterischer Einfluss überwiegt. Im Anhang „Übersicht verwendeter Hilfsmittel“ habe ich alle generativen KI-Tools gelistet, die verwendet wurden, und angegeben, wo und wie sie verwendet wurden. Für Textpassagen, die ohne substantielle Änderungen übernommen wurden, haben ich jeweils die von mir formulierten Eingaben (Prompts) und die verwendete IT- Anwendung mit ihrem Produktnamen und Versionsnummer/Datum angegeben.

Wien, 1. Mai 2025

Tobias Gallus Mathis

Danksagung

Ich danke meinen Betreuern, Univ.Prof. Thomas Lukasiewicz und Simon Frieder, für die schnellen Rückmeldungen und die stets reibungslose Kommunikation. Besonders danken möchte ich Simon Frieder für seine wertvollen Anregungen zur Arbeit.

Weiters möchte ich meinen Eltern für ihre finanzielle und persönliche Unterstützung über all die Jahre danken.

Schließlich gilt mein besonderer Dank meiner Freundin Gloria, die mir auch in schwierigeren Zeiten immer zur Seite stand und mich auf meinem Weg bestärkt hat.

Acknowledgements

I would like to thank my supervisors, Univ.Prof. Thomas Lukasiewicz and Simon Frieder, for their prompt feedback and smooth communication. I especially thank Simon Frieder for his dedicated support and valuable advice.

I am very grateful to my parents for their financial and personal support.

Finally, I sincerely thank my girlfriend Gloria for standing by my side even during difficult times and encouraging me throughout this journey.

Kurzfassung

Große Sprachmodelle (Large Language Models, LLMs) haben in jüngerer Zeit bemerkenswerte Fortschritte erzielt, unter anderem im Lösen mathematischer Aufgaben. Unter Forschenden bestehen jedoch Bedenken, dass diese Erfolge primär auf ein Phänomen namens Datenkontamination (Data Contamination) zurückzuführen ist. Datenkontamination tritt auf, wenn LLMs während des Trainings bereits mit jenen Beispielen konfrontiert wurden, auf Basis derer sie nachträglich getestet werden. Dieses Phänomen kann die Bewertung der Modelle in solchen Tests künstlich verbessern.

Diese Arbeit widmet sich der Untersuchung dieses Phänomens durch die Einführung eines neuen mathematischen Datensatzes *MathCONTA*, der speziell für die Evaluierung bestehender Methoden zur Detektion von Datenkontamination erstellt wurde. *MathCONTA* enthält sowohl mathematische Aufgaben, von denen bekannt ist, dass sie den Modellen während des Trainings vorlagen, sowie vollkommen neue Aufgaben. Im Gegensatz zu bisherigen Arbeiten, die Kontaminationseffekte meist durch künstliches Finetuning untersuchen, ermöglicht *MathCONTA* die Untersuchung unter natürlichen Kontaminationsbedingungen.

Anhand dieses Datensatzes werden verschiedene repräsentative Detektionsmethoden, darunter min$K\%$, n-gram accuracy und CDD evaluiert. Die Ergebnisse zeigen, dass keine dieser Methoden – auch nicht deren Kombination – ausreichend zuverlässig zwischen kontaminierten und nicht kontaminierten Aufgaben unterscheiden können.

Abschließend werden die tatsächlichen Auswirkungen von Datenkontamination auf die Lösung mathematischer Aufgaben in *MathCONTA* untersucht. Es zeigt sich, dass kontaminierte Beispiele im Durchschnitt zwar häufiger korrekt gelöst werden, dieser Unterschied jedoch nicht statistisch signifikant ist. Daraus folgt, dass Datenkontamination nicht der alleinige Grund für die jüngsten Erfolge von Sprachmodellen im Bereich des mathematischen Problemlösens sein kann.

Aus Transparenzgründen wird der Datensatz und der gesamte Code einschließlich aller Experimente öffentlich zugänglich gemacht.

Abstract

Large language models (LLMs) have demonstrated impressive capabilities in mathematical reasoning tasks. However, concerns persist around data contamination, where benchmark problems used for evaluation have appeared in the model's pretraining data. Such contamination can artificially inflate performance metrics, particularly in domains where genuine reasoning must be distinguished from memorization. This thesis introduces *MathCONTA*, a novel mathematical dataset for contamination detection. It spans multiple domains—including algebra, number theory, combinatorics, and integration—and covers various difficulty levels, from simple word problems to advanced math contest problems.

MathCONTA consists of two balanced subsets: one containing problems known to have been seen during pretraining (contaminated) and another with entirely novel problems (uncontaminated). In contrast to previous studies that simulate contamination via finetuning, *MathCONTA* reflects how contamination arises naturally during pretraining, offering a more realistic basis for evaluation. Using this dataset, we evaluate several representative detection methods, including min$K\%$, n-gram accuracy, and Contamination Detection via output Distribution (CDD), spanning confidence-based and memorization-based approaches. Our findings show, perhaps surprisingly, that none of the tested techniques reliably distinguish between contaminated and uncontaminated items. Moreover, combining these methods does not significantly improve detection performance. We hypothesize that this is because *MathCONTA* requires detecting subtle, incidental mathematical contamination arising naturally during large-scale pretraining, rather than the more obvious contamination introduced through fine-tuning.

Finally, we analyze the downstream impact of contamination on model accuracy and find that while it can lead to modest gains at specific difficulty levels, it is unlikely to be the primary factor behind recent advances in LLM-based mathematical reasoning. To support transparency and future research, *MathCONTA* and all accompanying code for experiments will be made openly available.

Contents

CHAPTER 1

Introduction

1.1 Problem Statement

Recent advances in large language models (LLMs) have led to significant performance gains across various natural language tasks, including mathematical reasoning. However, a critical concern has emerged regarding the integrity of these evaluations: the possibility that benchmark questions used for testing may have been seen during training. This phenomenon—known as *data contamination* [1]–[3]—can inflate performance metrics and obscure the true generalization capabilities of a model [4], particularly in specialized domains such as mathematics [5].

In the context of mathematical problem-solving, where distinguishing between genuine reasoning and simple memorization is essential, the risk of contamination is particularly high. Several methods have been proposed to detect such contamination, typically relying on token probability analysis or memorization-based heuristics. Yet, the effectiveness of these methods has not been thoroughly assessed in a controlled setting [6], particularly within the domain of mathematical benchmarks.

To address this gap, this thesis presents *MathCONTA*, a novel dataset for contamination detection in mathematics. It spans multiple domains—including algebra, number theory, combinatorics, and integration—and covers various difficulty levels, from simple word problems to advanced math contest problems. The dataset is designed to evaluate contamination detection methods under direct pretraining exposure. It contains an equal mix of questions seen during pretraining and entirely new problems. Importantly, unlike many prior studies that introduce contamination artificially via finetuning [7]–[9], *MathCONTA* reflects contamination as it arises *naturally* during pretraining. This distinction enhances the validity of our evaluation and ensures that our findings are applicable to real-world model deployment scenarios.

1.2 Research Questions

To guide this investigation and address current limitations in the field of data contamination detection, we define the following research questions:

- **RQ1:** How effectively can current closed-data contamination detection methods identify whether mathematical benchmark questions—drawn from a dataset with an equal mix of pretraining-seen and novel items—are contaminated or uncontaminated?
- **RQ2:** To what extent does pretraining contamination influence the accuracy of large language models on mathematical benchmark questions from the same dataset?

1.3 Definitions and Terminology

This section defines key terms and concepts central to data contamination detection in mathematical benchmarks to establish a common understanding.

- **Data Contamination:** Data Contamination describes the presence of overlapping content between training and evaluation datasets [1]–[3].
- **Direct Contamination:** Direct contamination is a form of data contamination where *identical* text sequences occur in both the training and evaluation sets [10].
- **Indirect Contamination:** In contrast, indirect contamination is a form of data contamination where semantically equivalent but syntactically altered content appears in both datasets [4]. This includes *paraphrased*, *reworded*, or otherwise *modified* instances of the same underlying information.
- **Memorization:** Memorization is a model's ability to exactly reproduce a sequence of tokens encountered during training [11], [12].
- **(Mathematical) Benchmark Data Contamination:** Benchmark data contamination is a specific form of data contamination that occurs when a benchmark question and at least one valid answer appear in the training corpus [1]–[3]. The presence of such contamination is considered independent of whether the model explicitly exploits it, such as through memorization. Benchmark data points are typically represented as tuples of the form $(\text{Question}, \text{Answer})$. In the context of mathematical benchmark data contamination, we extend this concept to tuples of the form $(\text{Mathematical Problem/Statement}, \text{Solution/Proof})$.

  In this work, we use the term *data contamination* exclusively to refer to *direct benchmark data contamination* if not otherwise specified.

- **Open-Data Contamination Detection Methods:** Methods that rely on the full availability of training data, enabling direct comparisons between known training and evaluation datasets.
- **Closed-Data Contamination Detection Methods:** Methods that detect data contamination without access to the training data, relying solely on abnormal patterns in model responses or token probabilities.
- **White-Box LLM Access:** White-box access refers to the ability to retrieve the full next-token probability distribution generated by a language model. This means accessing the complete probability vector for all possible next tokens instead of merely observing the output text.
- **Black-Box LLM Access:** Black-box access refers to restricted interaction with a language model, where only the generated text output is accessible.

1.4 Research Objectives

While benchmark data contamination is widely acknowledged as a pressing issue [4], [7], [13], existing detection methods have not been thoroughly validated in realistic, domain-specific scenarios [6], [14]. This thesis aims to fill that gap by evaluating the reliability of such methods using the *MathCONTA* dataset. The primary objectives are twofold:

1. To assess whether current contamination detection techniques can reliably differentiate between contaminated and uncontaminated items in a math-specific, closed-data setting.
2. To quantify the effect of pretraining contamination on model performance, determining whether it significantly contributes to observed improvements in mathematical reasoning.

A key contribution of this work is its emphasis on *naturally occurring* contamination during pretraining, as opposed to simulated contamination through finetuning. This design choice ensures that the resulting analysis reflects more realistic conditions under which contamination may influence model performance and evaluation integrity.

1.5 Methodological Approach

A high-level overview of the proposed methodology is provided in Figure 1.1. We start from the OpenWebMath (OWM) [15] dataset, sampling mathematical examples labeled as `Contaminated` if they originate from the training corpus, and as `Clean` if they are drawn from external sources (details are presented in 3.1).

For RQ1, we evaluate a selection of existing Contamination Detection Methods (CDMs), some of which require access to the full next-token probability distribution ($P_{\%}$) or the LLM's natural language answers (A_{LLM}). The goal is to identify the method that achieves the highest classification accuracy when predicting whether a given question (Q) and its ground truth answer (A_{gold}) are contaminated or clean. To ensure fair evaluation, we split *MathCONTA* into training and test sets, tune all methods using 5-fold stratified cross-validation on the training set, and report the final accuracy on the test set. Statistical significance compared to a random guessing baseline is assessed using McNemar's test [16].

For RQ2, the LLM's performance is evaluated separately on the Contaminated and Clean subsets using 1-shot chain-of-thought (CoT) prompting. Full points are awarded only when the model's output matches the reference solution. The resulting performance gap between the subsets is analyzed using the Mann–Whitney U test [17] to assess the effects of contamination.

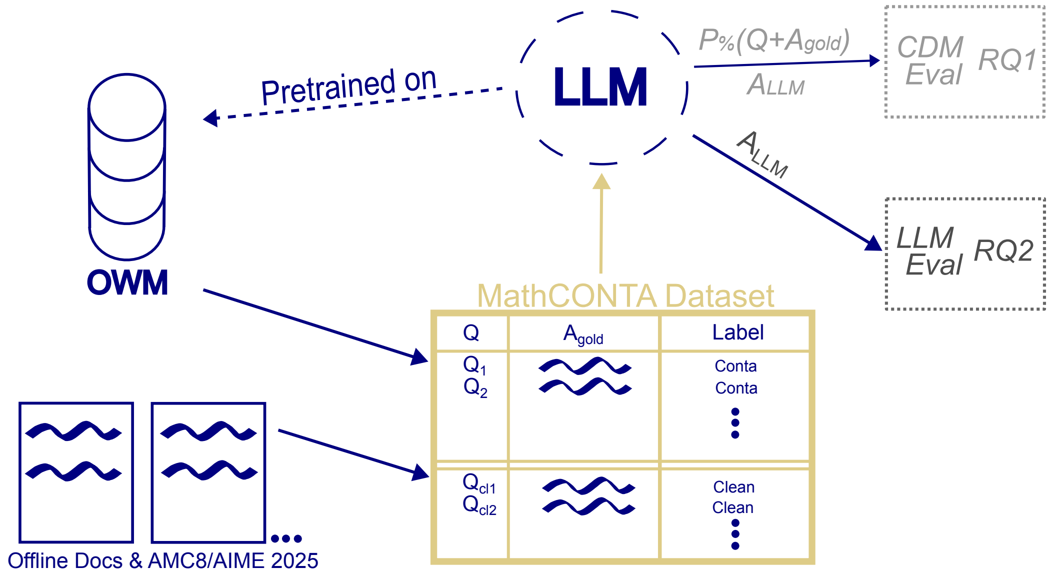


Figure 1.1: Overview of the methodological approach: Mathematical examples are sampled and labeled depending on their source (see section 3.1). For Research Question 1 (RQ1), a selection of Contamination Detection Methods (CDMs) (section 3.2) are evaluated on their ability to distinguish between Contaminated and Clean examples using LLM outputs, such as next-token probability distributions ($P_{\%}$) and natural language answers (A_{LLM}). Methods are trained and validated via 5-fold stratified cross-validation on the MathCONTA training set and tested separately (section 3.3). For Research Question 2 (RQ2), LLM performance is measured on the Contaminated and Clean subsets using 1-shot chain-of-thought prompting to quantify contamination impact (section 3.4).

1.6 Main Contributions

Our work makes the following key contributions:

- We introduce *MathCONTA*, a dataset of 100 high-quality mathematical question-answer pairs. It provides ground-truth contamination labels for all LLMs pretrained on OpenWebMath [15], enabling contamination detection and performance gap evaluation. The dataset and all accompanying experimental code are publicly available to support future research.[1,2]
- Using *MathCONTA*, we show that existing closed-data detection methods—such as min$K$ [18], CDD [9], n-gram accuracy [8], and their combinations—are insufficient to distinguish between contaminated and clean examples reliably.
- We further demonstrate that subtle contamination from mathematical pretraining has a small but statistically insignificant effect and does not account for recent LLM performance gains.

1.7 Structure of the Work

The remainder of this thesis is structured as follows. Chapter 2 introduces LLMs and reviews related work, positioning this study within the broader research landscape. Chapter 3 describes the proposed methodology, including dataset preparation, method selection and adaptation, and evaluation protocols. Chapter 4 presents the experimental results and key findings. Chapter 5 discusses the implications of the results and provides critical reflections. Chapter 6 outlines the limitations of the thesis and potential directions for future work. Finally, Chapter 7 summarizes the main contributions of the thesis.

[1]MathCONTA Dataset: `https://huggingface.co/datasets/Tobstar001/MathCONTA`
[2]Code and Raw Data: `https://github.com/Tobstar001/MathCONTA_MasterThesis.git`

CHAPTER 2

Background and Related Work

2.1 Data Usage in LLMs

This section provides a brief overview of the commonly used mechanisms within LLMs, followed by a discussion of the stages at which different training data is utilized, highlighting points where contamination may occur.

2.1.1 Inner Workings of LLMs

Despite the existence of thousands of different LLMs today, LLMs primarily utilize the decoder architecture [19, p. 28], which is based on the Transformer algorithm and its self-attention mechanism [20].

A defining feature of the decoder architecture is its access to only preceding tokens in a sequence, in contrast to the encoder architecture, which can incorporate both preceding and succeeding tokens.

At its core, a single layer in the Transformer algorithm consists of two primary components: (1) a multi-head self-attention mechanism, which captures token relationships within a sequence, and (2) a position-wise feed-forward neural network that processes the attended information.

For input embeddings $\mathbf{X} \in \mathbb{R}^{t \times d}$ the self-attention mechanism is defined as [20]:

$$\text{Attention}(\mathbf{Q}, \mathbf{K}, \mathbf{V}) = \text{softmax}\left(\frac{\mathbf{Q}\mathbf{K}^\top}{\sqrt{d_k}}\right)\mathbf{V},$$

where $\mathbf{Q} = \mathbf{X}\mathbf{W}_Q$, $\mathbf{K} = \mathbf{X}\mathbf{W}_K$, and $\mathbf{V} = \mathbf{X}\mathbf{W}_V$ represent the query, key, and value matrices, computed via learned weights $\mathbf{W}_Q, \mathbf{W}_K, \mathbf{W}_V \in \mathbb{R}^{d \times d_k}$.

To allow the model to focus on different aspects of the previous tokens, the multi-head attention mechanism processes multiple attention heads in parallel:

$$\text{MultiHead}(\mathbf{X}) = \text{Concat}(\text{head}_1, \ldots, \text{head}_h)\mathbf{W}_O,$$

where each head is defined as:

$$\text{head}_i = \text{Attention}(\mathbf{Q}_i, \mathbf{K}_i, \mathbf{V}_i),$$

using independent weight matrices for each head. The combined output is projected back to the model dimension via $\mathbf{W}_O$. Following this, a position-wise feed-forward network applies two fully connected layers with an activation function, transforming the attended data further.

The Transformer architecture typically consists of multiple such layers, stacked to increase the model's capacity to understand complex relationships in the token sequence, e.g. GPT-3 employs 96 such transformer layers stacked together [21].

2.1.2 Pretraining

Pretraining Task

The weight matrices of transformer networks are updated most commonly using an unsupervised pretraining task, known as language modeling [19, p. 26]: Token sequences $\mathbf{X} = \{x_1, x_2, \ldots, x_t\}$ are extracted from a large text corpus and the LLM aims to predict the next token x_j given the preceding tokens. This task optimizes the cumulative log-likelihood:

$$L_t = \sum_{j=1}^{t} \log p(x_j \mid x_1, x_2, \ldots, x_{j-1}).$$

The model's prediction is compared to the actual token, and the cross-entropy loss is computed as:

$$H = -\frac{1}{t} \sum_{j=1}^{t} \log p(x_j),$$

where $p(x_j)$ is the predicted probability of the true token x_j. This loss is minimized via backpropagation to update the model's weights.

A key advantage of this approach is its unsupervised nature, as the text itself inherently provides the true labels. The next section discusses the datasets commonly used for this pretraining process.

Commonly used Pretraining Sources

Commonly used sources for pretraining data can be broadly categorized into web pages, books, and code. For web pages, large-scale datasets such as *CommonCrawl* -which include snapshots of most web content on the Internet, although with significant amounts of noisy and low-quality information - are widely used. Filtered subsets of such datasets, like *C4*, *RedPajama-Data*, and *Wikipedia*, provide cleaner and more focused sources of information. Further specialized subsets, such as *OpenWebMath* [15], are available for specific domains, such as mathematics.

Books are another important category, offering high-quality data from datasets such as *BookCorpus* and *Project Gutenberg*, which include tens of thousands of books. Academic literature, such as the *arXiv Dataset*, provides additional high-quality training data in the form of scientific papers.

A third key category consists of code datasets sourced from public repositories, such as those available on *GitHub*.

Given the massive size of these datasets—often in the range of terabytes of raw text—there is a significant risk that benchmark data may inadvertently or intentionally be included in the model's training corpus, leading to potential data contamination.

Introduction to *OpenWebMath* [15] *OpenWebMath* (OWM) is a central resource in this thesis. It consists of approximately 6.3 million mathematical webpages, totaling 14.7 billion tokens ($\sim 52\,\text{GB}$), collected from Common Crawl snapshots between May 2013 and April 2023. The dataset was curated through a multi-stage filtering pipeline, including English language detection, a MathScore classifier for mathematical relevance, and perplexity-based quality filtering. Deduplication was performed using SimHash (threshold 0.7) followed by manual domain-level inspection.

Some of the largest domains in OWM are `stackexchange.com` (17.99% of all webpages), `physicsforums.com`, and `mathhelpforum.com`.

Models trained on OWM achieve strong results on mathematical benchmarks, outperforming models trained on larger general-domain corpora.[15]. While OWM was not designed for contamination analysis, we employ it as a reference: (Question, Answer) pairs found in OWM are labeled as Contaminated. Pairs not found in OWM are labeled as Clean if they additionally satisfy at least one of the following criteria: (1) sourced from a German offline book, (2) written or modified by the author, or (3) created after the year 2024. Section 3.1 provides details of the curation methodology.

2.1.3 Fine-tuning

After pretraining, it has been shown that LLMs develop a general understanding of the grammar and syntax of the language on which they were trained. However, the model remains unaligned at this stage, meaning it lacks safeguards to prevent generating toxic, dishonest, or harmful responses. Additionally, it is not designed to follow instructions or perform specific tasks. Addressing these limitations requires an additional training phase known as fine-tuning [19, p. 15].

