Detection of Large Language Model Contamination with Tabular Data

Benoît Ronval[1(✉)], Pierre Dupont[1], and Siegfried Nijssen[1,2]

[1] UCLouvain, ICTEAM, Louvain-la-Neuve, Belgium
{benoit.ronval,pierre.dupont,siegfried.nijssen}@uclouvain.be
[2] KU Leuven, DTAI, Leuven, Belgium

Abstract. Large Language Models (LLMs) are increasingly used to solve machine learning tasks on tabular data, such as classification tasks. Using standard benchmarks, recent studies have shown impressive performance for such tasks. However, in this paper we study a major concern with the use of standard benchmarks: LLMs may have also been trained using these benchmarks, that is, these LLMs are contaminated. While previous work mostly focused on the GPT models, we propose a methodology to evaluate whether a large range of LLMs is contaminated. We propose new tests to detect contamination with tabular data over two aspects: knowledge and memorization. We also design an algorithm to parse the answers of the LLMs and detect hints of contamination. Our experiments conclude that the bigger and more closed-source an LLM is, the more likely it is contaminated.

Keywords: Contamination Detection · LLMs · Tabular Data

1 Introduction

Large Language Models (LLMs) have shown impressive performance on various benchmarks [26] but also on tasks involving data that is not part of their training corpus [8]. Assessment of this last point is usually done through *zero* and *few-shots* settings where the LLM sees no to few examples of the task to perform. In either case, only the examples, if any, are given to the model and no fine-tuning is done. LLMs can also be used with data other than text. In this work, we focus on tabular data due to their use in data analysis in many domains in research and industry [7]. Once converted into text with a *serialization* process, tabular data can be processed by LLMs for a given task. Thus, an LLM can be used as classifier without any further training, as done with TabLLM [15], which claimed to obtain comparable performance to classical machine learning models.

However, **how can we be certain the LLM did not see the data used for evaluation during its training?** Indeed, if LLMs are *contaminated* with tabular data, i.e. have seen test data during their training, their classification performance may be overestimated. This motivates the need for techniques to determine the level at which models are contaminated. However, this is not

© The Author(s), under exclusive license to Springer Nature Switzerland AG 2025
G. Krempl et al. (Eds.): IDA 2025, LNCS 15669, pp. 234–245, 2025.
https://doi.org/10.1007/978-3-031-91398-3_18

straightforward as the training sets of most LLMs are not public. Models may also fail to provide hints of contamination if prompted incorrectly or even include filters limiting the output. Contamination can occur at different levels: models may have seen parts of the data, or know only the metadata of a dataset. Moreover, if prompting is used to determine contamination, an automatic analysis of the LLMs' output is needed to scale toward large number of datasets and LLMs.

Currently, companies developing LLMs have higher computing power than most universities. As part of their business plan, these companies claim to have increasingly powerful models. In this environment, we believe universities could play an objective role, study the proposed models and detect potential problems such as data contamination. We believe this area of study will grow in importance within academic research in the coming years.

In this work, we distinguish two types of contamination: *knowledge*, referring to metadata of the tabular dataset, and *memorization*, referring to the actual tabular examples [5]. We propose prompting techniques that allow to determine different forms and levels of contamination with new algorithms to analyze the LLMs' outputs. Our principal contributions are the following:

- We argue there are different forms and levels of contamination, and propose tests to detect knowledge and memorization contamination. Here we propose a test based on random perturbation of examples. A specific focus is to apply tests to a range of LLMs that is as wide as possible.
- We propose algorithms to detect hints of contamination in the output. They are designed for generalizable, automatic, fast, and reproducible testing.
- We perform a thorough study of contamination on many LLMs with many common tabular datasets. By doing so, we report which LLMs, closed and open-source, are more contaminated.
- We study the evolution of contamination between versions of LLMs and the impact of serialization.

We start this paper with the Related Work in Sect. 2 before explaining our Proposed Approach (Sect. 3) and Setup (Sect. 4) used in the work. Section 5 presents and analyzes the contamination results. We end with discussing the possible advantages of contamination in Sect. 6. The code is available on GitHub: https://github.com/bronval/LLM_tabular_contamination_detection.

