Impact of Pretraining Term Frequencies on Few-Shot Numerical Reasoning

Yasaman Razeghi♢ Robert L. Logan IV♡ Matt Gardner♠ Sameer Singh♢♣
♢University of California, Irvine  ♡Dataminr Inc.  ♠Microsoft Semantic Machines  ♣Allen Institute for AI
{yrazeghi,sameer}@uci.edu
rlogan@dataminr.com mattgardner@microsoft.com

Abstract

Pretrained Language Models (LMs) have demonstrated ability to perform numerical reasoning by extrapolating from a few examples in few-shot settings. However, the extent to which this extrapolation relies on robust reasoning is unclear. In this paper, we investigate how well these models reason with terms that are less frequent in the pretraining data. In particular, we examine the correlations between the model performance on test instances and the frequency of terms from those instances in the pretraining data. We measure the strength of this correlation for a multiple GPT-based language models (pretrained on the Pile dataset) on various numerical deduction tasks (e.g., arithmetic and unit conversion). Our results consistently demonstrate that models are more accurate on instances whose terms are more prevalent, in some cases above 70% (absolute) more accurate on the top 10% frequent terms in comparison to the bottom 10%. Overall, although LMs appear successful at few-shot numerical reasoning, our results raise the question of how much models actually generalize beyond pretraining data, and we encourage researchers to take the pretraining data into account when interpreting evaluation results.

1 Introduction

Large language models have demonstrated outstanding zero- and few-shot performance on various reasoning benchmarks (Brown et al., 2020; Radford et al., 2019). In particular, their high performance on numerical tasks, such as addition and multiplication, suggests that models may have learned the ability to perform the underlying reasoning operations simply through a combination of pretraining and model size (Lewkowycz et al., 2022). As these numerical reasoning tasks become increasingly prevalent for evaluating the performance of large language models (Chowdhery et al., 2022), it is crucial to understand the extent to which performance on these tasks reflects robust reasoning capabilities, especially since numerical reasoning is an essential skill needed to perform other complex reasoning tasks such as question answering through reading comprehension (Dua et al., 2019), and commonsense reasoning (Thawani et al., 2021; Lin et al., 2020a).

Current schemes for evaluating the reasoning of large language models, however, often neglect or underestimate the impact of data leakage from pretraining data. Although overlap between the training and evaluation splits of public datasets and its effect on the generalization of language models has been studied (Elangovan et al., 2021; Lewis et al., 2021a), the effect of the pretraining data has received less attention, and very few studies have at-

Figure 1: Multiplication Performance: Plot of GPT-J-6B’s 2-shot accuracy on multiplication (averaged over multiple multiplicands and training instances) against the frequency of the equation’s term in the pretraining corpus. Each point represents the average performance for that term (e.g., 24) multiplied by numbers 0-99 and 5 choices of random seeds. As in the example, the performance difference for the numbers 24 and 23 is more than 20%. We find a strong correlation between the accuracy for a number and its frequency in pretraining.
tempted to evaluate the effect of pretraining data on model’s performance (Elazar et al., 2022). Ideally, a model that has learned to reason in the training phase should be able to generalize outside of the narrow context that it was trained in. Specifically, if the model has learned to reason numerically, its performance on instances with less frequent numbers (based on pretraining data) should not be significantly lower than its performance on the instances with common numbers.

For illustration, consider the arithmetic task of multiplying two integers (shown in Figure 1). A model that has learned proper arithmetic skills should be able to answer the queries irrespective of the frequencies of the operands in the pretraining data. Therefore, it should have roughly equivalent performance when answering the queries $Q$: what is $24$ times $X$? and $Q$: what is $23$ times $X$? despite the fact that $24$ appears more frequently in the pretraining data. This is not the case with current LMs and we will study the effect of frequency terms in details through this paper. To show the effect of frequency, in this example, we plot the average accuracy of GPT-J-6B (Wang, 2021) on the numbers 0–99 (averaged over 0–99 as the other operand) against the frequency of the number in the pretraining data in Figure 1. We find a strong correlation between the term frequency and the model performance indicating that the model reasoning is not robust to these frequencies. Note that even “rare” terms still appear on the order of millions of times in the pretraining data.