Supervised Fine-Tuning (SFT) involves training the model on curated datasets containing high-quality examples, such as question-answer pairs, conversational exchanges (chat-style), or task completions. Unlike pretraining, which focuses on language modeling, SFT adjusts the model's weights using a sequence-to-sequence loss function, evaluating how well the output adheres to the given instruction. Despite relying on a smaller dataset compared to pretraining, SFT significantly enhances or unlocks the LLM's capacities [19, p. 34]. Common datasets for this stage include standard NLP task datasets such as *P3* and *FLAN*, conversational datasets like *ShareGPT* and *OpenAssistant*, and synthetic datasets generated semi-automatically by other LLMs, such as *Self-Instruct-52K* and *Alpaca* [19, p. 15].

Following fine-tuning, **Reinforcement Learning from Human Feedback (RLHF)** is commonly applied to align the LLM with human values, such as helpfulness, honesty, and harmlessness. RLHF involves training the model using feedback from human evaluators, who rank model outputs based on alignment with desired behaviors. This feedback is then incorporated using reinforcement learning algorithms to refine the model's responses [19, p. 38]. Common datasets for RLHF include *HH-RLHF*, *SHP*, and *Stack Exchange Preferences* [19, p. 15].

2.1.4 Specific LLMs

This section provides an overview of the large language models used in our experiments (Table 2.1). All selected models are decoder-only transformer architectures with publicly available weights, parameter sizes suitable for experiments on platforms such as Google Colab (7B–13B), and a particular focus on mathematical reasoning. Notably, all models include *OpenWebMath* (OWM) as part of their pretraining corpus.

OLMo-Series [22]

The OLMo series, developed by the Allen Institute for AI, is characterized by full transparency: model architecture, training corpora, and intermediate weights are openly documented and available. These models employ a transformer decoder architecture [20] with no biases, SwiGLU activations, and Rotary Positional Embeddings (RoPE). Tokenization is performed using a BPE-based tokenizer.

- **OLMo-0724-7B (OL) [22]:** This model includes 7 billion parameters, with 32 layers, 32 attention heads, and a hidden size of 4096. It features QKV clipping for training stability and an extended context window of 4096 tokens. Pretraining was conducted on the DOLMA v1.7 corpus [23], a 2.7 trillion-token dataset sourced from *Common Crawl*, *Wikipedia*, *GitHub*, *Reddit*, *Semantic Scholar*, and *Project Gutenberg*, with added high-quality content including *OpenWebMath*. Fine-tuning

Table 2.1: Overview of data usage and architectorial design choices of different open-source LLMs

Model (base)	**Size**	**Vocab Size**	**Pretraining Data**	**Fine-Tuning Data**	**Includes OWM**
OLMo-7B-0724	7B	50k	*DOLMA corpus 1.7* (4.5 TB)	*TULU 2 SFT mix* (0.5 TB)	Yes
OLMo-2	13B	100k	*OLMo-Mix-1124* (3.9T tokens) *Dolmino-Mix-1124* (843B tokens)	*TULU 3 SFT mix*	Yes
DeepSeekMath (DeepSeekCode)	7B	32k	*DeepSeekCoder Corpus* (800 GB) [Stage 1] *DeepSeekMath Corpus* (120B tokens) [Stage 2]	*DeepSeekMath SFT* (776K examples)	Yes* (as seed)
Lemma-7B (Code-LLaMA)	7B	32k	*LLaMA 2 corpus* (-) *Code LLaMA corpus* (-) *Proof-Pile-2* (200B tokens)	*muInstruct* (1.8k examples) *camel-ai/math* (50k examples)	Yes

was performed on the *TÜLU-2-SFT-MIX* dataset, comprising data from *FLAN*, *ShareGPT*, and *OpenAssistant*. The model was released in July 2024.

- **OLMo-2-13B (OL2) [24]:** This 13 billion parameter model incorporates architectural enhancements such as improved normalization and regularization [24, p. 7], and uses an expanded vocabulary of 100,000 tokens. Pretraining was carried out in two stages: first on *olmo-mix-1124*, which includes *DCLM-Baseline*, *Algebraic-stack*, and *OpenWebMath*; second on *dolmino-mix-1124*, a curated mix of high-quality, math-focused, and synthetic data. Three models were trained on separate subsets and merged using model souping. Fine-tuning used the *tulu-3-sft-olmo-2-mixture* dataset. The model was released in November 2024.

DeepSeekMath (DS) [25]

DeepSeekMath is a decoder transformer model with 7 billion parameters and a 32,000-token vocabulary. It is derived from continuous pretraining of the base LLM, DeepSeek-Code, which was initially trained on the *DeepSeekCoder Corpus* to specialize in coding skills. The model's training further utilized the 120 billion-token *DeepSeekMath Corpus*, constructed from filtered web pages in *CommonCrawl*. This corpus was curated with the assistance of *OpenWebMath*, taking it as an example of how high-quality math web pages should look. Although DeepSeekMath has open weights, the complete pretraining corpus is not publicly available. The model was released in February 2024.

Lemma-7B (LE) [26]

Lemma is a 7-billion-parameter decoder transformer model with a 32,000-token vocabulary. It builds on the base model *Code-LLaMA*, which was pretrained using the *LLaMA 2 corpus* and the *Code-LLaMA corpus*. Lemma's training included the 200 billion-token *Proof-Pile-2* dataset, supplemented by fine-tuning on smaller datasets such as *muInstruct* (1.8k examples) and *camel-ai/math* (50k examples). *OpenWebMath* is explicitly included in the *Proof-Pile-2* dataset. While Lemma's weights are open, details of the pretraining corpus for *LLaMA 2* are not publicly available. The model was released in September 2023.

2.2 Evaluation with Benchmarks

2.2.1 Introduction to LLM Evaluation

Language modeling, the core ability of LLMs, involves predicting the next token based on preceding tokens (see subsection 2.1.2). This capability primarily reflects a model's proficiency in basic language understanding and generation. Perplexity is a commonly used metric for assessing this ability [19, p. 56].

As LLMs have evolved from simple language models into general-purpose task solvers, evaluation criteria have also shifted. While human evaluators are often employed (for example, within a crowdsourcing platform called Chatbot Arena [27]), this approach lacks objectivity and is not always reproducible [19, p. 65]. Alternatively, other LLMs can serve as evaluators, offering a scalable and efficient solution [28]. However, they introduce certain biases, such as self-enhancement tendencies, i.e., the tendency to prefer answers from themselves or their model family and have a preference for verbose answers. Moreover, due to their current limitations in handling complex reasoning tasks, LLMs alone are insufficient as evaluators in specialized domains such as mathematics.

The most common approach for evaluating LLMs is the use of benchmarks. Benchmarks typically consist of a diverse set of test questions similar to those given to humans, covering areas such as language generation, knowledge utilization, reasoning, safety, and tool manipulation. Benchmarks are particularly valuable for monitoring model performance during pretraining and for comparing different LLMs. Consequently, new LLM releases are typically accompanied by standardized benchmark scores. However, benchmark-based evaluation has limitations: When evaluated automatically, LLMs are often sensitive to evaluation conditions, including prompt phrasing, zero-shot versus few-shot testing, and answer parsing methods. Additionally, publicly available benchmark datasets are susceptible to data contamination.

2.2.2 Mathematical Benchmarks

The following section will introduce the most common benchmarks for evaluating LLMs' mathematical abilities in natural language, distinguishing between static and dynamic evaluation settings. Static benchmarks are predefined datasets that remain fixed over time. Dynamic benchmarks, a relatively recent development (with the first appearing in 2023 [10]), aim to mitigate the risks of data contamination by continuously updating or adapting the questions.

Static Mathematical Benchmarks

Commonly used static or note-worthy mathematical benchmarks in natural language are:

- **GSM8K [29]** consists of grade-school level math word problems that typically require multi-step arithmetic reasoning (2-8 steps of basic operations). Each problem is written in natural language and is solvable by a bright middle school student. A step-by-step solution is provided. The dataset contains 8,500 problems in total (approximately 7,500 training and 1,000 test questions)
- **MATH [30]** is a collection of challenging competition-level mathematics problems, spanning algebra, geometry, number theory, and more at roughly high-school olympiad difficulty. Each question is accompanied by a full step-by-step solution. It contains 12,500 problems in total (with a standard split of 7,500 training and 5,000 test questions).
- **GHOSTS [31]** is a collection of graduate-level and research-level math questions in natural language curated by professional mathematicians. The full GHOSTS benchmark includes 728 prompts. For each prompt, ChatGPT/GPT-4's answer was evaluated by expert mathematicians manually, providing a scoring of the model's output.

Additional static mathematical benchmarks include **SVAMP** [32], **MATH 401** [33], **AGIEval** [34], **ARB** [35], **OlympiadBench** [36], and the **IMO Small Challenge** [37]. Recent benchmarks, such as **GSM1K** [38], employ private test sets to mitigate the risk of data contamination. Others propose generalizing existing datasets, for example extensions of **GSM8K** [39], [40], to analyze performance gaps relative to the original benchmark. For further details about performance analysis, see Section 2.3.2.

Dynamic Mathematical Benchmarks

Commonly used dynamic mathematical benchmarks in natural language are:

- **LiveBench [41] and MathArena [42]** LiveBench is a benchmark updated every six months, covering multiple domains. The mathematics category includes problems sourced from high school math competitions held in the past 12 months (e.g., AMC12, AIME, USAMO, IMO) as well as synthetically generated, more challenging problems from the AMPS dataset [30]. For proof-based questions, certain parts of the ground-truth solutions are masked, requiring the LLM to reconstruct the missing expressions in the correct positions. This approach ensures that all problems have easily verifiable ground-truth labels. The benchmark contains approximately 230 math questions in total. A related benchmark with examples from recent math competition provides MathArena [42].

- **UGMathBench [43]** focuses specifically on undergraduate-level mathematical reasoning. The dataset consists of 5,062 problems sourced from a university database, each with three randomized versions, with plans to introduce additional versions over time. They introduce two new evaluation metrics: *Effective Accuracy*, which measures the percentage of correctly solved problems across all randomized versions, and *Reasoning Gap*, which increases when the effective accuracy is lower than the overall average accuracy.

It is worth noting that examples from competitions (such as AMC [1], AIME [2], USAMO [3], and IMO [4]) are frequently used for a way of dynamically assess the reasoning capabilities of LLMs. We also highlight the AIMO Prize [5] as a recent initiative to encourage open development of AI models capable of advanced mathematical reasoning.

While dynamic benchmarks reduce the risk of data contamination by regularly introducing new problems, they also introduce variability in difficulty across iterations. Moreover, they cannot fully eliminate the possibility that some questions were seen during training. To address this, robust post-hoc contamination detection is essential, as it can help identify compromised items and inform the careful selection of future problems.

2.2.3 Datasets for Contamination Detection

Previous benchmarks target the performance evaluation of LLMs. In contrast, recent efforts introduce datasets specifically designed to assess contamination detection. It is worth noting that these datasets are not specific to math.

- *WikiMIA* [18]: Uses Wikipedia event page timestamps to separate seen (before January 1, 2023) and unseen data (after), evaluating contamination in QA tasks.

- *PatentMIA* [44]: Consists of Chinese patent documents partitioned by publication date (before January 1, 2023 vs. after March 1, 2024), targeting membership inference in patents.

[1]https://maa.org/student-programs/amc/
[2]https://maa.org/maa-invitational-competitions/
[3]https://maa.org/student-programs/amc/
[4]https://www.imo-official.org/
[5]https://aimoprize.com/

- *StackMIA* [45]: Derived from Stack Exchange posts, employing temporal splits around model-specific cutoff dates to detect contamination in code and technical QA tasks.
- *MIMIR* [14]: Samples members and non-members from the Pile's train and test split across seven diverse sources: general web (Pile-CC), knowledge bases (Wikipedia), academic papers (PubMed Central, ArXiv), dialogues (HackerNews), and specialized domains (DM Math or GitHub).
- *DetCon* [9]: Compares outputs from clean and contaminated LLMs, focusing on mathematical, logical, and coding tasks. Unlike the others, DetCon uses pre-generated outputs and is not intended for new LLM contamination experiments.

To complement these resources, we introduce *MathCONTA*, a dataset of 100 high-quality mathematical question-answer pairs. *MathCONTA* ensures ground-truth alignment for all LLMs pretrained on OpenWebMath [15] and release dates before 2025. It supports both contamination detection and model evaluation.

2.3 Data Contamination Detection Methods

2.3.1 Overview

We adopt a categorization of existing approaches similar to the survey by [1], distinguishing between open-data and closed-data settings:

- **Open Data:** Methods that rely on the full availability of training data, enabling direct comparisons between known training and evaluation datasets. Typical techniques include n-gram matching [21], [46], [47], embedding similarity search [48], and LLM-based paraphrase detection [4].
- **Closed Data:** Methods that evaluate models without access to their training data. Early work primarily focuses on membership inference attacks on LLMs [49] and on fundamental aspects of memorization and exploitation [50]. More recent approaches build on these insights, relying on behavioral signals, and can be further categorized into:
  - **Model Confidence:** Approaches that infer contamination based on statistical patterns such as token probabilities or output distributions. Representative techniques include MinK% [18], Conta Traces (named coined by the author of the thesis as no name was given) [51], perplexity-based comparisons over time [52] or with reference samples [49] and contamination detection via output distribution [9], [53].
  - **Model Memorization:** Methods that assess whether a model has memorized specific examples by measuring its ability to reproduce, complete, or

reconstruct text. These include n-gram accuracy [8], Test Slot Guessing (TS-Guessing) [54], and Guided Instruction [55].

– **Other Methods:** Other methods focus on better calibration techniques [44], [45] or use the information in hidden states of LLMs [7].

– **Performance Analysis:** This approach identifies contamination by detecting unusual performance changes on potentially contaminated samples. For instance, performance drops may arise due to training cut-off dates, as in code datasets analyzed in [56], [57], or through generalization measures, which can degrade performance in mathematical benchmarks [7], [38]–[40], [58].

Since this thesis focuses on evaluating methods within closed-data settings, the next section will introduce the techniques in greater detail.

2.3.2 Closed-Data Settings

Contamination detection presents a significant challenge in closed-data settings. Unlike open-data scenarios, which emphasize tasks such as retrieval and search, the focus here shifts to identifying anomalous statistical patterns in model behavior and outputs.

Two access settings are particularly relevant in this context: White-Box Access and Black-Box Access (see 1.3). Furthermore, methods may operate at either the instance level or the dataset level. The experiments using the *MathCONTA* dataset employ instance-level methods.

Model Confidence

This section outlines approaches that detect contamination by analyzing statistical patterns, such as token probabilities or output distributions.

- **MinK% Method [18] (White-Box)** The MinK% method is based on the hypothesis that if a text X was part of the training dataset, its words $x_i \in X$ would generally have higher probabilities in the output from an LLM. Accordingly, words with very low probabilities within a sequence can serve as indicators that X was not part of the training data. The method involves computing the probability of each token in the sequence X and selecting the lowest k% of tokens (with $k = 20$ empirically shown to yield strong results in prior work). This subset of tokens, referred to as the Min-K% set of X, is used to calculate a normalized sum of their probabilities. If this normalized sum exceeds a predefined threshold θ (which must be tailored to the specific use case), the sequence X is likely part of the training data. Mathematically, this is expressed as:

$$\text{Min-K\% Prob}(X) = \frac{1}{n} \sum_{x_i \in \text{Min-K\%}(X)} p(x_i \mid x_1, \ldots, x_{i-1}),$$

Name	**Technique**	**Open Data**	**LLM Access**	**Level**	**Hidden State**	**RQ1**
GPT4/3&LLama[21], [46], [47]	n-gram overlap	yes	no	instance	no	no
Platypus Decon [48]	embedding similarity	yes	no	instance	no	no
LLM Decontaminator [4]	LLM-based similarity	yes	no	instance	no	no
CDD [9]	peakedness of output distribution	no	black-box	instance	no	yes
MinK% [18]	minimum in token probs	no	white-box	instance	no	yes
Sharded Likelihood [59]	question ordering	no	white-box	benchmark	no	no
Method of Xu [8]	perplexity and n-gram output	no	white-box	benchmark and instance	no	partly
Conta Traces [51]	cum. log-probability	no	white-box	instance	no	yes
Guided Instruction [55]	n-gram output	no	black-box	benchmark	no	no
Ts Guessing [54]	n-gram output	no	black-box	instance	no	partly
Perplexity Comparison [49], [52], [60]	perplexity	no	white-box	benchmark and instance	no	partly
DICE [7]	reference llm, hidden state analysis	no	white-box+	instance	yes	no
minK%++ [53]	local maxima in output distribution	no	white-box	instance	no	yes*
CE comparison [44]	cross-entropy comparison	no	white-box	instance	no	yes*
PAC [45]	calibration through augmentation	no	white-box	instance	no	yes*

Table 2.2: Overview of Contamination Detection Methods, their key properties, and applicability to RQ1. Yes* indicates technical applicability but exclusion for scope reasons.

where:

– Min-K%(X) denotes the subset of tokens in X with the lowest $k\%$ probabilities.

– $p(x_i \mid x_1, \ldots, x_{i-1})$ represents the conditional probability of x_i given the preceding tokens.

– n the size of the Min-K%(X) subset.

In the study by [18], the authors not only identified copyrighted books used in pretraining OpenAI's language model `text-davinci3` — one of the early iterations of ChatGPT [19] — but also explored issues related to benchmark data contamination. They assessed their method's effectiveness across several benchmarks, including BoolQ, IMDB, TruthfulQA, and Commonsense QA, demonstrating superior performance compared to baseline metrics. However, the study did not evaluate the Min-K% method's effectiveness in identifying examples related to mathematical problem-solving, specifically.

Moreover, the model `text-davinci3`, which allowed access to the full next token probability despite being closed-source, has since been deprecated. Successor models, such as GPT-4 and GPT-4o, now operate with stricter limitations, offering only black-box access. This shift significantly reduces the applicability of methods like Min-K% to models that only offer black-box access.

- **Conta Traces [51] (White Box)**: The approach involves computing the cumulative log-likelihood of each token of a sequence and plotting it for each token in a graph. Let the sequence of tokens be $\mathbf{X} = \{x_1, x_2, \ldots, x_t\}$, and let $\log p(x_i \mid x_1, \ldots, x_{i-1})$ represent the log-likelihood of token x_i, given the preceding tokens. The **cumulative log-likelihood** up to the i-th token can be expressed as:

$$L_i = \sum_{j=1}^{i} \log p(x_j \mid x_1, \ldots, x_{j-1})$$

where:

– L_i is the cumulative log-likelihood after i tokens,

The hypothesis suggests that familiar text generates curves with greater curvature, whereas unfamiliar text results in nearly linear and steep curves. To compare curves, the authors use a two-parameter exponential model:

$$f(x) = -A\left(1 - \exp^{-Bx}\right)$$

where:

– A: Total cumulative log-likelihood.

  - B: Curvature of the function.
  - Higher B and lower A indicate contamination.

  With this method, the authors detect contamination in traditionally used reasoning tests for humans, such as the *Cognitive Reflection Test* and *Conjunction Fallacy Test*, requiring them to modify the examples for unbiased evaluation.

- **Perplexity Comparison [49], [52], [60] (White Box)**:

  Perplexity measures a model's uncertainty when predicting a sequence of tokens. For a sequence $\mathbf{X} = \{x_1, \ldots, x_t\}$, the cumulative log-likelihood is

$$L_i = \sum_{j=1}^{i} \log p(x_j \mid x_1, \ldots, x_{j-1}),$$

  and the perplexity (as defined in [8]) is

$$PPL(\mathbf{X}) = \exp\left(-\frac{L_t}{t}\right),$$

  where L_t is the cumulative log-likelihood up to token x_t.

  Li et al. [52] analyze perplexity across reading comprehension, multiple-choice, and summarization benchmarks over time. They compare perplexity against two baselines: (1) text likely included in model training and (2) text unlikely to have been included, based on dataset creation date relative to the model's training cutoff. For example, Wikipedia pages from 2016–2019 and post-July 2023 are used to evaluate benchmarks such as QuAC [61], BoolQ [62], and SQuAD_v2 [63], establishing a clean baseline for GPT-3 [21]. Benchmarks with perplexity closer to the contaminated baseline are flagged as suspicious. Results indicate contamination in GPT-3 and LLama models (7B, 13B, 30B) for reading comprehension and summarization tasks, while no correlation is observed in the MMLU dataset [30]. The study operates at the benchmark level and does not address individual examples or mathematical content.

  Carlini et al. [49] propose an alternative approach, comparing perplexity to lowercase perplexity or the zlib compression score [60].