2 Related Work

The field of contamination detection inside LLMs has mainly focused on text data and closed-source models, mostly GPT, with only few studies on open-source models. The most common approach is to use prompts designed to make the LLM output hints of contamination. Given the start of the sentence from a text example, the LLM is asked to complete it [13]. The output is then compared to the actual example through the BLEURT and ROUGE-L metrics [18,19] to decide if the LLM shows hints of contamination for this dataset. The authors discuss the use of another LLM which decides how similar the produced answer

is to the actual example. A variant of this approach is to use a quiz format [12]. The prompt is phrased as a question with multiple choices corresponding, for example, to the remaining parts of the truncated example put in the prompt. The LLM then selects the proposition corresponding to the correct answer. This approach allows for simpler parsing of the output and may also avoid copyright filters from closed-source LLMs. Another prompt-based method is to use the probabilities of the output tokens and assume that examples seen during training will have higher probabilities [21]. However, since not all LLMs provide output probabilities, this method limits the comparisons between them.

Regarding tabular data, few studies have been done on contamination detection. Using serialization, researchers fell back on the aforementioned prompt approach [5]. Using the examples as text, they designed tests to detect knowledge or memorization contamination. For example, they ask the LLM to output the names of the features for a given dataset. Using few-shot classification with the studied LLMs, they show that there may be a link between contamination and high classification scores. They also indicated that the models generally pass the knowledge tests, i.e. contamination is detected, but have more difficulties regarding the memorization part. However, this research is limited to GPT-3.5 and GPT-4. Moreover, the parsing of the LLMs' outputs for the tests is based on the Levenstein distance. They also keep a single serialization throughout their work and do not study the impact of different templates on the results.

Previous works focus on closed-source LLMs, often overlooking the serialization impact. The contamination decision typically relies on text metrics or another model. To the best of our knowledge, no study compares closed and open-source LLMs with an algorithm to process the test outpus.

3 Proposed Approach

Our approach is prompt-based and relies on the answer produced by the LLM, aiming to apply to the widest range of models. After careful prompt design using in-context examples, context, or indicators for the beginning of the answer, we select the dataset and the LLM for which we want to test the contamination. We also select the serialization template used to transform the tabular examples. The tests are then launched independently from each other and our decision algorithms parse the output. For each output, they determine if there is a hint of contamination or not, i.e. if there is information in the output that could hardly have been produced without having seen the dataset before. Furthermore, we study the link between contamination and classification performance by using the LLMs as classifiers for the studied datasets following TabLLM's design [15].

3.1 Contamination Tests Descriptions

We detail here the different contamination tests. We distinguish the knowledge tests (K) from the memorization tests (M) and separate tests inspired by the literature [5] from the ones designed in this work.

Tests Inspired by the Literature

- **Feature List (K)**: Given the dataset name, the LLM must output its list of features. Additionally to the literature, we consider two setups: an uninformed case, where we only give the dataset name, and an informed case, where we add a subset of features to the prompt.
- **Feature Values (M)**: For the uninformed case, given the dataset name and a feature, the LLM must give the values for the feature. The informed case, new to our work, adds 4 serialized examples from the dataset. We select the examples such that there is always at least one remaining value to guess.
- **Incomplete Example Completion (M)**: Given 4 full examples and one with the last 3 features removed, the LLM must complete this last example. The test can be made more (resp. less) difficult by removing more (resp. less) features. The literature uses consecutive rows while we sample randomly all the examples making the test more difficult if the dataset is ordered.
- **Full Example Completion (M)**: Given the dataset name, its features, and the 4 first examples, the LLM must generate the next example. This assumes the dataset is ordered, as webpages may display data without shuffling them.
- **Feature Completion (M)**: Given the dataset name, features, 4 random examples, and a random example with a missing feature value, the LLM must complete it. Each example should have a different value for this feature.

New Tests. Our proposed tests have different advantages. Following the idea of the quiz format, we try to reduce the output of the LLM to a short answer that is easy to parse. We also consider the reverse case of classical tests; instead of asking to produce examples, we ask if the model recognizes them.

- **Dataset Recognition from Features (K)**: Given the list of features, the LLM must output the name of the corresponding dataset.
- **Dataset Recognition from Examples (M)**: Given 4 random serialized examples, the LLM must output the name of the corresponding dataset.
- **Example Membership (M)**: The task is to classify an example as original, i.e. taken directly from the dataset, or modified. A modified example is created by swapping the value of a feature with another value from the same feature obtained in another example, a classical perturbation seen in previous work [14]. The LLM must output "Yes" if the example to classify is original and "No" otherwise. To help the model, we put two examples in the prompt: one original, taken from the dataset as well, and its modified version, with their respective categories (original or modified). The test is done with 100 examples to classify, 50 original and 50 modified. This test does not have a binary contamination result: its score, similar to the accuracy, is used to evaluate to which extent the model can distinguish examples directly taken from the dataset from noisy examples.