In this work, we investigate this impact of the frequency of test instance terms in a model’s pretraining data on the model’s performance. We experiment on numerical reasoning tasks of addition, multiplication, and unit conversion. We count occurrences of the numbers and units in instances of these tasks in the pretraining data, including co-occurrences of term pairs or triples within a fixed window. This procedure allows us to aggregate over instances in which these terms appear and observe the relationship between term frequency and model accuracy on instances that include those terms. We summarize this behavior through the performance gap between instances that have the most frequent terms and instances that have the least frequent terms. Intuitively, models that exhibit a high performance gap are much more accurate on instances that are more common in the pretraining data, suggesting that the model does not generalize appropriately and is affected by dataset overlap.

We present analysis on these numerical reasoning tasks for three sizes of the EleutherAI/GPT models pretrained on the Pile dataset (Gao et al., 2020), which has been publicly released and thus permits this kind of analysis (in contrast to the data that, e.g., GPT-3 (Brown et al., 2020) was trained on). Our results consistently show a large performance gap between highest- and lowest-frequency terms; in some cases there is a more than 70% average accuracy gap between the top and bottom 10% terms. We also investigate whether this performance gap can be explained by strong memorization effects, i.e. by instances that are memorized by the language model. To achieve this, we remove instances that contain frequent combinations of numbers from our analysis, and study the performance on the remaining instances. Even in this case, we still find a strong correlation between frequency of terms and average performance, indicating that our results cannot be explained solely by direct memorization.

These observations suggest that any evaluation of reasoning that does not take the pretraining data into account is difficult to interpret, and that we need to revisit evaluation of language models with respect to their pretraining data before making any conclusion about the models generalization abilities beyond the pretraining data.

2 Background and Methodology

Numerical reasoning has been essential part of complex multi-step reasoning tasks for natural language understanding (Dua et al., 2019; Wei et al., 2022). Recently, large language models have exhibited an ability to perform numerical reasoning tasks in few-shot settings without requiring any modifications to their parameters through a method called in-context learning (Brown et al., 2020; Chowdhery et al., 2022). Our goal is to evaluate this reasoning skill in-depth and with respect to the pretraining data. This section provides background information on in-context learning and introduces our method for measuring the performance gap of the models on numerical reasoning tasks based on differences in pretraining term frequency.

The demonstration of all experiments in this paper is available at https://nlp.ics.uci.edu/snoopy (Razeghi et al., 2022) and the code is available at https://github.com/yasamanrazeghi7/TermFrequency
2.1 In-context Learning

Brown et al. (2020) show that the large GPT-3 model is able to perform well on few-shot reasoning tasks without requiring any changes to its internal parameters, through the usage of a technique called in-context learning. In place of a typical learning procedure, in-context learning instead places training examples in a prompt format, which is subsequently fed to a language model as its input. Recently, a few studies have researched the role of prompt and investigated the aspects that make in-context learning successful (Min et al., 2022; Zhao et al., 2021; Chan et al., 2022).

Among numerous experiments, Brown et al. (2020) show that GPT3 performs well on a variety of arithmetic questions such as addition and subtraction with 2–5 digit numbers. For example, they show that the largest model can perform zero-shot 2-digit addition with 76.9% accuracy. Although impressive, due to the large volume of data GPT-3 is trained on, it is possible that the model is repeating answers seen during pretraining. To attribute this performance to the model’s reasoning capabilities, we need to make sure that the model is not affected by statistical overlaps between the terms of the arithmetic questions and the pretraining data.

In the following sections, we introduce metrics that we use to investigate the relationship between the frequency of terms in the pretraining data and the model performance on reasoning instances containing those terms. To assess this relation, we first define an approach for measuring term frequencies in a large pretraining dataset (Section 2.2). We connect these frequencies to reasoning performance by introducing the performance gap $\Delta$ (Section 2.3).

2.2 Frequency

We consider numerical reasoning tasks (Table 1) whose instances consist of input terms, $x = (x_1, \ldots, x_i, \ldots, x_n)$, and a derived output term $y$, where the $x_i$’s are either positive integers or units of time (e.g., 1, 2, hour, etc.) and $y$ is a positive integer. For example, for the task of multiplication, an instance might be $x = (23, 18)$ and $y = 414$, representing the equation $23 \times 18 = 414$.

For each instance, we extract counts of the number of times that a subset of its terms $X \subseteq \{x_1, \ldots, x_n, y\}$ appear within a specified window in the pretraining data. We refer to this count as the frequency, $\omega_{X}$, of $X$.