- **Sharded Likelihood [59] (White Box)**: This method uses the feature of a dataset, called *interchangeability*. Specifically, it examines whether an LLM shows a preference, as indicated by token probabilities, for the original ordering of questions in a benchmark over alternative orderings. A stronger preference for the original ordering would suggest the possibility of contamination.

  The authors conclude that test set contamination for a diverse selection of benchmarks, including the mathematical benchmark GSM8K is relatively uncommon in tested LLMs (all tested LLMs were below 10 B parameters).

As this method operates at the level of entire benchmarks, making it unsuitable for experiments requiring analysis of individual examples.

- **Contamination Detection via output Distribution (CDD) [9] (Black Box)**: This method is based on the hypothesis that limited variability in the responses of an LLM to a specific prompt x—quantified using an edit distance metric on a token basis—may indicate data contamination.

  To implement this, the edit distance between all sampled answers and the greedy search answer (obtained by setting the temperature $t = 0$ during sampling) is calculated. These distances are modeled using a distribution $\rho^*(d)$:

  $$\rho^*(d) = \frac{1}{n} \sum_{i=1}^{n} \mathbb{I}(\mathrm{ED}(s_i, s_{t=0}) = d),$$

  where:

  - $\mathrm{ED}(s_i, s_{t=0})$ represents the edit distance between a sampled response s_i and the greedy search response $s_{t=0}$,
  - and n is the number of samples (default: $n = 100$).

  The peakedness of the edit distance distribution is defined as:

  $$\mathrm{Peak}(x) = F(d \leq \alpha \cdot l) = \sum_{d=0}^{\alpha \cdot l} \rho^*(d),$$

  where:

  - F is the cumulative distribution function of ρ^*
  - $\alpha \in [0, 1]$ is a hyperparameter controlling the similarity threshold (default: $\alpha = 0.05$)
  - l is the maximum length of sampled responses, which is higher or equal to the maximum value of the distance d

  Using the peakedness score $\mathrm{Peak}(x) \in \mathbb{R}$ for each prompt x, contamination detection is conducted by applying a decision threshold θ (default: $\theta = 0.01$). Prompts with $\mathrm{Peak}(x)$ exceeding θ are flagged as potentially contaminated.

  CDD achieves state-of-the-art performance in detecting contamination in code generation and logical reasoning domains, surpassing other methods on their newly created experimental set-up. Evaluated on the DetCON dataset, which comprises 2224 tasks across two domains and two contamination forms (explicit and implicit, both induced through finetuning), CDD reliably identifies whether an LLM is contaminated.

Experiments on ChatGPT (versions 0613 and 1106) with hyperparameters set to $\alpha = 0$ and $\theta = 0.2$ were conducted on the coding benchmarks HumanEval [64] and CodeForces2305 (a coding benchmark created after the latest training cutoff from the CodeForces [6] website). The results indicate a high probability of contamination in HumanEval but none in CodeForces2305. For ChatGPT version 1106, approximately 40% of the examples in HumanEval are identified as contaminated.

- **minK%++ [53] (White Box)** This recent method shows that training data forms local maxima in the output distribution, complementing and reinforcing insights from CDD [9]. It also outperforms Min-K% [18], achieving superior results in their experiments on the WikiMIA [18] and MIMIR benchmark [14].

Model Memorization

This section presents approaches that detect contamination by assessing the model's ability to reproduce exact token sequences from the training data.

- **Method of Xu [8] (White Box)**:

  Xu et al. [8] propose a data detection pipeline that integrates two "atomic metrics", perplexity and n-gram accuracy, alongside a synthesized reference benchmark.

  The pipeline first computes metric scores on the training splits from GSM8K and MATH and on linguistically modified versions generated using ChatGPT. The exact process is repeated for the test split. All scores are normalized, and the differences from the reference scores are calculated. These differences are then used to derive a ranking score of $\delta_{\text{train-test}}$. A positive $\delta_{\text{train-test}}$ indicates greater familiarity with the training data. At the same time, a value near zero suggests the training and test data were either not seen during pretraining or were seen at similar rates.

  Although the pipeline operates at the benchmark level, the n-gram accuracy metric might also be used for identifying instance-level mathematical contamination [8, p. 10].

  The n-gram accuracy metric is formally defined as follows [8]: For a token sequence $\mathbf{X}$, K random starting points start_j are selected. The text preceding each starting point serves as a prompt, and the subsequent n-gram is used as the target for model prediction. The n-gram accuracy for a single starting point j is computed as

$$\text{N-gram Accuracy}(\mathbf{X}_j) = \mathbb{I}\left(X_{\text{start}_j:\text{start}_j+n}, \hat{X}_{\text{start}_j:\text{start}_j+n}\right),$$

  where:

  – $X_{\text{start}_j:\text{start}_j+n}$ is the true n-gram,

  – $\hat{X}_{\text{start}_j:\text{start}_j+n}$ is the model's prediction and

[6] https://codeforces.com

   – the indicator function $\mathbb{I}$ equals 1 if the prediction matches the true n-gram, and 0 otherwise.

Averaging over all K starting points yields the overall n-gram accuracy:

$$\text{N-gram Accuracy}(\mathbf{X}) = \frac{1}{K}\sum_{j=1}^{K}\text{N-gram Accuracy}(\mathbf{X}_j).$$

An n-gram accuracy above 0.5 indicates that most n-grams are predicted correctly, suggesting the model may have encountered specific samples during training. Notably, n-gram accuracy relies solely on the model's verbal outputs, making it applicable even to black-box LLMs.

Their case study, in which 31 open-source LLMs are investigated on GSM8K and MATH suggests that several LLMs might have been trained on the training sets. However, no evidence indicates that these models were trained on the corresponding test sets. It should be noted that this conclusion is constrained by the limitations of their method, which cannot definitively determine scenarios where models are simultaneously trained on both splits.

- **Test Slot Guessing (TS-Guessing) [54] (Black Box)** The TS-Guessing method prompts an LLM to reconstruct a benchmark question's exact word or phrasing by masking a pivotal word, which captures the sentence's core meaning. The authors employed ChatGPT with 5-shot in-context learning for this task. In the case of multiple-choice questions, one incorrect, non-trivial answer was masked.

  Closed data experiments evaluated state-of-the-art LLMs on knowledge-based question answering and reasoning benchmarks. Notably, on the MMLU benchmark [65], ChatGPT and GPT-4 [46] achieved exact match rates of 52% and 57%, respectively, when predicting missing options. The TS-Guessing method has limitations, relying on LLMs' ability to interpret unconventional instructions. Tests with open-source LLMs revealed frequent failures to identify the masked incorrect option, as many models defaulted to predicting the correct answer.

- **Guided Instruction [55] (Black Box)** Similar to TS Guessing [54], this method prompts the LLM to complete specific parts of benchmark instances. It provides additional context, such as the benchmark name and split type, and operates at the benchmark level. As a baseline, the model is also prompted without this context. If the contextual prompt improves completion quality, the instance is flagged as contaminated. To assess overlap with the reference, the method uses BLEURT [66], ROUGE L [67], and GPT 4 [46]. Results show that GPT 4 is contaminated with AG News, WNLI, and XSum.

Other Closed-Data Methods

- **DICE [7] (White Box +)** DICE is a method proposed for detecting contamination in LLMs fine-tuned on mathematical reasoning datasets such as GSM8K [29]. It identifies the layer in the fine-tuned model that deviates most from its counterpart in the original, non-fine-tuned model. A multilayer perceptron binary classifier is then trained on the weights of this layer to detect contamination. DICE demonstrates strong performance in their setup for in-distribution contamination, where fine-tuning and evaluation use data samples from the same distribution, achieving near-perfect AUC and outperforming methods like MinK. Despite its effectiveness during fine-tuning, DICE requires an uncontaminated reference model for each tested LLM. In our setting, where contamination is not introduced post-hoc via fine-tuning, such a reference model is unavailable. Therefore, DICE cannot be integrated into the MathCONTA framework.

- **Cross Entropy comparison [44] (White Box)** A recent adaptation of the Min-K% method [18] was developed by [44], introducing a divergence-based calibration approach. They improved pretraining data detection by computing the cross-entropy between token probability and frequency distributions. Their method, validated on both English datasets and the newly proposed Chinese benchmark *PatentMIA*, outperformed the original technique in their experiments.

- **Polarized Augment Calibration (PAC) [45] (White/Black Box)** PAC detects overfitted training samples by leveraging confidence discrepancies in local regions through data augmentation. It introduces a new evaluation score, polarized distance, to correct biases in global confidence metrics. A black-box probability extraction algorithm enables PAC to even operate on LLMs with restricted probability access.

Performance Analysis

Performance analysis does not introduce distinct methods but examines performance differences between potentially contaminated and clean benchmarks. This section briefly overviews existing studies in this area and differentiates between general NLP benchmark analysis and math and coding benchmark analysis.

- **General NLP Benchmark Performance Analysis**
  Li et al. [68] show that LLMs, particularly the GPT-3 series, achieve significantly higher performance on non-mathematical NLP QA tasks released prior to their training cutoff, surpassing majority baselines in over 80% of cases. However, their performance declines substantially on tasks introduced after the cutoff. Jiang et al. [69] further demonstrate that retraining GPT-2 models on datasets with different contamination levels yields modest but consistent performance gains. Similar behavior can be observed with medical data [70].

- **Math and Coding Benchmark Performance Analysis**
  Roberts et al. [56] show that GPT-4 fails to solve any Project Euler problems released after its training cutoff, highlighting a strong dependence on pretraining exposure. Similarly, Ranaldi et al. [57] observe comparable effects for GPT-3.5

on Text-to-SQL tasks. Mirzadeh et al. [40] propose *GSM-Symbolic*, a variant of *GSM8K*, to study mathematical reasoning under controlled perturbations. Models perform substantially worse on *GSM-Symbolic*, suggesting potential contamination or benchmark-specific training for *GSM8K*. Similarly, Zhang et al. [38], using GSM1K, report that while many LLMs exhibit significant performance drops, frontier models such as GPT-4 and Gemini 1.5 Pro are less affected. Hosseini et al. [39], using compositional GSM, find that factors beyond contamination may also contribute to performance degradation. Supplementary analyses conducted by LLM providers themselves report that contamination has little to no impact on mathematical benchmark performance [46], [47], [71].

In contrast, malicious performance boosting appears to be relatively easy, as shown by Tu et al. [7], even in the presence of existing decontamination methods [4], [13].

2.3.3 Limitation of current Data Contamination Detection Approaches

Although various contamination detection methods have been proposed, their effectiveness in identifying mathematical benchmark contamination remains unclear, especially from inadvertent inclusion during pretraining. Most studies simulate contamination through finetuning [7]–[9], [58] rather than analyzing natural contamination from large unsupervised pretraining datasets. Others are not rigorously tested on mathematical text. [18], [49], [51].

Duan et al. [14] find that existing methods for detecting contamination in the Pile corpus [72] using Pythia LLMs [73] perform scarcely better than random guessing. Similarly, Fu et al. [6] demonstrate that key assumptions underlying closed contamination detection methods often fail in practice, raising concerns about their reliability.

To address these limitations, we introduce MathCONTA, a dataset designed to examine mathematical contamination in realistic pretraining settings. Our study proceeds along two lines of inquiry. In **RQ1**, we evaluate the performance of existing data detection methods on MathCONTA. In contrast, **RQ2** shifts the focus from detection to impact and investigates how subtle pretraining contamination affects answer correctness.

CHAPTER 3

Methodology

3.1 Dataset Preparation - MathCONTA

3.1.1 Overview and Purpose

In this section, we introduce *MathCONTA*, a carefully curated dataset consisting of 100 mathematical question-answer pairs. *MathCONTA* is designed to study contamination phenomena in LLMs, specifically focusing on how models behave when presented with known versus novel problems.

The dataset is evenly split into two subsets:

- **50 contaminated samples** that are known to appear in an existing pretraining dataset, namely OpenWebMath [15] (OWM). For more information on OWM please see section 2.1.2.
- **50 clean samples** that are absent from the same dataset and highly unlikely to appear in any current training corpora.

Both subsets are balanced across four categories of mathematical problems and cover a range of difficulty levels. MathCONTA aims to serve as a controlled environment for investigating LLM response patterns when dealing with potentially contaminated examples versus clean examples.

3.1.2 Constructing Contaminated Samples

The contaminated subset of MathCONTA comprises 50 question-solution pairs, spanning diverse mathematical problems—from straightforward word problems to challenging contest-level questions—all exhibiting an exact 1:1 overlap with content in OWM.

For **word problems**, we employed a keyword-based filtering strategy, initially targeting terms such as “Question”, “Solution”, and “word problems” within OWM. However, identifying valid pairs proved challenging, as some documents (each document ID corresponds to a URL on the web) contain solutions behind inaccessible interactive elements (e.g., hidden sections or expandable links), complicating direct extraction. We refer to Appendix (Findings on OWM) for details.

Therefore, we prioritized reputable sources in OWM known for their complete and well-structured question-solution pairs. For contest mathematics, specifically **AMC 8**[1] and **AIME**[2] problems, we primarily utilized documents from *Art of Problem Solving (AoPS)*[3], which offers consistently formatted content, including recent competition problems. This consistency supports stylistic uniformity and facilitates comparative analyses in our contamination experiments. Thus, our first three sample categories are **word problems**, **AMC 8** and **AIME**.

The fourth sample category, **Forum**, was drawn from the *Math StackExchange*[4] domain within OWM. We selected problems with clear, fully worked-out solutions, excluding open-ended or unresolved discussions. Since forum threads typically follow the structure [Question, Attempt1, CommentA, CommentB, . . ., Solution1, Solution2, . . .], we reduced them to [Question, Solution1], with the omitted content stored separately in the column `L_omit` of the *MathCONTA* dataset.

Across all categories, we applied the following inclusion criteria:

- Problems were required to be in English and self-contained, solvable without external resources.
- Each problem included a step-by-step, written solution; final answers alone were deemed insufficient.
- Problems heavily reliant on diagrams or graphs were excluded
- All texts were extracted verbatim, preserving LaTeX syntax, formatting quirks, and typographical inconsistencies.

3.1.3 Constructing Clean Samples

The clean set consists of 50 question-answer pairs, specifically selected to be absent from OWM and unlikely to appear in standard pretraining datasets, in particular those on which the LLMs we investigated were trained on. These clean samples match the difficulty levels of their corresponding contaminated samples. Common LaTeX formatting quirks observed in the contaminated examples—such as the use of double “$$” signs or LaTeX

[1] `https://maa.org/student-programs/amc/`
[2] `https://maa.org/maa-invitational-competitions/`
[3] `https://artofproblemsolving.com/`
[4] `https://math.stackexchange.com/`

commands (e.g., `frac`) within double backslashes—were also replicated in the clean samples. To minimize stylistic discrepancies, we aimed to sample the clean examples from the same sources as the contaminated ones. However, for **word problems**, this was not feasible; consequently, all clean examples (and their solutions) were authored by a non-native English speaker.

Clean Sample Criteria: Cleanness was verified via a naive exact match search against OWM. However, each sample meets at least one of the following additional criteria specified in the column *CLEAN_REASON* in *MathCONTA* (a description of all columns can be found in the Appendix (Column Descriptions for MathCONTA)):

1. Sourced from offline, German-language textbooks
2. Written or adapted by the thesis author (including translations).
3. Taken from online posts published after 2024 and, therefore, after the training data cutoff of investigated LLMs (exact dates provided).

Table 3.1: Distribution of Examples Across Groups (Total = 100)

Category	**Typical Difficulty**	**Contaminated**	**Clean**
Word Problems	1-2	12	12
AMC 8	2-3	18	18
AIME	4	10	10
Forum	3-4	10	10

Table 3.2: Description of Difficulty Levels

Level	**Description**
1	Basic arithmetic and simple word problems (~Grade < 8)
2	Lower-intermediate contest problems (~Grade < 10)
3	Higher-intermediate contest problems (~Grade < 12)
4	Advanced contest problems (~Grade ≥ 12)

A critical reflection on the inherent limitation of our approach to constructing the *MathCONTA* dataset can be found in chapter 6.

3.2 Contamination Detection Methods Under Evaluation

While the pretraining corpus in our experiments is available and accessible, we focus exclusively on methods that operate without its use, known as closed-data methods. This allows us to better simulate real-world scenarios where access to training data may be restricted due to privacy concerns, proprietary constraints, or practical limitations. Access to the training data is only required to determine the ground-truth labels in the dataset (see 3.1); however, the methods themselves do not utilize this data.

We selected methods based on the following criteria: they must (1) be closed-data, (2) operate at the instance level, and (3) not require access to hidden model states. To reduce redundancy, we include only one representative per core approach. Some methods are adapted or simplified to better fit our dataset and experimental goals, with details provided in the method descriptions below. While our selection aims to cover a diverse and representative set, it is not intended to be comprehensive.

We evaluate each method (A) on its standalone effectiveness for mathematical content and (B) in combination with others using majority voting to assess potential gains in accuracy. Here, we categorize the methods based on their access settings (White-Box vs Black-Box 1.3).

3.2.1 White-Box Access Methods

This section outlines the white-box access methods used in our evaluation, as defined in 1.3. These methods assume access to the model's next-token probability distribution. We briefly summarize their implementation and any modifications made to our experiments. For detailed theoretical descriptions, we refer to Section 2.3.2.

All white-box methods rely on access to token-level probability information. In our setup, we obtained these probabilities directly from the model's output logits, which were converted to probabilities via the softmax function. Depending on the method, we use the raw token probability $p(x_i \mid x_1, \ldots, x_{i-1})$ or its logarithm (log-likelihood).

- **MinK% Method [18]**

  We implement the MinK% method as originally proposed, without modifications. The method computes token-level probabilities from the model and identifies the lowest $k\%$ of tokens in a given sequence. These low-probability tokens are used to estimate the likelihood that the sequence was part of the model's training data.

  We evaluate the method using multiple values of $k \in \{5, 10, 20, 30\}$ and we optimize the decision threshold θ using stratified random cross-validation (Experimental Setup in 3.3).

  No adaptations were required for our use case, and all token probabilities were extracted directly from the model's output using softmax-normalized logits.

- **Conta Traces [51]**

  We implemented the Conta Traces method by computing the cumulative log-likelihood of each token in a sequence and fitting an exponential curve of the form $f(x) = -A(1 - e^{-Bx})$, as proposed by the original authors. Contrary to their setup, we experiment with applying the method to both questions and answers. A high value of B or a low value of A indicates contamination.

(Adapted Method) To explore the effect of curve-fitting assumptions, we also introduce a variant based on linear fitting. In this version, we fit a linear function to the cumulative log-likelihood curve and extract a single slope parameter m. This approach, referred to as *ContaTracesLinear*, serves as a simplified alternative to the exponential model.

For both variants, we fit the parameters A, B and m and optimize their respective decision threshold using cross-validation.

Since the cumulative log-likelihood is directly related to perplexity (as shown in section 2.3.2), we consider Conta Traces representative of all perplexity-based methods for our experiments.

- **N-gram Log-Likelihood (White-Box Variant) (Adapted Method)**

  This method is based on the n-gram accuracy metric introduced by [8]. In our work, we adapt the metric for white-box access by replacing model-generated predictions with the log-likelihood of the ground-truth n-gram, computed using the model's internal token probability distribution.

  Concretely, for each sequence $\mathbf{X}$ and a set of $n_{\text{starts}} = 20$ randomly sampled starting positions start$_j$, we extract the subsequent n-gram and compute the average log-likelihood:

$$\text{AvgLL}_j = \frac{1}{n} \sum_{i=0}^{n-1} \log p(x_{\text{start}_j+i} \mid x_1, \ldots, x_{\text{start}_j+i-1})$$

  This yields a per-n-gram score that reflects the model's familiarity with the sequence. Normalizing by n ensures comparability across different n-gram lengths and reduces scale variance in the resulting scores.

  We test this method using $n \in \{1, 2, 3, 5, 7\}$ as a hyperparameter and optimize a threshold θ on the average log-likelihood using cross-validation. We hypothesize that this variant allows for more nuanced detection of memorization than its counterpart `N-gram accuracy` because it better leverages the available white-box access.

3.2.2 Black-Box Access Methods

Below, we discuss black-box access methods as defined in 1.3.

- **Contamination Detection via Output Distribution (CDD) [9]**

  We implement the CDD method following the original formulation, using a sample size of 50 responses per prompt. The method computes the edit distance between each sampled response and a greedy response (temperature $t = 0$) and then models these distances as a discrete distribution. The cumulative distribution function calculates a peakedness score, a contamination indicator.

We experiment with $\alpha \in \{0, 0.1, 0.2, 0.3\}$ to control the maximum distance threshold relative to response length and optimized θ as a decision threshold for contamination. Both parameters are selected using cross-validation. Furthermore, we limit model responses to 200 tokens for this method for computing efficiency.