3.2 Decision Algorithm for Contamination

Here, we explain the decision algorithm we propose. Given the LLMs' test outputs, the algorithm indicates if there is a hint of contamination or not. It distinguishes two scenarios: one for regular text output and one when the test output

Algorithm 1. Regular Text Decision Algorithm

Input: output (LLM's output), truth_words (the expected full answer of the test)
Output: True if contamination detected, False otherwise

```
unknown_words := { word ∈ truth_word | word ∉ English dictionary }
if unknown_words is empty then
    return ∀ word ∈ truth_words: word ∈ output
else
    return ∃ word ∈ unknown_words: word ∈ output
end if
```

involves examples. Note that in both cases, we limit the number of output tokens depending on the test to prevent false positive. A summary is provided in Table 1.

The first scenario is detailed in Algorithm 1 and is applied only to categorical features. It relies on an external English dictionary[1] to distinguish two cases: 1) there are words in the example that are specific to the dataset or 2) all the words in the example are actual English words (and thus included in the dictionary). In the first case, we consider a hint of contamination if at least one of these non-English words appears in the LLM's output as it is unlikely that a word not in the dictionary is accidentally generated by an uncontaminated LLM. For the second case, all the expected words have to appear to indicate the contamination. The test on which the algorithm is applied is passed, i.e. contamination is detected, if the algorithm returns True at least one time.

The second, simpler scenario checks that the value of every missing feature appears in the LLM's output. If one of these values does not appear with an exact match in the output, the test is failed. The test is passed if all missing values are in the output. Note that different features in the datasets we use have generally different values, reducing the risk of false positives.

For classification and membership tests, we do not use the decision algorithm as the LLMs are asked only to output "yes" or "no".

3.3 Serialization Details

To convert tabular data to text using serialization, we consider this template (whitespaces added for readability):

[specifier] [feature name] [value separator] [feature value] [feature separator]

The *feature name* and *feature value* are taken from the tabular example. The other elements have to be decided according to the sentence we aim to build. Here we consider 4 templates to assess the impact of this process: 1) "The [f. name] is [f. value]," [6], the default template for our tests, 2) "[f. name] = [f. value]," with an empty specifier, 3) "[f. name]: [f. value]", similar to a Python dictionary, and 4) "The [f. name] is equal to [f. value];", similar to a real sentence.

[1] https://pyenchant.github.io/pyenchant/.

Table 1. List of contamination tests and the algorithm used to assess their output.

Test Id	Test Name	Algorithm	Max. Output Tokens	Type
1	Feature list (inf.)	1	256	Knowledge
2	Feature list (uninf.)	1	256	Knowledge
3	Recognition (feat.)	1	16	Knowledge
4	Recognition (ex.)	1	16	Memorization
5	Feature values (inf.)	1	128	Memorization
6	Feature values (uninf.)	1	128	Memorization
7	Incomplete completion	2	256	Memorization
8	Full completion	2	512	Memorization
9	Feature completion	2	64	Memorization

4 Experiments Setup

4.1 LLMs

We separate the LLMs into 3 categories: the closed-source (no access to the model architecture and the training sets), the open-source without access to the training sets (we refer to them as semi-open-source), and the open-source where we have access to everything. For the first category, we access the models through the APIs. Otherwise, we use HuggingFace and consider the smallest version of the LLMs due to hardware limitations.

Closed-Source LLMs: We use 3 models from the GPT family: GPT-3.5 Turbo [8], GPT-4 [2], and GPT-4o[2]. More precisely, we use the checkpoints *gpt3-3.5-turbo-0125*, *gpt-4-0125-preview*, and *gpt-4o-2024-08-06*. We also use the Gemini-1.0 and Gemini-1.5 models [4].

Semi-open-source LLMs: From the Llama family, we use Llama 2 [24], Llama 3 [3], and Llama 3.1 [10], respectively with 7B, 8B, and 8B parameters. From the Gemma family, we use Gemma with 7B parameters [22] and Gemma 2 with 9B parameters [23]. We also use Phi-2 (2.7B parameters) [16] and Phi-3 (3.8B parameters) [1]. Finally, we consider Mistral (7B parameters) [17].