In this paper, we restrict our attention to frequencies involving three or less input terms, e.g., $x = (x_1, x_2)$ or $(x_1, x_2, x_3)$ and optionally the output term $y$, e.g.:

- $\omega_{x_1}$: the number of times that $x_1$ (one of the terms, e.g., 23) appears in the pretraining data.
- $\omega_{x_1, x_2}$: the number of times that the input terms $x_1$ (e.g., 23) and $x_2$ (e.g., 18) appear in the pretraining data within a specific window size.
- $\omega_{x_1, y}$: the number of times that the first input term $x_1$ (e.g., 23) and the output term $y$ (e.g., 414) appear in the pretraining data within a specific window size.

Note that our usage of set notation in the subscript is deliberate; although $x = (x_1, x_2)$ and $x' = (x_2, x_1)$ are not necessarily the same (e.g., order is important when representing the task instance), frequency is symmetric (e.g., $\omega_{x_1, x_2} = \omega_{x_2, x_1}$).

2.3 Performance Gap

We want to measure how much more accurate the model is on instances containing more versus less frequent terms in the pretraining data. We do this by calculating the differences in average accuracies of the instances in the top and bottom quantiles of the distribution over term frequencies, which we call the performance gap.

Formally, let $\{(X^{(n)}, \omega_{X}^{(n)})\}$, $n \in [1, N]$, be a set of terms for a task and their associated term frequencies in the pretraining corpus. Given a task (e.g., addition), we create reasoning instances for each element of this set by instantiating values of $x_i$ and deriving $y$. We then measure the LM’s accuracy $a^{(n)}$ over the set of instances, and repeat this process for all $n \in [1, N]$, producing a set $\Omega = \{(\omega_{X}^{(n)}, a^{(n)})\}$. The formula for the performance gap is then given by:

$$\Delta(\Omega) = \text{Acc}(\Omega_{>90%}) - \text{Acc}(\Omega_{<10%}) \quad (1)$$

where $\Omega_{>90%}$ is the top 10% of elements in $\Omega$ ordered by frequency, $\Omega_{<10%}$ is the bottom 10%, and $\text{Acc}(\Omega')$ is the average accuracy of elements in $\Omega'$. We introduce the following convenient abuses of notation $\Delta_1, \Delta_{1,2}, \Delta_{1,y} \ldots$, to denote the performance gap over the frequency distributions of $\omega_{x_1}, \omega_{x_1, x_2}, \omega_{x_1, y}, \ldots$, respectively.

Concretely, for the multiplication example from Figure 1, $x = (x_1, x_2)$ and we consider the performance gap over frequencies $\omega_{x_1}$. For each number (say 23), we count the number of times it appears in the pretraining corpus ($\omega_{23}$), and
Figure 2: Pipeline for Data Construction: We use the term counts processed from the pretraining data to develop the reasoning queries and render them with prompts templates to a proper language model input format.

Table 1: Prompt templates and the number of test cases (#) investigated for each numerical reasoning task.

<table>
<thead>
<tr>
<th>Task</th>
<th>Prompt Template</th>
<th>#</th>
</tr>
</thead>
<tbody>
<tr>
<td>Arithmetic</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Multiplication</td>
<td>Q: What is $x_1$ times $x_2$? A: $y$</td>
<td>$10^4$</td>
</tr>
<tr>
<td>Addition</td>
<td>Q: What is $x_1$ plus $x_2$? A: $y$</td>
<td>$10^4$</td>
</tr>
<tr>
<td>Operation Inference</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Mult. #</td>
<td>Q: What is $x_1$ # $x_2$? A: $y$</td>
<td>$10^4$</td>
</tr>
<tr>
<td>Add. #</td>
<td>Q: What is $x_1$ # $x_2$? A: $y$</td>
<td>$10^4$</td>
</tr>
<tr>
<td>Time Unit Inference</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Min→Sec</td>
<td>Q: What is $x_1$ minutes in seconds? A: $y$</td>
<td>79</td>
</tr>
<tr>
<td>Hour→Min</td>
<td>Q: What is $x_1$ hours in minutes? A: $y$</td>
<td>100</td>
</tr>
<tr>
<td>Day→Hour</td>
<td>Q: What is $x_1$ days in hours? A: $y$</td>
<td>100</td>
</tr>
<tr>
<td>Week→Day</td>
<td>Q: What is $x_1$ weeks in days? A: $y$</td>
<td>100</td>
</tr>
<tr>
<td>Month→Week</td>
<td>Q: What is $x_1$ months in weeks? A: $y$</td>
<td>100</td>
</tr>
<tr>
<td>Year→Month</td>
<td>Q: What is $x_1$ years in months? A: $y$</td>
<td>100</td>
</tr>
<tr>
<td>Decade→Year</td>
<td>Q: What is $x_1$ decades in years? A: $y$</td>
<td>100</td>
</tr>
</tbody>
</table>