- **N-gram Accuracy [8]**

  We implement the n-gram accuracy metric as proposed by [8], which evaluates contamination based on the model's ability to reproduce short n-gram sequences given a prefix prompt. For each sequence $\mathbf{X}$, we randomly sample $n_{\text{starts}} = 20$ starting positions. At each position, the preceding text is used as the prompt, and the model is queried to predict the following n tokens. The prediction is compared to the ground truth n-gram using an exact match criterion.

  While the original work employs this metric as part of a larger pipeline that compares benchmark-level differences between training and test data, we isolate and adapt the atomic N-gram Accuracy component to evaluate contamination at the instance level. Specifically, we treat the match threshold—originally fixed at 0.5—as a tunable parameter θ and optimize it via cross-validation.

  Additionally, we assess the method's sensitivity across multiple n-gram lengths, using $n \in \{1, 2, 3, 5, 7\}$.

- **N-gram CDD (Adapted Method)**

  We propose a variant of the CDD method that operates on short n-gram continuations rather than full-length outputs, allowing for finer-grained analysis of response variability.

  For each sequence $\mathbf{X}$, we choose $n_{starts} = 3$ prompt positions, one in the first part of $\mathbf{X}$, one in the middle, and one in the last part.

  1. **Sampling Continuations:** From each prompt position, we generate 50 model continuations of a maximal length of n (we use $n \in \{10, 20, 30\}$) with the model's default generation parameters. Additionally, we generate a greedy continuation from the same prompt using temperature $t = 0$.
  2. **Edit Distance Distribution:** We then compute the token-level edit distance d between each sampled n-gram and the greedy n-gram. Let $\rho_i^*(d)$ denote the empirical probability distribution over these distances for the i-th prompt position.
  3. **Peakedness Score:** We define the cumulative peakedness score exactly as proposed for CDD [9] for each prompt as the proportion of sampled continuations whose edit distance to the greedy baseline falls within a relative threshold:

  $$\text{Peak}_{n,i}(\mathbf{X}) = \sum_{d=0}^{\lfloor \alpha \cdot n \rfloor} \rho_i^*(d),$$

  where $\alpha \in \{0, 0.1, 0.2, 0.3\}$ controls the maximum acceptable deviation.

4. **Aggregation:** At last, we compute the final contamination score for $\mathbf{X}$ by averaging over all n_{starts} positions:

$$\mathrm{CDD}_n(\mathbf{X}) = \frac{1}{n_{\text{starts}}} \sum_{i=1}^{n_{\text{starts}}} \mathrm{Peak}_{n,i}(\mathbf{X}).$$

A contamination decision is made by comparing $\mathrm{CDD}_n(\mathbf{X})$ to a threshold θ, which is selected via cross-validation.

Name	**Box Type**	**Parameters**	**Type**
MinK% Method [18]	White	$k \in \{5, 10, 20, 30\}$; threshold θ (cross-validated)	Unmodified
Conta Traces [51]	White	A, B for exponential fit; thresholds θ_A, θ_B cross-validated	Unmodified
Conta Traces (Linear)	White	Slope m of linear fit; cross-validated	Adapted
N-gram Log-Likelihood	White	$n \in \{1, 2, 3, 5, 7\}$; threshold θ (cross-validated)	Adapted
CDD [9]	Black	$\alpha \in \{0, 0.1, 0.2, 0.3\}$; threshold θ (cross-validated)	Unmodified
N-gram Accuracy [8]	Black	$n \in \{1, 2, 3, 5, 7\}$; match threshold θ (cross-validated)	Unmodified
N-gram CDD	Black	$\alpha \in \{0, 0.1, 0.2, 0.3\}$, $n \in \{10, 20, 30\}$; threshold θ (cross-validated)	Adapted

Table 3.3: Evaluated contamination detection methods, categorized by model access type and parameterization. The *Type* column indicates whether the method is sourced from prior work or adapted by the thesis author.

3.3 Detection Experiments on MathCONTA

This section outlines the experimental setup used to evaluate contamination detection methods on the MathCONTA dataset. Our goal is to assess method performance for individual LLMs and across LLMs in combination. We also investigate whether combining detection methods improves results.

3.3.1 Dataset Split and Cross-Validation

The MathCONTA dataset consists of 100 carefully constructed examples. Before applying any methods, we perform a single stratified 70/30 split, resulting in 70 examples for finding the best parameters and 30 for testing. This split is fixed and used across all LLMs, meaning that each method produces predictions on the same 30 test examples.

Across all experiments, we use 5-fold stratified cross-validation on the 70 training examples to tune method-specific parameters (e.g., thresholds, n-gram sizes, k-parameter). The configuration that yields the highest average validation accuracy is selected. This process is repeated independently for each LLM and method family.

3.3.2 Per-LLM Evaluation

The following four LLMs are used in our evaluation and are described in detail in Section 2.1.4. Since each model has been pretrained at least once on *OWM*, the labels in *MathCONTA* are valid for all of them:

- **DS**: `deepseek-ai/deepseek-math-7b-instruct`
- **LE**: `EleutherAI/llemma_7b`
- **OL**: `allenai/OLMo-7B-0724-Instruct-hf`
- **OL2**: `allenai/OLMo-2-1124-13B-Instruct`

We apply the detection methods to the 70 training examples for each of the four LLMs and evaluate performance on the shared 30 test examples. We select the best-performing method based on cross-validation accuracy from each of the six method families (both Conta Traces methods are grouped). These six models are refitted on all 70 training samples and evaluated on the 30 test examples for that LLM.

We also evaluate a simple majority-vote ensemble of the six selected models. Each model casts one vote, and a majority determines the predicted label, with ties resolved at random.

For each LLM, we report the mean validation accuracy with standard deviation from cross-validation, the test accuracy based on 30 test examples, and 95% confidence intervals estimated via bootstrapping ($n = 1000$). Using McNemar's test [16], statistical significance is assessed against a simple baseline (random guessing). Methods are marked as significant if the p-value is below $\frac{0.05}{6} \approx 0.01$, accounting for the multiple testing problem.

3.3.3 Combined Evaluation Across LLMs

To assess generalization across LLMs, we construct a combined dataset of **400 samples** by pairing each of the 70 examples with outputs from four different LLMs. We apply the same 70/30 split at the example level, yielding 280 training samples and 120 test samples in total.

Using the 280 training samples, we perform 5-fold stratified cross-validation to tune parameters and select the best-performing method from each family. These six models are refitted on all 280 samples and evaluated on the 120 held-out test samples.

3.4 LLM Performance Gap Evaluation on MathCONTA

In addition to evaluating contamination detection methods, we assess the mathematical problem-solving performance of five LLMs on the MathCONTA dataset. The four open-source models previously analyzed are included in this evaluation, now extended with GPT-4o [46] — a proprietary model representing the current state of the art of non-reasoning models.

Unlike the open-source models, for which contamination with MathCONTA's "contaminated" subset is confirmed, no such confirmation is possible for GPT-4o due to the lack of transparency in their training data or procedure. However, given that the contaminated subset consists of widely available, high-quality web content, it remains a possibility that GPT-4o was indirectly exposed to this material as well. Therefore, this remains speculative, and we treat the clean/contaminated split with caution when interpreting GPT-4o's results.

All models are evaluated using a consistent 1-shot Chain-of-Thought prompting setup, where a single solved example precedes the target problem to encourage step-by-step reasoning. Responses are manually assessed, and a model receives one point only if the final answer is exactly correct—no partial credit is awarded.

We report accuracy scores separately for the contaminated and clean subsets. This allows us to assess overall performance and potential differences in behavior across models when solving problems with or without prior exposure. We compute the relative performance gap between the two subsets for each model and assess statistical significance using the Mann–Whitney U test [17].

CHAPTER 4

Results

4.1 Contamination Detection Results on MathCONTA

In this chapter, we present the results of evaluating several contamination detection methods (see Table 3.3 for details) on the *MathCONTA* dataset. We conduct all experiments on the following four LLMs, for which we know the labels in *MathCONTA* are valid, as they are all pretrained at least once on *OWM*. For details on the LLMs, we refer to Chapter 2.1.4:

- DS: `deepseek-ai/deepseek-math-7b-instruct`
- LE: `EleutherAI/llemma_7b`
- OL: `allenai/OLMo-7B-0724-Instruct-hf`
- OL2: `allenai/OLMo-2-1124-13B-Instruct`

All experiments follow a 70/30 train-test split, using stratified sampling to preserve the distribution of clean and contaminated samples across categories. Model performance is evaluated using 5-fold CV on the training split. We select the best-performing configuration for each method family based on mean validation accuracy and then test it on the held-out set. Confidence intervals (95%) are computed using bootstrapping, and statistical significance is assessed via McNemar's test [16]. A p-value below 0.01 is considered significant, with multiple testing adjustments applied. We set the baseline to 0.5 accuracy, which is achieved by only predicting one class or in expectation with random guessing.

4.1.1 Combined Evaluation Across LLMs

To assess the generalization of detection methods across model architectures, we constructed a combined dataset of 400 samples. This was achieved by pairing each of 70 unique examples with generations from the four LLMs: DS (7B), LE (7B), OL (7B), and OL2 (13B). Using 280 samples for training and 120 for testing, we evaluated the best-performing methods based on the mean validation accuracy from CV from each family.

Across the combined evaluation (Table 4.1), the highest test accuracy was achieved by the min$K$ method at 56%, followed closely by n-gram accuracy at 53%. However, none of the methods yielded a p-value below 0.01, and confidence intervals for all methods overlapped with the 0.5 baseline, indicating no statistically significant improvement over the 0.5 baseline.

Table 4.1: Across-LLM results: 5-fold CV validation accuracy ± standard deviation, test accuracy, 95% CI, and McNemar p-value vs. baseline (test set, $n = 120$).

Method	CV Acc ±	Test Acc	95% CI	p (McN)
baseline	-	0.50	-	-
minK	0.55 ± 0.07	0.56	[0.48–0.64]	0.337
ContaTraces	0.56 ± 0.04	0.50	[0.33–0.67]	1.000
CDD	0.53 ± 0.02	0.48	[0.39–0.58]	0.925
ngram-acc	0.60 ± 0.08	0.53	[0.39–0.58]	0.925
ngram-loglike	0.63 ± 0.03	0.49	[0.40–0.58]	1.000
ngram-cdd	0.55 ± 0.14	0.48	[0.38–0.57]	0.581

4.1.2 Per-LLM Evaluation

Each LLM was evaluated independently using 70 training and 30 test examples in the per-model setting. The same contamination detection methods were applied and tuned separately for each model. While we expected improved performance due to method adaptation to specific LLMs, this hypothesis is not supported, as no method significantly outperforms the 0.5 accuracy baseline (see Table 4.2 and Figure 4.1)

Methods on DS showed the highest performance overall, with ngram-loglike achieving 0.60 and minK reaching 0.63 test accuracy, though neither was statistically significant.

Similarly, the best-performing methods on LE are minK and ngram-loglike, which achieved both 0.63 test accuracy.

Methods on OL and OL2 performed worse overall. On OL, the best accuracy was 0.53 (e.g., ngram-cdd), and on OL2, no method exceeded 0.53. ContaTraces on OL performed poorly, with a test accuracy of just 0.30.

Notably, CDD often returned `peak_CDD` values of 0 (across all tested α), effectively defaulting to a single-class prediction due to lack of separability.

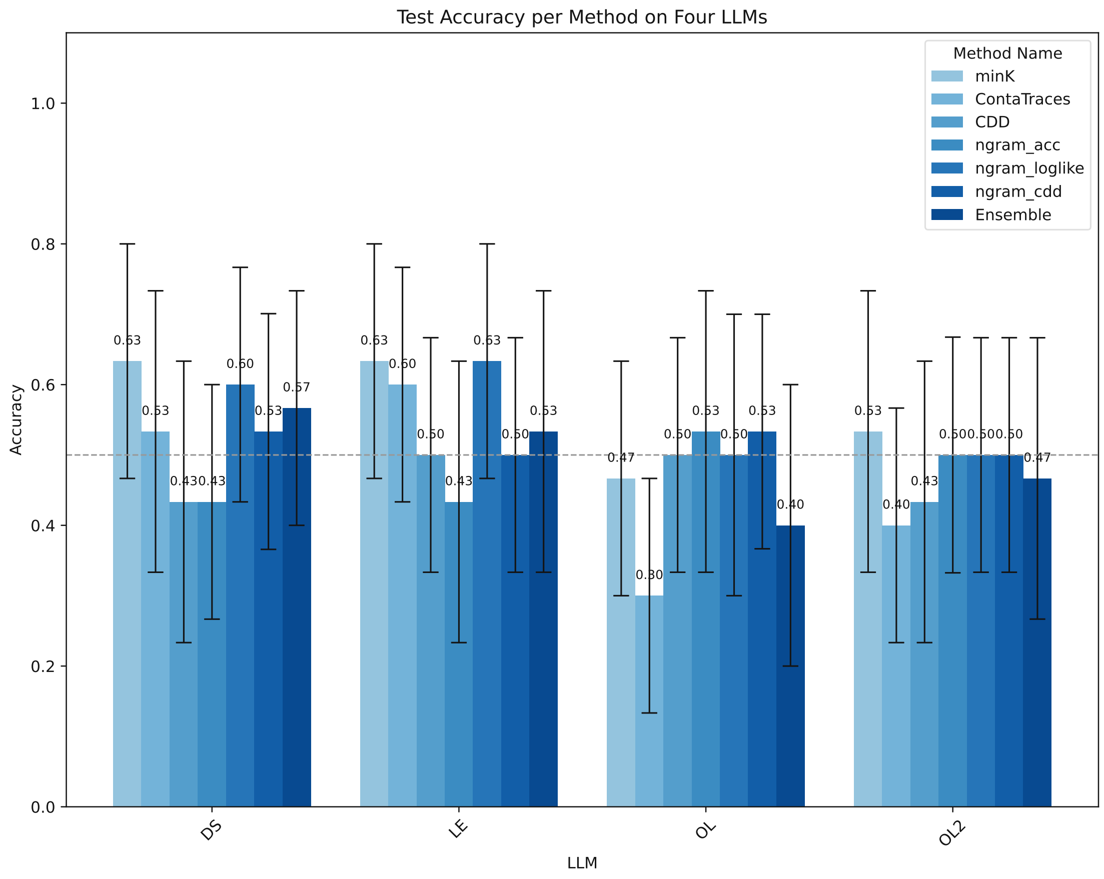


Figure 4.1: Test set accuracy (95% CI) of the methods evaluated separately on four LLMs (test set, $n = 30$). The baseline accuracy of 0.5 is shown as a dotted line, which lies within all confidence intervals.

We refer to Chapter 5.1 for discussions and implications of these results.

4.2 LLM Performance Evaluation on MathCONTA

This section presents the evaluation of LLM performance on the MathCONTA dataset, now treated as a mathematical benchmark. In contrast to the previous section, which focused on contamination detection, we now investigate the practical impact of subtle pretraining contamination on problem-solving abilities. The four open-source LLMs under evaluation are:

- DS: `deepseek-ai/deepseek-math-7b-instruct`
- LE: `EleutherAI/llemma_7b`

Table 4.2: Per-LLM results: 5-fold CV validation accuracy ± standard deviation, test accuracy, 95% CI, and McNemar p-value vs. baseline (test set, $n = 30$).

LLM	**Method**	**CV Acc ±**	**Test Acc**	**95% CI**	**p (McN)**
-	baseline	-	0.50	-	-
DS	minK	0.66 ± 0.08	0.63	[0.47–0.80]	0.388
	ContaTraces	0.54 ± 0.07	0.53	[0.37–0.73]	1.000
	CDD	0.56 ± 0.18	0.43	[0.27–0.63]	0.824
	ngram-acc	0.63 ± 0.05	0.43	[0.27–0.60]	0.754
	ngram-loglike	0.67 ± 0.13	0.60	[0.43–0.77]	0.549
	ngram-cdd	0.64 ± 0.14	0.53	[0.37–0.70]	1.000
	Ensemble	–	**0.57**	**[0.40–0.73]**	**0.727**
LE	minK	0.74 ± 0.09	0.63	[0.47–0.80]	0.388
	ContaTraces	0.61 ± 0.12	0.60	[0.43–0.77]	0.581
	CDD	0.50 ± 0.00	0.50	[0.33–0.67]	1.000
	ngram-acc	0.66 ± 0.09	0.43	[0.27–0.63]	0.727
	ngram-loglike	0.66 ± 0.08	0.63	[0.47–0.80]	0.289
	ngram-cdd	0.60 ± 0.12	0.50	[0.30–0.67]	1.000
	Ensemble	–	**0.53**	**[0.37–0.73]**	**1.000**
OL	minK	0.53 ± 0.07	0.47	[0.30–0.63]	1.000
	ContaTraces	0.56 ± 0.12	0.30	[0.17–0.47]	0.263
	CDD	0.50 ± 0.00	0.50	[0.33–0.67]	1.000
	ngram-acc	0.47 ± 0.07	0.53	[0.37–0.73]	1.000
	ngram-loglike	0.59 ± 0.07	0.50	[0.33–0.70]	1.000
	ngram-cdd	0.57 ± 0.06	0.53	[0.37–0.70]	1.000
	Ensemble	–	**0.40**	**[0.23–0.60]**	**0.607**
OL2	minK	0.69 ± 0.10	0.53	[0.37–0.73]	1.000
	ContaTraces	0.49 ± 0.11	0.40	[0.23–0.57]	0.581
	CDD	0.54 ± 0.14	0.43	[0.27–0.63]	0.832
	ngram-acc	0.49 ± 0.03	0.50	[0.33–0.67]	1.000
	ngram-loglike	0.60 ± 0.07	0.50	[0.33–0.67]	1.000
	ngram-cdd	0.51 ± 0.05	0.50	[0.30–0.67]	1.000
	Ensemble	–	**0.47**	**[0.30–0.67]**	**1.000**

- OL: `allenai/OLMo-7B-0724-Instruct-hf`
- OL2: `allenai/OLMo-2-1124-13B-Instruct`

4.2.1 Combined Performance Evaluation

The MathCONTA dataset comprises 100 diverse mathematical problems, resulting in a total of $n = 400$ evaluations (200 per subset: Contaminated and Clean). Accuracies are reported alongside 95% CIs, calculated using the standard error of the mean (SEM). The results are aggregated across all four open-source LLMs.

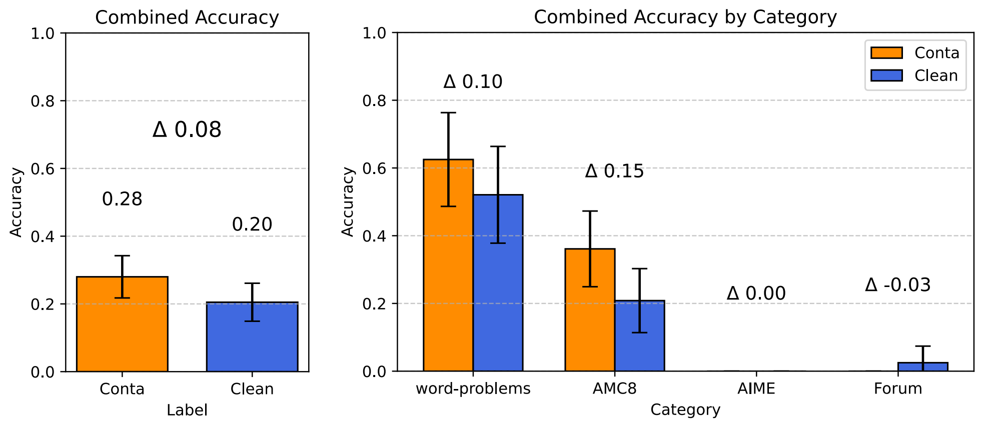


Figure 4.2: Combined accuracy shows a performance gap of 0.08 on all four LLMs (left). The gap is larger in easier problem categories and negligible in more difficult ones (right).

The overall accuracy across models is 0.24. When broken down by contamination status, models achieve an accuracy of 0.28 on the contaminated problems and 0.20 on the clean problems, corresponding to a performance gap of 0.08.

Accuracy also varies substantially by problem category: models attain an accuracy of 0.57 on word problems, 0.28 on AMC8 problems, 0.01 on Forum problems, and 0.00 on AIME problems. As anticipated, performance declines with increasing problem difficulty.

For the easier categories (word problems and AMC8), a small but consistent performance gap between the contaminated and clean subsets is observed. This effect disappears for the more challenging categories (AIME and Forum), with models exhibiting near-zero performance across both subsets.

The results are summarized visually in Figure 4.2. Although the confidence intervals overlap, suggesting that the observed gap is not statistically significant, the pattern remains robust across problem types.

4.2.2 Per LLM Performance Evaluation

Table 4.3: Performance results across models: Accuracy on contaminated and clean datasets, performance gap, and p-value (Mann–Whitney U test).

