Assessing Contamination in Large Language Models: Introducing the LogProber method

Anonymous ACL submission

Abstract

In machine learning, "contamination" refers to situations where testing data leak into the training set. The issue is particularly relevant for the evaluation of the performance of Large Language Models (LLMs), which are generally trained on gargantuan, and generally opaque, corpora of text scraped from the world wide web. Developing tools to detect contamination is therefore crucial to be able to fairly and properly track the evolution of the performance of LLMs. Most recent works in the field are not tailored to quantify contamination on short sequences of text like we find in psychology questionnaires. In the present paper we introduce LogProber, a novel, efficient, algorithm that we show able to detect contamination using token probability in given sentences. In the second part we investigate the limitations of the method and discuss how different training methods can contaminate models without leaving traces in the token probabilities.

1 Introduction

Large language models (LLMs) are deep-learning systems trained on huge corpora or textual data, which size and complexity makes impossible to predict *ex ante* the extent and depth of their capabilities (Brown et al., 2020a). The situation is made even more complex by the fact that their competences span quite different domains, ranging from content creation to translation and coding. This is why there has been a proliferation of studies proposing new benchmarks and tools aimed at assessing their capabilities (Hendrycks et al., 2021; Srivastava et al., 2023).

Most of these evaluation methods rely on asking questions to the LLM and evaluating whether the resulting answers are correct. Crucially, this method is valid if, and only if, the model was not trained on the same material used in the benchmark items. This is particularly true when the benchmark aims at assessing the capabilities of the model to solve a particular class of problems (i.e., its cognitive abilities) and not its capacity to retrieve factually accurate information (i.e., the accuracy of its knowledge). In machine learning, this situation where part of the testing set is leaked in the training phase is referred to as "contamination". When contamination occurs, a model's performance in a given task may not reflect the true capability of a model in a given domain, but rather its capacity to retrieve training material.

This contamination issue is particularly relevant in a field where models are trained on such a vast amount of text, where it is very difficult to check the presence of benchmark items in the training set (Brown et al., 2020b; Zhou et al., 2023). Accordingly, several recent models showing impressive scores on benchmarks have been suspected of contamination on these specific benchmarks, but this criticism remains unclear due to the lack of transparency about the training details of these models (Gunasekar et al., 2023; Li et al., 2023; Jiang et al., 2024; Balloccu et al., 2024). The contamination issue is also particularly relevant when LLMs' performance is evaluated using cognitive science tools (Binz and Schulz, 2023; Yax et al., 2024), because, as opposed to machine learning benchmarks, tests in psychology and cognitive science are generally older, thus allowing ample time to permeate the existing corpora, and generally involve a fewer number of questions (Coda-Forno et al., 2024). In addition, in machine learning, keeping test sets and training sets as independent as possible is a major concern that seems not to apply to psychology items making the study of machine psychology very difficult and requires the development of adapted tools.

Various methods have already been proposed to detect the presence of contamination. Some methods, such as n-gram matches present the obvious limitation that they require access to the train-

ing data - which is rarely available (Brown et al., 2020b; Raffel et al., 2023; Touvron et al., 2023b; Hendrycks et al., 2021; Touvron et al., 2023a). Other methods have been developed that generally work on the assumption that contaminated text should be associated with highest completion probability. Some approaches focus on finding if a part of a document was included in the training corpus by looking if its token probability is particularly high (Meeus et al., 2023) or by verifying if a permutation of the lines in the document is more likely than others (Oren et al., 2024). However, this approach is ill-suited to evaluate contamination for some LLM benchmarks or more specifically cognitive tests, because they usually involve only a few very short questions as they require a lot of text to be efficient. Another very relevant approach, also based on token probability analysis is more precise and can be used with rather short texts (Dong et al., 2024). It focuses on quantifying how many different completion a LLM is likely to generate to a given question, the lower the more likely the LLM is to be contaminated on the given question. However, we believe the fact that a LLM completes (or responds) always in the same way does not necessarily imply contamination, but could rather reflect higher confidence for a specific completion or response (which is often the case for answers provided in a question/answer format, such as the one often implemented in some benchmarks and often in cognitive tasks).

The first aim of this paper is to present a new algorithm (LogProber) that allows for disentangling contamination from confidence in a way that is computationally cheap and adapted to the short question/answer format, such as the one usually used in machine learning benchmarks and cognitive tests. After illustrating the underlying principles and mechanisms behind LogProber, we validate it in dedicated experiments where we finetune a LLM (LLAMA (Touvron et al., 2023b)) with specific items of a recently published cognitive test (Yax et al., 2024). The results show that the method is effective in detecting contamination. However, in additional model training experiments, where the model doesn't fit on all the question and answer tokens but only on the answer tokens, we found that LogProber was incapable of detecting the contamination. As such, the second aim of the paper is to demonstrate that not all types of contamination can be detected by simple analysis of token probability of a given benchmark or cognitive test question.

Intriguingly, our fine-tuning experiments also show that LLMs can perfectly memorize the answer to a given question without showing signs of memorization of the question itself.

2 Methods

2.1 What is *contamination* and how does it relate to *confidence*?

Benchmarks and cognitive tasks are usually collections of questions ('Q') associated with a specific correct answer ('A'). LLMs' performance is usually assessed giving 'Q' as input (or context, prompt) and taking the resulting completed text as the response of the model. The performance of the LLM is then assessed by comparing its response with the known correct answer ('A').