compute the average accuracy of the model over all instances where one of the operands is 23. The performance gap w.r.t. to $\omega_{\{23\}}$ for this task is the difference between the average accuracy over the top 10% and the bottom 10% most frequent numbers in the pretraining corpus. We picked 10% as the threshold to have a simple, intuitive metric that captures how accuracy differs between the most and least frequent terms. We also provide the plots to show the full distribution in the frequency range.

3 Experiment Setup

In this section, we describe our setup to measure the effect of pretraining data on the few-shot evaluation of a number of numerical reasoning tasks for different language models.

Language Models We experiment on the following models from EleutherAI: GPT-J-6B (Wang, 2021), and GPT-Neo-1.3B, GPT-Neo-2.7B (Black et al., 2021). These models are publicly available, but more importantly, they are among the few models that their pretraining corpus has also been released. These language models are trained on the Pile dataset (Gao et al., 2020), a large-scale language modeling dataset consisting of English documents in 22 academic or other professional data sources. We count the frequency of all integers with less than seven digits using a slightly modified version of Spacy English tokenizer (Honnibal and Montani, 2017). To calculate the frequencies of the numbers we use Amazon Elastic Map Reduce (EMR) platform. We use the HuggingFace1 Transformer integration of the models for experiments.

Numerical Reasoning Tasks We create three types of datasets that target the mathematical capabilities of language models since solving mathematical questions is a useful reasoning capability of the models (Brown et al., 2020).

- Arithmetic, 2 tasks As the first task, we consider simple arithmetic operations: addition $x_1 + x_2 \rightarrow y$ and multiplication $x_1 \times x_2 \rightarrow y$. In both cases, the both operands ($x_1$ and $x_2$) are numbers less than 100 (these numbers are in the top 200 most frequent numbers in the pretraining data).

- Operation Inference, 2 tasks Instead of directly specifying the operation, we also create a variation where the model needs to infer, from a few examples, the operation itself, as well as the result, as introduced in the evaluation of Megatron-Turing model.2 We replace the arithmetic operation with a “#”, with the same operations and operands as previous, to create these datasets.

- Time Unit Conversion, 7 tasks Apart from direct arithmetic expressions, we are also interested in evaluating model capability to implicitly reason about these operations. To this end, we construct a unit conversion dataset by identifying the most frequent numbers that co-occur with time unit words (“second”, “minute”, “hour”, “day”, “week”, “month”, “year”, and “decade”) as the primary operand $x_1$, the time units themselves as additional operands ($x_2 \rightarrow x_3$), i.e. converting 24

---

1 Source code at https://huggingface.co/EleutherAI

2https://turing.microsoft.com/
hours to minutes is represented as (24, “hours”, 60). We expect converting time values to be mathematically more straightforward than two-digit multiplication since the model need only multiply with the same (implicit) second operand, e.g., $\times 60$ for converting hours to minutes.

The pipeline for creating instances in our evaluation is illustrated in Figure 2. We compute occurrences and co-occurrences (assuming a window of 5) of the terms in the corpus, i.e. the time units and numbers. We generate instances for the reasoning tasks using the most frequent terms with less than 3 digits (the top 200) as operands. We focus on the top terms since we expect the models to have a fairly reliable and robust representations for these words. Each reasoning instance is rendered as a natural language query using the prompt templates from Table 1, and input to the language model to generate the answer. For example, to create a multiplication instance given the terms ($x_1 = 23$, $x_2 = 18$), we use the instance template to create a natural language input for the model as “Q: What is 23 times 18? A: __”, with the goal of producing “414” ($y = 23 \times 18 = 414$). For few-shot evaluation, we prompt the language models with $k = 0, 2, 4, 8, 16$ shots, and average performance over five random selection of the prompt instances.