Open-Source LLMs: We use T0 [20] with 11B parameters. This LLM has been trained on different language modeling datasets, listed in the original paper. We also use GPT-J [25] with 6B parameters. It has been trained on The Pile [11], a dataset that combines 22 language modeling datasets.

4.2 Tabular Datasets

We consider common tabular datasets used in machine learning: Adult income, Bank marketing, California housing, German credit, Iris, Wine, and Titanic. We

[2] https://openai.com/index/hello-gpt-4o/.

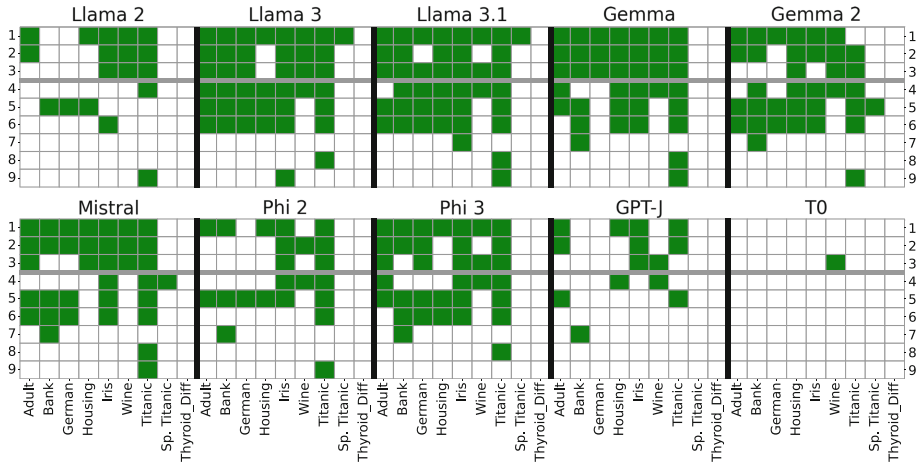

Fig. 1. Results for open-source (GPT-J, T0) and semi-open source LLMs. Green squares mark when contamination is detected. The bold, vertical black lines separate the models. The bold gray line splits the knowledge tests (above) from the memorization tests (below). (Color figure online)

also consider 2 more recent datasets: Spaceship Titanic, released just before the training cut-off of most closed-source models, and the Differentiated Thyroid Cancer Recurrence dataset, released long after the training cut-off of any LLM.

4.3 Classification Tests

We evaluate the classification accuracy of LLMs with 0 and 1-shot settings, giving one example of each class in this second case. The test is done over 100 random examples for each dataset. Following the approach of TabLLM, the prompts used are dataset specific. We ask the LLM to answer by "Yes" or "No" depending on the class of the output except in two cases: 1) for the housing dataset, we ask if the house price is higher than the median, and 2) for the iris and wine datasets, we ask the original class as they both contain 3 classes.

5 Results

In this section, we present the results of the tests described in Sect. 3. We present the contamination results and comment on the evolution between models from the same family (GPT, Llama, Gemma, Phi). Then, we discuss the impact of the serialization and display the contamination ranking based on the results.

5.1 Knowledge Tests

We observe that open-source models, GPT-J and T0 in Fig. 1, have little to no information about the datasets. On the contrary, most semi-open models (Fig. 1)

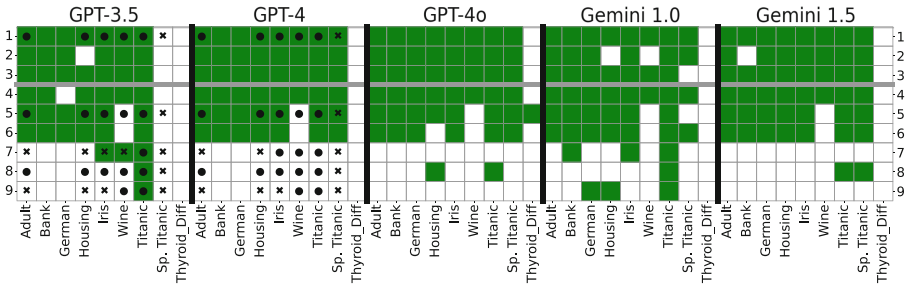

Fig. 2. Results for closed-source LLMs. Green squares mark when contamination is detected. The bold, vertical black lines separate the models. The bold gray line splits the knowledge (above) and the memorization tests (below). For GPT-3.5 and GPT-4, results from [5] are shown by a circle (contamination detected) or a cross (no detection). (Color figure online)