Contamination broadly refers to the fact that the testing materials (benchmark questions or cognitive tasks) were present in the training corpus. LLM's performance is then considered to be driven directly by training data rather than from its generalization (or emergent cognitive) capacities. In the context of benchmarks and cognitive tasks, the question of contamination boils down to answering the question of whether or not the model has been trained on the test material ('Q' and 'A'), because if the sample 'Q-A' appears in the training dataset, the model will learn to "automatically" predict A from context 'Q'. In the first part of the paper, we assume that the model is trained of the full sequence 'Q-A' as opposed to only finding the question 'Q' in the training data for example without the associated answer. We will challenge this assumption later in the paper and discuss what happens if this is not the case.

Contamination (i.e., the presence of a given 'Q-A' in the training corpus), will make the generation of the response 'A' very probable after the context 'Q'. In analogy with human cognitive science (Efklides, 2006), LLM's response probability to a given query is considered to reflect the *confidence* of the model. It is therefore important to clarify here the difference and the relation between these key concepts of contamination and confidence. First, contamination is a state of the model (said to be *contaminated*) deriving from the relation between its training corpus and the testing material. Second, confidence is a feature of the model's response to a given answer (said to be *confident*). Crucially, while it is true that responses 'A' to contaminated 'Q-A' pairs will generally appear very confident,

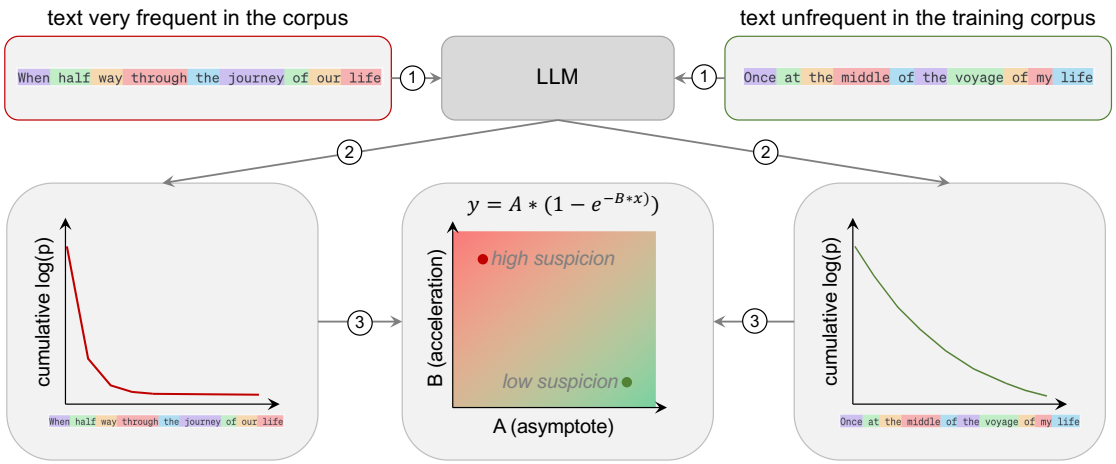

Figure 1: **LogProber algorithm.** (1) A LLM is presented with a sequence of tokens and returns the log-probabilities of each token (in the example we are using the actual *incipit* of the Dante's *Divina Commedia* (1321) and a sematically equivalent sentence. (2) We plot the cumulative log-probability along the sequence. If the sequence is known by the model, the curve should become horizontal (i.e., reach the asymptote) quickly (red), on the other hand if it is not known by the model its increase should be more gradual (green). (3) By fitting an exponential model parameterised by 2 parameters A (asymptote) and B (acceleration) on the curve we can quantify the "horizontality" of the curve and provide a measure of how much contaminated the model is on the given sequence.

higher confidence can only be achieved by other means. To understand this, consider the case where a LLM is asked the following:

"*What is the next character in the sequence 17817817817817817?*".

Many models will complete with "8" with a very high probability (i.e., *confidence*). However, it is very unlikely that this particular 'Q-A' was present in the training corpus.

2.2 A way to measure contamination in LLMs: the LogProber algorithm

Most models are accessed in a 'black box' setting, where we do not have access to the training corpus nor the model's weights, we can only infer contamination from the probabilities of the generated tokens. If the sequence 'Q-A' appears in the training data, the model is contaminated by it, thus the probability of responding 'A' from 'Q' should be high. Such probability could be estimated either by directly accessing the tokens' log(probabilities) or by running multiple queries and deriving "empirical" distribution of 'A'. Then, as proposed in (Dong et al., 2024) one could study the completion distribution and infer the presence of contamination from a peaked completion distribution.

However, this approach does not rule out that a LLM responds 'A' to a given 'Q' with high confidence because of its inferential and generalization capabilities. Indeed the 'A' tokens probabilities mix confidence and contamination behaviors making the measure unreliable to disentangle between the two. To tackle this issue, our approach proposes an alternative method that switches the focus on 'Q' instead of 'A'. Indeed while it is possible to find a peak in the generation distribution of 'A' due to a high confidence (and not because of contamination), it is more unlikely that the model is able to predict the question 'Q' confidently without having been explicitely trained on it.