4 Results

With the three types of numerical reasoning tasks (consisting of 11 total datasets), we present an evaluation of the effect of pretraining term frequency on the performance of the language models. For each dataset, we measure the performance gap on instances that consist of rarer (relatively) terms, for a few different choices of what to compute frequency over (different combinations of the instance terms). We also investigate the effect of the model size on this performance gap and do a case study to further clarify if all this impact is due to memorization.

**Arithmetic** We first study the performance on simple addition and multiplication of numbers. The results for the GPT-J-6B model is provided in Table 2, with performance gap computed just for $x_1$ (any of the multiplicands), for $(x_1, x_2)$ (both multiplicands), and for $(x_1, y)$ (any of the multiplicands and the golden answer). In *multiplication*, we observe a very high performance gap for all these definitions of frequencies, suggesting a strong effect of frequency in the pretraining data on the model’s ability to perform multiplication. For better illustration of the performance gap, we plot the mean accuracy across the frequency of $x_1$ in Figure 3b. The plot demonstrates the strong correlation between the models accuracy on specific instances, and the instance element frequency in the pretraining data. For *addition*, we observe an overall higher performance of the GPT-J-6B model in comparison to the multiplication experiments. However, the performance gap on all of the definitions of the instance frequencies still shows an strong effect on the models accuracy. As shown in Figure 3a, the average accuracy of the model still has a positive slope, indicating the effect of instance frequencies.

**Operation Inference** These tasks aim to assess the model capability to both infer the math operation and to perform the actual computation. As we see in Table 2, the model is much less accurate here as compared to the arithmetic experiments. However, the model has better performance on the frequent instances even for these low performance tasks (see detailed trend in Figures 3d and 3c).
performance gap here suggests that the effect of pretraining is not only for tasks that the model is accurate on, but even for operation inference that is more challenging and require deeper reasoning. Moreover, the lower accuracy here as compared to addition experiments in the previous section suggests that the model is unable to infer the operation from the few-shot prompts, and it may be performing some form of pattern matching based on the pretraining data on the common instances.

**Time-Unit Conversion**  The performance gap evaluated on the time unit conversion experiments is in Table 3. We first observe a relatively high performance gap on all the tasks except the conversion from decade to year. We also observe a general pattern of increase in the performance gap as the number of shots (training examples in the prompt) increases. These results suggest that even though the model gets more accurate, the improvements focus on more frequent instances of the task. (Example figures for time units experiments is provided in Figures 4, 5)

*Decades to years:* As we observe in Table 3, the model performs nearly perfectly on this task with as few as 8 shots, and we only see very small performance gap. This is likely due to the task being quite simple (appending a “0” to the input number) so, the model is able to generalize in the manner we are evaluating it. However, it is also possible that we are simply not identifying the right frequency statistics for this task, and there is an effect that our current evaluation setup does not capture.

**Studying the Size of Language Models**  To further study the impact of language models sizes on the performance gap caused by the instance frequencies, we perform the arithmetic experiments for 2, 8 shots using the smaller models (GPT-Neo-1.3B and GPT-Neo-2.7B). We can see the trends of the average accuracy of the models in Figures 6. The smaller models overall are less accurate on the arithmetic tasks, which is consistent with observations in related work (Brown et al., 2020). However, their success is still focused on the more frequent terms from the pretraining corpus, suggesting that even the smaller models show the effect of reliance on the pretraining data, although to a much lower extent than the larger ones.
Table 3: GPT-J-6B results on Time-Unit Conversion: $\Delta_{1.2}$, $\Delta_{1.2,3}$ and $\Delta_{1.2,y}$ represent the performance gap over the frequency distributions of $\omega_{\{x_1, x_2\}}$, $\omega_{\{x_1, x_2, x_3\}}$ and $\omega_{\{x_1, x_2,y\}}$, respectively, where $x_1$ is the number operand, $x_2$ is the source unit, $x_3$ is the number operand needed for performing the conversion and $y$ is the true answer.