and all closed-source ones in Fig. 2 have extensive knowledge of the data. The informed case when the prompt contains a subset of the features tends to lead to more contamination detection. This confirms the idea that this task is not trivial and sensitive to many false negatives. Indeed, directing LLMs towards a certain dataset may result in an indication of contamination that did not appear with an uninformed prompt. The only exception is the Gemma 2 model with the Titanic dataset where we do not detect hints of contamination with the feature informed test. This shows that doing both informed and uninformed cases is useful in assessing the contamination of a model.

As expected due to their recent release dates, the Spaceship Titanic dataset is only known by the recent models (the closed-source) and the thyroid one is unknown to all models. This shows that our approach is unlikely to produce false positives and the results indicate true hints of contamination for the studied models and datasets. Compared to previous work that uses GPT models [5], we see similar results for the relevant tests and datasets. However, our experiments detect contamination in GPT-4 for the Spaceship Titanic dataset, hinting that the model contamination continues to evolve.

5.2 Memorization Tests

As it appears from the results, memorization tests are more difficult than the knowledge ones. This comes from the expected exact match to validate the completion tests as explained in Sect. 3. One exception is the Titanic dataset. We suppose these tests work well because of the *PassengerId* that has a unique value for each example and because it is a famous dataset from Kaggle[3]. Such combination of factors is difficult to have for other datasets, explaining why we detect less contamination with these completion tests. The difficulty of these tests varies with the number of features to complete and with the number of

[3] https://www.kaggle.com/competitions/titanic.

Table 2. Scores of LLMs. Mean accuracy over all datasets is reported for the membership, 0, and 1-shot tests. LLMs are ordered according to their contamination score, reported from the above figures, and scaled between 0 and 1.

	Gemini 1.0	GPT-4o	Gemini 1.5	GPT-4	GPT-3.5	Llama 3	Llama 3.1	Gemma	Mistral	Gemma 2	Phi 3	Phi 2	Llama 2	GPT-J	T0
Contamination	**0.60**	0.59	0.58	0.57	0.53	0.51	0.51	0.47	0.43	0.43	0.42	0.27	0.22	0.17	0.01
Membership	0.51	0.60	**0.70**	0.59	0.59	0.49	0.49	0.17	0.53	0.47	0.50	0.48	0.45	0.49	0.52
0-shot	0.56	0.59	0.51	**0.72**	0.64	0.47	0.49	0.44	0.34	0.15	0.41	0.02	0.23	0.03	0.58
1-shot	0.68	0.77	0.68	**0.78**	0.68	0.52	0.59	0.31	0.57	0.14	0.00	0.05	0.12	0.16	0.43

possible values for each feature. For example, the completion for the *income* feature (2 possible values) of the Adult income dataset will surely be easier than for the *education* feature (16 possible values).

The results for the membership test, shown in Table 2, are split into two parts. The open-source LLMs are close to the random baseline (0.5 accuracy) while the closed-source models perform better. This hints that these larger models can recognize easier examples that truly belong to the dataset. It also mostly matches the results of the other memorization tests where we detect more contamination for the closed-source models. We notice it is possible that LLMs sometimes output an invalid answer, i.e. other than "yes" or "no", thus obtaining a score largely below the random baseline.

As for the knowledge tests, the more an LLM is closed-source, the more it memorizes the data. Compared to previous work, the results are mostly the same [5]. However, GPT-4 fails more completion tests in our case. We recall that these tests are not performed the exact same way and that we expect an exact match while the other work is based on the Levenstein distance.

5.3 Contamination Evolution

We focus now on the differences in the contamination results between versions of models from the same family. We look at GPT-3.5, GPT-4, and GPT-4o, Gemini-1.0 and Gemini-1.5 (Fig. 2), Llama 2, Llama 3, and Llama 3.1, Gemma and Gemma 2, and Phi-2 and Phi-3 (Fig. 1). We observe that for most families, the contamination increases in the latest versions. We explain this phenomenon with three (not mutually exclusive) factors: 1) We suppose the training sets are larger for the latest versions, or at least contain new information compared to the older version; 2) Due to their increased capabilities, the newest LLMs may better model the task and thus answer more precisely; 3) The newest versions have probably been initialized with the weights of the previous model, leading to some contamination being transferred to the next version of the model.

