A Taxonomy for Data Contamination in Large Language Models

Medha Palavalli

CMU-CS-24-111

May 2024

Computer Science Department
School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213

Thesis Committee:
Matt Gormley, Chair
Lori Levin

Submitted in partial fulfillment of the requirements for the degree of Master of Science in Computer Science.

Keywords: Large Language Models, Data Contamination, Taxonomy, Contamination Ratio

For my family.

Abstract

Large language models pretrained on extensive web corpora demonstrate remarkable performance across a wide range of downstream tasks. However, a growing concern is data contamination, where evaluation datasets may unintentionally be contained in the pretraining corpus, inflating model performance. Not all contamination manifests in the evaluation form when encountered within the pretraining data; these contaminants may originate from altered versions of the test set, evading detection during decontamination. Despite these concerns, how different types of contamination impact the performance of language models on downstream tasks is not fully understood. In this thesis, we present a taxonomy that categorizes the various types of contamination encountered by LLMs during the pretraining phase and identify which types pose the highest risk. We analyze the impact of contamination on two key NLP tasks— summarization and question answering— revealing how different types of contamination influence task performance during evaluation. Our findings yield concrete recommendations for prioritizing data decontamination for pretraining.

Acknowledgments

I am deeply grateful to my advisors, Dr. Matt Gormley and Amanda Bertsch, for their invaluable guidance and unwavering support throughout my academic journey. Their mentorship has been instrumental in shaping my study, research, and personal development. I also extend my sincere appreciation to Dr. Lori Levin for serving on my thesis committee.

I am profoundly grateful to my family for their unwavering encouragement and belief in my abilities. Their love and support have been a constant source of motivation. Lastly, I express my appreciation to my friends and colleagues for their camaraderie and understanding throughout this endeavor.

This thesis would not have been possible without the collective support and encouragement of all those mentioned above. Thank you for being a part of this important chapter in my academic journey.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Advancements in machine learning have traditionally relied on benchmark datasets to evaluate and compare model performance [17, 40]. With the surge of LLMs in recent years, these benchmarks are now leveraged to showcase remarkable abilities across diverse tasks, including open-ended text and code generation. Yet these static, publicly available benchmarks are typically sourced from the internet, and contemporary LLMs often integrate internet text into their training datasets.

For natural language data, the most popular objective is causal language modeling [36]. In general, pretrained large language models (LLMs) perform substantially better than models that are solely trained on the downstream task [11, 34, 39].

Data contamination, also referred to as data leakage, occurs when test data is inadvertently included in a model's training dataset [28]. This can lead to the model exhibiting exceptional performance on this leaked test data, often while failing to generalize effectively to new, unseen data. This can result in misleadingly high performance metrics during testing, giving a false impression of the model's capabilities. The use of contaminated LLMs unknowingly in research can lead to incorrect conclusions or misleading results.

Consequently, accurately assessing model performance and generalization capabilities becomes increasingly challenging, as the influence of test set contamination becomes more subtle and difficult to discern. This not only undermines the reliability of model evaluation but also poses significant implications for real-world applications where trustworthy predictions are essential. Effectively addressing this issue is paramount to ensuring the robustness and trustworthiness of LLMs in practical scenarios, where accurate evaluation is crucial for informed decision-making and reliable performance.

The pervasive use of webscraping to build pretraining corpora for LLMs introduces additional complexities and challenges for evaluating model performance. As LLMs increasingly rely on vast amounts of unlabeled data scraped from the internet for pretraining, including evaluation datasets, ensuring fair and unbiased evaluations becomes inherently challenging. Concerns arise regarding LLMs simply memorizing test samples encountered during pretraining. The very nature of scraping pretraining data from the internet raises fundamental questions about the fairness and validity of evaluation scores obtained on such datasets. With potential overlap between pretraining and evaluation data, there arises a concern regarding the model's familiarity with evaluation examples during pretraining, potentially leading to biased performance metrics. Consequently, the integrity of evaluation scores and our ability to truly comprehend the model's capabilities come under scrutiny. In essence, the presence of internet-sourced data in both pretraining and evaluation datasets challenges the notion of fair evaluation, prompting critical reflection on the methodologies employed to assess LLM performance and the extent to which we can glean insights into the model's true generalization abilities.

Due to the extensive size and diverse sources of pre-trained datasets used for LLMs, they are particularly vulnerable to data contamination, which can be categorized into two main situations:

1. Existing Benchmark Datasets: These are more prone to leakage due to the inclusion of extensive text excerpts, reused code, and synthetic data in the training data of LLMs.

2. Upcoming Benchmark Datasets: Newly constructed test data may unintentionally overlap with the continuously evolving training data of LLMs, as users often lack awareness of the specific composition of LLMs' training datasets.

Consequently, preventing data contamination in LLMs presents significant challenges.

Both GPT-3 and the C4 training corpus were found to have high amounts of data contamination for popular downstream task datasets [5, 12, 39]. Ensuring the absence of contamination is challenging due to two potential sources of contamination: directly from consuming the official version of a dataset (which is easier to manage), and indirectly through duplicated data encountered on the web (which is nearly impossible to manage). This complicates the ability for researchers to comprehensively evaluate the performance of models, especially the GPT-3/3.5 family [5] and GPT-4 [32], on downstream tasks and in turn raises concerns regarding the validity of current evaluations performed on these models [8, 26].

The impact of indirect and approximate test set contamination from Internet-scale pretraining data is not well understood. In this thesis, we taxonomize data contamination from pretraining and measure the impact of different contamination types on downstream performance. Specifically, we explore contamination during pretraining for two case studies: summarization and question & answering. For each case study, we design experiments to assess the effects of indirect and approximate test set contamination on model performance. We observe that both Noise (see Section 2.1.2) and Distribution (see Section 2.1.1) Contamination yield a similar performance enhancement during evaluation as exposure to the entire test dataset during pretraining. However, Mask Contamination (see Section 2.1.2) exhibits inferior performance compared to simply fine-tuning on the training dataset.

Removing all sources of indirect and approximate contamination is challenging, and could significantly limit the pretraining corpus. Due to computational limitations, we employ continued pretraining as a surrogate for pretraining the model from scratch. For each type of approximate contamination, we quantify the impact on performance downstream. This enables us to make recommendations about what types of contamination are most important to exclude or check for when using pretrained LLMs.

Chapter 2

Taxonomy

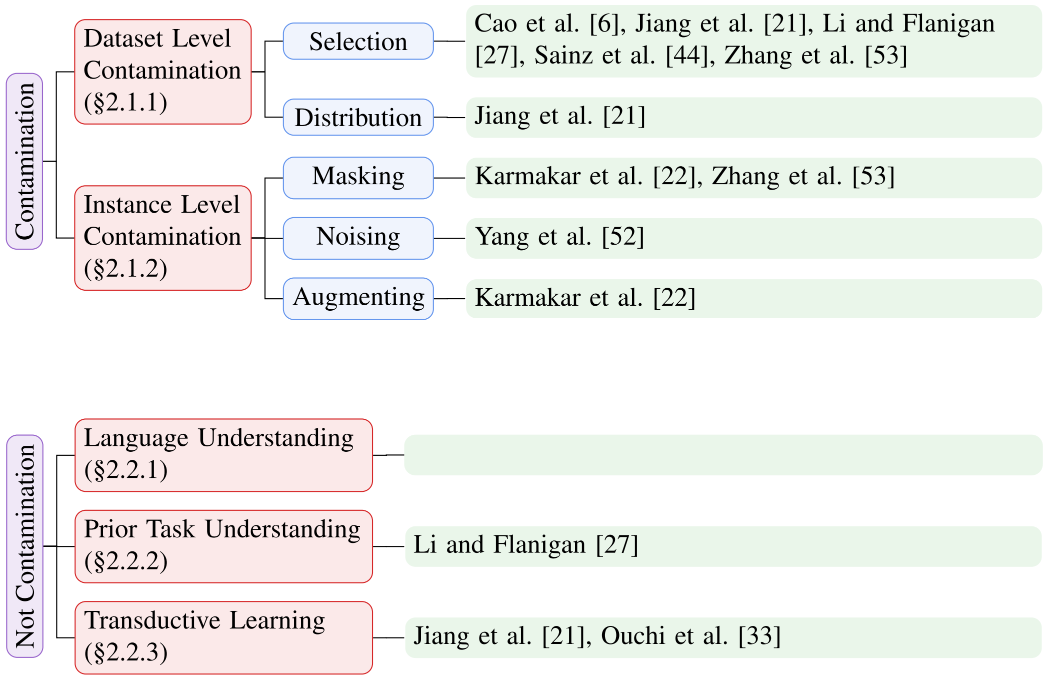
Discussing data contamination can be challenging due to varying beliefs regarding what is considered contamination. To address this, we devised a taxonomy to distinguish between different forms of contamination. By categorizing contamination types, we hope that researchers can better understand and mitigate their effects on data quality and analysis outcomes.

Establishing a taxonomy for LLM pretraining data serves three main purposes. First, it fosters standardization and consensus building, providing a common language and framework for the research community. This enhances clarity and coherence in communication and collaboration among researchers and practitioners.

Second, a well-defined taxonomy enables precise detection and characterization of contamination in pretraining data, crucial for managing the vast volume and varied sources of data used in training LLMs. By categorizing contamination, tailored detection strategies can be employed, improving the reliability and generalization capabilities of LLMs.

Finally, the taxonomy informs the prioritization of mitigation efforts by categorizing contamination based on its impact on model performance. This allows researchers to allocate resources effectively, focusing on addressing the most critical sources of bias or error, thereby enhancing the overall robustness and performance of LLMs.

We believe that the establishment of a systematic taxonomy for categorizing pretraining data contamination in LLM will encourage the community to discuss and evolve standards to deal with contamination in all forms. In particular, identifying the most detrimental types of contamination outlined in our taxonomy enables the development of customized detection strategies for effective decontamination.



Contamination
Dataset Level Contamination (§2.1.1)
Selection
Cao et al. [6], Jiang et al. [21], Li and Flanigan [27], Sainz et al. [44], Zhang et al. [53]
Distribution
Jiang et al. [21]
Instance Level Contamination (§2.1.2)
Masking
Karmakar et al. [22], Zhang et al. [53]
Noising
Yang et al. [52]
Augmenting
Karmakar et al. [22]
Not Contamination
Language Understanding (§2.2.1)
Prior Task Understanding (§2.2.2)
Li and Flanigan [27]
Transductive Learning (§2.2.3)
Jiang et al. [21], Ouchi et al. [33]

Figure 2.1: Taxonomy of Contamination, with some example representative works in the literature that fit in each category.

Consider a model $M : \mathcal{X} \rightarrow \mathcal{Y}$ which, given an input of some type $x \in \mathcal{X}$, outputs text $\widehat{y} \in \mathcal{Y}$. While x can be of any format, we will restrict ourselves to cases where y is in the space of the *natural language* ($\mathcal{Y} \subseteq \Sigma^*$ for some alphabet in Σ). Let D be the *test* set, consisting of $|D|$ examples $(x_i, \widehat{y}_i)$.

2.1 Contamination

We define **contamination** as any leakage of information that provides a signal for the correct label for at least one example in the test set D. We characterize types of contamination by their *dataset-level* and *example-level* properties.

2.1.1 Dataset-level properties

When contamination occurs, some subset of the pretraining data can be characterized as the result of a function $f(D)$. In the simplest case, f is the identity function; this is the leakage of a full test set, e.g. from scraping a file containing the test set examples and labels.