<table>
<thead>
<tr>
<th>$k$</th>
<th>Min→Sec</th>
<th>Hour→Min</th>
<th>Day→Hour</th>
<th>Week→Day</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>$\Delta_{1.2}$</td>
<td>$\Delta_{1.2,3}$</td>
<td>$\Delta_{1.2,y}$</td>
<td>$\Delta_{1.2}$</td>
</tr>
<tr>
<td>0</td>
<td>1.3</td>
<td>0.0</td>
<td>0.0</td>
<td>12.5</td>
</tr>
<tr>
<td>2</td>
<td>25.5</td>
<td>60.0</td>
<td>62.5</td>
<td>62.5</td>
</tr>
<tr>
<td>4</td>
<td>35.5</td>
<td>60.7</td>
<td>65.0</td>
<td>50.6</td>
</tr>
<tr>
<td>8</td>
<td>49.9</td>
<td>72.7</td>
<td>82.3</td>
<td>42.1</td>
</tr>
<tr>
<td>16</td>
<td>58.4</td>
<td>87.5</td>
<td>88.5</td>
<td>64.2</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Shots, $k$</th>
<th>Month→Week</th>
<th>Year→Month</th>
<th>Decade→Year</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>$\Delta_{1.2}$</td>
<td>$\Delta_{1.2,3}$</td>
<td>$\Delta_{1.2,y}$</td>
</tr>
<tr>
<td>2</td>
<td>1.0</td>
<td>0.0</td>
<td>0.0</td>
</tr>
<tr>
<td>4</td>
<td>30.1</td>
<td>9.0</td>
<td>13.0</td>
</tr>
<tr>
<td>8</td>
<td>63.3</td>
<td>21.5</td>
<td>25.5</td>
</tr>
<tr>
<td>16</td>
<td>80.9</td>
<td>37.5</td>
<td>24.0</td>
</tr>
</tbody>
</table>

Figure 6: The effect of model size on performance
Smaller models only perform well on instances with more frequent terms in the pretraining data. $k$ represents the number of shots.

Impact Due to Memorization In this section, we will study the extent to which the impact of the pretraining term frequencies on model performance can be explained due to pure memorization. To tease apart direct memorization, we perform similar analysis as above, but do so without the instances that have already been memorized by the language model in their entirety. It is worth mentioning that finding such instances is not trivial. In other words, it is not trivial to identify purely memorized instances. Prior work (Magar and Schwartz, 2022; Carlini et al., 2022) has shown that the models perform more accurately on the instances with higher exact match counts from the pertaining data; we first verify this trend by observing a similar trend for our setting by plotting $\omega_{\{x_1, x_2\}}$, the co-occurrence of all three numbers in Appendix Figure 9. Based on these studies and results, we treat the number triples with the highest average accuracy (more than 85%), as the ones the model has most likely memorized. Specifically, we remove these memorized instances from our evaluation to see the impact of lower order term frequencies on the remaining instances. As shown in Figure 7, the dependency of model performance on lower order frequencies ($\omega_{\{x_1\}}$ and ($\omega_{\{x_1, x_2\}}$) is still very high even after removing the memorized instances. These observations suggest that the impact of term frequencies on model performance is beyond pure memorization of the numerical terms.

Summary Overall, we observe high positive performance gap for almost all of the experiments on the three definition levels of the frequency for each task. This suggests a strong effect of frequency of the instances in the pretraining data on the model performance. In particular, evaluation using performance gap with $\omega_{\{x_1\}}$ shows that even the unigram statistics of the instances have strong correlation with the models performance on the instance.

Other than some exceptional cases, we observe an increasing trend in the performance gap as we put more training examples in the prompt (the number of shots); this can be a further indication that the model is directed through the patterns in the pretraining data to answer the reasoning questions. Our experiments with the smaller sizes of the model also show that they can only solve the frequent instances of the tasks, which further supports our observation that model performance is correlated with the term frequencies.
5 Related Work

A large and growing body of literature has investigated a number of related concerns with large language models (for discussion of more tangentially related work see Appendix A.1).

Numeracy and Temporal Reasoning in LMs

Our work contributes to the larger body of work studying numeracy in word embeddings and language models (Spithourakis and Riedel, 2018; Wallace et al., 2019). Geva et al. (2020), Zhou et al. (2020) Zhou et al. (2022) and Lewkowycz et al. (2022) propose training schemes to help improve LMs’ temporal and numerical reasoning capabilities. Patel et al. (2021) show that NLP math solvers rely on simple heuristics to answer math questions. We expect that the performance gap metric proposed in this work will be useful to better understand the impact of such schemes.