Assuming we can access the probability of each consecutive token in a given 'Q' sequence, we can plot the cumulative log-probability of this sequence. This graph can be understood as the 'surprise' the model has when generating a new token after all the previous ones. Crucially, for 'Q' present in the corpus we can expect the probability of the subsequent tokens to approach p=1 (signaling that the token is predicted with high *confidence*) and therefore the log-probability will quickly achieve a plateau. To understand this point see for example what the shape of a very frequent phrase (e.g., the first line of a very famous and old poem Divine Comedy " *When half way through the journey of our life*"; Figure 1) should look like compared to a (semantically equivalent), but not as frequent, sentence ("*Once at the middle of the voyage of my life*" Figure 1; right). In the second case, successive tokens will be comparatively more surprising (or predicted with less *confidence*) (see 1).

Once obtained the log-probability curves of 'Q' sequences, their shape can be analysed in order to quantify how much they plateau to get insight into whether they were the model was 'Q' was present in the corpus or not. More specifically, we propose to fit a very simple 2 parameters function to curve:

$$f(x) = -A(1 - e^{-Bx})$$

with 2 variables A (asymptote) and B (acceleration). If the model is surprised by each new individual token in the sequence, it should result in a linear graph that will make the model slightly diverge by returning very high *A* and low *B*. Conversely, if the model predicts each consecutive token with high probability resulting curve "saturates" very quickly (low *A* and high *B*), therefore meaning that is likely that the 'Q' sequence was present in the training corpus (*contamination*). The full procedure is recapped in Alg 1 in Appendix C, illustrated in Figure 1 and further training details are provided in appendix A.

2.3 Predictions and training experiment

By returning the A and B parameters, LogProber quantifies the shape of the log-probability curve for each sentence and quantitative insights related to how much the model is familiar with the given question. These parameters can be very high or very low depending on the graph they are fitted on. As such we will consider their logarithm to reduce the variance : log(A) and log(B). This quantity is then used as a proxy for contamination on the answer (even if indeed having seen the question in the training corpus doesn't mean the model has been exposed to its answer - we will discuss this later).

To test the validity of this method and the LogProber algorithm, we designed several experiments. First, we compared the *log(A)* and *log(B)* parameters obtained by fitting 'Q' in a cognitive test that has been published decades ago (the Cognitive Reflection Test; oldCRT; (Toplak et al., 2013; Frederick, 2005a)) to those of structurally and conceptually similar 'Q' which have been designed after the creation of the model and cannot have been included in the training corpus (newCRT; (Yax et al., 2024)) (see Table 2 in appendix B). We predicted that the *log(A)* and *log(B)* parameters of the oldCRT and the newCRT should be distinguishable (e.g., lay in different regions of the *log(A)* and *log(B)* plane).

Second, we compared *log(A)* and *log(B)* parameters of the newCRT items in a "native" LLM (i.e., uncontaminated by the 'Q-A' pairs; llama 1 7B model (Touvron et al., 2023b)) with a version of the same model fine-tuned and contaminated (using the Alpaca pipeline (Taori et al., 2023)). Hyperparameters are the default from the Alpaca git repository (Taori et al., 2023) except that we add items from the new CRT questionnaires 10 times each. The questionnaire contains 7 questions and the model is trained for 3 epochs meaning the model will see each of the 7 question-answer pair 30 times during the training process. In addition, we simplified the prompting and we saved the models after each epoch (see Appendix A for more details). We predicted that the *log(A)* and *log(B)* parameters of the newCRT items should change after fine-tuning, more specifically, as the fine-tuned model becomes contaminated with 'Q-A' the *log(A)* parameter should decrease and the *log(B)* parameter should increase (QA-training).

Finally, we run two additional fine-tuning experiments, featuring hybrid fine-tuning scenarios in which, for each item of the cognitive test, the model was only fine-tuned in the question 'Q' (Q- training) or the 'A' (A- training) (see Table 1). These experiments further allowed us to explore the complex relationship between contamination and performance of a model.

3 Results

3.1 Can LogProber discriminate between newly designed and old 'Q-A' items ?

The Cognitive Reflection Text (CRT) (Frederick, 2005b) is one of most popular psychological questionnaires used to investigate human reasoning and typically associated to the demonstration of two modes of reasoning, the fast "system 2" and slow "system 1" (Kahneman, 2011). The test is widely used in the scientific literature and can be found even beyond in numerous journal articles, blogs and forums. Therefore it is very likely that LLMs trained on web data containing the CRT. To validate LogProber, we deployed it on "classical" items of the CRT (oldCRT) and to more recently developed ones (newCRT), which has been published only once and so recently so they should not be represented in the training corpus of the considered LLMs (Yax et al., 2024). The analysis shows that the A and B parameters for the 7 CRT items lied in linearly separable sectors of the

Type of training	Description	Example
STD-Training	is the original Alpaca training (Taori et al., 2023) with no data from new CRT. We only simplified the instruction prompting (see Appendix A).	Q: A bat and a ball cost £1.10 in total. The bat costs £1.00 more than the ball. How much does the ball cost? A: The ball costs £0.05.
QA-Training	involves the alpaca training with new CRT pairs of questions and answers. Thus the model is trained to predict each token of the question as well as each token of the answer given the question.	**Q: A bat and a ball cost £1.10 in total. The bat costs £1.00 more than the ball. How much does the ball cost? A: The ball costs £0.05.**
A-Training	only fits on the answer given the question. Thus the model is not optimized the predict the question's tokens but only the answer's.	Q: A bat and a ball cost £1.10 in total. The bat costs £1.00 more than the ball. How much does the ball cost? **A: The ball costs £0.05.**
Q-Training	only learns the question's tokens and therefore is only trained to predict the question's tokens and not the answer.	**Q: A bat and a ball cost £1.10 in total. The bat costs £1.00 more than the ball. How much does the ball cost?** A: The ball costs £0.05.