- **Selection:** $f(D)$ is a selection of some group of examples $D' \subset D$, such that only a subset of the test set is leaked. This is likely when the test data is drawn from several sources, only some of which appear in the pretraining data; when some of the test data is more recent than other data and the pretraining data contains an older snapshot of the contamination source;

or when the data is contained in several documents and the cleaning of the pretraining data only removes some of these documents. Figure 2.3 shows an example of Selection.

- **Distribution:** $f(D)$ is a function which combines the contaminated data D with some additional, non-contaminating documents, such that the examples from D are not all sequential in the pretraining data. This can occur during data shuffling, or if the contamination comes from multiple documents. Practically, this means that the contaminated region of the pretraining data $f(D)$ contains more tokens. Figure 2.4 shows an example of Distribution.

2.1.2 Instance-level properties

We consider functions applied to each individual leaked example $f((x_i, \widehat{y_i}))$ for $i \in |D|$[1]. A few representative examples in this class are enumerated below:

- **Masking:** A function that removes some or all of the input, e.g. $f((x_i, \widehat{y_i})) = \widehat{y_i}$. *This only qualifies as contamination for generation tasks;* for a classification task, leaking the label-space in advance may not be a concern if the labels don't have inherent contextual value without the input, such as binary labels like 0s and 1s or positives and negatives. However, if the labels carry meaningful information on their own, their premature disclosure would indeed constitute contamination. Figure 2.5 shows an example of Masking. Also note that masking *all of the output*, leaving only the inputs from the test set, is generally considered to be a type of transductive learning, *not* contamination; see 2.2 for more discussion.

- **Noising:** A function that modifies the surface form of the example, e.g. by paraphrasing the inputs or outputs, or by using silver rather than gold labels for each example. Note that this can also take the form of *alternate correct answers* being present in the pretraining data: for instance, in book summarization, a different summary of the book being present in the pretraining data is still contamination. Figure 2.6 shows an example of Noising.

- **Augmenting:** A function that adds additional context, which may or may not be relevant to the example. For instance, for a task where the model must answer an open-ended question at test time, an *augmented* contaminated example in pretraining would be a multiple-choice test with the same questions. While this provides the correct answer, it also introduces new (distractor) information that is not present at test time. Note the difference between example-level augmenting and dataset-level distribution. Figure 2.7 shows an example of Augmenting.

It's worth noting that certain transformations may initially seem to fall under a different category of contamination, but upon closer examination, they align more closely with either augmentation or noising. For example, reformatting the order of inputs and outputs could be considered noising, while converting a free-form question and answer format into a multiple-choice question format would be considered augmentation. Therefore, while considering reformatting as a type

[1]Note that this is a strict subset of all functions applied to the leaked dataset, $f(D)$; however, we distinguish this set of functions that operate on individual examples.

of contamination, it's essential to carefully assess the specific transformation and categorize it accordingly based on its effect on the example's surface form and context.

Examples of Dataset and Instance Level Contamination

We will present illustrative instances of various contamination types delineated within the taxonomy, utilizing a sample drawn from the HumanEval dataset [9].

```
< HumanEval/0 >
< HumanEval/1 >
< HumanEval/2 >
def below_zero(operations: List[int]) -> bool:
  /* You're given a list of deposit and withdrawal operations on a bank
  account that starts with zero balance. Your task is to detect if at
  any point the balance of account falls below zero, and at that point
  function should return True. Otherwise, it should return False.
  >>> below_zero([1,2,3])
  False
  >>> below_zero([1,2,-4,5])
  True */

  balance = 0
  for op in operations:
    balance += op
    if balance < 0:
      return True
  return False
< HumanEval/3 >
...
< HumanEval/99 >
```

Figure 2.2: Example of Selection. First 100 samples of HumanEval are seen during pre-training

```
< some web text >
def below_zero(operations: List[int]) -> bool:
  /* You're given a list of deposit and withdrawal operations on a bank
  account that starts with zero balance. Your task is to detect if at
  any point the balance of account falls below zero, and at that point
  function should return True. Otherwise, it should return False.
  >>> below_zero([1,2,3])
  False
  >>> below_zero([1,2,-4,5])
  True */

  balance = 0
  for op in operations:
    balance += op
    if balance < 0:
      return True
  return False
< some more web text >
```

Figure 2.3: Example of Distribution. Samples of the Human Eval Dataset are shuffled with Open Web Text.

```
from typing import List
def below_zero(operations: List[int]) -> bool:
  /* You're given a list of deposit and withdrawal operations on a bank
  account that starts with zero balance. Your task is to detect if at
  any point the balance of account falls below zero, and at that point
  function should return True. Otherwise, it should return False.
  >>> below_zero([1,2,3])
  False
  >>> below_zero([1,2,-4,5])
  True */
  balance = 0
  for op in operations:
    balance += op
    if balance < 0:
      return True
  return False
```

Figure 2.4: Sample HumanEval/3 from the HumanEval Dataset.

```
from typing import List
def below_zero(operations: List[int]) -> bool:
  balance = 0
  for op in operations:
    balance += op
    if balance < 0:
      return True
  return False
```

Figure 2.5: Example of Masking. The function specifications have been masked, presenting only the function itself

```
def below_zero(operations: List[int]) -> bool:
  /* You're given a list of deposit and withdrawal operations on a bank
  account that starts with zero balance. Your task is to detect if at
  any point the balance of account falls below zero, and at that point
  function should return True. Otherwise, it should return False.
  >>> below_zero([1,2,3])
  False
  >>> below_zero([1,2,-4,5])
  True */

    balance, i, n = 0, 0, len(operations)

    while i < n:
        balance += operations[i]
        if balance < 0:
            return True
        i += 1

    return False
```

Figure 2.6: Example of Noising. This example demonstrates another function body that accomplishes the same task.

```
def below_zero(operations: List[int]) -> bool:
  /* You're given a list of deposit and withdrawal operations on a bank
  account that starts with zero balance. Your task is to detect if at
  any point the balance of account falls below zero, and at that point
  function should return True. Otherwise, it should return False.
  >>> below_zero([1,2,3])
  False
  Explanation: balance = [1,3,6]
  >>> below_zero([1,2,-4,5])
  True
  Explanation: balance = [1,3,-1,4]*/

  balance = 0
  for op in operations:
    balance += op
    if balance < 0:
      return True
  return False
```

Figure 2.7: Example of Augmenting. In this example, an explanation has been included alongside the sample test cases in the specifications.

2.2 Phenomena that aren't Contamination

For clarity, we describe several phenomena that lead to improved performance on test sets downstream but are *not* considered contamination under our taxonomy.

2.2.1 Language Understanding

Pretraining enables models to produce (generally) fluent text and encodes some representation of meaning for words commonly used in task definitions; for instance, the model has some representation of meaning for the labels "positive" and "negative" in sentiment analysis. While these are both helpful for performing downstream tasks, this is clearly not contamination as long as the pretraining text does not overlap with the test set.

2.2.2 Prior Task Understanding

We define prior task understanding as an ability to perform a task learned from non-contamination sources. For instance, fine-tuning a model on a training dataset for the task is clearly not contamination of the test set, although it generally improves performance on that test set; likewise, pretraining on other related datasets is not contamination for a given test set. However, prior task understanding does generally violate the assumption of "zero-shot" performance: that the model has not seen training data for that task.

In tasks like closed-book question answering (QA) and those requiring world knowledge, prior task understanding is essential. Closed-book QA tasks demand answering questions without external resources, relying solely on the model's training. Models with prior task understanding excel here, leveraging their training on similar question-answer pairs or related datasets. A broad understanding of the world further enhances performance, ensuring more accurate and contextually appropriate responses across various tasks and domains. However, careful consideration of the training data's sources and nature is crucial to maintain the model's integrity and generalizability.

2.2.3 Transductive Learning

Transductive learning [49] involves incorporating an unlabeled test set into the training phase. This means that during training, the raw text inputs of the test set can be utilized, although the labels are never incorporated. The model, once trained, is then evaluated on the same test set during the test phase. While simple, transductive LM fine-tuning has shown to consistently improve neural models in both in-domain and out-of-domain settings [33].

Thus, we generally do not consider pretraining on the *inputs* of the test set to be contamination, although we note that this will likely improve performance in the same manner than pretraining on training set text improves downstream performance [18, 25]: by providing some domain adaptation to the testing domain. Some previous work [21, 33] describes the presence of inputs-only in the pretraining data as contamination for classification tasks; however, under our taxonomy, we consider this a type of transductive learning.

A key exception is tasks where the input is not easily separable from the output. In tasks like perplexity evaluation, where the input sequence also serves as the target output, pretraining on text from the test set can introduce contamination into the training process. This contamination arises from the model inadvertently memorizing sequences from the test set, leading to artificially inflated performance metrics. Consequently, caution is warranted when applying transductive learning techniques in such scenarios to maintain the integrity of the evaluation process.

Despite its potential benefits, transductive learning has sparked substantial discussion and controversy within the machine learning community. It has been argued that by incorporating unlabeled test data, transductive learning may inadvertently introduce biases or overfitting, compromising the model's ability to generalize to unseen data [21]. Additionally, concerns have been raised regarding the fairness and transparency of models trained using this approach, particularly in sensitive or high-stakes applications. Furthermore, ongoing debate surrounds the extent to which transductive learning blurs the line between training and evaluation, challenging traditional notions of model validation and performance assessment. Nonetheless, proponents highlight its potential to improve model robustness and adaptability, particularly in scenarios with scarce or unrepresentative training data [3, 33]. As the field navigates these nuanced considerations, further research and discourse are necessary to elucidate the benefits and limitations of transductive learning and its integration within the broader landscape of machine learning methodologies.

2.3 Mapping Prior Work into this Taxonomy

Sainz et al. [43] establishes a taxonomy that delineates three types of data contamination: guideline contamination, raw text contamination, and annotation contamination. Guideline contamination occurs when the annotation guidelines for a dataset are exposed to the model, while raw text contamination involves the model encountering the original, unannotated text. Annotation contamination occurs when the annotations (labels) of the target benchmark are exposed to the model during training. It is noteworthy that guideline and raw text contamination fall under the transductive category of our taxonomy, whereas annotation contamination aligns with the contamination category. However, while Sainz et al. [43] distinguish between broader levels of contamination, they do not address the specific types of contamination that we define in our taxonomy, highlighting the need for further exploration and clarification in this area.

Zhang et al. [53] explores selection by introducing the concept of counterfactual memorization, which examines how a model's predictions are affected by excluding specific documents during training. Different subsets of the test data were viewed during pretraining. Through empirical analysis, it identifies and explores memorized training examples in standard text datasets, estimating their influence on the validation set and generated texts. It further reveals that the model predictions for examples from both the validation set and the model-generated texts vary significantly depending on the presence or absence of specific training examples with high memorization.

Through running experiments by retraining GPT-2 from scratch, Jiang et al. [21] found that ground-truth contamination during pre-training does not outperform input text contamination (transductive learning) in text classification tasks, as these tasks predominantly depend on the model's comprehension of the input text. The ground truth contamination model they trained is similar to our full contamination setting (3.2.2), which is essentially selecting all of the data in the test dataset to be seen during pretraining.