Impact of Frequency on LM Performance

Kassner et al. (2020) and Wei et al. (2021) perform controlled experiments varying pretraining data to characterize the extent to which pretraining affects LMs’ ability to memorize and reason with facts as well as learn generalizable syntax rules. In line with our results, both of these find that frequency is a distinguishing factor in whether or not the model memorizes a particular fact or syntactic rule for a verb form. Sinha et al. (2021) further demonstrate that shuffling word order during pretraining has minimal impact on an LMs’ accuracy on downstream tasks, and, concurrent with this work, Min et al. (2022) similarly find that shuffling labels in in-context learning demonstrations has minimal impact on few-shot accuracy. These results further suggest that LMs’ performance is largely driven by their ability to model high-order word co-occurrence statistics. Data privacy researchers have shown that LMs may memorize sensitive sequences occurring in training data even if they are rare (Carlini et al., 2019; Song and Shmatikov, 2019).

Memorization

Feldman (2020) provide a theoretical definition of memorization as the difference between the accuracy of a model on a training data point when that point is included vs. excluded from training. They also develop an approach for approximating memorization using influence functions Feldman and Zhang (2020). This framework is applied to study memorization in language models by Zhang et al. (2021), who find that training examples that are memorized by the LM tend to have high influence of LM predictions on similar validation instances. Their result may provide a plausible explanation that the frequency effects observed in this work are due to memorization.

6 Discussion

In this work, we consider how to conduct few-shot evaluations in light of the analysis with the pretraining data. Prior work has attempted to control for overlap between pretraining data and the test instances, but as we have seen, those methods are insufficient. For example, Brown et al. (2020) measure the impact of removing instances from evaluation datasets that share 13-gram overlap with their pretraining data on GPT-3’s accuracy, and also argue that the low occurrence of exact phrases such as “NUM1 + NUM2 =” and “NUM1 plus NUM2” in the pretraining data indicate that the model’s strong performance on arithmetic tasks is likely due to factors other than memorization. However, we show that LM performance is impacted by much simpler statistical patterns, as small as unigram overlap with the pretraining data.

For these reasons, we strongly recommend that evaluation of reasoning capabilities should take the
pretraining corpus into account, and any claims of reasoning can only be made after demonstrating robustness to the effect of pretraining. Current LM benchmarks, that are dissociated from the model’s pretraining data, make it impossible to interpret few-shot reasoning performance results. It is worth mentioning that, even a performance gap of 0 is likely not sufficient to claim reasoning capabilities—what exactly constitutes “reasoning” remains ill-defined—but it may be a necessary condition, and one that current models do not meet.

In this study, we are not making a causal claim, and in general, there may be confounders that we have not eliminated in our setting. Recently, Elazar et al. (2022) introduced a causal framework based on pretraining data statistics for understanding language model’s factual predictions. To be able to use the causal inference techniques they construct and assume a causal graph for the task of extracting factual knowledge from pretrained language models. We recommend further research in the proposed direction for other NLP tasks such as reasoning and interventions during training to provide finer-grained analysis of the effect of pretraining.

One potential concern is that our experiments do not distinguish whether incorrect answers are due to lack of reasoning or lack of recognition, i.e. it is possible that the model has the ability to multiply but the embeddings for rare terms are not adapted to that algorithm. However, recognizing numbers is a prerequisite to numerical reasoning, thus if the models lack the ability to identify numbers, this still means that they lack numerical reasoning skills. That said, we also suspect that the errors are not due to recognition. Even the most infrequent terms in our experiments have been seen millions of times—they are not unknown tokens.

## 7 Conclusion

We show that in-context language model performance on numerical reasoning tasks can be impacted significantly by low-order co-occurrence statistics in the pretraining data, raising questions on the extent to which these models are actually reasoning to solve these tasks. These observations suggest the necessity for reconsidering and redefining the reasoning evaluation schemes for the large language models. Further characterizing the impacting factors on the models reasoning capacities is also an important tasks for the community. Most importantly, we suggest that the community should not treat the pretraining data of the large language models as unknown black boxes. Overlooking the impact of the pretraining data can be misleading in evaluating the model reasoning skills.

## Acknowledgements

We would like to thank the members of UCI-NLP, Yanai Elazar, Mukund Sundarajan, Marco Tulio Ribeiro, Eric Wallace, Shivanshu Gupta, Navid Salehnamadi, Pouya Pezeshkpour, and Dylan Slack for valuable discussions and feedback on this work. This material is sponsored in part by the DARPA MCS program under Contract No. N660011924033 with the United States Office Of Naval Research, by an Amazon Research Award, and by awards IIS-2046873 and IIS-2040989 from the National Science Foundation.