Model	Acc Conta	Acc Clean	Perf Gap	p (MWU)
DS	0.36	0.30	0.06	0.528
LE	0.22	0.10	0.12	0.105
OL	0.12	0.06	0.06	0.300
OL2	0.42	0.36	0.06	0.543
GPT4o*	0.64	0.56	0.08	0.419

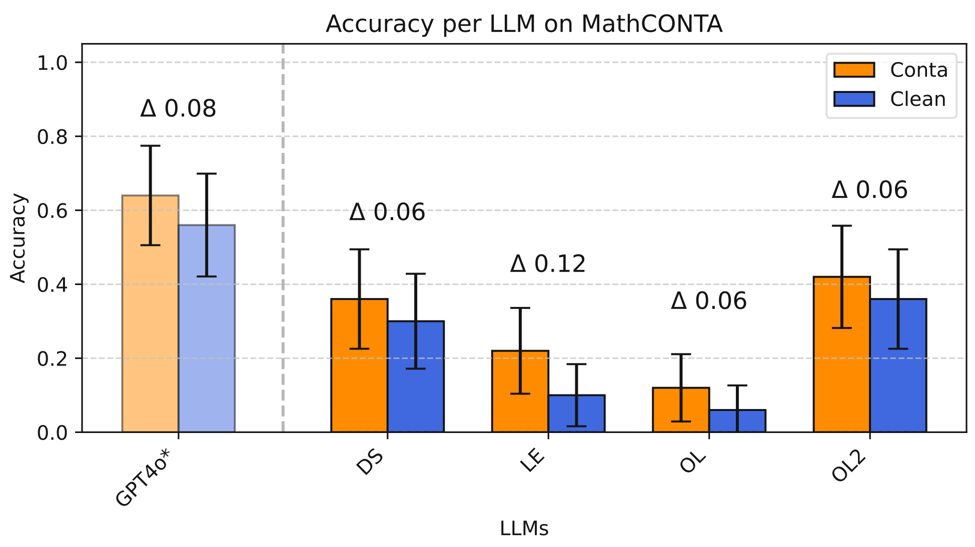


Figure 4.3: Accuracy per LLM on MathCONTA. Modest but consistent performance gains are observed for contaminated samples, though not statistically significant. GPT-4o* is a special case, as its potential pretraining on OWM is unconfirmed, leaving the contamination status uncertain.

In this section, the performance of each LLM is individually analyzed, now including GPT-4o as a representative of the current state of the art. As noted in Section 3.4, GPT-4o's results must be interpreted cautiously, as we cannot definitively confirm or rule out its exposure to the contaminated subset.

Figure 4.3 and Table 4.3 summarize the model-specific results. For each model, we report the accuracies on the contaminated and clean subsets, the corresponding performance gap, and the p-value from a Mann-Whitney U (MWU) test assessing the statistical significance of the gap.

Across all models, performance on the contaminated subset exceeds that on the clean subset. However, the observed gaps are modest, ranging from 6 to 12 percentage points, and none reach statistical significance ($p > 0.05$ for all models).

Among the open-source models, LE exhibits the largest performance gap (12 percentage points), with a marginal p-value ($p = 0.105$). DS, OL, and OL2 display smaller, less variable gaps of approximately 6 percentage points.

For GPT-4o, overall accuracy is substantially higher, achieving 0.64 on the contaminated subset and 0.56 on the clean subset. The performance gap remains modest (8 percentage points) and is not statistically significant ($p = 0.419$).

We refer to Chapter 5.2 for discussions and implications of these results.

CHAPTER 5

Discussion

5.1 Discussion of Detection Results

Our results indicate that existing closed-data contamination detection methods struggle to identify subtle pretraining contamination in the *MathCONTA* dataset, as illustrated by two examples on OL2 shown in the Appendix (Sample Instances from MathCONTA). Across both combined and per-LLM evaluations, no method significantly outperformed the baseline classifier, which always predicts only one class.

This is surprising given the prior success of methods such as min$K$, CDD, and n-gram accuracy. However, earlier work typically addressed more noticeable forms of contamination, such as detecting entire books [18] or scenarios with a higher contamination count inserting through fine-tuning [9]. **In contrast, *MathCONTA* targets subtle, incidental mathematical contamination arising naturally in large-scale pretraining.**

Addressing **RQ1:** *How effectively can current closed-data contamination detection methods identify whether mathematical benchmark questions—drawn from a dataset with an equal mix of pretraining-seen and novel items—are contaminated or uncontaminated?*

The findings indicate that, under these conditions, evaluated methods cannot distinguish between contaminated and uncontaminated items reliably. This highlights the need for new approaches specifically designed to handle low-signal contamination settings. It also supports the recommendation by the authors of [74] that developers should report train-test overlap statistics when pretraining data cannot be disclosed, as they acknowledge the difficulty of detecting contamination in closed-data settings.

The additional analysis points to underfitting as a key issue: cross-validation train and validation accuracies were similar. They remained near the baseline for most contamination methods, indicating the models failed to learn a meaningful contamination signal. We refer to all cross-validation results in the Appendix (Contamination Detection CV Results Details).

In our experiments, increasing model size had minimal impact. Methods tested on OL2 (13B) slightly outperformed those on OL (7B) but did not consistently surpass other 7B models. While larger models may be more prone to memorization [11], potentially aiding contamination detection, our results do not conclusively support this. Future investigation with even larger models could be insightful.

CDD failed on LE and OL, where excessive edit distances resulted in `peak_CDD` values of zero, regardless of contamination status.

Our own adaptations of existing methods, as well as the ensembles, did not improve performance. This is likely because they inherit the same core assumptions as the original methods—which are poorly suited for the subtle contamination scenarios posed by *MathCONTA*. As such, the limitations of existing approaches also extend to our proposed variants.

In summary, current investigated closed-data detection methods fail to identify subtle mathematical pretraining contamination in benchmarks like *MathCONTA*. This underscores the need for new methods suited to low-signal settings, consistent with prior findings [6], [14].

5.2 Discussion of LLM Performance Evaluation

Our evaluation on the MathCONTA dataset shows that subtle pretraining contamination can improve performance on mathematical benchmarks, but the effect is limited. The contaminated subset contains problems with valid solutions copied directly from the LLMs' pretraining corpus. In principle, no reasoning or paraphrasing is required—verbatim reproduction would suffice for correct answers. Nevertheless, even under these conditions, the observed accuracy gain was modest (6–12 percentage points) and statistically insignificant ($p > 0.05$).

Addressing **RQ2:** *To what extent does pretraining contamination influence the accuracy of large language models on mathematical benchmark questions from the same dataset (used in RQ1)?*

Our results suggest that pretraining contamination has a relatively limited effect on overall model accuracy. While it leads to a consistent modest numerical improvement, this effect is neither high nor statistically significant. Thus, contamination alone does not account for most model performance on MathCONTA and similar mathematical benchmarks.

Furthermore, contamination primarily benefits model performance at difficulty levels where models already demonstrate some capacity to solve problems. Our results indicate that contamination helps slightly at easier or moderately challenging problems but does not substantially assist models in addressing significantly more complicated mathematical tasks. Factors such as pretraining data size, computational resources, and model scale are known to have a major impact on model performance [75]. This is further supported by our observation that models trained on different corpora and with different procedures—despite all including OWM—show substantial performance variability, indicating that contamination alone cannot account for the differences.

Interestingly, GPT-4o, despite its substantially larger size and greater resource investment, achieves only about twice the accuracy of the smaller, more resource-efficient open-source model OL2. This suggests limitations in even large-scale, multifunctional LLMs regarding extremely difficult mathematical tasks. Such limitations align with prior literature, where even advanced models like GPT-4o achieve modest accuracy ($\approx 9\%$) on challenging tasks, such as those in AIME competitions [76], regardless of testing on openly available competition data.

As of early 2025, a new paradigm of reasoning-focused models has emerged, demonstrating significantly improved performance on complex mathematical tasks. For instance, DeepSeek-R1 achieves approximately 79.8% accuracy on the AIME 2024 competition [76], indicating considerable progress compared to previous models. Investigating how detection methods perform on these advanced reasoning models within the MathCONTA framework represents an intriguing and valuable direction for future research (see Chapter 6 on current limitations).

CHAPTER 6

Limitations and Future Work

This chapter discusses the limitations of the thesis and suggests directions for future work.

Limitations of *MathCONTA*

- **Clean Sample Validity:** Exact-match searches confirm that the wording of clean samples does not appear in OWM. Although their presence in other datasets cannot be entirely excluded, it is unlikely due to an additional clean criterion: the sample must be (1) sourced from a German offline book, (2) written or modified by the author, or (3) created after 2024. Paraphrased variants may remain undetected, as noted in prior work [4] and online observations.[1] Clean samples may also face future contamination following the release of *MathCONTA*.

- **Formatting Variations across Examples:** Despite efforts to standardize formatting between clean and contaminated samples, minor inconsistencies may persist due to differences in source documents and the author's non-native English usage. These stylistic factors could influence detection accuracy. We encourage future research to evaluate how such variation affects model behavior by testing against diverse styles (e.g., LLM-generated, native, and non-native solutions).

- **Contamination Frequency:** Some contaminated examples (e.g., math contest problems) appear multiple times in OWM. Our current evaluation does not account for how repeated exposure affects detection; this remains an open question for future study.

- **Manual Curation Effort:** The benchmark consists of 100 carefully curated examples. This limited size reflects the labor-intensive process of locating valid

[1]`https://x.com/DimitrisPapail/status/1888325914603516214`

contaminated examples. Clean sample selection involved manual topic and difficulty alignment, followed by validation. The decision to exclude LLMs from construction was intentional to reduce bias. While this ensured high quality, it limited scalability. Future work could explore partial automation of the pipeline to expand the dataset.

- **Domain-Specific Scaling:** A promising direction is to focus on a single mathematical domain aligned with model failure boundaries. This could enable more controlled experiments (see Appendix (Contamination Detection - Output Metric Distribution)). However, due to our limited dataset size per category, such domain-specific analysis was not feasible in the present study.

Variability in Pretraining Procedures We based our analysis on the reported presence of OWM in the pretraining corpora. However, we did not account for potential variability in how contamination occurred, such as the specific training stage or learning rate. Although some information was available, a deeper investigation would have required access to training codebases and detailed data tracking, which was not possible for most of the evaluated models. Fully controlled pretraining experiments, which would require retraining models from scratch, would provide greater comparability but were beyond the computational resources available for this study.

Model Scale and Selection We evaluated four relatively small open-source models (ranging from 7B to 13B parameters). Future work should expand testing to larger models. At the time of writing (early 2025), new reasoning-focused models such as DeepSeek-R1 [76] have demonstrated significant improvements in mathematical reasoning tasks. Evaluating contamination detection methods on these models represents an important direction for future research.

Detection Methods and Model Accessibility Our selection of contamination detection methods relies solely on natural language responses and token probability distributions, enabling broad applicability to both open- and closed-source models. Recent work, such as DICE [7], leverages hidden states to detect contamination during fine-tuning, and similar techniques may be effective for pretraining. As contamination detection remains an active research area, further methods should be explored, including new calibration methods, such as Cross-Entropy-based calibration [44] or Polarized Augment Calibration [45], and improvements of existing methods like minK%++ [53].

Inadvertent vs. Malicious Contamination This study addresses subtle contamination scenarios, where contamination happens inadvertently due to the massive size of pretraining data sources. Future work should also investigate detection methods for malicious contamination, where actors intentionally manipulate training data to inflate perceived model performance on publicly known benchmarks, as demonstrated in [4] using rephrased samples.

CHAPTER 7

Conclusion

This thesis examined the effectiveness of closed-data contamination detection methods and the influence of subtle pretraining contamination on LLMs using the *MathCONTA* dataset.

For **RQ1**, we found that existing methods such as min$K$, CDD, and n-gram accuracy failed to detect subtle mathematical contamination, often performing no better than a trivial baseline. This aligns with observations from prior work [14], [74], indicating that current techniques, designed for detecting more obvious contamination, are poorly suited for low-signal scenarios like *MathCONTA*, in which contamination occurs naturally during pretraining.

For **RQ2**, our evaluation showed that while contamination provides a modest accuracy gain (6–12 percentage points), the improvement is statistically insignificant and primarily limited to easier problems.

However, several limitations affect the generality of these conclusions. The small size of the MathCONTA dataset (100 examples) and variability in pretraining practices constrain the scope of our findings. Additionally, evaluations were limited to smaller models (7B–13B parameters), excluding the newest reasoning-specialized systems like DeepSeek-R1 [76]. Future work should scale up datasets, investigate contamination in larger or more advanced models, and explore new methods using hidden states and advanced calibration techniques [7], [53].

Despite the insight that the impact of undetectable contamination on overall model performance appears to be limited, benchmark scores—especially static ones—should be interpreted cautiously. As noted in several studies [4], [13], malicious contamination remains a genuine threat to the credibility of benchmark-based evaluations.

At the same time, state-of-the-art models continue to perform strongly across both static and dynamic benchmarks [27], [41]–[43]. Their increasing success in real-world applications suggests that their capabilities extend beyond memorization and may reflect emergent reasoning or intelligence [77].

To conclude, it is clear that we are only at the beginning of a profound transformation. Or, as Wharton professor Ethan Mollick put it in his book Co-Intelligence, we should “assume that this AI is the worst AI you will ever use” [78].

With this thesis, I hope to have contributed a small piece to the ongoing effort to understand better how these models work and how we might evaluate them more reliably.

Overview of Generative AI Tools Used

Generative AI tools were used solely as supportive instruments in this thesis (and, of course, as test objects in the experiments).

GPT-4o by OpenAI (version dated 2024-11-20)[1] and **Grammarly**[2] were employed for grammar checking and for rephrasing individual sentences to enhance clarity and academic style, without altering the author's original arguments or ideas.

In addition, **GPT-4o** (version dated 2024-11-20) provided assistance with debugging, generating visualization code, and writing function docstrings.

All AI-generated output was used cautiously and served only as a basis for further refinement by the author.

[1] `https://www.chatgpt.com`
[2] `https://www.grammarly.com`

List of Figures

List of Tables

Bibliography

[1] M. Ravaut, B. Ding, F. Jiao, H. Chen, X. Li, R. Zhao, *et al.*, *How Much are LLMs Contaminated? A Comprehensive Survey and the LLMSanitize Library*, arXiv:2404.00699 [cs], Mar. 2024. [Online]. Available: `http://arxiv.org/abs/2404.00699` (visited on 07/14/2024).

[2] C. Deng, Y. Zhao, Y. Heng, Y. Li, J. Cao, X. Tang, *et al.*, *Unveiling the Spectrum of Data Contamination in Language Models: A Survey from Detection to Remediation*, arXiv:2406.14644 [cs], Jun. 2024. [Online]. Available: `http://arxiv.org/abs/2406.14644` (visited on 07/03/2024).

[3] C. Xu, S. Guan, D. Greene, and M.-T. Kechadi, *Benchmark Data Contamination of Large Language Models: A Survey*, en, arXiv:2406.04244 [cs], Jun. 2024. [Online]. Available: `http://arxiv.org/abs/2406.04244` (visited on 06/26/2024).

[4] S. Yang, W.-L. Chiang, L. Zheng, J. E. Gonzalez, and I. Stoica, *Rethinking Benchmark and Contamination for Language Models with Rephrased Samples*, en, arXiv:2311.04850 [cs], Nov. 2023. [Online]. Available: `http://arxiv.org/abs/2311.04850` (visited on 05/31/2024).

[5] Z. Zhou, S. Liu, M. Ning, W. Liu, J. Wang, D. F. Wong, *et al.*, *Is Your Model Really A Good Math Reasoner? Evaluating Mathematical Reasoning with Checklist*, arXiv:2407.08733 [cs], Jul. 2024. [Online]. Available: `http://arxiv.org/abs/2407.08733` (visited on 07/14/2024).

[6] Y. Fu, O. Uzuner, M. Yetisgen, and F. Xia, *Does Data Contamination Detection Work (Well) for LLMs? A Survey and Evaluation on Detection Assumptions*, arXiv:2410.18966 [cs], Oct. 2024. DOI: `10.48550/arXiv.2410.18966`. [Online]. Available: `http://arxiv.org/abs/2410.18966` (visited on 02/28/2025).

[7] S. Tu, K. Zhu, Y. Bai, Z. Yao, L. Hou, and J. Li, *DICE: Detecting In-distribution Contamination in LLM's Fine-tuning Phase for Math Reasoning*, arXiv:2406.04197 [cs], Sep. 2024. [Online]. Available: `http://arxiv.org/abs/2406.04197` (visited on 10/31/2024).

[8] R. Xu, Z. Wang, R.-Z. Fan, and P. Liu, *Benchmarking Benchmark Leakage in Large Language Models*, arXiv:2404.18824 [cs], Apr. 2024. [Online]. Available: `http://arxiv.org/abs/2404.18824` (visited on 08/15/2024).

[9] Y. Dong, X. Jiang, H. Liu, Z. Jin, B. Gu, M. Yang, *et al.*, *Generalization or Memorization: Data Contamination and Trustworthy Evaluation for Large Language Models*, arXiv:2402.15938 [cs], May 2024. [Online]. Available: `http://arxiv.org/abs/2402.15938` (visited on 10/06/2024).

[10] S. Chen, Y. Chen, Z. Li, Y. Jiang, Z. Wan, Y. He, *et al.*, *Recent Advances in Large Langauge Model Benchmarks against Data Contamination: From Static to Dynamic Evaluation*, arXiv:2502.17521 [cs], Feb. 2025. DOI: `10.48550/arXiv.2502.17521`. [Online]. Available: `http://arxiv.org/abs/2502.17521` (visited on 02/28/2025).

[11] K. Tirumala, A. H. Markosyan, L. Zettlemoyer, and A. Aghajanyan, *Memorization Without Overfitting: Analyzing the Training Dynamics of Large Language Models*, arXiv:2205.10770 [cs], Nov. 2022. DOI: `10.48550/arXiv.2205.10770`. [Online]. Available: `http://arxiv.org/abs/2205.10770` (visited on 04/13/2025).

[12] N. Carlini, D. Ippolito, M. Jagielski, K. Lee, F. Tramer, and C. Zhang, *Quantifying Memorization Across Neural Language Models*, arXiv:2202.07646 [cs], Mar. 2023. [Online]. Available: `http://arxiv.org/abs/2202.07646` (visited on 06/01/2024).

[13] J. Dekoninck, M. N. Müller, M. Baader, M. Fischer, and M. Vechev, *Evading Data Contamination Detection for Language Models is (too) Easy*, arXiv:2402.02823 [cs], Feb. 2024. [Online]. Available: `http://arxiv.org/abs/2402.02823` (visited on 06/26/2024).

[14] M. Duan, A. Suri, N. Mireshghallah, S. Min, W. Shi, L. Zettlemoyer, *et al.*, *Do Membership Inference Attacks Work on Large Language Models?*, arXiv:2402.07841 [cs], Sep. 2024. DOI: `10.48550/arXiv.2402.07841`. [Online]. Available: `http://arxiv.org/abs/2402.07841` (visited on 04/27/2025).

[15] K. Paster, M. D. Santos, Z. Azerbayev, and J. Ba, *OpenWebMath: An Open Dataset of High-Quality Mathematical Web Text*, arXiv:2310.06786 [cs], Oct. 2023. [Online]. Available: `http://arxiv.org/abs/2310.06786` (visited on 11/03/2024).

[16] Q. McNemar, "Note on the Sampling Error of the Difference Between Correlated Proportions or Percentages", en, *Psychometrika*, vol. 12, no. 2, pp. 153–157, Jun. 1947, ISSN: 0033-3123, 1860-0980. DOI: `10.1007/BF02295996`. [Online]. Available: `https://www.cambridge.org/core/product/identifier/S0033312300045178/type/journal_article` (visited on 04/28/2025).

[17] H. B. Mann and D. R. Whitney, "On a Test of Whether one of Two Random Variables is Stochastically Larger than the Other", *The Annals of Mathematical Statistics*, vol. 18, no. 1, pp. 50–60, Mar. 1947, Publisher: Institute of Mathematical Statistics, ISSN: 0003-4851, 2168-8990. DOI: `10.1214/aoms/1177730491`. [Online]. Available: `https://projecteuclid.org/journals/annals-of-mathematical-statistics/volume-18/issue-1/On-a-Test-of-Whether-one-of-Two-Random-Variables/10.1214/aoms/1177730491.full` (visited on 04/27/2025).

[18] W. Shi, A. Ajith, M. Xia, Y. Huang, D. Liu, T. Blevins, *et al.*, *Detecting Pretraining Data from Large Language Models*, arXiv:2310.16789, Mar. 2024. [Online]. Available: `http://arxiv.org/abs/2310.16789` (visited on 10/23/2024).