Jiang et al. [21] explores distribution by investigating the impact of sample frequency in training data on model performance by contaminating the dataset with varying amounts of repetitions and observing the subsequent changes in model performance. The test samples are shuffled in with the Pile [14], which is defined as distribution under our taxonomy. The findings reveal that while performance initially improves with increased contamination, it begins to decline once the contamination factor reaches approximately 10 repetitions, suggesting a threshold effect in the relationship between sample frequency and model performance.

Li and Flanigan [27] investigates the performance of GPT-3 series models and other recent open-sourced LLMs on datasets released before and after their training data creation dates. The study finds that there exists task contamination (prior task understanding) on zero-shot and few-shot evaluation for datasets, leading to their improved performance. They also show that models perform better on datasets created before the pretraining data was collected which is an example of selection.

We can visualize how these research papers map into our existing taxonomy in Figure 2.1

2.3.1 Contamination Detection Methodologies

Yang et al. [52] demonstrate the limitations of existing decontamination methods by showcasing their susceptibility to circumvention tactics, such as rephrasing multiple-choice questions (noisy pretraining data) to evade n-gram overlap or embedding similarity checks. Additionally, the authors propose a novel language model (LLM)-based decontamination method, which leverages embedding similarity search followed by scrutiny from a robust LLM, such as GPT-4, to identify and mitigate instances of contamination effectively.

Through conducting zero-shot prompting experiments on the Codex model using HackerRank problems, Karmakar et al. [22] investigates the effects of full-sample prompting, prompts with missing input/output specifications (masking), and prompts with altered objectives (augmenting). They observe clear signs of memorization in Codex, as it consistently produces complete and accurate code even when essential information is absent from the prompts. Moreover, when the problem objectives change, the model exhibits notably diminished performance, suggesting challenges in generalization.

Recent studies have explored data contamination using domain specific features with Chang et al. [7] using a name cloze membership inference query to infer book memorization in ChatGPT and GPT-4. Cao et al. [6] distinguishes between ”contaminated data,” created before the models' cutoff date, and ”cleansed data,” where countermeasures are applied. It assesses the effectiveness of these countermeasures by examining the performance disparity of CLMs on contaminated versus cleansed data resulting from various countermeasures.

Shi et al. [45] introduces a new detection method MIN-K% PROB based on a simple hypothesis: an unseen example is likely to contain a few outlier words with low probabilities under the LLM, while a seen example is less likely to have words with such low probabilities; a benefit of this methodology being that it can be applied without any knowledge about the pretraining corpus or any additional training.

Sainz et al. [44] attempts to calculate data contamination on LLMs with closed-source pretraining corpora by operating with the underlying assumption that if an LLM can reproduce dataset instances, it must have been trained using that particular split. Golchin and Surdeanu [16] has seen success in identifying pretraining corpora contamination with their two step methodology: identifying potential contamination at the instance level using guided instruction, then assessing wider contamination at the partition level by evaluating the difference between the average overlap score for guided instruction and the general instruction to the reference instances.

Chapter 3

Methodology

We investigate the impact of data contamination by training models exposed to various forms of contamination during pretraining and subsequently evaluating their performance.

3.1 Model

In all our experiments, we employ nanoGPT [24], a lightweight PyTorch re-implementation of OpenAI's GPT (Generative Pretrained Transformer) training. nanoGPT is designed to efficiently train or fine-tune medium-sized GPT models, offering a straightforward and swift approach. Derived from minGPT [23], nanoGPT emphasizes simplicity and speed. Although it remains a work in progress, nanoGPT features a concise training loop of approximately 300 lines and a similarly compact GPT model definition, making it user-friendly and adaptable.

The repository has been able to successfully train a GPT-2 (124M) model on OpenWebText running on a single 8xA100 40GB node in about 4 days of training. For our experiments, we will be loading the 'gpt2-large'[38] weights from OpenAI as our initial weights. We will refer to this model as our *initial model*.

Furthermore, it's important to acknowledge potential data contamination when initializing with OpenAI's GPT-2 weights. The GPT-2 pretraining corpus is not publicly released, which raises concerns about the integrity of the training data used by the model. Consequently, the outcomes of our experiments will serve as a conservative estimate or lower bound on the effects of data contamination, highlighting the need for transparency and caution in model evaluation and deployment.

3.2 General Setup

3.2.1 Data Split

To mitigate potential bias in model performance stemming from dataset diversity, a consistent approach was employed in dataset preparation for both training and testing phases. Specifically, the train and test datasets were curated to contain an identical number of samples. This was achieved by randomly selecting a subsample from the training dataset, using a fixed seed to ensure reproducibility, matching the size of the test dataset. This uniform sampling strategy was applied across all datasets examined in the experiments, aiming to establish a fair and comparable evaluation environment for model performance assessment.

In the context of continued pretraining for the models under evaluation, each dataset incorporates a specific split of the Open AI's WebText dataset [38]. To mitigate any potential recency bias associated with continued pretraining, we incorporate an additional 10,000 samples of WebText into the continued pretraining data. This split plays a crucial role in assessing the impact of distribution (§2.1.1) on model performance. For model's testing the impact of distribution, during continued pretraining, the WebText split is interleaved with the evaluation dataset to assess the model's adaptability to varying data distributions.

However, for models that undergo continued pretraining without a specific focus on distribution, WebText data split is partitioned into two distinct sets. The first set is presented to the model at the beginning of continued pretraining, while the second set is reserved for later stages. This approach enables a systematic exploration of model performance across different types of contamination, while maintaining consistency and comparability across experiments.

3.2.2 Training Settings

The following methodology is applied to each of the datasets to calculate how different types of contamination affect model performance on test data. The following are the types of settings that will be used in our experiments:

1. **Zero Shot Setting:** This zero-shot experimentation will help us establish a baseline in terms of the models ability to perform the task without having seen the task structure or any data related to the evaluation dataset.

2. **Baseline Setting:** The baseline setting for our experiment is a straightforward approach that involves training a language model without any introduced contamination during pretraining. The model with this setting is initialized with 'gpt2-large' weights and is then fine-tuned exclusively on the training dataset without any additional modifications. By using this baseline, we aim to establish a foundational understanding of model performance under standard training conditions, serving as a reference point for evaluating the impact of contamination on subsequent training settings.

3. **Cheating Setting:** The cheating setting represents an extreme scenario where the model is fine-tuned directly on the test data of the evaluation dataset, bypassing the standard training process. This approach aims to assess the upper bound of model performance by leveraging information directly from the evaluation dataset, thereby providing insight into the model's potential when given access to unseen data during training.

4. **In Domain Setting:** The in-domain setting adopts a specialized training approach by conducting continued pretraining on data that specifically belongs to the domain of interest, sourced from the evaluation dataset but distinct from the datasets used for other settings. Following this continued pretraining, the model is fine-tuned on the established training split of the dataset used in our experiment. This variation aims to evaluate the impact of

domain-specific continued pretraining on model performance and generalization capabilities within the targeted domain.

5. **Contaminated Setting:** The contaminated setting explores the effects of various types of contamination introduced during the pretraining phase. Through experimentation with different forms of contamination as defined by our taxonomy, the model with this setting undergoes continued pretraining on the contaminated data. Subsequently, it is fine-tuned on the designated training data, allowing us to assess how different types of contamination affect the model's performance and robustness across diverse tasks and domains.

3.2.3 Training Strategy

Current State of the Art (SOTA) models are being trained on one or a few epochs as they are typically trained on a large corpus of text data (trillions of tokens) in an unsupervised manner [10, 20, 48]. To replicate the nature of pretraining LLMs while optimizing for efficiency and computational resources, we decided to perform continued pretraining on nanoGPT models over one epoch of our dataset. This approach is motivated by the desire to strike a balance between model training comprehensiveness and computational resources.

For each setup, three different models trained on data with varied shuffling schemes were evaluated to mitigate the impact of data ordering randomness on model performance.[1]

For fine-tuning our models, we employed a training strategy that involved training the models over 10 epochs to facilitate comprehensive adaptation to the target task or dataset. To manage computational resources efficiently and prevent overfitting, we incorporated early stopping into our training process. Early stopping was employed to halt training if the model's performance on a validation set ceased to improve, thereby preventing unnecessary computation and ensuring that the model generalized well.

In our training process, we implemented a mechanism to save the best-performing model based on its performance on the validation dataset.

3.3 Contamination Ratio

We introduce a novel metric that calculates the 'ratio of approximate contamination to full contamination performance,' aimed at providing a standardized measure for evaluating model performance across diverse datasets. This metric offers a normalized comparison of model performance by considering the full contamination setting, typically regarded as achieving optimal results during evaluation, and the baseline setting specific to each dataset. By normalizing all metrics relative to these benchmarks, our metric facilitates a comprehensive assessment of each setting's efficacy across datasets.

$$\text{Contamination Ratio} = \frac{contaminated\_setting - baseline}{full\_contamination - baseline}$$

[1]Thesis Repository: `https://github.com/medhap02/nanoGPT-case_studies`

We devised the metric wherein the full contamination setting is assigned a score of 1, while the baseline setting is rated 0. Setting that achieved negative scores perform worse than the baseline, whereas those exceeding 1 outperform the full contamination setting.

This normalization process enables direct comparisons of metrics across datasets, shedding light on the relative performance of the model under varying types of contamination.

Chapter 4

Case Study: Summarization

Summarization models aim to generate concise summaries that capture the main ideas and important details of the original text, often leveraging techniques like extractive (selecting and rearranging existing text) or abstractive (generating new text) summarization methods.

Regarding specific datasets used in NLP summarization research, prominent datasets like XSum, SAMSum, and CNN/Daily Mail have been instrumental in advancing the state-of-the-art in summarization. XSum [31] is a large-scale dataset specifically designed for the task of extreme summarization, consisting of articles paired with single-sentence summaries that capture the most salient information from the source text. SAMSum [15] focuses on abstractive dialogue summarization, providing conversations paired with human-generated summaries. The CNN/Daily Mail dataset [30] comprises news articles paired with multi-sentence summaries, facilitating research in large-scale document summarization. These datasets serve as benchmarks for evaluating summarization models across different domains and summarization types, contributing to advancements in NLP research and applications.

4.1 Experimental Setup

For this case study, we ran the baseline setting, full contamination setting, in-domain setting, zero-shot setting, and 5 different contaminated settings.

- **Full Contamination Setting:** The model underwent continued pretraining on 1 epoch of the test split, maintaining the format encountered during evaluation, followed by 10 epochs of fine-tuning on the established train split.
- **Distribution Contaminated Setting:** The model underwent continued pretraining on 1 epoch of the test split, which included data mixed with open web text, before proceeding to 10 epochs of fine-tuning on the established train split.
- **Masked Contaminated Setting:** The model underwent continued pretraining only on the outputs of 1 epoch of the test split, focusing solely on summaries without considering the associated documents. It was subsequently proceeded by 10 epochs of fine-tuning on the established train split.

- **Noised Contaminated Setting:** The model, underwent continued pretraining on 1 epoch of the test split, encountering summaries generated by prompting Chat GPT-3.5, rather than the ground truth summaries. This was then proceeded by 10 epochs of fine-tuning on the established train split.
- **Reformatted Contaminated Setting:** The model underwent continued pretraining on 1 epoch of the test split. During continued pretraining, it experienced the summary before the document, deviating from the standard document-summary order. Subsequently, it was fine-tuned on 10 epochs of the established train split.