We highlight some exceptions as the Gemini and Gemma models show less detected contamination. For the Gemini models, a look at the outputs indicates that more filters have been put in place in the model as Gemini-1.5 often claims to be unable to answer due to a lack of information while Gemini-1.0 passed the test. For Gemma models, the main difference in the results is visible in the feature recognition test. The absence of conclusive results is due to the

model generating code to answer the question. We also observe that there is no significant difference between GPT-4 and GPT-4o, and Llama 3 and Llama 3.1.

5.4 Classification Scores and Ranking

Table 2 displays the ranking of the LLMs and their classification scores. We see that the most contaminated LLMs are often the largest ones. It also appears that the more a model is closed-source, the more it is contaminated. These two observations are linked but explain the result for T0, an LLM with 11B parameters with only two tests passed.

For the classification tests, the more contaminated a model is, the more likely it is to perform well. However, other aspects can impact these results. As for the membership test, LLMs may give an invalid answer and thus obtain a score below a random baseline. Since the most contaminated models are also the biggest ones, the results may be impacted by the capabilities of the model. It also depends on the inherent difficulty of the dataset for the classification task.

5.5 Influence of the Serialization

To study the impact of serialization, we use the templates described in Sect. 3 for GPT-4, Llama 3, Mistral, and Gemma 2. We only consider the memorization tests since only they require examples in the prompt. Our experiments show that there were little to no differences in the contamination results between the serialization templates. We recommend that future researchers use the template of their choice, without losing contamination results. The general template of the prompt appears to be more important than the serialization template.

6 Discussion

In this section, we comment on the results of our tests and on the contamination itself. First, a person or company with bad intentions may avoid contamination detection by tweaking the data during the training phase [9]. However, we believe this is not the case for the LLMs studied in this work given the obtained results.

Regarding the contamination results, we must be cautious about false positives and, more importantly, false negatives. False positives can be reduced by improving the decision algorithm or handling the number of maximum outputted tokens. However, our current tests may fail to detect certain forms of contamination. We would for example need to use other prompts to change the output probabilities or do the tests a sufficient number of times with high temperature values. The risk resulting from the false negative tests is to unintentionally bias the later tests (for example, classification) and report overestimated results.

We acknowledge that the best method to assess our approach would be to train LLMs in two settings, with and without contamination. However, hardware limitations prevent this method for the models used in this paper.

An interesting point is whether the contamination of LLMs is necessarily bad. As detailed before, contamination in the context of evaluating LLMs as classifier or similar tasks has to be avoided to prevent biased results. However, contamination, seen as prior knowledge of a dataset, may be useful in other cases. If we consider data generation [6], having prior knowledge about a dataset may lead to better synthetic data or reduce the time of fine-tuning.

7 Conclusion and Future Work

In this work, we study the detection of two types of tabular data contamination, knowledge, and memorization, for various LLMs with different levels of openness. To do this, we propose new tests to detect contamination and algorithms to decide if the LLM show hints of contamination in their outputs. We then apply these tests and algorithms to many different LLMs and tabular datasets.

We show that many LLMs have knowledge about the datasets we consider and some have memorized parts of the data. We highlight that the larger and the more closed-source a model is, the more likely it is to be contaminated with tabular data. This shows the need for careful use of LLMs for tasks such as classification, where the LLMs may know the benchmark under use.

As guidelines for future works, we highly recommend performing a contamination check if the task under study may be impacted by prior knowledge of the data. Any LLM could be used with the condition that our tests do not detect contamination to ensure that the results would not be biased. A possibility is also to use an LLM that is not contaminated, such as T0 used in TabLLM [15]. One important goal of future research for contamination detection is the reduction of false negative results. As mentioned in Sect. 6, this would require either new prompt designs or entirely new approaches.

Acknowledgments. Computational resources have been provided by the supercomputing facilities of the Université catholique de Louvain (CISM/UCL) and the Consortium des Équipements de Calcul Intensif en Fédération Wallonie Bruxelles (CÉCI) funded by the Fond de la Recherche Scientifique de Belgique (F.R.S.-FNRS) under convention 2.5020.11 and by the Walloon Region.