[19] W. X. Zhao, K. Zhou, J. Li, T. Tang, X. Wang, Y. Hou, *et al.*, *A Survey of Large Language Models*, arXiv:2303.18223 [cs], Nov. 2023. [Online]. Available: `http://arxiv.org/abs/2303.18223` (visited on 04/13/2024).

[20] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, *et al.*, *Attention Is All You Need*, arXiv:1706.03762 [cs], Aug. 2023. DOI: `10.48550/arXiv.1706.03762`. [Online]. Available: `http://arxiv.org/abs/1706.03762` (visited on 01/06/2025).

[21] T. B. Brown, B. Mann, N. Ryder, M. Subbiah, J. Kaplan, P. Dhariwal, *et al.*, *Language Models are Few-Shot Learners*, arXiv:2005.14165, Jul. 2020. DOI: `10.48550/arXiv.2005.14165`. [Online]. Available: `http://arxiv.org/abs/2005.14165` (visited on 12/27/2024).

[22] D. Groeneveld, I. Beltagy, P. Walsh, A. Bhagia, R. Kinney, O. Tafjord, *et al.*, *OLMo: Accelerating the Science of Language Models*, arXiv:2402.00838 [cs], Jun. 2024. DOI: `10.48550/arXiv.2402.00838`. [Online]. Available: `http://arxiv.org/abs/2402.00838` (visited on 09/26/2024).

[23] L. Soldaini, R. Kinney, A. Bhagia, D. Schwenk, R. Authur, B. Bogin, *et al.*, *Dolma: An Open Corpus of Three Trillion Tokens for Language Model Pretraining Research*, arXiv:2402.00159 [cs], Jun. 2024. DOI: `10.48550/arXiv.2402.00159`. [Online]. Available: `http://arxiv.org/abs/2402.00159` (visited on 09/26/2024).

[24] T. OLMo, P. Walsh, L. Soldaini, D. Groeneveld, K. Lo, S. Arora, *et al.*, *2 OLMo 2 Furious*, arXiv:2501.00656 [cs], Dec. 2024. DOI: `10.48550/arXiv.2501.00656`. [Online]. Available: `http://arxiv.org/abs/2501.00656` (visited on 01/09/2025).

[25] Z. Shao, P. Wang, Q. Zhu, R. Xu, J. Song, X. Bi, *et al.*, *DeepSeekMath: Pushing the Limits of Mathematical Reasoning in Open Language Models*, arXiv:2402.03300 [cs], Apr. 2024. DOI: `10.48550/arXiv.2402.03300`. [Online]. Available: `http://arxiv.org/abs/2402.03300` (visited on 09/26/2024).

[26] Z. Azerbayev, H. Schoelkopf, K. Paster, M. D. Santos, S. McAleer, A. Q. Jiang, *et al.*, *Llemma: An Open Language Model For Mathematics*, arXiv:2310.10631 [cs], Mar. 2024. DOI: `10.48550/arXiv.2310.10631`. [Online]. Available: `http://arxiv.org/abs/2310.10631` (visited on 01/11/2025).

[27] W.-L. Chiang, L. Zheng, Y. Sheng, A. N. Angelopoulos, T. Li, D. Li, *et al.*, *Chatbot Arena: An Open Platform for Evaluating LLMs by Human Preference*, arXiv:2403.04132 [cs], Mar. 2024. DOI: `10.48550/arXiv.2403.04132`. [Online]. Available: `http://arxiv.org/abs/2403.04132` (visited on 03/01/2025).

[28] L. Zheng, W.-L. Chiang, Y. Sheng, S. Zhuang, Z. Wu, Y. Zhuang, *et al.*, *Judging LLM-as-a-Judge with MT-Bench and Chatbot Arena*, arXiv:2306.05685 [cs], Dec. 2023. DOI: 10.48550/arXiv.2306.05685. [Online]. Available: http://arxiv.org/abs/2306.05685 (visited on 03/01/2025).

[29] K. Cobbe, V. Kosaraju, M. Bavarian, M. Chen, H. Jun, L. Kaiser, *et al.*, *Training Verifiers to Solve Math Word Problems*, en, arXiv:2110.14168 [cs], Nov. 2021. [Online]. Available: http://arxiv.org/abs/2110.14168 (visited on 06/05/2024).

[30] D. Hendrycks, C. Burns, S. Kadavath, A. Arora, S. Basart, E. Tang, *et al.*, *Measuring Mathematical Problem Solving With the MATH Dataset*, en, arXiv:2103.03874 [cs], Nov. 2021. [Online]. Available: http://arxiv.org/abs/2103.03874 (visited on 05/26/2024).

[31] S. Frieder, L. Pinchetti, A. Chevalier, R.-R. Griffiths, T. Salvatori, T. Lukasiewicz, *et al.*, *Mathematical Capabilities of ChatGPT*, arXiv:2301.13867 [cs], Jul. 2023. [Online]. Available: http://arxiv.org/abs/2301.13867 (visited on 04/13/2024).

[32] A. Patel, S. Bhattamishra, and N. Goyal, *Are NLP Models really able to Solve Simple Math Word Problems?*, arXiv:2103.07191 [cs], Apr. 2021. DOI: 10.48550/arXiv.2103.07191. [Online]. Available: http://arxiv.org/abs/2103.07191 (visited on 04/25/2025).

[33] Z. Yuan, H. Yuan, C. Tan, W. Wang, and S. Huang, *How well do Large Language Models perform in Arithmetic tasks?*, arXiv:2304.02015 [cs], Mar. 2023. DOI: 10.48550/arXiv.2304.02015. [Online]. Available: http://arxiv.org/abs/2304.02015 (visited on 04/25/2025).

[34] W. Zhong, R. Cui, Y. Guo, Y. Liang, S. Lu, Y. Wang, *et al.*, *AGIEval: A Human-Centric Benchmark for Evaluating Foundation Models*, arXiv:2304.06364 [cs], Sep. 2023. DOI: 10.48550/arXiv.2304.06364. [Online]. Available: http://arxiv.org/abs/2304.06364 (visited on 04/25/2025).

[35] T. Sawada, D. Paleka, A. Havrilla, P. Tadepalli, P. Vidas, A. Kranias, *et al.*, *ARB: Advanced Reasoning Benchmark for Large Language Models*, arXiv:2307.13692 [cs], Jul. 2023. DOI: 10.48550/arXiv.2307.13692. [Online]. Available: http://arxiv.org/abs/2307.13692 (visited on 04/25/2025).

[36] C. He, R. Luo, Y. Bai, S. Hu, Z. L. Thai, J. Shen, *et al.*, *OlympiadBench: A Challenging Benchmark for Promoting AGI with Olympiad-Level Bilingual Multimodal Scientific Problems*, arXiv:2402.14008 [cs], Jun. 2024. DOI: 10.48550/arXiv.2402.14008. [Online]. Available: http://arxiv.org/abs/2402.14008 (visited on 04/25/2025).

[37] S. Frieder, M. Olšák, J. Berner, and T. Lukasiewicz, "The IMO Small Challenge: Not-Too-Hard Olympiad Math Datasets for LLMs", en, Mar. 2024. [Online]. Available: https://openreview.net/forum?id=HYuKXhgzjm (visited on 04/25/2025).

[38] H. Zhang, J. Da, D. Lee, V. Robinson, C. Wu, W. Song, *et al.*, *A Careful Examination of Large Language Model Performance on Grade School Arithmetic*, arXiv:2405.00332 [cs], Nov. 2024. DOI: 10.48550/arXiv.2405.00332. [Online]. Available: http://arxiv.org/abs/2405.00332 (visited on 03/02/2025).

[39] A. Hosseini, A. Sordoni, D. Toyama, A. Courville, and R. Agarwal, *Not All LLM Reasoners Are Created Equal*, arXiv:2410.01748 [cs], Oct. 2024. DOI: 10.48550/arXiv.2410.01748. [Online]. Available: http://arxiv.org/abs/2410.01748 (visited on 04/25/2025).

[40] I. Mirzadeh, K. Alizadeh, H. Shahrokhi, O. Tuzel, S. Bengio, and M. Farajtabar, *GSM-Symbolic: Understanding the Limitations of Mathematical Reasoning in Large Language Models*, arXiv:2410.05229 [cs], Oct. 2024. DOI: 10.48550/arXiv.2410.05229. [Online]. Available: http://arxiv.org/abs/2410.05229 (visited on 10/18/2024).

[41] C. White, S. Dooley, M. Roberts, A. Pal, B. Feuer, S. Jain, *et al.*, *LiveBench: A Challenging, Contamination-Free LLM Benchmark*, arXiv:2406.19314 [cs], Jun. 2024. DOI: 10.48550/arXiv.2406.19314. [Online]. Available: http://arxiv.org/abs/2406.19314 (visited on 03/01/2025).

[42] I. Petrov, J. Dekoninck, L. Baltadzhiev, M. Drencheva, K. Minchev, M. Balunović, *et al.*, *Proof or Bluff? Evaluating LLMs on 2025 USA Math Olympiad*, arXiv:2503.21934 [cs], Apr. 2025. DOI: 10.48550/arXiv.2503.21934. [Online]. Available: http://arxiv.org/abs/2503.21934 (visited on 05/04/2025).

[43] X. Xu, J. Zhang, T. Chen, Z. Chao, J. Hu, and C. Yang, *UGMathBench: A Diverse and Dynamic Benchmark for Undergraduate-Level Mathematical Reasoning with Large Language Models*, arXiv:2501.13766 [cs], Feb. 2025. DOI: 10.48550/arXiv.2501.13766. [Online]. Available: http://arxiv.org/abs/2501.13766 (visited on 03/01/2025).

[44] W. Zhang, R. Zhang, J. Guo, M. d. Rijke, Y. Fan, and X. Cheng, *Pretraining Data Detection for Large Language Models: A Divergence-based Calibration Method*, arXiv:2409.14781 [cs] version: 1, Sep. 2024. DOI: 10.48550/arXiv.2409.14781. [Online]. Available: http://arxiv.org/abs/2409.14781 (visited on 04/27/2025).

[45] W. Ye, J. Hu, L. Li, H. Wang, G. Chen, and J. Zhao, *Data Contamination Calibration for Black-box LLMs*, arXiv:2405.11930 [cs], Jun. 2024. DOI: 10.48550/arXiv.2405.11930. [Online]. Available: http://arxiv.org/abs/2405.11930 (visited on 04/27/2025).

[46] OpenAI, J. Achiam, S. Adler, S. Agarwal, L. Ahmad, I. Akkaya, *et al.*, *GPT-4 Technical Report*, arXiv:2303.08774 [cs], Mar. 2024. [Online]. Available: http://arxiv.org/abs/2303.08774 (visited on 10/05/2024).

[47] Llama-Team, A. Dubey, A. Jauhri, A. Pandey, A. Kadian, A. Al-Dahle, *et al.*, *The Llama 3 Herd of Models*, arXiv:2407.21783 [cs], Aug. 2024. [Online]. Available: http://arxiv.org/abs/2407.21783 (visited on 10/05/2024).

[48] A. N. Lee, C. J. Hunter, and N. Ruiz, *Platypus: Quick, Cheap, and Powerful Refinement of LLMs*, arXiv:2308.07317, Mar. 2024. DOI: `10.48550/arXiv.2308.07317`. [Online]. Available: `http://arxiv.org/abs/2308.07317` (visited on 10/31/2024).

[49] N. Carlini, F. Tramer, E. Wallace, M. Jagielski, A. Herbert-Voss, K. Lee, *et al.*, *Extracting Training Data from Large Language Models*, arXiv:2012.07805 [cs], Jun. 2021. DOI: `10.48550/arXiv.2012.07805`. [Online]. Available: `http://arxiv.org/abs/2012.07805` (visited on 03/12/2025).

[50] I. Magar and R. Schwartz, "Data Contamination: From Memorization to Exploitation", in *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, S. Muresan, P. Nakov, and A. Villavicencio, Eds., Dublin, Ireland: Association for Computational Linguistics, May 2022, pp. 157–165. DOI: `10.18653/v1/2022.acl-short.18`. [Online]. Available: `https://aclanthology.org/2022.acl-short.18/` (visited on 04/25/2025).

[51] N. Yax, H. Anlló, and S. Palminteri, "Studying and improving reasoning in humans and machines", en, *Communications Psychology*, vol. 2, no. 1, p. 51, Jun. 2024, ISSN: 2731-9121. DOI: `10.1038/s44271-024-00091-8`. [Online]. Available: `https://www.nature.com/articles/s44271-024-00091-8` (visited on 10/25/2024).

[52] Y. Li, *Estimating Contamination via Perplexity: Quantifying Memorisation in Language Model Evaluation*, arXiv:2309.10677 [cs], Sep. 2023. [Online]. Available: `http://arxiv.org/abs/2309.10677` (visited on 06/01/2024).

[53] J. Zhang, J. Sun, E. Yeats, Y. Ouyang, M. Kuo, J. Zhang, *et al.*, *Min-K%++: Improved Baseline for Detecting Pre-Training Data from Large Language Models*, arXiv:2404.02936 [cs], Feb. 2025. DOI: `10.48550/arXiv.2404.02936`. [Online]. Available: `http://arxiv.org/abs/2404.02936` (visited on 04/27/2025).

[54] C. Deng, Y. Zhao, X. Tang, M. Gerstein, and A. Cohan, *Investigating Data Contamination in Modern Benchmarks for Large Language Models*, arXiv:2311.09783 [cs], Apr. 2024. [Online]. Available: `http://arxiv.org/abs/2311.09783` (visited on 10/30/2024).

[55] S. Golchin and M. Surdeanu, *Time Travel in LLMs: Tracing Data Contamination in Large Language Models*, arXiv:2308.08493 [cs], Feb. 2024. [Online]. Available: `http://arxiv.org/abs/2308.08493` (visited on 06/26/2024).

[56] M. Roberts, H. Thakur, C. Herlihy, C. White, and S. Dooley, *Data Contamination Through the Lens of Time*, arXiv:2310.10628 [cs], Oct. 2023. [Online]. Available: `http://arxiv.org/abs/2310.10628` (visited on 06/26/2024).

[57] F. Ranaldi, E. S. Ruzzetti, D. Onorati, L. Ranaldi, C. Giannone, A. Favalli, *et al.*, "Investigating the Impact of Data Contamination of Large Language Models in Text-to-SQL Translation", in *Findings of the Association for Computational Linguistics ACL 2024*, arXiv:2402.08100 [cs], 2024, pp. 13 909–13 920. DOI: 10.18653/v1/2024.findings-acl.827. [Online]. Available: http://arxiv.org/abs/2402.08100 (visited on 04/25/2025).

[58] J. Dekoninck, M. N. Müller, and M. Vechev, *ConStat: Performance-Based Contamination Detection in Large Language Models*, arXiv:2405.16281 [cs], May 2024. [Online]. Available: http://arxiv.org/abs/2405.16281 (visited on 07/03/2024).

[59] Y. Oren, N. Meister, N. Chatterji, F. Ladhak, and T. B. Hashimoto, *Proving Test Set Contamination in Black Box Language Models*, arXiv:2310.17623 [cs], Nov. 2023. [Online]. Available: http://arxiv.org/abs/2310.17623 (visited on 06/01/2024).

[60] J.-l. Gailly and M. Adler, "Zlib compression library", Dec. 2004.

[61] E. Choi, H. He, M. Iyyer, M. Yatskar, W.-t. Yih, Y. Choi, *et al.*, *QuAC : Question Answering in Context*, arXiv:1808.07036, Aug. 2018. DOI: 10.48550/arXiv.1808.07036. [Online]. Available: http://arxiv.org/abs/1808.07036 (visited on 12/27/2024).

[62] C. Clark, K. Lee, M.-W. Chang, T. Kwiatkowski, M. Collins, and K. Toutanova, *BoolQ: Exploring the Surprising Difficulty of Natural Yes/No Questions*, arXiv:1905.10044, May 2019. DOI: 10.48550/arXiv.1905.10044. [Online]. Available: http://arxiv.org/abs/1905.10044 (visited on 12/27/2024).

[63] P. Rajpurkar, R. Jia, and P. Liang, "Know What You Don't Know: Unanswerable Questions for SQuAD", en, in *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, Melbourne, Australia: Association for Computational Linguistics, 2018, pp. 784–789. DOI: 10.18653/v1/P18-2124. [Online]. Available: http://aclweb.org/anthology/P18-2124 (visited on 12/27/2024).

[64] M. Chen, J. Tworek, H. Jun, Q. Yuan, H. P. d. O. Pinto, J. Kaplan, *et al.*, *Evaluating Large Language Models Trained on Code*, arXiv:2107.03374, Jul. 2021. DOI: 10.48550/arXiv.2107.03374. [Online]. Available: http://arxiv.org/abs/2107.03374 (visited on 12/27/2024).

[65] D. Hendrycks, C. Burns, S. Basart, A. Zou, M. Mazeika, D. Song, *et al.*, *Measuring Massive Multitask Language Understanding*, arXiv:2009.03300 [cs], Jan. 2021. DOI: 10.48550/arXiv.2009.03300. [Online]. Available: http://arxiv.org/abs/2009.03300 (visited on 12/28/2024).

[66] T. Sellam, D. Das, and A. Parikh, "BLEURT: Learning Robust Metrics for Text Generation", in *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, D. Jurafsky, J. Chai, N. Schluter, and J. Tetreault, Eds., Online: Association for Computational Linguistics, Jul. 2020, pp. 7881–7892. DOI: `10.18653/v1/2020.acl-main.704`. [Online]. Available: `https://aclanthology.org/2020.acl-main.704` (visited on 12/28/2024).

[67] C.-Y. Lin, "ROUGE: A Package for Automatic Evaluation of Summaries", in *Text Summarization Branches Out*, Barcelona, Spain: Association for Computational Linguistics, Jul. 2004, pp. 74–81. [Online]. Available: `https://aclanthology.org/W04-1013` (visited on 12/28/2024).

[68] Y. Li, F. Guerin, and C. Lin, *An Open Source Data Contamination Report for Large Language Models*, arXiv:2310.17589 [cs], Jan. 2024. [Online]. Available: `http://arxiv.org/abs/2310.17589` (visited on 07/03/2024).

[69] M. Jiang, K. Z. Liu, M. Zhong, R. Schaeffer, S. Ouyang, J. Han, *et al.*, *Investigating Data Contamination for Pre-training Language Models*, arXiv:2401.06059 [cs], Jan. 2024. [Online]. Available: `http://arxiv.org/abs/2401.06059` (visited on 10/06/2024).

[70] A. Elangovan, J. He, and K. Verspoor, "Memorization vs. Generalization : Quantifying Data Leakage in NLP Performance Evaluation", in *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume*, P. Merlo, J. Tiedemann, and R. Tsarfaty, Eds., Online: Association for Computational Linguistics, Apr. 2021, pp. 1325–1335. DOI: `10.18653/v1/2021.eacl-main.113`. [Online]. Available: `https://aclanthology.org/2021.eacl-main.113/` (visited on 04/27/2025).

[71] H. Touvron, L. Martin, K. Stone, P. Albert, A. Almahairi, Y. Babaei, *et al.*, *Llama 2: Open Foundation and Fine-Tuned Chat Models*, arXiv:2307.09288 [cs], Jul. 2023. DOI: `10.48550/arXiv.2307.09288`. [Online]. Available: `http://arxiv.org/abs/2307.09288` (visited on 04/24/2025).

[72] L. Gao, S. Biderman, S. Black, L. Golding, T. Hoppe, C. Foster, *et al.*, *The Pile: An 800GB Dataset of Diverse Text for Language Modeling*, arXiv:2101.00027 [cs], Dec. 2020. DOI: `10.48550/arXiv.2101.00027`. [Online]. Available: `http://arxiv.org/abs/2101.00027` (visited on 04/27/2025).

[73] S. Biderman, H. Schoelkopf, Q. Anthony, H. Bradley, K. O'Brien, E. Hallahan, *et al.*, *Pythia: A Suite for Analyzing Large Language Models Across Training and Scaling*, arXiv:2304.01373 [cs], May 2023. DOI: `10.48550/arXiv.2304.01373`. [Online]. Available: `http://arxiv.org/abs/2304.01373` (visited on 04/27/2025).

[74] A. K. Zhang, K. Klyman, Y. Mai, Y. Levine, Y. Zhang, R. Bommasani, *et al.*, *Language model developers should report train-test overlap*, arXiv:2410.08385 [cs], Oct. 2024. DOI: `10.48550/arXiv.2410.08385`. [Online]. Available: `http://arxiv.org/abs/2410.08385` (visited on 04/24/2025).

[75] J. Kaplan, S. McCandlish, T. Henighan, T. B. Brown, B. Chess, R. Child, *et al.*, *Scaling Laws for Neural Language Models*, arXiv:2001.08361 [cs, stat], Jan. 2020. [Online]. Available: `http://arxiv.org/abs/2001.08361` (visited on 04/14/2024).