Table 1: **Different types of finetuning**. The example column shows the tokens the model fits on in bold.

log(A)/log(B) plane (Figure 2). More specifically log(A) was significantly higher (oldCRT: -1.04 newCRT: 1.28 t(11)=6.29, p<0.001) and log(B) was significantly lower (oldCRT: 3.38 newCRT: 0.37 t(9)=5.83, p<0.001) in the oldCRT compared to the newCRT test.

Looking more in details, LogProber returns higher scores for the first 3 oldCRT items, which are the first having being published and more widespread in the literature (Frederick, 2005a; Toplak et al., 2013). To sum up, LogProber was able to differentiate oldCRT items (present in the training corpus) from newCRT items (absent in the training corpus). These first results therefore provide first observational evidence of the effectiveness of LogProber.

3.2 Can LogProber discriminate between newly designed 'Q-A' items before and after contamination ?

In the previous analysis, we showed promising evidence that LogProber is capable of discriminating between contaminated items and some conceptually equivalent, non-contaminated, versions (*observational* experiment). In the present section, we aim to provide *experimental* support to our claim by running dedicated fine-tuning experiments. To do so, we finetuned Llama 7B using both the questions and its answers ('Q-A' tuning, see Table 2) the newCRT items, which corresponds to contaminating the model with them.

Results are reported in Figure 3 and show that the log(A) and log(B) parameter of LogProber lie in very different sector of the the plane, before and after fine-tuning.

The fine-tuning lead to high contamination log scores in the contaminated experiment (QA) as well as very high accuracies. On the other hand, in the reference experiment scores are very low and accuracy is average. This experiment validates the fact that contamination scores capture heavy model contamination if the model is contaminated in a Q-A manner. More specifically log(A) was significantly higher (STD=2.09, QA=-1.38, t(9)=5.19, p<0.001) and log(B) was significantly lower (STD=-0.43, QA=3.50, t(11)=4.45, p<0.001) in the fine-tuned compared to the native LLM test.

These results show that in a carefully controlled experiment, where we explicitly manipulated the training of a model to include the newCRT items (Q-A), LogProber was capable to detect that the model was finetuned with the newCRT Q-A items.

3.3 What is the simultaneous impact of different fine-tuning strategies on accuracy and contamination markers?

Our first finetuning experiment featured a 'Q-A' type of training, where the model was forced to

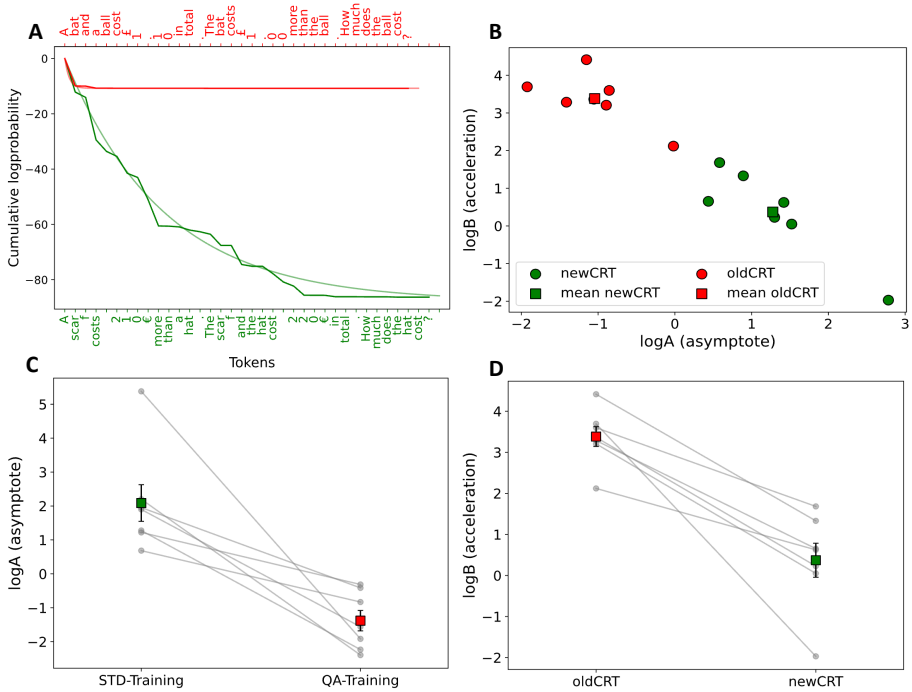

Figure 2: LogProber results for Llama 1 7B on the original and new CRT items. **A** is the log-graph showing the cumulative logprobability of the bat and the ball CRT item for the original and the new version of the questionnaire. **B** is the scatter plot of all parameters A and B from each CRT item from the original and the new questionnaire. Larger points are the average over all points either from the original or from the new version of the questionnaires. **C** and **D** are error bar plots showing the distribution of A and B in both the old and the new version of the questionnaire.