Sample	Conversation: Anita: I'm at the station in Bologna Jenny: No problems so far? Anita: no, everything's going smoothly Tomy: good! Summary: Anita is at Bologna station.
Distribution	⟨ *some open web text* ⟩ Conversation: Anita: I'm at the station in Bologna Jenny: No problems so far? Anita: no, everything's going smoothly Tomy: good! Summary: Anita is at Bologna station. ⟨ *some more open web text* ⟩
Masking	Summary: Anita is at Bologna station.
Noising	Conversation: Anita: I'm at the station in Bologna Jenny: No problems so far? Anita: no, everything's going smoothly Tomy: good! Summary: Anita confirms her location at the Bologna station to Jenny and Tomy, reassuring them that everything is running smoothly.
Reformatting	Summary: Anita is at Bologna station. Conversation: Anita: I'm at the station in Bologna Jenny: No problems so far? Anita: no, everything's going smoothly Tomy: good!

Table 4.1: Applying the different contamination techniques to a sample from the SAMSum dataset.

4.2 Results and Analysis

Dataset	Model	Contaminated Pretraining Data	Contaminated Fine-tuning Data	ROUGE-1	ROUGE-2	ROUGE-L	ROUGE-LSUM
	Zero-Shot	-	-	21.98 ± 0.26	5.076 ± 0.01	13.63 ± 0.10	18.51 ± 0.10
	Baseline	-	×	27.22 ± 0.53	7.436 ± 0.03	18.15 ± 0.43	24.90 ± 0.36
	Cheating	-	✓	**33.60** ± 0.58	**10.198** ± 0.06	**20.52** ± 0.33	**29.61** ± 0.32
	In-Domain	×	×	29.81 ± 0.48	9.277 ± 0.04	18.93 ± 0.25	26.88 ± 0.31
CNN	Full Contamination	✓	×	29.84 ± 0.48	9.488 ± 0.04	19.50 ± 0.38	26.98 ± 0.40
	Distribution	✓	×	29.73 ± 0.33	*9.557* ± 0.03	*19.50* ± 0.22	27.12 ± 0.26
	Masked	✓	×	28.34 ± 0.22	8.326 ± 0.03	18.01 ± 0.29	25.96 ± 0.20
	Noised	✓	×	*31.31* ± 0.52	8.821 ± 0.05	19.19 ± 0.32	*28.85* ± 0.30
	Reformatted	✓	×	29.21 ± 0.28	8.887 ± 0.03	18.88 ± 0.29	26.27 ± 0.30
	Zero-Shot	-	-	11.73 ± 0.14	1.357 ± 0.01	8.377 ± 0.19	9.33 ± 0.16
	Baseline	-	×	32.95 ± 0.57	10.22 ± 0.05	25.83 ± 0.32	25.59 ± 0.29
	Cheating	-	✓	**36.36** ± 0.53	**12.31** ± 0.04	**28.41** ± 0.33	**28.48** ± 0.33
	In-Domain	✓	×	33.61 ± 0.46	10.27 ± 0.09	26.39 ± 0.30	26.46 ± 0.35
SAMSum	Full Contamination	×	×	*34.34* ± 0.45	*10.76* ± 0.06	*26.98* ± 0.40	*27.04* ± 0.38
	Distribution	✓	×	33.73 ± 0.51	10.32 ± 0.05	26.48 ± 0.31	26.56 ± 0.33
	Masked	✓	×	33.05 ± 0.46	10.46 ± 0.05	25.77 ± 0.30	25.81 ± 0.28
	Noised	✓	×	33.62 ± 0.43	10.27 ± 0.06	26.50 ± 0.37	26.49 ± 0.38
	Reformatted	✓	×	33.63 ± 0.39	10.25 ± 0.05	26.37 ± 0.31	26.46 ± 0.34
	Zero-Shot	-	-	12.52 ± 0.11	2.059 ± 0.00	9.035 ± 0.16	10.27 ± 0.17
	Baseline	-	×	26.28 ± 0.48	6.424 ± 0.02	19.80 ± 0.32	19.81 ± 0.33
	Cheating	-	✓	**29.87** ± 0.41	**8.334** ± 0.03	**22.97** ± 0.43	**22.98** ± 0.42
	In-Domain	×	×	26.43 ± 0.52	6.745 ± 0.05	19.99 ± 0.39	19.99 ± 0.40
XSum	Full Contamination	✓	×	26.53 ± 0.51	6.820 ± 0.02	20.08 ± 0.33	20.03 ± 0.37
	Distribution	✓	×	*26.61* ± 0.42	*6.885* ± 0.03	*20.12* ± 0.37	*20.11* ± 0.37
	Masked	✓	×	24.50 ± 0.46	5.677 ± 0.02	18.16 ± 0.29	18.39 ± 0.31
	Noised	✓	×	26.16 ± 0.39	6.599 ± 0.02	19.72 ± 0.35	19.72 ± 0.35
	Reformatted	✓	×	26.27 ± 0.43	6.623 ± 0.02	19.86 ± 0.29	19.86 ± 0.30

Table 4.2: Results for all nine models trained on XSum, SAMSum, and CNN/Daily Mail Datasets. The table showcases evaluation metrics, with the best-performing model scores bolded and the second best italicized.

In this section, we consider the overall performance of each contamination method across summarization datasets.

Consistently, the cheating setting outperforms all others; this is expected, given that deliberately finetuning on the test data is an extreme form of contamination.

Overall, continued pretraining with the approximate contamination methods improves performance above the baseline, often by several standard deviation. This suggests that exposure to these forms of contamination during pretraining can impact the reliability of evaluations on this data downstream.

The distribution, full contamination, and noised settings generally outperform the in-domain setting. This suggests that the boost when pretraining on contaminated splits is not merely from seeing in-domain data, but specifically from seeing the test data.

However, it is important to note that while the majority of these settings have metrics the fall within one standard deviation of each other, there are exceptions. For instance, in the case of the XSum dataset, the noised setting fails to surpass the baseline. This discrepancy can be attributed