[76] DeepSeek-AI, D. Guo, D. Yang, H. Zhang, J. Song, R. Zhang, *et al.*, *DeepSeek-R1: Incentivizing Reasoning Capability in LLMs via Reinforcement Learning*, arXiv:2501.12948 [cs], Jan. 2025. DOI: `10.48550/arXiv.2501.12948`. [Online]. Available: `http://arxiv.org/abs/2501.12948` (visited on 03/08/2025).

[77] D. Yu, S. Kaur, A. Gupta, J. Brown-Cohen, A. Goyal, and S. Arora, *Skill-Mix: A Flexible and Expandable Family of Evaluations for AI models*, arXiv:2310.17567 [cs], Oct. 2023. [Online]. Available: `http://arxiv.org/abs/2310.17567` (visited on 10/04/2024).

[78] E. Mollick, *Co-intelligence: living and working with AI*, eng. London: W H Allen, 2024, ISBN: 978-0-7535-6077-8.

Appendix

Column Descriptions for MathCONTA

Table 1: Column Descriptions for MathCONTA verbatim

Name	**Explanation**
ID	Unique identifier for the problem.
LABEL	Indicates if the problem is in OWM $\rightarrow$ {Conta}, otherwise $\rightarrow$ {Clean}.
SOURCE	Where the problem was found.
CATEGORY	Broad classification of the problem type.
SUBCATEGORY	More detailed classification.
PROBLEM	Full problem statement.
SOLUTION_OWM	Solution as found in OWM.
SOLUTION_CLEAN	*Clean* solution (criteria in CLEAN_REASON_S).
EXTRACTED_ANSWER	Final answer extracted from the solution.
L_OMIT	Indicates if non-essential text between question and answer was omitted (e.g., omitting comments or attempts), this text can be found here.
L_ANS	Indicates if the solution explains all steps but does not explicitly state the final answer (e.g., missing "The solution is: $x = 10$").

Continued on next page

Name	Explanation
CLEAN_REASON_Q	Reason *clean* question (1 - Offline Book in German, 2 - Written/modified by author, 3 - Created after training cut-off, clean till #dd-mm-YYYY, 4 - checked overlap with OWM).
CLEAN_REASON_S	Reason for *clean* solution (1 - Offline Book in German, 2 - Written/modified by author, 3 - Created after training cut-off, 4 - checked overlap with OWM).
DIFFICULTY_SCORE_A	Difficulty estimation (1-4).
COMMENT_A	Additional comments or notes.

Findings on OWM during MathCONTA creation

While assembling our dataset, we encountered numerous instances where question-answer extraction was not feasible. The table below lists a few random examples of documents excluded due to structural or content-related issues. These examples illustrate common challenges, such as missing answers, questions listed without solutions, or solutions obscured behind interactive elements.

These cases represent only a small subset. *OWM* [15] contains over six million documents, many with valid and well-structured question-answer pairs. Quantifying this effect systematically was beyond the scope of this thesis, but it might be interesting for future research.

Additionally, the phenomenon—where another question rather than an answer follows a question—raises interesting questions for future work: Studying how LLMs are affected when pretraining data frequently contains unanswered questions would be worthwhile. A promising direction could be automatically identifying such instances within large-scale corpora and inserting plausible answers using an LLM back into the corpus. Investigating whether this approach improves mathematical reasoning in pretraining could yield valuable insights.

ID (URL)	Problem Encountered
`https://brilliant.org/problems/a-simple-sum/`	No solution given. Among 60,000 pages from brilliant.org, the first 100 checked contained no answers.
`https://prepfortests.com/gmat/tutorials/wordproblems/tables`	No solution given.
`https://www.mathallstar.org/Practice/Search?Category=TheSpecialValueMethod`	Only questions listed.
`https://www.greenemath.com/Algebra1/24/SystemsofEquationsWordProblemsPracticeTest.html`	Problems and solutions listed separately.
`https://webwork.libretexts.org/webwork2/html2xml?...`	No solution given.
`https://www.calculatorsoup.com/calculators/wordproblems/`	Only questions listed.
`https://www.cheenta.com/read/`	Solutions are cut off with "…" before completion.
`https://www.cheenta.com/amc-10-algebra-previous-year-questions-year-wisenumerates`	Only questions listed.
`https://www.math10.com/problems/polynomial-vocabulary/easy/`	Solutions hidden behind buttons.
`https://www.math10.com/problems/integrals-trigonometric-substitutions/easy/`	Solutions hidden behind buttons.

Table 2: Random Selection of documents in OWM with missing or inaccessible solutions.

Sample Instances from *MathCONTA*

This section presents two examples, one from the word problem category and one from the AIME category. All remaining examples are available on Hugging Face.[3] The examples are quoted verbatim but formatted for clarity. Both are taken from the test split of *MathCONTA*.

Notably, on OLMO-2-13B (OL2), CDD, n-gram accuracy, and n-gram cdd failed to correctly classify `clean-word-7` as clean. Conversely, `clean-aime-7` was falsely identified as contaminated by all methods except CDD. It is also noted that although the methods do not perform better than random guessing, they are still correct half the time despite lacking predictive power.

[3]MathCONTA Dataset: `https://huggingface.co/datasets/Tobstar001/MathCONTA`

Example for word problem (ID clean-word-7)

Question: In the 1st game you scored 2 goals, in the 2nd game you scored 5 goals, in the 3rd game you scored none and in the 4th and 5th games you scored the same number of goals. How many goals n did you score in the last game if you scored an average of 3 goals per game in the last 5 games in total?

Solution:
We can calculate the average as follow:

$$\frac{2+5+0+n+n}{5} = 3$$

This represents the sum of all five values divided by 5, equaling the given average of 3.

Multiplying both sides by 5 to eliminate the fraction and simplifying:

$$7 + 2 \times n = 15$$

and therefore

$$n = 4.$$

4

Example for AIME (ID clean-aime-7)

Question: Find the sum of all positive integers n such that $n+2$ divides the product $3(n+3)(n^2+9)$.

Solution 1

$$\frac{3(n+3)(n^2+9)}{n+2} \in \mathbb{Z}$$

$$\Rightarrow \frac{3(n+2+1)(n^2+9)}{n+2} \in \mathbb{Z}$$

$$\Rightarrow \frac{3(n+2)(n^2+9)+3(n^2+9)}{n+2} \in \mathbb{Z}$$

$$\Rightarrow 3(n^2+9)+\frac{3(n^2+9)}{n+2} \in \mathbb{Z}$$

$$\Rightarrow \frac{3(n^2-4+13)}{n+2} \in \mathbb{Z}$$

$$\Rightarrow \frac{3(n+2)(n-2)+39}{n+2} \in \mathbb{Z}$$

$$\Rightarrow 3(n-2)+\frac{39}{n+2} \in \mathbb{Z}$$

$$\Rightarrow \frac{39}{n+2} \in \mathbb{Z}$$

Since $n+2$ is positive, the positive factors of 39 are $1, 3, 13, 39$.

Therefore, $n = -1, 1, 11, 37$.

Since n is positive, $n = 1, 11, 37$.

$$1+11+37 = \boxed{049}$$

Contamination Detection Results across OLMo-7B-0724 Model Variants

OLMo-7B-0724 [22] is released in three variants: **Base**, **SFT**, and **Instruct** (referred to as **OL** in the main experiments). The **Base** model is pretrained on the Dolma 1.7 dataset [23] for general language understanding. **SFT** extends the Base model through supervised fine-tuning on the Tulu 2 SFT Mix dataset. **Instruct** (OL) further refines SFT by applying Direct Preference Optimization (DPO) and additional fine-tuning on a cleaned UltraFeedback dataset, optimizing it for instruction-following tasks such as chatbot development.

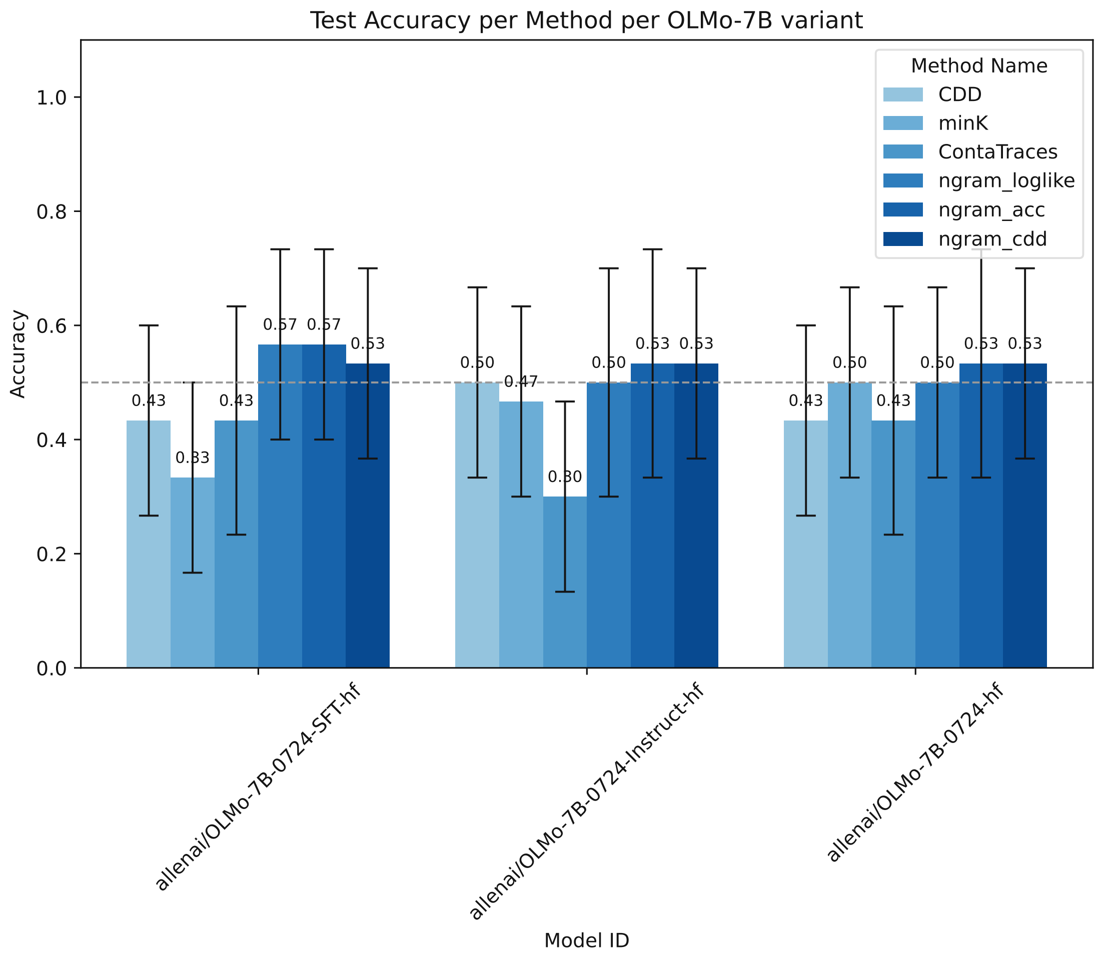


Figure 1: Test set accuracy across OLMo-7B-0724 model variants.

It might be expected that successive fine-tuning stages would make sample detection more challenging. However, our small-scale experiment with the OLMo-7B-0724 variants does not support this hypothesis (see Figure 1). Detection accuracy varies minimally across all models. Overall, performance remains poor, with no model exceeding a baseline accuracy of 0.5, equivalent to random guessing.

Contamination Detection CV Results Details

All experimental results from our cross-validation are presented in Tables 3 and 4. Each entry reports the mean accuracy and the standard deviation (±).

method_name_with_param	**OL2**	**OL**	**DS**	**LE**
minK_k=5	0.69 ± 0.024	0.56 ± 0.017	0.68 ± 0.02	0.77 ± 0.021
ngram_loglike_n=5	0.65 ± 0.018	0.6 ± 0.021	0.69 ± 0.013	0.73 ± 0.028
minK_k=20	0.61 ± 0.017	0.58 ± 0.018	0.67 ± 0.013	0.76 ± 0.021
minK_k=10	0.62 ± 0.024	0.57 ± 0.011	0.69 ± 0.018	0.75 ± 0.021
minK_k=30	0.61 ± 0.026	0.58 ± 0.018	0.65 ± 0.017	0.77 ± 0.02
ngram_loglike_n=2	0.64 ± 0.018	0.58 ± 0.017	0.68 ± 0.03	0.69 ± 0.038
ngram_loglike_n=7	0.6 ± 0.007	0.6 ± 0.027	0.66 ± 0.029	0.7 ± 0.02
ngram_loglike_n=1	0.6 ± 0.033	0.6 ± 0.051	0.69 ± 0.037	0.71 ± 0.026
ContaTraces_fit=exponential	0.57 ± 0.016	0.62 ± 0.032	0.59 ± 0.035	0.69 ± 0.026
ngram_loglike_n=3	0.59 ± 0.031	0.61 ± 0.024	0.67 ± 0.009	0.66 ± 0.021
ContaTraces_fit=linear	0.6 ± 0.009	0.59 ± 0.013	0.63 ± 0.013	0.67 ± 0.013
ngram_acc_n=2	0.53 ± 0.021	0.57 ± 0.021	0.67 ± 0.024	0.7 ± 0.029
ngram_acc_n=1	0.52 ± 0.007	0.55 ± 0.02	0.67 ± 0.014	0.72 ± 0.035
ngram_cdd_alpha=0.3_n=10	0.54 ± 0.02	0.57 ± 0.016	0.61 ± 0.038	0.61 ± 0.029
ngram_acc_n=7	0.53 ± 0.024	0.55 ± 0.016	0.65 ± 0.046	0.61 ± 0.041
ngram_cdd_alpha=0.2_n=10	0.53 ± 0.007	0.56 ± 0.017	0.61 ± 0.038	0.61 ± 0.029
ngram_acc_n=5	0.52 ± 0.007	0.52 ± 0.017	0.63 ± 0.018	0.67 ± 0.035
ngram_cdd_alpha=0.1_n=10	0.51 ± 0.007	0.53 ± 0.018	0.64 ± 0.034	0.59 ± 0.021
ngram_cdd_alpha=0.0_n=10	0.5 ± 0.017	0.54 ± 0.0	0.66 ± 0.04	0.59 ± 0.02
ngram_cdd_alpha=0.3_n=20	0.53 ± 0.007	0.54 ± 0.029	0.62 ± 0.045	0.58 ± 0.021
ngram_cdd_alpha=0.2_n=20	0.51 ± 0.013	0.53 ± 0.024	0.64 ± 0.046	0.56 ± 0.033
ngram_acc_n=3	0.53 ± 0.009	0.52 ± 0.016	0.66 ± 0.013	0.65 ± 0.024
ngram_cdd_alpha=0.1_n=20	0.52 ± 0.024	0.52 ± 0.02	0.62 ± 0.045	0.56 ± 0.027
ngram_cdd_alpha=0.3_n=30	0.51 ± 0.009	0.53 ± 0.009	0.61 ± 0.039	0.55 ± 0.029
ngram_cdd_alpha=0.2_n=30	0.51 ± 0.009	0.52 ± 0.011	0.6 ± 0.041	0.54 ± 0.021
ngram_cdd_alpha=0.1_n=30	0.5 ± 0.007	0.53 ± 0.009	0.58 ± 0.035	0.54 ± 0.02
ngram_cdd_alpha=0.0_n=20	0.51 ± 0.013	0.5 ± 0.007	0.59 ± 0.039	0.54 ± 0.025
CDD_alpha=0.3	0.54 ± 0.035	0.5 ± 0.0	0.6 ± 0.035	0.5 ± 0.0
ngram_cdd_alpha=0.0_n=30	0.51 ± 0.017	0.52 ± 0.011	0.55 ± 0.026	0.51 ± 0.007
CDD_alpha=0.2	0.52 ± 0.011	0.5 ± 0.0	0.59 ± 0.029	0.5 ± 0.0
CDD_alpha=0.1	0.51 ± 0.009	0.5 ± 0.0	0.54 ± 0.027	0.5 ± 0.0
CDD_alpha=0.0	0.5 ± 0.007	0.5 ± 0.0	0.51 ± 0.014	0.5 ± 0.0

Table 3: Mean CV Train Accuracy for various methods/models

Contamination Detection - Output Metric Distribution

Analysis of minK and ngram_loglike on DeepseekMath

As the methods min$K$ and n-gram loglike performed reasonably well on DeepseekMath (DS) (test accuracy around 0.6) with a slight, though not significant, improvement, we provide a detailed overview of the statistics and value distributions across categories for this model and these methods. Due to the small number of examples per category (approximately 25 per category), this analysis is intended purely for informational purposes and to inform future work. No statistical tests were applied. All metrics are based on the full dataset using the optimal hyperparameter settings for each method. Thus, results should be interpreted cautiously: abnormal patterns may arise due to randomness or selection bias.

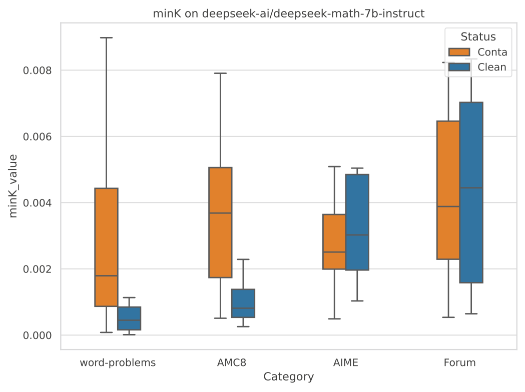
method_name_with_param	OL2	OL	DS	LE
minK_k=5	0.69 ± 0.097	0.53 ± 0.073	0.63 ± 0.07	0.74 ± 0.086
ngram_loglike_n=5	0.56 ± 0.114	0.59 ± 0.07	0.64 ± 0.064	0.66 ± 0.083
minK_k=20	0.53 ± 0.057	0.5 ± 0.09	0.66 ± 0.053	0.73 ± 0.083
ngram_loglike_n=2	0.6 ± 0.073	0.51 ± 0.105	0.59 ± 0.095	0.63 ± 0.177
minK_k=30	0.56 ± 0.105	0.53 ± 0.073	0.57 ± 0.045	0.69 ± 0.097
ngram_acc_n=1	0.49 ± 0.029	0.47 ± 0.073	0.59 ± 0.095	0.66 ± 0.095
ngram_loglike_n=3	0.5 ± 0.078	0.54 ± 0.097	0.56 ± 0.07	0.61 ± 0.097
ngram_loglike_n=1	0.51 ± 0.139	0.44 ± 0.153	0.67 ± 0.132	0.64 ± 0.09
minK_k=10	0.54 ± 0.097	0.46 ± 0.073	0.66 ± 0.083	0.67 ± 0.057
ngram_loglike_n=7	0.54 ± 0.073	0.5 ± 0.111	0.63 ± 0.159	0.64 ± 0.078
ngram_cdd_alpha=0.3_n=10	0.51 ± 0.053	0.57 ± 0.064	0.51 ± 0.114	0.6 ± 0.125
ngram_cdd_alpha=0.2_n=10	0.5 ± 0.0	0.56 ± 0.07	0.59 ± 0.199	0.57 ± 0.101
ngram_cdd_alpha=0.0_n=10	0.46 ± 0.035	0.5 ± 0.0	0.64 ± 0.136	0.54 ± 0.097
ngram_acc_n=2	0.46 ± 0.057	0.47 ± 0.097	0.59 ± 0.105	0.64 ± 0.101
ContaTraces_fit=exponential	0.49 ± 0.114	0.56 ± 0.123	0.49 ± 0.165	0.61 ± 0.125
ngram_cdd_alpha=0.1_n=10	0.5 ± 0.0	0.47 ± 0.035	0.63 ± 0.131	0.46 ± 0.035
ContaTraces_fit=linear	0.47 ± 0.073	0.43 ± 0.0	0.54 ± 0.073	0.59 ± 0.029
CDD_alpha=0.3	0.54 ± 0.14	0.5 ± 0.0	0.56 ± 0.183	0.5 ± 0.0
ngram_acc_n=5	0.44 ± 0.053	0.4 ± 0.073	0.59 ± 0.095	0.61 ± 0.167
ngram_cdd_alpha=0.3_n=20	0.5 ± 0.0	0.5 ± 0.078	0.51 ± 0.159	0.47 ± 0.116
ngram_cdd_alpha=0.1_n=30	0.46 ± 0.086	0.53 ± 0.035	0.5 ± 0.181	0.47 ± 0.132
ngram_cdd_alpha=0.2_n=20	0.43 ± 0.064	0.47 ± 0.035	0.51 ± 0.194	0.47 ± 0.132
ngram_cdd_alpha=0.2_n=30	0.44 ± 0.053	0.46 ± 0.057	0.53 ± 0.132	0.47 ± 0.097
CDD_alpha=0.2	0.46 ± 0.057	0.5 ± 0.0	0.56 ± 0.114	0.5 ± 0.0
ngram_acc_n=7	0.41 ± 0.083	0.47 ± 0.035	0.6 ± 0.19	0.51 ± 0.131
CDD_alpha=0.0	0.49 ± 0.029	0.5 ± 0.0	0.46 ± 0.057	0.5 ± 0.0
CDD_alpha=0.1	0.47 ± 0.035	0.5 ± 0.0	0.5 ± 0.064	0.5 ± 0.0
ngram_acc_n=3	0.41 ± 0.053	0.43 ± 0.064	0.63 ± 0.053	0.57 ± 0.078
ngram_cdd_alpha=0.3_n=30	0.47 ± 0.035	0.51 ± 0.029	0.5 ± 0.136	0.46 ± 0.116
ngram_cdd_alpha=0.0_n=30	0.39 ± 0.14	0.47 ± 0.035	0.46 ± 0.116	0.49 ± 0.029
ngram_cdd_alpha=0.0_n=20	0.39 ± 0.116	0.47 ± 0.057	0.51 ± 0.159	0.47 ± 0.073
ngram_cdd_alpha=0.1_n=20	0.41 ± 0.083	0.43 ± 0.078	0.56 ± 0.171	0.47 ± 0.107

Table 4: Mean CV Validation Accuracy for various methods/models

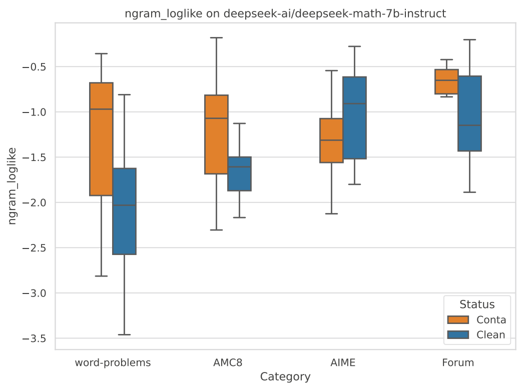
For the AMC8 category and word problems, the differences in values are noticeably more pronounced than in more difficult categories such as AIME and Forum (see Figure 2). This suggests contamination detection may be easier at levels where the language model performs more confidently.