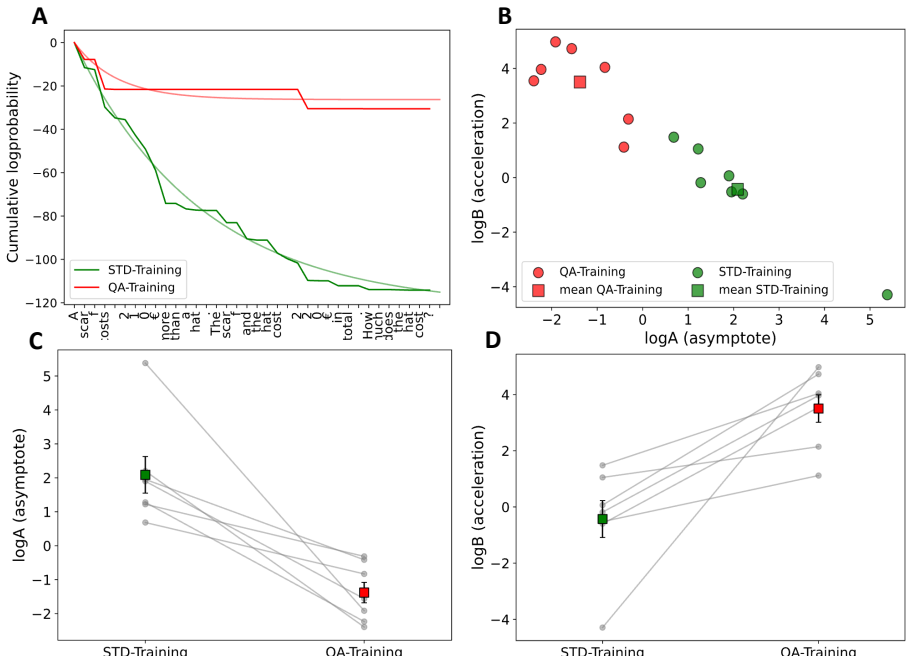

Figure 3: LogProber results for both the STD and QA trainings on the new CRT items. **A** is the log-graph showing the cumulative logprobability of the bat and the ball new CRT item for STD and QA trainings. **B** is the scatter plot of all parameters A and B from each CRT item from the STD and QA trainings. Larger points are the average over all points either from the original or from the new version of the questionnaires. **C** and **D** are error bar plots showing the distribution of A and B in both the STD and QA trainings on the new CRT questionnaire.

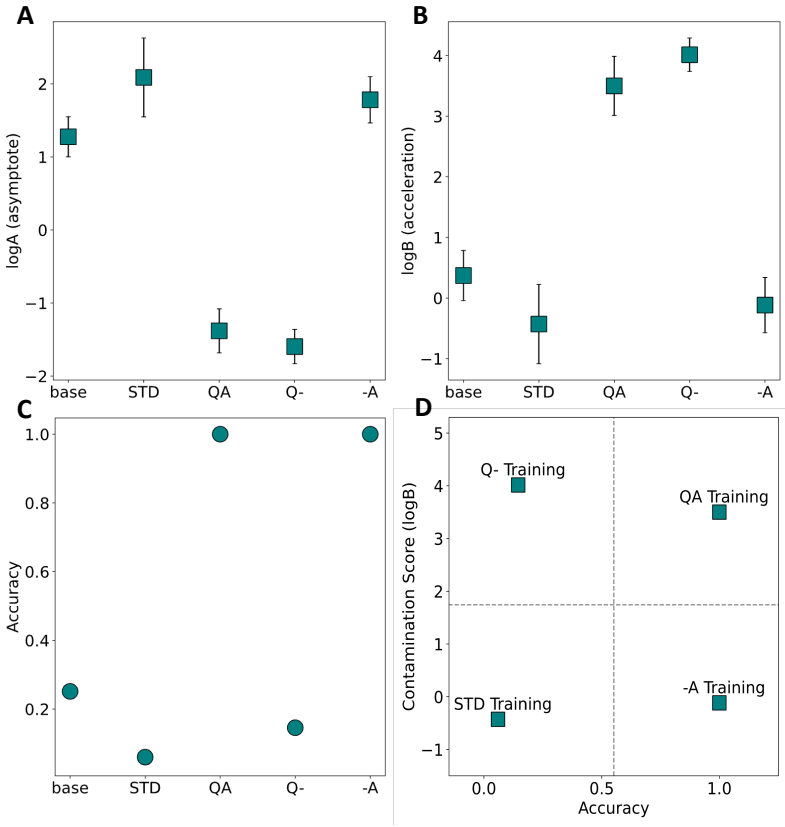

Figure 4: LogProber in newCRT for different types of finetuning. **A** shows the asymptote parameter, **B** represents the acceleration parameter and finally **C** is the accuracy of the model.

learn to predict both the question and the answer of each individual CRT item. However, this represents only one way in which a model can be contaminated (See Table 2). For instance, the model can be trained on predicting the answer, based on the question (without further training the question per se). Such '-A' training strategy is very common to fine-tune the model on instructions (Zheng et al., 2023; Taori et al., 2023). Conversely, a model could be trained only to predict the question, without necessarily including the (correct) answer in the string. Such 'Q-' training strategy style would correspond to the presence in the corpus of a given question without the corresponding answer, for example when it is quoted to explain it or in research papers.