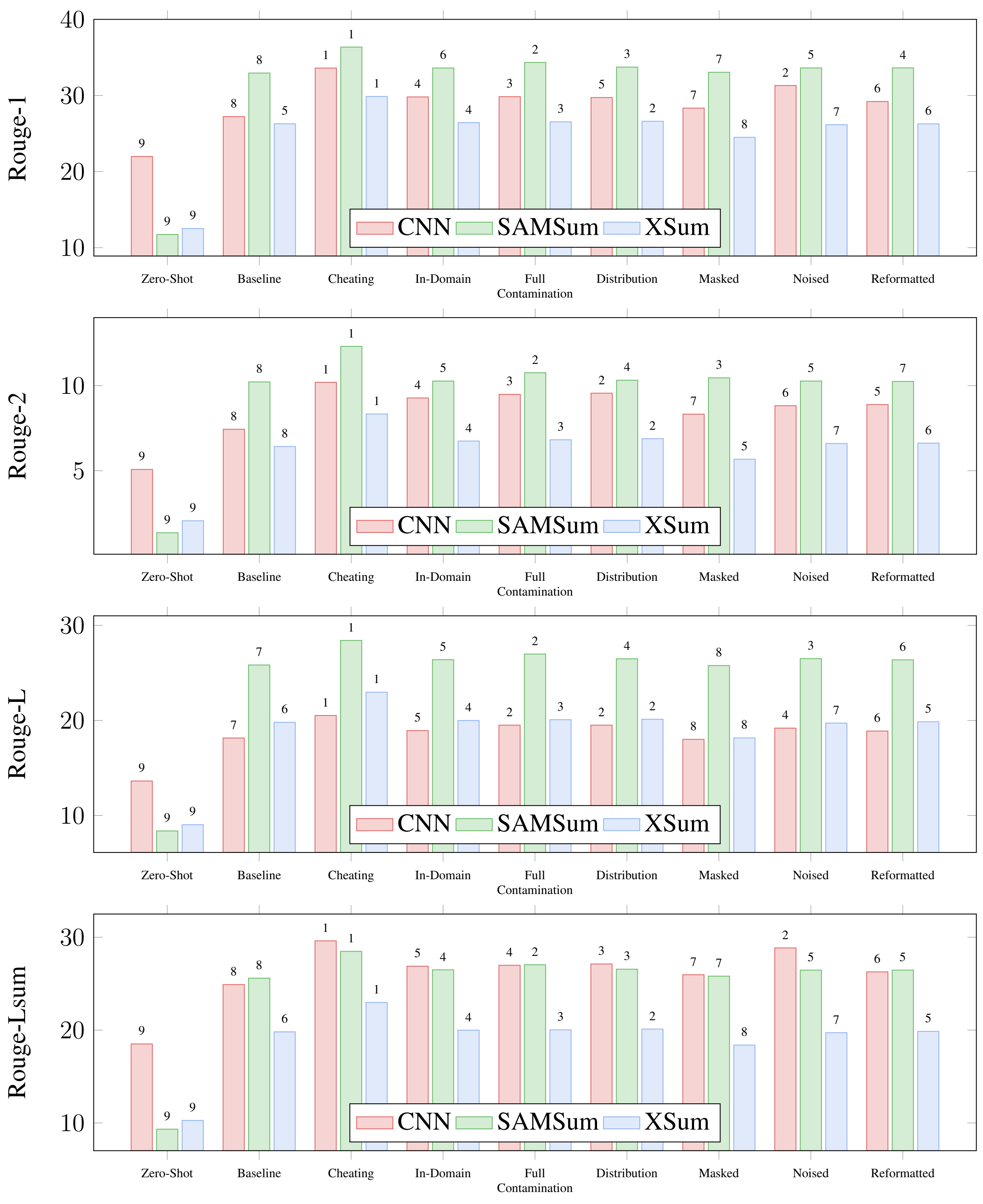


Figure 4.1: Barchart of all models compared for each metric

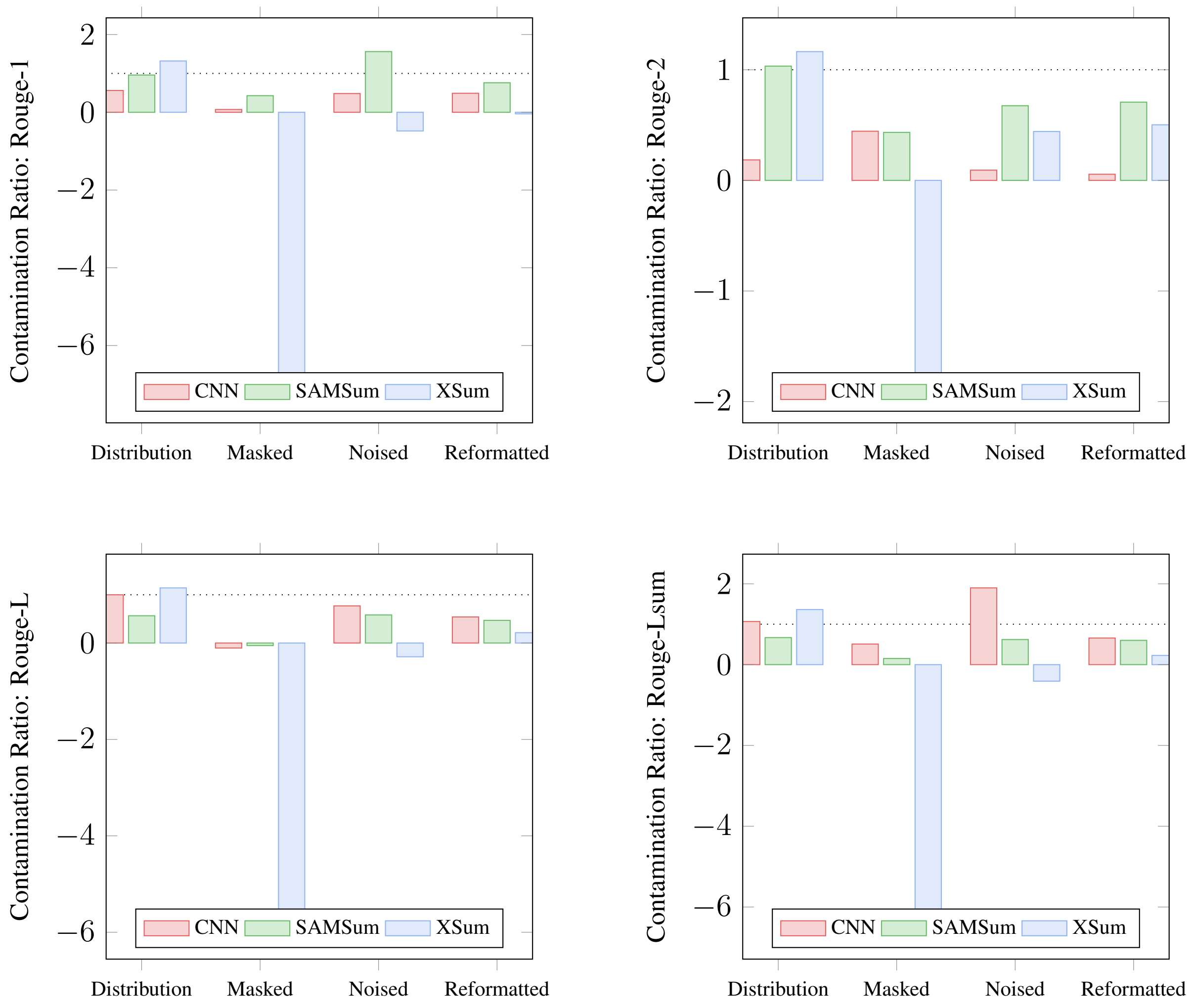


Figure 4.2: Ratio of Contamination Ratio to full contamination performance for the contaminated models.

to the idiosyncrasies of the XSum dataset, where ground truth summaries may deviate significantly from typical summaries, thus posing a challenge for the model in generating accurate outputs. The summaries generated by GPT-3.5 for the XSum dataset have lower Rouge scores than the other two datasets.

We observe that the reformatted setting demonstrates improved performance compared to the baseline, but it doesn't perform as well as the full contamination setting. This observation could be due to GPT models, whose training objective involves next token prediction, highlighting the importance of capturing sequential dependencies in text generation tasks like summarization. Effectively capturing these dependencies is crucial for the model to perform better on downstream tasks, emphasizing the significance of order information processing in natural language processing.

Additionally, underperformance of the mask contamination setting relative to the baseline across all datasets warrants attention. This disparity suggests that exposure solely to summaries

Dataset	Rouge-1	Rouge-2	Rouge-L	Rouge-Lsum
CNN	38.70	14.14	24.90	32.11
SAMSum	37.92	13.78	28.73	28.75
XSum	24.21	4.89	16.60	16.60

Table 4.3: Rouge scores for summaries generated by GPT-3.5

during the pretraining phase may impose limitations on the model's learning capacity, hindering its ability to effectively capture and generalize from diverse textual inputs.

In summary, the observed trends underscore the multifaceted impact of different contamination methodologies on model performance. These findings not only contribute to our understanding of model robustness and generalization but also highlight the distribution, full contamination, and noisy data have a positive impact on model evaluation scores.

Chapter 5

Case Study: Question and Answering

Question answering (QA) models are designed to automatically generate responses to questions posed in natural language. These models come in various forms, catering to different types of questions and tasks. Some QA models focus on answering multiple-choice questions, where they select the most appropriate answer from a set of options. These models typically employ techniques such as machine learning and natural language processing to analyze the question and candidate answers before making a selection. On the other hand, there are QA models that tackle free-response questions, where they generate textual answers based on the input question. These models often use sophisticated language generation algorithms, including transformer-based architectures, to produce coherent and contextually relevant responses. Overall, question answering models play a crucial role in automating information retrieval and comprehension tasks, facilitating efficient access to knowledge and insights from large volumes of textual data.

The Stanford Question Answering Dataset (SQuAD) [41] stands as a benchmark dataset in the field of question answering, renowned for its rich collection of context-question-answer triples sourced from a diverse range of articles. SQuAD is structured as a freeform question answering task, where models are tasked with generating precise answers to questions based on a given passage of text.

In contrast, Children's Book Test (CBT) [19] was meticulously crafted to gauge the ability of language models to grasp the nuanced meaning embedded within children's literature. Unlike conventional language modeling benchmarks, CBT goes beyond merely predicting syntactic function words by also assessing the comprehension of lower-frequency words, which convey richer semantic content and play a pivotal role in understanding text. CBT adopts a multiple-choice question answering format, wherein models must select the correct answer from a set of options provided for each comprehension question.

This distinction underscores the varied nature of question answering tasks, with SQuAD emphasizing open-ended response generation and CBT focusing on selecting the most appropriate answer among predefined choices. By encompassing both free-response and multiple-choice question answering paradigms, SQuAD and CBT offer complementary perspectives on the challenges and nuances inherent in understanding and processing textual information.

5.1 Experimental Setup

For this case study, we ran the baseline setting, full contamination setting, in-domain setting, zero-shot setting, and 6 different contaminated settings.

- **Full Contamination Setting:** The model underwent continued pretraining on 1 epoch of the test split, maintaining the format encountered during evaluation, followed by 10 epochs of fine-tuning on the established train split.

- **Distribution Contaminated Setting:** The model underwent continued pretraining on 1 epoch of the test split, which included data mixed with open web text, before proceeding to 10 epochs of fine-tuning on the established train split.

- **Masked Contaminated Setting:** The model underwent one epoch of continued pretraining on the established test split where context sentences were masked, allowing the model to only observe questions and answers, followed by 10 epochs of fine-tuning on the established train split.

- **Noised Contaminated Setting:** The model underwent one epoch of continued pretraining on the established train split, viewing answers generated by GPT-3.5 in response to context and questions, before proceeding to 10 epochs of fine-tuning on the established train split.

- **Reformatted Contaminated Setting:** The model underwent one epoch of continued pretraining on the established test split. For the SQuAD dataset, free-form QA was converted to multiple-choice QA with GPT-3.5's assistance, while the CBT dataset was converted into free-form QA with the answer options not being presented. It then underwent 10 epochs of fine-tuning on the established train split.

- **Augmented Contaminated Setting:** The model underwent one epoch of continued pretraining on the established test split. Additional information was included in the context preceding the question and answers by prompting GPT-3.5 to generate related information, before proceeding to 10 epochs of fine-tuning on the established train split.

Sample	Context: The Bey Hive is the name given to Beyoncé's fan base. Fans were previously titled "The Beyontourage", (a portmanteau of Beyoncé and entourage). The name Bey Hive derives from the word beehive, purposely misspelled to resemble her first name, and was penned by fans after petitions on the online social networking service Twitter and online news reports during competitions. Question: Beyonce has a fan base that is referred to as what? Answer: The Bey Hive
Distribution	⟨ *some open web text* ⟩ Context: The Bey Hive is the name given to Beyoncé's fan base. Fans were previously titled "The Beyontourage", (a portmanteau of Beyoncé and entourage). The name Bey Hive derives from the word beehive, purposely misspelled to resemble her first name, and was penned by fans after petitions on the online social networking service Twitter and online news reports during competitions. Question: Beyonce has a fan base that is referred to as what? Answer: The Bey Hive ⟨ *some more open web text* ⟩
Masking	Question: Beyonce has a fan base that is referred to as what? Answer: The Bey Hive
Noising	Context: The Bey Hive is the name given to Beyoncé's fan base. Fans were previously titled "The Beyontourage", (a portmanteau of Beyoncé and entourage). The name Bey Hive derives from the word beehive, purposely misspelled to resemble her first name, and was penned by fans after petitions on the online social networking service Twitter and online news reports during competitions. Question: Beyonce has a fan base that is referred to as what? Answer: Bey Hive
Reformatting	Context: The Bey Hive is the name given to Beyoncé's fan base. Fans were previously titled "The Beyontourage", (a portmanteau of Beyoncé and entourage). The name Bey Hive derives from the word beehive, purposely misspelled to resemble her first name, and was penned by fans after petitions on the online social networking service Twitter and online news reports during competitions. Question: Beyonce has a fan base that is referred to as what? Options: A) The Beehivers B) The Bey Hive C) The Beyontourage D) The Bey Flock Answer: The Bey Hive
Augmenting	Context: The Bey Hive is the name given to Beyoncé's fan base. Fans were previously titled "The Beyontourage", (a portmanteau of Beyoncé and entourage). The name Bey Hive derives from the word beehive, purposely misspelled to resemble her first name, and was penned by fans after petitions on the online social networking service Twitter and online news reports during competitions. This fervent fan base actively engages with Beyoncé's music, performances, and philanthropic endeavors. Question: Beyonce has a fan base that is referred to as what? Answer: The Bey Hive

Table 5.1: Applying the different contamination techniques to a sample from the SQuAD dataset.

5.2 Results and Analysis

Dataset	Model	Contaminated Pretraining Data	Contaminated Fine-tuning Data	Exact Match	F1 Score
SQuAD	Zero-shot	-	-	0.178 ± 0.01	4.180 ± 0.01
	Baseline	-	×	41.76 ± 0.31	55.72 ± 0.35
	Cheating	-	✓	**55.73** ± 0.34	**66.47** ± 0.30
	In-Domain	×	×	52.44 ± 0.29	64.52 ± 0.30
	Full Contamination	✓	×	53.38 ± 0.24	65.07 ± 0.26
	Distribution	✓	×	52.76 ± 0.29	64.92 ± 0.28
	Masked	✓	×	38.77 ± 0.26	51.93 ± 0.28
	Noised	✓	×	52.72 ± 0.29	64.65 ± 0.28
	Reformatted	✓	×	48.08 ± 0.31	61.85 ± 0.28
	Augmented	✓	×	*53.58* ± 0.28	*65.51* ± 0.28
CBT	Zero-shot	-	-	0.192 ± 0.01	3.290 ± 0.01
	Baseline	-	×	19.41 ± 0.29	33.11 ± 0.30
	Cheating	-	✓	**54.27** ± 0.25	**71.20** ± 0.26
	In-Domain	×	×	44.19 ± 0.24	62.12 ± 0.26
	Full Contamination	✓	×	*52.06* ± 0.28	*69.20* ± 0.29
	Distribution	✓	×	25.49 ± 0.27	41.45 ± 0.27
	Masked	✓	×	46.51 ± 0.24	64.34 ± 0.23
	Noised	✓	×	49.59 ± 0.26	67.06 ± 0.26
	Reformatted	✓	×	51.46 ± 0.23	68.82 ± 0.26
	Augmented	✓	×	27.51 ± 0.30	44.08 ± 0.29

Table 5.2: Results for all 10 models trained on the SQuAD and CBT dataset. The table showcases evaluation metrics, with the best-performing model scores bolded and the second best italicized.