However, two alternative explanations should be considered. First, word problems may be easier to detect because all clean examples were written by a non-native speaker, potentially introducing unintended linguistic biases. This would support the hypothesis that formulation differences are more easily detected than the actual contamination status. However, this explanation can be ruled out for AMC8 since clean and contaminated AMC8 examples were sourced from the same websites. We hypothesize that DS may have been specifically trained on AMC8 examples and has likely encountered them multiple times.

This observation motivates the creation of additional datasets designed to evaluate contamination in AMC8 problems. We intend to pursue this in future work.



minK



ngram_loglike

Figure 2: Boxplots showing the distribution of metric values produced by minK (left) and ngram-loglike (right), grouped by category and label (clean vs. contaminated) on DS. Clearer separation between the groups indicates better discriminative power and facilitates threshold selection.

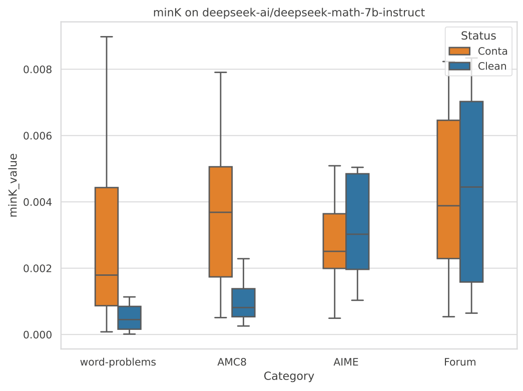
Distribution of Metric Outputs (all)

Figures 3–6 show boxplot visualizations of all evaluated methods across the four LLMs, grouped by the four categories of the *MathCONTA* dataset.

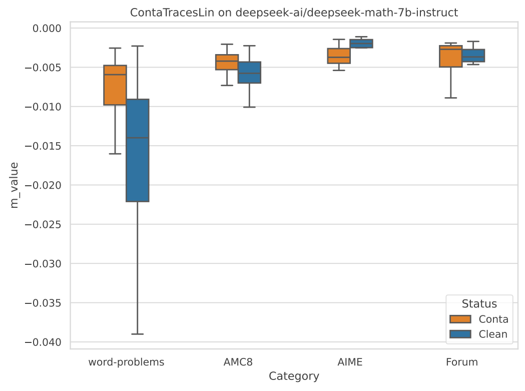
Despite differences in solution length, formulation style, and mathematical difficulty, all categories share a common defining property: each instance is either seen during pretraining or entirely novel. This is the core signal that contamination detection methods aim to capture. However, it is evident that, in general, these methods fail to do so—even without applying any train/test splits.

The trend observed for DS (discussed in the section above) also appears in other models: the word problems and AMC8 categories are generally easier to distinguish than the more challenging AIME and Forum problems—at least for methods such as min$K$, n-gram loglike, and ContaTraces (linear). An exception is seen in OL2, where the category Forum exhibits the highest distributional drift under min$K$.

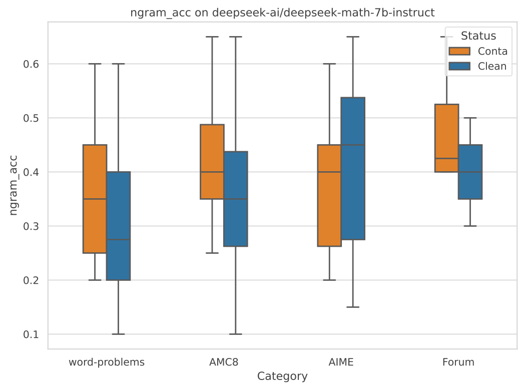
The visualizations also highlight the failure of CDD to provide meaningful results. Our proposed variant, n-gram cdd, shows some improvement but still does not reliably distinguish between contaminated and uncontaminated examples.



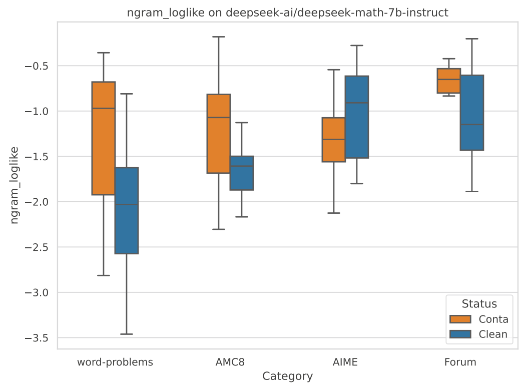
minK



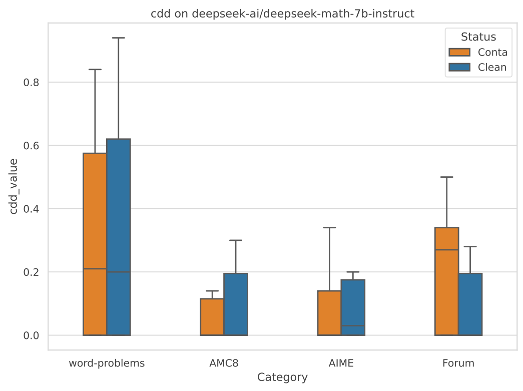
Conta Traces



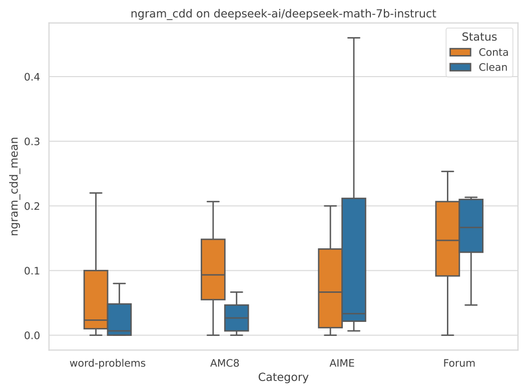
ngram_accuracy



ngram_loglike

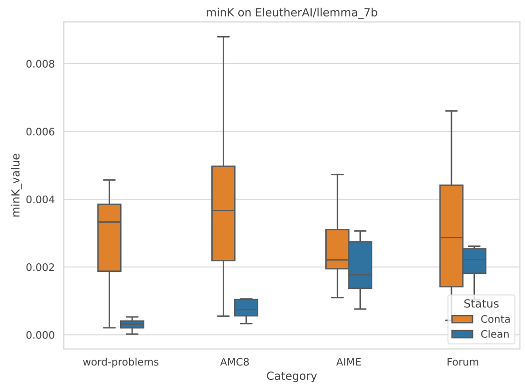


cdd

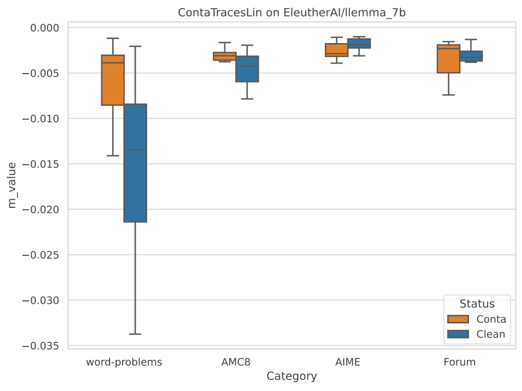


ngram_cdd

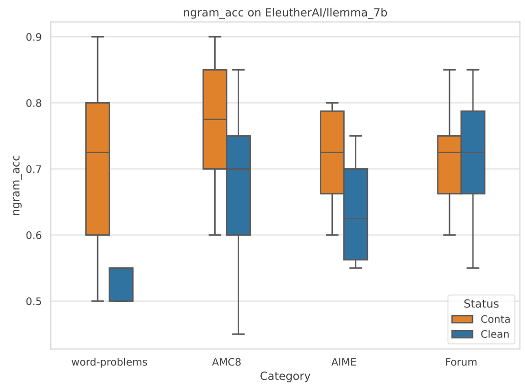
Figure 3: Boxplots showing the distribution of metric values produced by each method, grouped by category and label (clean vs. contaminated) on DS. Clearer separation between the groups indicates better discriminative power and facilitates threshold selection.



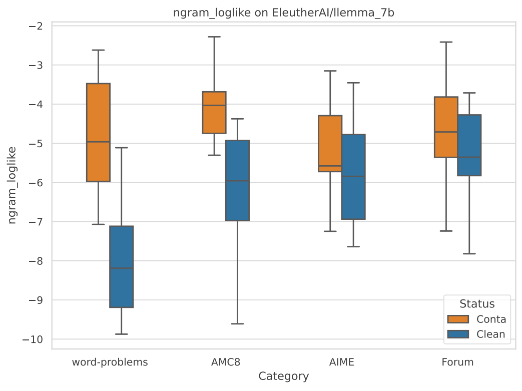
minK



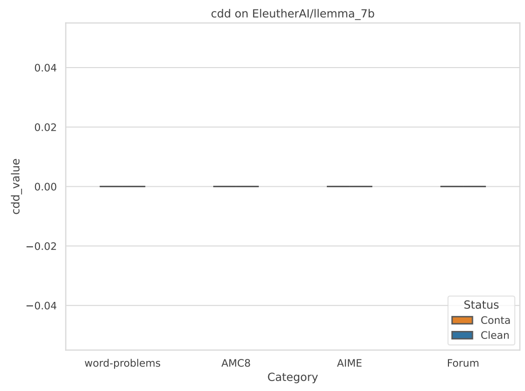
Conta Traces



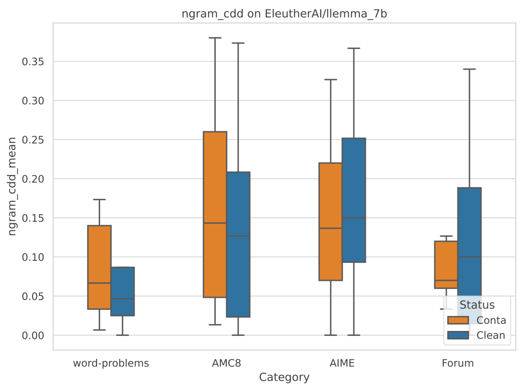
ngram_accuracy



ngram_loglike

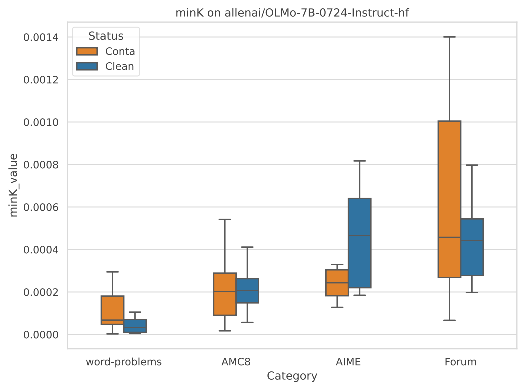


cdd

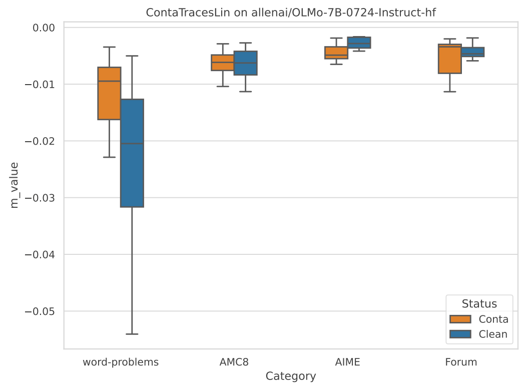


ngram_cdd

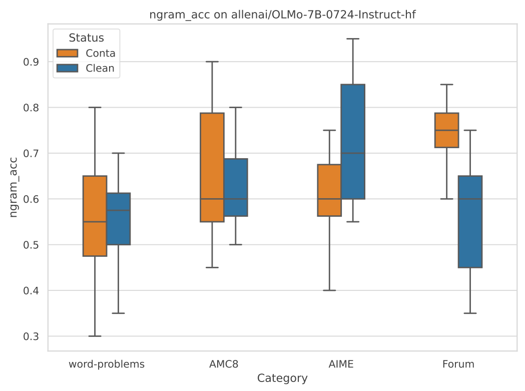
Figure 4: Boxplots showing the distribution of metric values produced by each method, grouped by category and label (clean vs. contaminated) on LE. Clearer separation between the groups indicates better discriminative power and facilitates threshold selection.



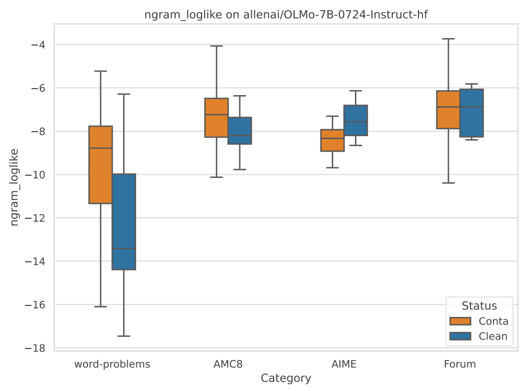
minK



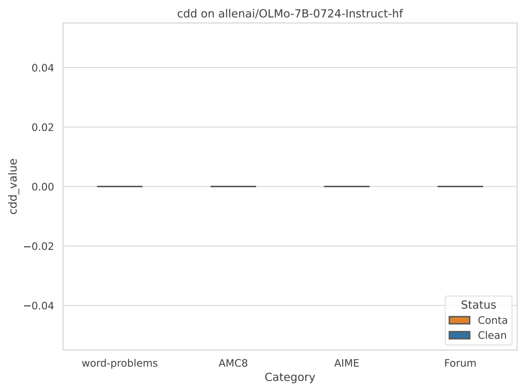
Conta Traces



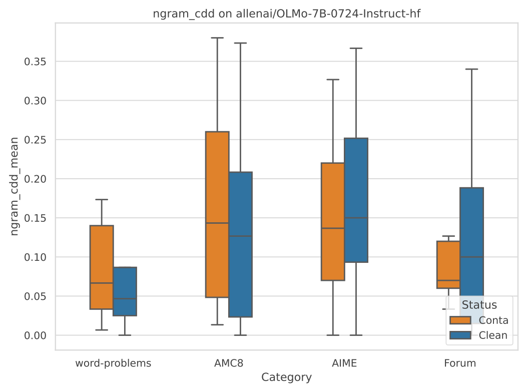
ngram_accuracy



ngram_loglike

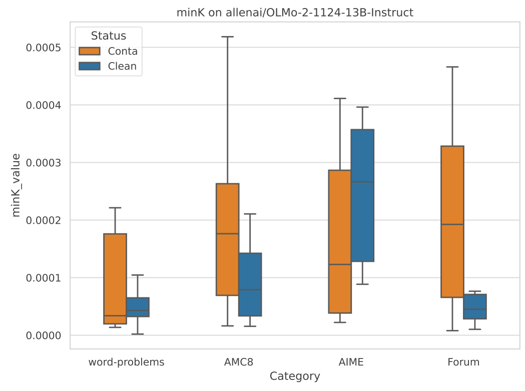


cdd

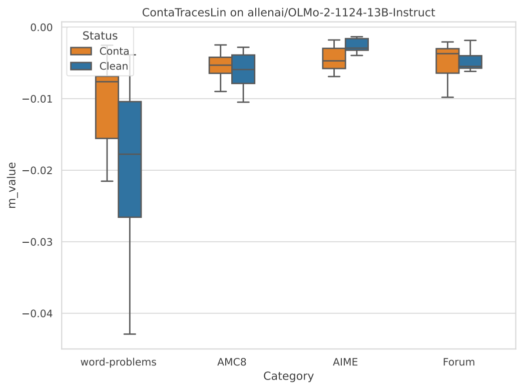


ngram_cdd

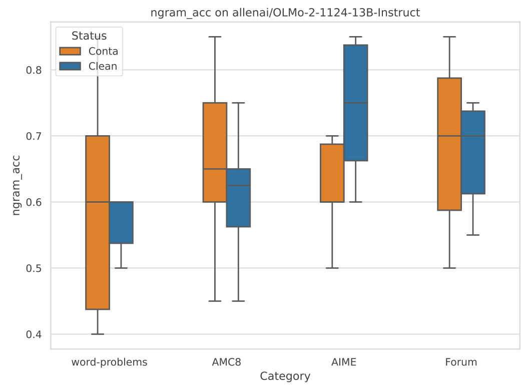
Figure 5: Boxplots showing the distribution of metric values produced by each method, grouped by category and label (clean vs. contaminated) on OL. Clearer separation between the groups indicates better discriminative power and facilitates threshold selection.



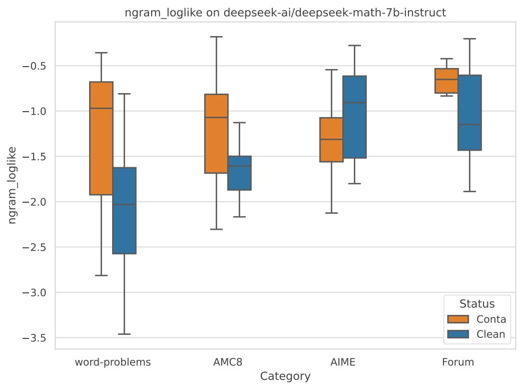
minK



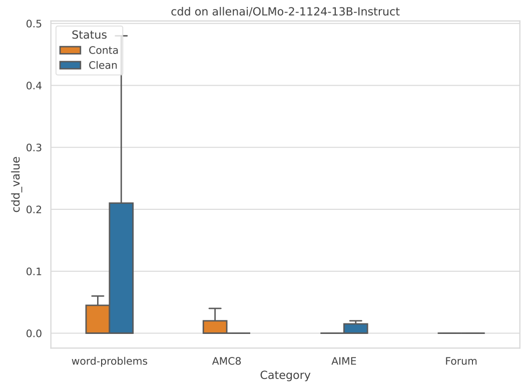
Conta Traces



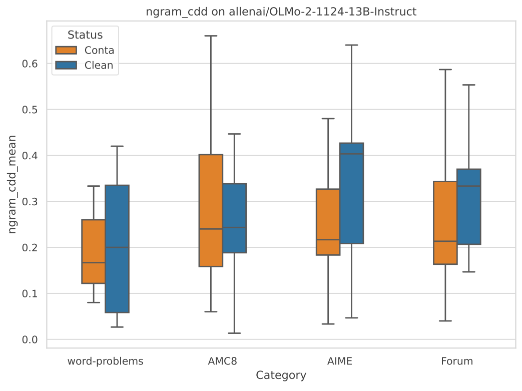
ngram_accuracy



ngram_loglike



cdd



ngram_cdd

Figure 6: Boxplots showing the distribution of metric values produced by each method, grouped by category and label (clean vs. contaminated) on OL2. Clearer separation between the groups indicates better discriminative power and facilitates threshold selection.