In Figure 4 we report contamination scores for both these types of training as well as accuracy and found that, indeed, the contamination log-scores do not capture the contamination for the -A type of training as the model doesn't memorize the question. As for the Q- style of training, We found that the contamination scores are very high while the model's accuracy doesn't increase with the training (the model is contaminated on the question and not the answers).

This series of experiments show the complex relationship between markers of contamination (as derived from LogProber algorithm) and accuracy. On one side, (and somehow unsurprisingly) the accuracy is high whenever the model is trained to predict 'A' given 'Q', regardless of whether or not the 'Q' itself is contained in the training. A language model can know by heart the answer to a question without knowing the question itself and can even show signs of surprise when reading the question. On the other side, LogProber detects contamination whenever the model has been trained to predict 'Q', regardless of whether or not the 'A' was included.

4 Discussion

In this paper we propose and validate a very simple algorithm, LogProber, which is capable to spot common forms of contamination in LLMs. More specifically, LogProber can be applied to detect contamination of text in a question-answer (Q-A) format, which is commonly used both in bench-

marks used in machine learning, as well as in psychology tests (Srivastava et al., 2023; Yax et al., 2024; Coda-Forno et al., 2024).

LogProber analyses the log-probability of the question in a Q-A sequence with a low cost, compared to other approaches, as only requires a single forward pass on the question. As comparison, Dong et al. (2024) requires several generations of the answer (but doesn't require access to the log-probabilities).

We demonstrated the effectiveness of LogProber by leveraging both *observational* and *experimental* evidence. Concerning observational evidence, we demonstrated the capacity of LogProber to differentiate between the classic items of a popular cognitive experiment (the Cognitive Reflection Test; (Frederick, 2005a; Toplak et al., 2013)) and newly designed versions of the same items (Yax et al., 2024). More specifically, the contamination scores of the new items were significantly different from those of the old items, which were, in turn, much better predicted by the LLMs. Concerning experimental evidence, we ran dedicated experiments where we trained (fine-tuned) a model to predict 'Q-A' sequences of the newly designed items of the Cognitive Reflection Test. These experiments showed that LogProber was able to detect the changes in LLM training (i.e., it was capable of detecting its *contamination*).

For the sake of completeness, we ran other experiments where the additional only concerned either learning to predict only the question ('Q-' training) or only the answer ('-A' training). Critically, and somehow unsurprisingly, our results showed that LogProber was able to detect contaminations only when the question was included in the training.

This result suggests that our approach is very effective in detecting contamination at the level of pre-training, since it is likely to occur in the 'Q-A' format, and not in fine-tuning (likely to occur in the '-A' format: answer only) (Wang et al., 2022; Taori et al., 2023; Zheng et al., 2023). This limitation does not substantially reduce the utility of LogProber, since in the LLM field pretraining dataset are rarely shared or very large to search a query in them. LogProber makes it possible to account for pretraining contamination at a very cheap cost.

When training a model using a full language modeling method - a common approach for pretraining autoregressive decoder-only LLM (Wang et al., 2022) — the training data can follow three scenarios. Either the model is trained only on the question (Q) if the data does not include the answer, the model is trained on both the question and the answer (QA) if they appear consecutively in the data or neither the question nor the answer is included in the training data (STD).

In this context, the QA scenario indicates data contamination. A low contamination score implies that neither Q nor QA are likely scenarios, suggesting the STD scenario. Therefore, if a query results in a low contamination score, it is likely safe for use. Conversely, a high contamination score indicates that the model has been exposed to the question, suggesting potential contamination and that the model might be unreliable for that query.

In the open access community finetuning datasets are often shared and a lot smaller making it possible to know explicitly if the model was trained on the given query.

In short, LogProber can efficiently account for contamination acquired during pretraining (assuming the model is trained in a full language modelling - which is nearly always the case for modern LLMs) but may fail to capture finetuning contamination. This issue is mitigated by the fact that open access LLM finetuning datasets are often shared and fairly small making it possible to known explicitly if the model was trained on the given query.

Incidentally, these additional experiments also illustrate a very important, but often overlooked, fact: LLMs can be (over) trained to respond to a question, making them very familiar with answering the question without necessarily showing signs of knowing the question itself. Indeed, in our experiment '-A' training resulted in no sign of contamination, while the accuracy on the questions was extremely high. This also shows that LLM cognition is very different from human cognition especially in terms of how these two systems learn. Indeed, we can choose tokens on which LLMs learn while it is not possible to constraint the human learning mechanisms on specific words only.

In conclusion, we propose and validate a method to estimate the likelihood of contamination for string of text, based on a computationally cheap analysis of the log-probabilities of the sequence. Our approach, LogProber, is particularly suited to detect contamination that should occur at the level of pretraining and can be easily applied in a domain-general manner, as soon as the benchmarks questions are framed in the 'Q-A' manner.

5 Limitations

This work also puts into perspective inherent limitations of quantifying contamination in LLMs in a black box setting on a given short sequence. Focusing only on the answer leads to mixing confidence and contamination while using the question solve this issue but only works in specific training settings. Further research could focus on doing both at the same time in order to increase the probability to get it right but we believe these methods would still not be foolproof and might require additional information such as neuronal activations from the network to accurately classify contaminated items.