Disclosure of Interests. The authors have no competing interests to declare that are relevant to the content of this article.

References

1. Abdin, M., et al.: Phi-3 technical report: a highly capable language model locally on your phone. arXiv preprint: arXiv:2404.14219 (2024)
2. Achiam, J., et al.: GPT-4 technical report. arXiv preprint: arXiv:2303.08774 (2023)
3. AI, M.: Introducing meta Llama 3: the most capable openly available LLM to date (2024). https://ai.meta.com/blog/meta-llama-3/
4. Anil, R., et al.: Gemini: a family of highly capable multimodal models. arXiv preprint: arXiv:2312.11805 (2023)

5. Bordt, S., Nori, H., Caruana, R.: Elephants never forget: testing language models for memorization of tabular data. arXiv preprint: arXiv:2403.06644 (2024)
6. Borisov, V., Seßler, K., Leemann, T., Pawelczyk, M., Kasneci, G.: Language models are realistic tabular data generators. arXiv preprint: arXiv:2210.06280 (2022)
7. van Breugel, B., van der Schaar, M.: Why tabular foundation models should be a research priority. arXiv preprint: arXiv:2405.01147 (2024)
8. Brown, T., et al.: Language models are few-shot learners. In: Advances in Neural Information Processing Systems, vol. 33, pp. 1877–1901 (2020)
9. Dekoninck, J., Müller, M.N., Baader, M., Fischer, M., Vechev, M.: Evading data contamination detection for language models is (too) easy. arXiv preprint: arXiv:2402.02823 (2024)
10. Dubey, A., et al.: The Llama 3 herd of models. arXiv preprint: arXiv:2407.21783 (2024)
11. Gao, L., et al.: The Pile: An 800gb dataset of diverse text for language modeling. arXiv preprint: arXiv:2101.00027 (2020)
12. Golchin, S., Surdeanu, M.: Data contamination quiz: a tool to detect and estimate contamination in large language models. arXiv preprint: arXiv:2311.06233 (2023)
13. Golchin, S., Surdeanu, M.: Time travel in Llms: tracing data contamination in large language models. arXiv preprint: arXiv:2308.08493 (2023)
14. Hanhijärvi, S., Ojala, M., Vuokko, N., Puolamäki, K., Tatti, N., Mannila, H.: Tell me something i don't know: randomization strategies for iterative data mining. In: Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 379–388 (2009)
15. Hegselmann, S., Buendia, A., Lang, H., Agrawal, M., Jiang, X., Sontag, D.: TabLLM: few-shot classification of tabular data with large language models. In: International Conference on Artificial Intelligence and Statistics, pp. 5549–5581. PMLR (2023)
16. Javaheripi, M., et al.: Phi-2: the surprising power of small language models. Microsoft Research Blog (2023)
17. Jiang, A.Q., et al.: Mistral 7B. arXiv preprint: arXiv:2310.06825 (2023)
18. Lin, C.Y.: Rouge: A package for automatic evaluation of summaries. In: Text Summarization Branches Out, pp. 74–81 (2004)
19. Papineni, K., Roukos, S., Ward, T., Zhu, W.J.: BLEU: a method for automatic evaluation of machine translation. In: Proceedings of the 40th annual meeting of the Association for Computational Linguistics, pp. 311–318 (2002)
20. Sanh, V., et al.: Multitask prompted training enables zero-shot task generalization. arXiv preprint: arXiv:2110.08207 (2021)
21. Shi, W., et al.: Detecting pretraining data from large language models. arXiv preprint: arXiv:2310.16789 (2023)
22. Team, G., et al.: Gemma: open models based on Gemini research and technology. arXiv preprint: arXiv:2403.08295 (2024)
23. Team, G., et al.: Gemma 2: improving open language models at a practical size. arXiv preprint: arXiv:2408.00118 (2024)
24. Touvron, H., et al.: Llama 2: open foundation and fine-tuned chat models. arXiv preprint: arXiv:2307.09288 (2023)
25. Wang, B., Komatsuzaki, A.: GPT-J-6B: A 6 Billion Parameter Autoregressive Language Model (2021). https://github.com/kingoflolz/mesh-transformer-jax
26. Zhao, W.X., et al.: A survey of large language models. arXiv preprint: arXiv:2303.18223 (2023)