In this section, we consider the overall performance of each contamination method across Question Answering datasets

Consistently, the cheating setting outperforms all others by a noticable margin; this is expected, given that deliberately finetuning on the test data is an extreme form of contamination.

Furthermore, with the exception of the masked setting for the SQuAD dataset, all contaminated settings exhibit better performance compared to the baseline setting. Particularly noteworthy are the full contamination and noised settings, which outperform the baseline by a considerable margin across both datasets. It is important to note that while the difference in performance of these contaminated settings compared to the cheating setting is statistically significant, it is small.

Dataset	Exact Match	F1 Score
SQuAD	74.56	88.15
CBT	77.21	87.78

Table 5.3: Exact Match and F1 scores for answers generated by GPT-3.5

Note that the noised setting performs almost as well as the full contamination setting. This can be attributed to the exact match and F1 scores of the answers generated by GPT-3.5 being fairly high.

Exposure to in-domain data during pretraining appears to improve model performance. However, our results show that the different contaminated setting such as noise, full contamination,

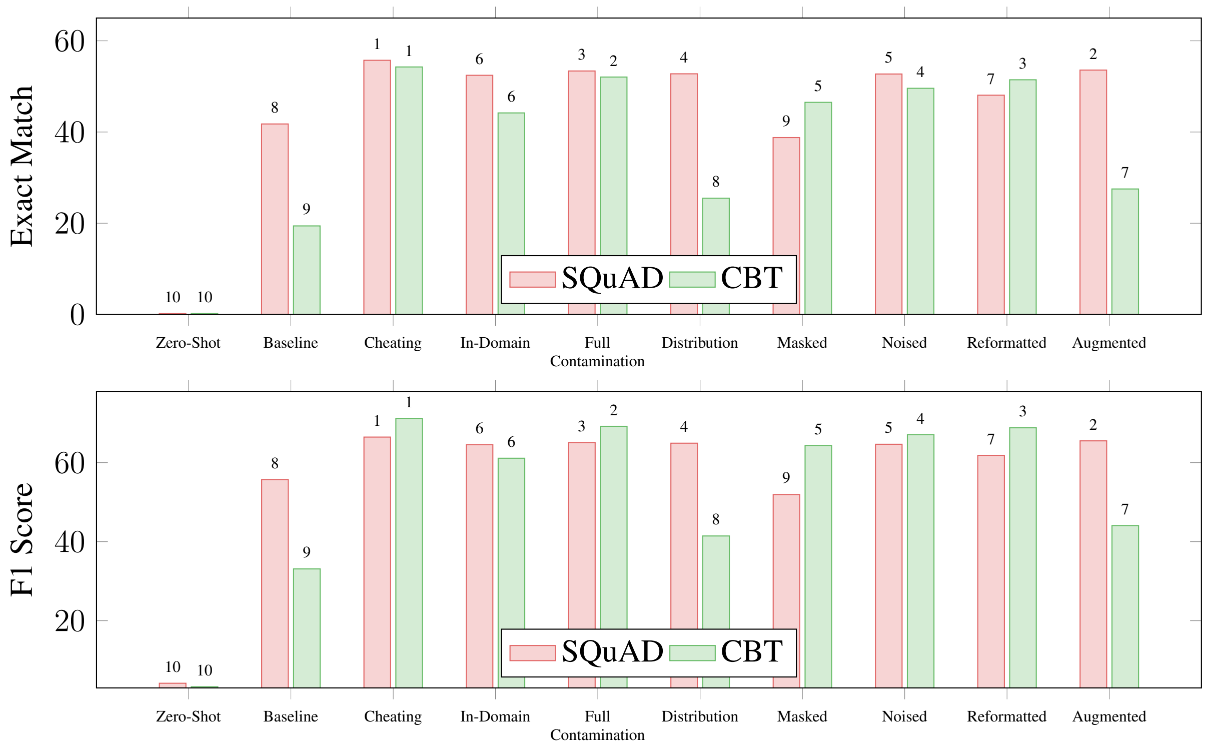


Figure 5.1: Bar Chart of all SQuAD and CBT models compared for each metrics

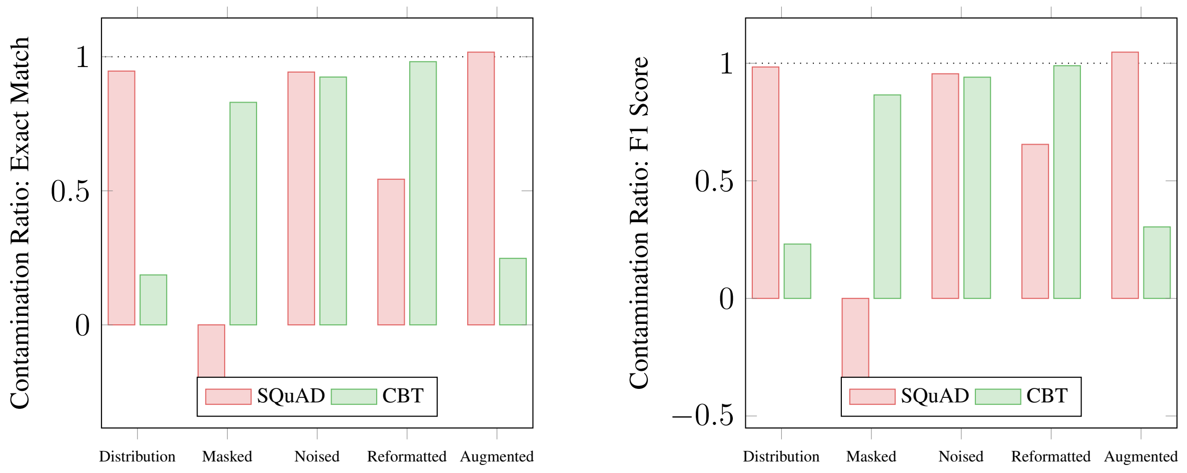


Figure 5.2: Ratio of Contamination Ratio to full contamination performance for the contaminated models.

and reformatting tend to outperform the in-domain setting during evaluation. This suggests that seeing data from the test set, even if it isn't in the correct format, can positively impact model performance for question answering tasks.

Reformatting free-form questions from SQuAD into multiple-choice answers during pre-training appears to have a detrimental effect on model performance, albeit not to the extent of matching the baseline. Conversely, converting multiple-choice questions from CBT into free-form questions during pretraining yields positive results, with the reformatting setting outperforming most other contaminated settings.

Furthermore, we observe variations in the performance of augmented and distribution settings across the two datasets. While these settings perform well for SQuAD, their performance is not as impressive for CBT. For the augmentation setting, this discrepancy may be attributed to the nature of data augmentation, where the additional information provided for SQuAD is more relevant and beneficial to the wikipedia paragraphs compared to the irrelevant introductions, such as 'once upon a time' style introductions generated by GPT-3.5 for these book excerpts, added to CBT stories. It is important to note that since this information doesn't significantly contribute to the task, this form of augmentation falls in a blurry space between distribution and augmentation sections of the taxonomy. It could also be viewed as unrelated information being added between samples during pretraining, complicating its categorization.

Overall, our findings highlight the significant impact of exposing the model to test data during pretraining on their performance during evaluation. Regardless of the contamination type, the mere exposure to test data appears to substantially enhance model performance compared to the baseline setting, underscoring the importance of careful consideration of dataset characteristics and contamination methodologies in model training and evaluation.

Chapter 6

Analysis

In this analysis, we explore the performance of the summarization and QA model for all of the training settings across diverse datasets.

We consistently observe that the cheating setting outperforms the baseline in summarization and QA tasks, emphasizing the risk of overfitting and inflated metrics when fine-tuning on the test dataset. This underscores the importance of adhering to best practices in model training for reliable performance.

Most contaminated settings exhibit performance levels that lie between those of the cheating and baseline settings. This suggests that exposure to evaluation data during pretraining indeed enhances model performance. Specifically, the full contamination setting consistently outperforms the baseline across all datasets, indicating that exposure to pretraining data in the exact format as evaluation data leads to better performance during evaluation. Similarly, the distribution setting performs well, implying that interleaved test data with other irrelevant information still benefits model performance during evaluation.

Furthermore, we find that noisy data tends to contribute to improved performance, primarily due to the high evaluation metric values of GPT-3.5 generated summaries/answers compared to ground truth summaries/answers. This incorporation of noisy labels during pretraining aids in the model's correct understanding of the task, reflected in its improved performance during evaluation.

The consistent outperformance of the in-domain setting compared to the baseline underscores the advantages of exposure to related samples in the correct format during pretraining [25]. However, one potential reason behind this performance disparity could be attributed to the fact that the model with the in-domain setting sees a more diverse set of data during its overall training process. Unlike the established train split used for the baseline setting, the in-domain pretraining split introduces additional diversity, potentially enriching the model's understanding and adaptability. However, it is noteworthy that despite experiencing similar diversity in sample data during training, some contaminated settings outperform the in-domain setting during evaluation. This highlights the significant impact of exposure to test split data on model performance and tells us that contamination can play a crucial role in enhancing model effectiveness during evaluation.

Some types of contamination have task-dependent effects. For example, the masked setting boosts the model's evaluation scores more in the QA task than in the summarization task. The disparity may arise from the fact that during pretraining for the summarization task, the model is exposed solely to summaries, whereas during evaluation, it encounters complete documents. However, in the QA task, the model observes both questions and answers during pretraining and receives context along with a question during evaluation. This overlap enables the model to better recall memorized answers when presented with questions.

Note however, that the model with the masking setting generally performs around the same or worse than the baseline setting. This unexpected outcome suggests that contamination may potentially degrade model performance during evaluation rather than enhance it, contrary to prevailing expectations in the field of pretraining large language models. However, it is possible that this observed phenomenon is merely a characteristic of the datasets or tasks under scrutiny and may not hold true across the board.

In summary, our comprehensive analysis sheds light on the intricate interplay between contamination methodologies and model performance in both summarization and QA tasks. These insights underscore the importance of careful consideration in model training and evaluation processes to ensure reliable and robust performance across various datasets and tasks.

Chapter 7

Related Work

7.1 Detecting Data Contamination

Over the past three years, research into data contamination has increased significantly. Among the early research exploring data contamination in LLMs, most primarily adopt a methodology similar to using high-order n-grams to detect overlapping content between the pre-training data and the evaluation dataset [5, 37, 48, 50]. The primary limitations to this methodology, however, is that the pretraining data must be open-source and may require substantial computational resources or manual labor. [44] attempts to calculate data contamination on LLMs with closed-source pretraining corpora by operating with the underlying assumption that if an LLM can reproduce dataset instances, it must have been trained using that particular split. [16] has seen success in identifying pretraining corpora contamination with their two step methodology: identifying potential contamination at the instance level using guided instruction, then assessing wider contamination at the partition level by evaluating the difference between the average overlap score for guided instruction and the general instruction to the reference instances.

The shelf life of benchmarks is incredibly low with [42] demonstrating that newer models with updated training cutoff dates are iteratively rendering existing benchmarks stale. Utilizing OpenAI's GPT-3.5 and GPT-4, [1] revealed that these models have been globally exposed to approximately 4.7 million samples from 263 benchmarks within the first year after their release.