## Limitations

There are a few limitations to our study that open up avenues for future research. First, our approach aggregates fairly simple patterns and the effect we observe might be stronger if a wider variety and complexity of patterns is considered in the pretraining corpus. Similarly, our work is limited to simple numerical reasoning tasks, and it would be worthwhile to study how much other reasoning evaluations and more complex quantitative reasoning tasks such as GSM8K (Cobbe et al., 2021) are impacted by the same effect, which could be measured using the performance gap metric introduced here. Defining appropriate instance terms for other reasoning tasks such as commonsense reasoning will be a challenging but important direction for future work. Lastly, we do not propose a solution for changing language models to robust reasoners. We hope that the insights in this work inspire further studies into the effect of pretraining on language model’s performance, improvements in evaluation schemes, and better training mechanisms for more robust language models with true generalization capabilities.

## References


A Appendix

A.1 Additional Related Work

In this section, we further discuss the related work.

Prompting

Prompting has been widely applied to study the factual (Petroni et al., 2019), commonsense (Davison et al., 2019; Weir et al., 2020; Lin et al., 2020b), mathematical (Saxton et al., 2019), and other NLP task-related (Radford et al., 2019; Shin et al., 2020) knowledge LMs acquire during pretraining. In this work, we focus on the in-context learning setup of Brown et al. (2020), who use prompts that include training examples to diagnose LMs’ few-shot learning capabilities.

Training Artifacts Challenge Evaluation

Our results raise the issue that in-context learning probes may overestimate an LM’s ability generalize from few examples when biases are present in the training data. This is consistent with prior work that has exposed the similar effects of biases from: lexical cues in natural language inference datasets (Gururangan et al., 2018; Poliak et al., 2018; McCoy et al., 2019), question-passage overlap and entity cues in reading comprehension datasets (Chen et al., 2016; Sugawara et al., 2018; Jia and Liang, 2017; Lewis et al., 2021b), gender cues in coreference resolution datasets (Rudinger et al., 2018), popularity in named entity disambiguation (Chen et al., 2021), similarity between training and test instances in information extraction and sentiment analysis datasets (Elangovan et al., 2021), and effects of how data is split (Gorman and Bedrick, 2019; Søgaard et al., 2021). Relatively, data poisoning research studies how to adversarially introduce artifacts into training data to produce unwanted model behaviors (Nelson et al., 2008; Chan et al., 2020; Wallace et al., 2021). A general statistical procedure to test for artifacts is presented in Gardner et al. (2021), who also theoretically show that large datasets are almost certain to contain artifacts under reasonable assumptions. Techniques for mitigating biases in the presence of dataset artifacts are covered by Romanov et al. (2019) and Karimi Mahabadi et al. (2020).

Documenting Pretraining Data

To better understand the risks of dataset artifacts, there has been a call to better document the characteristics and intended uses of datasets (Gebru et al., 2021; Bender et al., 2021). However, due to the sheer size of LM pretraining datasets—which range from 100’s of GBs to 10’s of TBs—doing so can pose a substantial challenge. Despite this, researchers have been able to estimate word frequencies, topics, and proportions of toxic text (Sharoff, 2020), as well as proportions of toxic text (Gehman et al., 2020) appearing in OpenWebText (Gokaslan and Cohen, 2019). Similar efforts have been made to characterize the top-level domains, amount of hate speech, and censored text appearing in the C4 corpus (Raffel et al., 2020; Dodge et al., 2021; Luccioni and Viviano, 2021). Our work documents co-occurrence statistics of numbers and dates of documents appearing in the Pile dataset.

A.2 Examples of time unit conversion plots

We provide the figures showing the dependence between the average accuracy and the $\omega_{x_1,x_2}$ for time unit experiments of Minute, Year and Decade in Figures 10, 4 and 5, respectively.
Figure 9: The impact of $\omega\{x_1, x_2, y\}$ (the frequency of all numbers $(x_1, x_2, y)$ in an arithmetic instance) on GPT-J-6B’s 2-shot performance, the high dependence of models average accuracy on $\omega\{x_1, x_2, y\}$ may be due to memorization specifically in highest frequencies.

Figure 10: GPT-J-6B performance on Minute$\rightarrow$Second: The interpolation lines show the correlation between the average accuracy and the $\omega\{x_1, x_2\}$. $k$ is the number of shots.