Another limitation of our approach is that it requires having access to the log-probabilities for the tokens and the question of the Q-A pair (typically used in benchmark and cognitive tasks). It could be, in principle, extended to cases where this information is not present by sampling each token in a sequence and deriving an "empirical" distribution. We note however that in that way the method would become more computationally expensive.

We would also like to emphasize on the fact that LogProber focuses on how easily the LLM can predict a question, which is common when the model is contaminated. However, LLMs might also predict questions through in-context learning. For example, take the 178178... question —while the start is hard to predict, once the pattern emerges, a LLM can predict the end, resulting in a flatter line on the graph. This can be an issue for sequences with strong patterns, but in practice, such questions still score lower in contamination than truly contaminated ones, as the graph plateaus more slowly.

Finally, LogProber presents other limitations, for instance, the algorithm returns a scores that can be difficult to interpret *per se*. To be interpreted they have to be compared with other sequences to put it in perspective and decide how likely the model is contaminated on the sequence. In our paper, we focus on an unambiguous case, where newly designed were compared to structurally equivalent old CRT, which served as an obvious reference point. Designing new items could not be easily scalable to large benchmarks. However, by comparing LogProber scores across similar benchmarks, it will nonetheless be possible to make hypotheses about the relative likelihood of contamination. Furthermore, future experiments featuring more questions and LLMs could provide in future insights concerning the mapping between the LogProber scores and the likelihood of contamination.

References

Simone Balloccu, Patrícia Schmidtová, Mateusz Lango, and Ondřej Dušek. 2024. Leak, cheat, repeat: Data contamination and evaluation malpractices in closed-source llms. *Preprint*, arXiv:2402.03927.

Marcel Binz and Eric Schulz. 2023. Using cognitive psychology to understand gpt-3. *Proceedings of the National Academy of Sciences*, 120(6).

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. 2020a. Language models are few-shot learners. *Preprint*, arXiv:2005.14165.

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. 2020b. Language models are few-shot learners. *Preprint*, arXiv:2005.14165.

Julian Coda-Forno, Marcel Binz, Jane X. Wang, and Eric Schulz. 2024. Cogbench: a large language model walks into a psychology lab. *Preprint*, arXiv:2402.18225.

Yihong Dong, Xue Jiang, Huanyu Liu, Zhi Jin, and Ge Li. 2024. Generalization or memorization: Data contamination and trustworthy evaluation for large language models. *Preprint*, arXiv:2402.15938.

Anastasia Efklides. 2006. Metacognition and affect: What can metacognitive experiences tell us about the learning process? *Educational Research Review*, 1(1):3–14.

Shane Frederick. 2005a. Cognitive reflection and decision making. *Journal of Economic Perspectives*, 19(4):25–42.

Shane Frederick. 2005b. Cognitive reflection and decision making. *Journal of Economic Perspectives*, 19:25–42.

Suriya Gunasekar, Yi Zhang, Jyoti Aneja, Caio César Teodoro Mendes, Allie Del Giorno, Sivakanth Gopi, Mojan Javaheripi, Piero Kauffmann, Gustavo

de Rosa, Olli Saarikivi, Adil Salim, Shital Shah, Harkirat Singh Behl, Xin Wang, Sébastien Bubeck, Ronen Eldan, Adam Tauman Kalai, Yin Tat Lee, and Yuanzhi Li. 2023. Textbooks are all you need. *Preprint*, arXiv:2306.11644.

Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. 2021. Measuring massive multitask language understanding. *Preprint*, arXiv:2009.03300.

Minhao Jiang, Ken Ziyu Liu, Ming Zhong, Rylan Schaeffer, Siru Ouyang, Jiawei Han, and Sanmi Koyejo. 2024. Investigating data contamination for pre-training language models. *Preprint*, arXiv:2401.06059.

Daniel Kahneman. 2011. *Thinking, fast and slow*. Farrar, Straus and Giroux, New York.

Yuanzhi Li, Sébastien Bubeck, Ronen Eldan, Allie Del Giorno, Suriya Gunasekar, and Yin Tat Lee. 2023. Textbooks are all you need ii: phi-1.5 technical report. *Preprint*, arXiv:2309.05463.

Matthieu Meeus, Shubham Jain, Marek Rei, and Yves-Alexandre de Montjoye. 2023. Did the neurons read your book? document-level membership inference for large language models. *Preprint*, arXiv:2310.15007.

Yonatan Oren, Nicole Meister, Niladri S. Chatterji, Faisal Ladhak, and Tatsunori Hashimoto. 2024. Proving test set contamination for black-box language models. In *The Twelfth International Conference on Learning Representations*.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. 2023. Exploring the limits of transfer learning with a unified text-to-text transformer. *Preprint*, arXiv:1910.10683.

Aarohi Srivastava et al. 2023. Beyond the imitation game: Quantifying and extrapolating the capabilities of language models. *Preprint*, arXiv:2206.04615.

Rohan Taori, Ishaan Gulrajani, Tianyi Zhang, Yann Dubois, Xuechen Li, Carlos Guestrin, Percy Liang, and Tatsunori B. Hashimoto. 2023. Stanford alpaca: An instruction-following llama model. https://github.com/tatsu-lab/stanford_alpaca.

Maggie Toplak, Richard West, and Keith Stanovich. 2013. Assessing miserly information processing: An expansion of the cognitive reflection test. *Thinking and Reasoning*, 20:147–168.

Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, Aurelien Rodriguez, Armand Joulin, Edouard Grave, and Guillaume Lample. 2023a. Llama: Open and efficient foundation language models. *Preprint*, arXiv:2302.13971.

Hugo Touvron et al. 2023b. Llama 2: Open foundation and fine-tuned chat models. *Preprint*, arXiv:2307.09288.

Thomas Wang, Adam Roberts, Daniel Hesslow, Teven Le Scao, Hyung Won Chung, Iz Beltagy, Julien Launay, and Colin Raffel. 2022. What language model architecture and pretraining objective work best for zero-shot generalization? *Preprint*, arXiv:2204.05832.

Nicolas Yax, Hernan Anllo, and Stefano Palminteri. 2024. Studying and improving reasoning in humans and machines. *Communications Psychology*, 2.

Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric P. Xing, Hao Zhang, Joseph E. Gonzalez, and Ion Stoica. 2023. Judging llm-as-a-judge with mt-bench and chatbot arena. *Preprint*, arXiv:2306.05685.

Kun Zhou, Yutao Zhu, Zhipeng Chen, Wentong Chen, Wayne Xin Zhao, Xu Chen, Yankai Lin, Ji-Rong Wen, and Jiawei Han. 2023. Don't make your llm an evaluation benchmark cheater. *Preprint*, arXiv:2311.01964.

A Training details

The classic Alpaca finetuning procedure is composed of 52k examples obtained from text-davinci-003 using the self-instruct pipeline (Taori et al., 2023). They are composed of 3 fields : Instruction, Input and Response prompted with the following format :

> Below is an instruction that describes a task, paired with an input that provides further context. Write a response that appropriately completes the request.
>
> ### Instruction:
> {instruction}
>
> ### Input:
> {input}
>
> ### Response:
> {response}

The model only fits on tokens from the Response field. The input field is omitted if no input is provided for the given instruction.

To contaminate the model on newCRT we simplified the prompting to :

> {instruction}
> {input}
> {response}

and included both the question and the answer in the response field so that the model fits on both of them. The point of this procedure is to make the model fit on the newCRT tokens from scratch without prompting coming before as we don't want the model to condition the generation of newCRT items to the alpaca instruction prompting scheme.

While the alpaca default training data only fit on the response (like the paper) we formatted the newCRT data in different manner for the 3 training methods. For the QA-Training we included both question and answer in the response field so that the model fits on both. For the A-Training we included the question in the instruction field and the answer in the response field. Finally for the Q-Training we only included the question in the response field without the answer so that the model only fits on the question tokens.

Hyperparameters are the default hyperparameters for Llama 7B taken from the stanford alpaca git repository with a learning rate of 2e-5 and 3 epochs.

B Old CRT and new CRT

The list of items referred to as old CRT and new CRT are borrowed from (Yax et al., 2024) where the authors used original items from (Frederick, 2005a) which language models are likely to be trained on and made new fresh ones after the training of these models (thus LLMs trained before that time cannot be contaminated on them). Items are shown in Tab 2.

oldCRT items	**newCRT items**
A bat and a ball cost £1.10 in total. The bat costs £1.00 more than the ball. How much does the ball cost?	A scarf costs 210€ more than a hat. The scarf and the hat cost 220€ in total. How much does the hat cost?
If it takes 5 machines 5 minutes to make 5 widgets, how long would it take 100 machines to make 100 widgets?	How long would it take 80 carpenters to repair 80 tables, if it takes 8 carpenters 8 hours to repair 8 tables?
In a lake, there is a patch of lily pads. Every day, the patch doubles in size. If it takes 48 days for the patch to cover the entire lake, how long would it take for the patch to cover half of the lake?	An entire forest was consumed by a wildfire in 40hours, with its size doubling every hour. How long did it take to burn 50% of the forest?
If John can drink one barrel of water in 6 days, and Mary can drink one barrel of water in 12 days, how long would it take them to drink one barrel of water together?	If Andrea can clean a house in 3 hours, and Alex can clean a house in 6 hours, how many hours would it take for them to clean a house together?
Jerry received both the 15th highest and the 15th lowest mark in the class. How many students are in the class?	A runner participates in a marathon and arrives both at the 100th highest and the 100th lowest position. How many participants are in the marathon?
A man buys a pig for £60, sells it for £70, buys it back for £80, and sells it finally for £90. How much has he made?	A woman buys a second-hand car for $1000, then sells it for $2000. Later she buys it back for $3000 and finally sells it for $4000. How much has she made?
Simon decided to invest £8,000 in the stock market one day early in 2008. Six months after he invested, on July 17, the stocks he had purchased were down 50%. Fortunately for Simon, from July 17 to October 17, the stocks he had purchased went up 75%. How much money does he have after this?	Frank decided to invest $10,000 into bitcoin in January 2018. Four months after he invested, the bitcoin he had purchased went down 50%. In the subsequent eight months, the bitcoin he had purchased went up 80%. What is the value of Frank's bitcoin after one year?

Table 2: **oldCRT and newCRT items**.

C LogProber algorithm

The pseudo code for LogProber is provided in Algorithm 1.

Algorithm 1 LogProber Algorithm

Require: $LLM, sequence$
$lp \leftarrow get_logprobs(LLM, sequence)$
$lg \leftarrow cum_sum(lp) \,/\, len(lp)$
$A, B \leftarrow fit(lg)$
return $log(A), log(B)$