Data sketching, storing compressed or approximate views of large datasets, has long been used to enable efficient analysis of large datasets [4]. In attempt to reason over large pretraining copora, [29] created Data Portraits, a document artifact that performs datasketching on the Pile [13] using membership inference with a strided Bloom filter [2] and enables answering questions about test set leakage and model plagiarism. Search engines, such as the ROOTS Search Tool [35] allow users to search over the entire ROOTS corpus offering both fuzzy and exact search capabilities, allowing for qualitative analysis over pretraining data.

7.2 Self Pretraining

In [25], the idea of self-pretraining, where models are trained directly on downstream corpora using self-supervised objectives, is explored. Research has indicated that additional self-supervised pretraining on downstream data can yield further gains [18, 25]. The advantages of self-pretraining cannot be ascribed to knowledge transfer from the upstream corpus since the data required for self-pretraining is the same as that for finetuning in the evaluated downstream tasks. The gains from this approach stem solely from the pretraining objective, which may be more effective at capturing certain inductive biases than the finetuning objective, such as linguistic knowledge [47]. Additionally, it is possible that self-pretraining simply sets up the network parameters in a way that results in superior optimization during finetuning [51]. Self-pretraining also offers advantages in structured output prediction tasks, including question answering and commonsense inference, frequently yielding over a 50% enhancement in performance compared to conventional pretraining [25].

Chapter 8

Conclusion

In conclusion, our research has provided valuable insights into the performance of contaminated summarization and QA models across diverse datasets and contamination methods. Through a comprehensive analysis, we have observed notable trends that highlight the impact of various contamination methodologies on model efficacy. It is increasingly evident that assessing models on these benchmarks undermines our capacity to accurately compare and evaluate modern models.

Our findings reinforce the widely accepted notion that fine-tuning on the test dataset, as exemplified by the cheating setting, can lead to overfitting and inflated evaluation metrics. This underscores the importance of adhering to best practices in model training to ensure robust and reliable performance.

Moreover, our analysis uncovered that exposure to evaluation data during pretraining improves model performance, as indicated by the performance of the model with the contaminated settings falling between those of the cheating and baseline settings. The performance of the full contamination setting and noisy setting emphasizes the significance of exposure to evaluation data in enhancing model performance.

Additionally, our research highlighted the significant impact of exposure to test split data on model performance. The observed performance disparity between the in-domain setting and some contaminated settings suggests that contamination can play a crucial role in enhancing model effectiveness during evaluation, particularly when considering the diversity of sample data during training.

Our analysis revealed the positive impact of exposure to test data in the correct format on model performance. Notably, for all summarization datasets, reformatting the data resulted in poorer performance during evaluation compared to the full contamination setting. Our findings indicate that the format of contaminated data during pretraining influences the model's performance during evaluation, highlighting the importance of meticulously decontaminating properly formatted contamination data from datasets.

Additionally, our findings highlight the task-specific nature of model performance. For instance, the distribution contaminated setting exhibits better performance in the summarization task compared to the QA task during evaluation. This indicates that the effectiveness of certain contamination methods varies depending on the specific task requirements, emphasizing the importance of tailoring decontamination strategies to the characteristics of the task at hand.

Finally, the unexpected outcome of masking out context during continued pretraining highlights the potential risks of contamination in degrading model performance during evaluation, contrary to expectations in the field.

In addition to our research findings, it is essential to acknowledge the current limitations and gaps in existing decontamination practices for pretraining data used in LLMs. While efforts such as those employed in decontaminating the Dolma corpus [46] utilize techniques such as paragraph-based decontamination, our results from the summarization case study reveal that contaminated data distributed throughout the dataset can have a comparable impact to instances where all contaminated data is grouped together, as in a paragraph. This highlights a critical gap in current decontamination practices, particularly in identifying and mitigating instance-level contamination effectively.

The recognition of this gap underscores the need for further advancements in decontamination techniques tailored to address specific types of contamination at the instance level. Future research in this area could focus on developing more sophisticated decontamination strategies that take into account the nuanced nature of contamination and its diverse manifestations in pretraining data. By addressing these gaps and developing more effective decontamination techniques, researchers can enhance the reliability and robustness of LLMs and minimize the potential adverse effects of contamination on model performance and generalization capabilities. This avenue of research represents a promising direction for advancing the field of decontamination and ensuring the integrity of pretraining data for future LLM development and deployment.

In summary, our research sheds light on the intricate interplay between contamination methodologies and model performance in both summarization and QA tasks. These insights underscore the importance of careful consideration in model training and evaluation processes to ensure reliable and robust performance across various datasets and tasks. Moving forward, our findings provide valuable guidance for practitioners and researchers in the development and evaluation of language models for natural language understanding tasks.

Chapter 9

Limitations

Several limitations warrant consideration. First, due to constraints in computational resources and time, our investigation focuses solely on continued pretraining. Consequently, we cannot investigate the impact of encountering contaminated data randomly throughout pretraining; instead, we consistently encounter contaminated data towards the pretraining's conclusion. This may influence our findings, potentially introducing a recency bias.

Additionally, this study primarily examines the performance of a single language model, limiting the generalizability of our findings to other models. Therefore, caution should be exercised when extrapolating our results to different language models, as variations in architecture, training procedures, and dataset characteristics may lead to different results. Further research involving multiple models and comprehensive evaluations is necessary to establish more robust conclusions regarding the impact of different types of contamination across diverse language models and tasks.

Chapter 10

Future Work

Our results suggest several avenues for future investigation. We hope that our work, in combination with future work in this direction, will advance our understanding of the effects of contamination during pretraining, refine model development practices, and enhance the reliability and robustness of language models across a multitude of tasks and domains.

Pretraining from Scratch

One avenue for future research involves pretraining the model with different settings from scratch rather than initializing from GPT-2 weights and conducting continued pretraining. By starting from scratch, researchers gain greater control over the pretraining corpora, allowing for a more comprehensive examination of the effects of different types of contamination during pretraining. Additionally, pretraining from scratch eliminates potential recency bias associated with continued pretraining, where contaminated data encountered last may disproportionately influence model behavior.

Exploration Across Model Architectures

Extending our methodology to other types of models with different architectures is another promising direction for future investigation. Exploring whether the effects observed extend beyond the GPT family can provide valuable insights into the generalizability of our findings and the impact of contamination across diverse model architectures.

Expansion to Diverse Tasks

Furthermore, future research should explore the impacts of different types of contamination on a wider range of tasks. While our study focused on summarization and question answering, language models are applied to various tasks beyond these domains. Investigating how models perform with different types of contamination during pretraining across diverse tasks can offer a more comprehensive understanding of contamination effects and inform best practices for model development and evaluation in real-world applications.

Bibliography

[1] Simone Balloccu, Patrícia Schmidtová, Mateusz Lango, and Ondrej Dusek. Leak, cheat, repeat: Data contamination and evaluation malpractices in closed-source LLMs. In Yvette Graham and Matthew Purver, editors, *Proceedings of the 18th Conference of the European Chapter of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 67–93, St. Julian's, Malta, March 2024. Association for Computational Linguistics. URL `https://aclanthology.org/2024.eacl-long.5`.

[2] Burton H. Bloom. Space/time trade-offs in hash coding with allowable errors. *ACM 13(7)*, pages 422–426, 1970. URL `https://dl.acm.org/doi/10.1145/362686.362692`.

[3] Arthur Brazinskas, Mengwen Liu, Ramesh Nallapati, Sujith Ravi, and Markus Dreyer. Transductive learning for abstractive news summarization. *CoRR*, abs/2104.09500, 2021. URL `https://arxiv.org/abs/2104.09500`.

[4] A.Z. Broder. On the resemblance and containment of documents. In *Proceedings. Compression and Complexity of SEQUENCES 1997 (Cat. No.97TB100171)*, pages 21–29, 1997. doi: 10.1109/SEQUEN.1997.666900.

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

[6] Jialun Cao, Wuqi Zhang, and Shing-Chi Cheung. Concerned with data contamination? assessing countermeasures in code language model, 2024.

[7] Kent K. Chang, Mackenzie Cramer, Sandeep Soni, and David Bamman. Speak, memory: An archaeology of books known to chatgpt/gpt-4, 2023.

[8] Yupeng Chang, Xu Wang, Jindong Wang, Yuan Wu, Linyi Yang, Kaijie Zhu, Hao Chen, Xiaoyuan Yi, Cunxiang Wang, Yidong Wang, Wei Ye, Yue Zhang, Yi Chang, Philip S. Yu, Qiang Yang, and Xing Xie. A survey on evaluation of large language models, 2023.

[9] Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Pondé de Oliveira Pinto, Jared Kaplan, Harrison Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, Alex Ray, Raul Puri, Gretchen Krueger, Michael Petrov, Heidy Khlaaf, Girish Sastry, Pamela Mishkin, Brooke Chan, Scott Gray, Nick Ryder, Mikhail Pavlov, Alethea Power, Lukasz Kaiser, Mohammad Bavarian, Clemens Winter, Philippe Tillet, Felipe Petroski Such, Dave Cummings, Matthias Plappert, Fotios Chantzis, Elizabeth Barnes, Ariel Herbert-Voss, William Hebgen Guss, Alex Nichol, Alex Paino, Nikolas Tezak, Jie Tang, Igor Babuschkin, Suchir Balaji, Shantanu Jain, William Saunders, Christopher Hesse, Andrew N. Carr, Jan Leike, Joshua Achiam, Vedant Misra, Evan Morikawa, Alec Radford, Matthew Knight, Miles Brundage, Mira Murati, Katie Mayer, Peter Welinder, Bob McGrew, Dario Amodei, Sam McCandlish, Ilya Sutskever, and Wojciech Zaremba. Evaluating large language models trained on code. *CoRR*, abs/2107.03374, 2021. URL `https://arxiv.org/abs/2107.03374`.

[10] Aakanksha Chowdhery, Sharan Narang, Jacob Devlin, Maarten Bosma, Gaurav Mishra, Adam Roberts, Paul Barham, Hyung Won Chung, Charles Sutton, Sebastian Gehrmann, Parker Schuh, Kensen Shi, Sasha Tsvyashchenko, Joshua Maynez, Abhishek Rao, Parker Barnes, Yi Tay, Noam Shazeer, Vinodkumar Prabhakaran, Emily Reif, Nan Du, Ben Hutchinson, Reiner Pope, James Bradbury, Jacob Austin, Michael Isard, Guy Gur-Ari, Pengcheng Yin, Toju Duke, Anselm Levskaya, Sanjay Ghemawat, Sunipa Dev, Henryk Michalewski, Xavier Garcia, Vedant Misra, Kevin Robinson, Liam Fedus, Denny Zhou, Daphne Ippolito, David Luan, Hyeontaek Lim, Barret Zoph, Alexander Spiridonov, Ryan Sepassi, David Dohan, Shivani Agrawal, Mark Omernick, Andrew M. Dai, Thanumalayan Sankaranarayana Pillai, Marie Pellat, Aitor Lewkowycz, Erica Moreira, Rewon Child, Oleksandr Polozov, Katherine Lee, Zongwei Zhou, Xuezhi Wang, Brennan Saeta, Mark Diaz, Orhan Firat, Michele Catasta, Jason Wei, Kathy Meier-Hellstern, Douglas Eck, Jeff Dean, Slav Petrov, and Noah Fiedel. Palm: Scaling language modeling with pathways, 2022.

[11] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota, June 2019. Association for Computational Linguistics. doi: 10.18653/v1/N19-1423. URL `https://aclanthology.org/N19-1423`.

[12] Jesse Dodge, Maarten Sap, Ana Marasović, William Agnew, Gabriel Ilharco, Dirk Groeneveld, Margaret Mitchell, and Matt Gardner. Documenting large webtext corpora: A case study on the colossal clean crawled corpus. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 1286–1305, Online and Punta Cana, Dominican Republic, November 2021. Association for Computational Linguistics. doi: 10.18653/v1/2021.emnlp-main.98. URL `https://aclanthology.org/2021.emnlp-main.98`.

[13] Leo Gao, Stella Biderman, Sid Black, Laurence Golding, Travis Hoppe, Charles Foster,

Jason Phang, Horace He, Anish Thite, Noa Nabeshima, Shawn Presser, and Connor Leahy. The pile: An 800gb dataset of diverse text for language modeling, 2020.

[14] Leo Gao, Stella Biderman, Sid Black, Laurence Golding, Travis Hoppe, Charles Foster, Jason Phang, Horace He, Anish Thite, Noa Nabeshima, Shawn Presser, and Connor Leahy. The pile: An 800gb dataset of diverse text for language modeling. *CoRR*, abs/2101.00027, 2021. URL `https://arxiv.org/abs/2101.00027`.

[15] Bogdan Gliwa, Iwona Mochol, Maciej Biesek, and Aleksander Wawer. SAMSum corpus: A human-annotated dialogue dataset for abstractive summarization. In *Proceedings of the 2nd Workshop on New Frontiers in Summarization*, pages 70–79, Hong Kong, China, November 2019. Association for Computational Linguistics. doi: 10.18653/v1/D19-5409. URL `https://aclanthology.org/D19-5409`.

[16] Shahriar Golchin and Mihai Surdeanu. Time travel in llms: Tracing data contamination in large language models, 2023.

[17] Sireesh Gururaja, Amanda Bertsch, Clara Na, David Widder, and Emma Strubell. To build our future, we must know our past: Contextualizing paradigm shifts in natural language processing. In Houda Bouamor, Juan Pino, and Kalika Bali, editors, *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 13310–13325, Singapore, December 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.emnlp-main.822. URL `https://aclanthology.org/2023.emnlp-main.822`.

[18] Suchin Gururangan, Ana Marasović, Swabha Swayamdipta, Kyle Lo, Iz Beltagy, Doug Downey, and Noah A. Smith. Don't stop pretraining: Adapt language models to domains and tasks. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 8342–8360, Online, July 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.acl-main.740. URL `https://aclanthology.org/2020.acl-main.740`.

[19] Felix Hill, Antoine Bordes, Sumit Chopra, and Jason Weston. The goldilocks principle: Reading children's books with explicit memory representations, 2016.

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

[21] Minhao Jiang, Ken Ziyu Liu, Ming Zhong, Rylan Schaeffer, Siru Ouyang, Jiawei Han, and Sanmi Koyejo. Investigating data contamination for pre-training language models, 2024.

[22] Anjan Karmakar, Julian Aron Prenner, Marco D'Ambros, and Romain Robbes. Codex hacks hackerrank: Memorization issues and a framework for code synthesis evaluation, 2022.

[23] Andrej Karpathy. mingpt, 2020. URL `https://github.com/karpathy/minGPT`.

[24] Andrej Karpathy. nanogpt, 2023. URL `https://github.com/karpathy/nanoGPT`.

[25] Kundan Krishna, Saurabh Garg, Jeffrey P. Bigham, and Zachary C. Lipton. Downstream datasets make surprisingly good pretraining corpora, 2023.

[26] Katherine Lee, Daphne Ippolito, Andrew Nystrom, Chiyuan Zhang, Douglas Eck, Chris Callison-Burch, and Nicholas Carlini. Deduplicating training data makes language models better. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 8424–8445, Dublin, Ireland, May 2022. Association for Computational Linguistics. doi: 10.18653/v1/2022.acl-long.577. URL `https://aclanthology.org/2022.acl-long.577`.

[27] Changmao Li and Jeffrey Flanigan. Task contamination: Language models may not be few-shot anymore, 2023.

[28] Inbal Magar and Roy Schwartz. Data contamination: From memorization to exploitation, 2022.

[29] Marc Marone and Benjamin Van Durme. Data portraits: Recording foundation model training data, 2023.

[30] Ramesh Nallapati, Bowen Zhou, Cicero dos Santos, Çağlar Gulçehre, and Bing Xiang. Abstractive text summarization using sequence-to-sequence RNNs and beyond. In *Proceedings of the 20th SIGNLL Conference on Computational Natural Language Learning*, pages 280–290, Berlin, Germany, August 2016. Association for Computational Linguistics. doi: 10.18653/v1/K16-1028. URL `https://aclanthology.org/K16-1028`.

[31] Shashi Narayan, Shay B. Cohen, and Mirella Lapata. Don't give me the details, just the summary! topic-aware convolutional neural networks for extreme summarization. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 1797–1807, Brussels, Belgium, October-November 2018. Association for Computational Linguistics. doi: 10.18653/v1/D18-1206. URL `https://aclanthology.org/D18-1206`.

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

[33] Hiroki Ouchi, Jun Suzuki, and Kentaro Inui. Transductive learning of neural language models for syntactic and semantic analysis. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 3665–3671, Hong Kong, China, November 2019. Association for Computational Linguistics. doi: 10.18653/v1/D19-1379. URL `https://aclanthology.org/D19-1379`.

[34] Matthew E. Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. Deep contextualized word representations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 2227–2237, New Orleans, Louisiana, June 2018. Association for Computational Linguistics. doi: 10.18653/v1/N18-1202. URL `https://aclanthology.org/N18-1202`.

[35] Aleksandra Piktus, Christopher Akiki, Paulo Villegas, Hugo Laurençon, Gérard Dupont, Alexandra Sasha Luccioni, Yacine Jernite, and Anna Rogers. The roots search tool: Data transparency for llms, 2023.

[36] Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. Improving language understanding by generative pretraining, 2018.

[37] Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. Language models are unsupervised multitask learners, 2019.

[38] Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, et al. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9, 2019.

[39] Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. Exploring the limits of transfer learning with a unified text-to-text transformer. *CoRR*, abs/1910.10683, 2020. URL `http://arxiv.org/abs/1910.10683`.

[40] Inioluwa Deborah Raji, Emily M. Bender, Amandalynne Paullada, Emily Denton, and Alex Hanna. Ai and the everything in the whole wide world benchmark, 2021.

[41] Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. SQuAD: 100,000+ questions for machine comprehension of text. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 2383–2392, Austin, Texas, November 2016. Association for Computational Linguistics. doi: 10.18653/v1/D16-1264. URL `https://aclanthology.org/D16-1264`.

[42] Manley Roberts, Himanshu Thakur, Christine Herlihy, Colin White, and Samuel Dooley. Data contamination through the lens of time, 2023.

[43] Oscar Sainz, Jon Campos, Iker García-Ferrero, Julen Etxaniz, Oier Lopez de Lacalle, and Eneko Agirre. NLP evaluation in trouble: On the need to measure LLM data contamination for each benchmark. In Houda Bouamor, Juan Pino, and Kalika Bali, editors, *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 10776–10787, Singapore, December 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.findings-emnlp.722. URL `https://aclanthology.org/2023.findings-emnlp.722`.

[44] Oscar Sainz, Jon Ander Campos, Iker Garcia-Ferrero, Julen Etxaniz, , and Eneko Agirre. Did chatgpt cheat on your test?, 2023. URL `https://hitz-zentroa.github.io/lm-contamination/blog`.

[45] Weijia Shi, Anirudh Ajith, Mengzhou Xia, Yangsibo Huang, Daogao Liu, Terra Blevins, Danqi Chen, and Luke Zettlemoyer. Detecting pretraining data from large language models, 2023.

[46] Luca Soldaini, Rodney Kinney, Akshita Bhagia, Dustin Schwenk, David Atkinson, Russell Authur, Ben Bogin, Khyathi Chandu, Jennifer Dumas, Yanai Elazar, Valentin Hofmann, Ananya Harsh Jha, Sachin Kumar, Li Lucy, Xinxi Lyu, Nathan Lambert, Ian Magnusson, Jacob Morrison, Niklas Muennighoff, Aakanksha Naik, Crystal Nam, Matthew E. Peters, Abhilasha Ravichander, Kyle Richardson, Zejiang Shen, Emma Strubell, Nishant Subramani, Oyvind Tafjord, Pete Walsh, Luke Zettlemoyer, Noah A. Smith, Hannaneh Hajishirzi, Iz Beltagy, Dirk Groeneveld, Jesse Dodge, and Kyle Lo. Dolma: an open corpus of three trillion tokens for language model pretraining research, 2024.

[47] Ian Tenney, Dipanjan Das, and Ellie Pavlick. BERT rediscovers the classical NLP pipeline. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4593–4601, Florence, Italy, July 2019. Association for Computational Linguistics. doi: 10.18653/v1/P19-1452. URL `https://aclanthology.org/P19-1452`.

[48] Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, Dan Bikel, Lukas Blecher, Cristian Canton Ferrer, Moya Chen, Guillem Cucurull, David Esiobu, Jude Fernandes, Jeremy Fu, Wenyin Fu, Brian Fuller, Cynthia Gao, Vedanuj Goswami, Naman Goyal, Anthony Hartshorn, Saghar Hosseini, Rui Hou, Hakan Inan, Marcin Kardas, Viktor Kerkez, Madian Khabsa, Isabel Kloumann, Artem Korenev, Punit Singh Koura, Marie-Anne Lachaux, Thibaut Lavril, Jenya Lee, Diana Liskovich, Yinghai Lu, Yuning Mao, Xavier Martinet, Todor Mihaylov, Pushkar Mishra, Igor Molybog, Yixin Nie, Andrew Poulton, Jeremy Reizenstein, Rashi Rungta, Kalyan Saladi, Alan Schelten, Ruan Silva, Eric Michael Smith, Ranjan Subramanian, Xiaoqing Ellen Tan, Binh Tang, Ross Taylor, Adina Williams, Jian Xiang Kuan, Puxin Xu, Zheng Yan, Iliyan Zarov, Yuchen Zhang, Angela Fan, Melanie Kambadur, Sharan Narang, Aurelien Rodriguez, Robert Stojnic, Sergey Edunov, and Thomas Scialom. Llama 2: Open foundation and fine-tuned chat models, 2023.

[49] Vladimir Vapnik. Statistical learning theory. *John Wiley Sons*, 1998.

[50] Jason Wei, Maarten Bosma, Vincent Y. Zhao, Kelvin Guu, Adams Wei Yu, Brian Lester, Nan Du, Andrew M. Dai, and Quoc V. Le. Finetuned language models are zero-shot learners. *CoRR*, abs/2109.01652, 2021. URL `https://arxiv.org/abs/2109.01652`.

[51] Yuhuai Wu, Felix Li, and Percy Liang. Insights into pre-training via simpler synthetic tasks, 2022.

[52] Shuo Yang, Wei-Lin Chiang, Lianmin Zheng, Joseph E. Gonzalez, and Ion Stoica. Rethinking benchmark and contamination for language models with rephrased samples, 2023.

[53] Chiyuan Zhang, Daphne Ippolito, Katherine Lee, Matthew Jagielski, Florian Tramèr, and Nicholas Carlini. Counterfactual memorization in neural language models, 2023.