The Heap: A Contamination-Free Multilingual Code Dataset for Evaluating Large Language Models

Jonathan Katzy
Delft University of Technology
Delft, The Netherlands
0009-0005-9574-2414

Razvan Mihai Popescu
Delft University of Technology
Delft, The Netherlands
0009-0003-6251-770X

Arie van Deursen
Delft University of Technology
Delft, The Netherlands
0000-0003-4850-3312

Maliheh Izadi
Delft University of Technology
Delft, The Netherlands
0000-0001-5093-5523

***Abstract*—The recent rise in the popularity of large language models has spurred the development of extensive code datasets needed to train them. This has left limited code available for collection and use in the downstream investigation of specific behaviors, or evaluation of large language models without suffering from data contamination. To address this problem, we release *The Heap*, a large multilingual dataset covering 57 programming languages that has been deduplicated with respect to other open datasets of code, enabling researchers to conduct fair evaluations of large language models without significant data cleaning overhead.**

***Index Terms*—Dataset, Evaluation, Large Language Models, Open Science, Data Contamination, Multilingual**

I. Introduction

The data-intensive training process of Large Language Models (LLMs) has driven the release of numerous large-scale datasets, particularly for code, to facilitate the development of new models. This rapid increase in the amount of training data used to pre-train LLMs has resulted in extensive datasets covering almost all publicly available code [1]–[3].

To assess the success of such LLMs in downstream tasks, *fresh* data not seen during training is needed. Otherwise such evaluations are *contaminated*, possibly resulting in overly optimistic results. Unfortunately, obtaining such non-contaminated data is increasingly difficult. In fact, a recent study establishes that only 10% of investigations involving LLMs deduplicate their data with respect to the training data in order to avoid contamination [4].

To address this, we propose *The Heap*, a dataset of not previously used code that can be used for contamination-free multilingual evaluation of LLMs in downstream tasks. We address contamination in two ways. First, we select code with a *non-permissive* license, such as the GNU General Public License. Using such code for *training* is unattractive, as it may require the end user to publicly release *all* code in their code bases. Second, we pre-conduct computationally expensive near and exact *deduplication*, removing code that is used in other datasets widely used for training such as The Stack [1].

II. Collection

Using the search API, we collect our dataset from GitHub, a commonly used online platform for sharing code repositories. This collection process mimics the data distribution of other large-scale datasets [3], [5]–[8], minimizing the probability of including confounding factors in the dataset, such as drifts in the representations of data [9].

A. Programming Languages

We aim to compile a representative dataset that encompasses a wide range of programming languages. To achieve this, we select languages based on several criteria. Our selection includes languages with diverse syntactic structures, such as LISP, C, Python, Haskell, and Assembly. We also select different programming paradigms, such as COBOL, Pascal, and C for procedural languages, Java, C#, Python, for object-oriented languages, and Haskell and Clojure for functional languages. To cover more specific use cases, we also include domain-specific languages such as Mathematica, Emacs-Lisp, and Coq. A complete list of all languages included in the dataset is presented in Table I.

B. Query

We focus on repositories that have one of the targeted languages as the main language of the repository. We further select only repositories that are licensed under non-permissive licenses. We choose non-permissive licenses as an initial filter for repositories, as many large-scale datasets focus on exclusively unlicensed or permissively licensed code [2], [3], [5]. The reasons for the exclusion of non-permissively licensed code in other datasets come from potential licensing issues that may be related to the output of models trained on non-permissively licensed data [10]. *The Heap* is not intended for pre-training models that are aimed at end users, but rather for exclusive use in a research setting. The inclusion of exclusively non-permissively licensed code has the added benefit that it acts as a deterrent for developers to train LLMs on *The Heap*, ensuring it remains a relevant source of data for downstream tasks. We provide an overview of the licenses used in this work in Table II.

C. Scraping

For each programming language, we scrape up to 50,000 repositories or as many as are available. Our dataset contains code from repositories created between January 2008 and August 2024. For each selected language, we extract repositories sorted by star count in descending order; this has been used as a loose quality metric before [11]. To

TABLE I
LANGUAGES INCLUDED IN THE DATASET

Language	Repositories	Raw Files	Unique Files
Ada	676	41,367	35,425
Agda	142	5,483	5,113
ANTLR	101	564	541
Apex	253	17,833	7,641
Assembly	7,100	208,896	104,901
C	50,000	16,585,280	4,960,192
C#	50,000	5,906,716	3,770,829
C++	50,000	14,891,856	4,811,620
Clojure	27,107	380,567	273,181
Cobol	341	2,242	1,208
Common Lisp	796	45,083	16,968
Coq	477	54,137	26,175
Crystal	368	11,606	7,300
Cuda	1,191	26,948	13,359
D	1,185	185,630	126,111
Dart	11,907	484,935	413,203
EJS	1,475	15,513	12,884
Elixir	2,371	643,856	127,910
Emacs Lisp	377	8,260	7,963
Erlang	1,240	55,932	32,049
F#	876	22,152	16,015
Forth	222	28,287	7,932
Go	50,000	8,506,379	2,355,716
Groovy	2,198	60,299	48,353
Hack	1,379	84,916	37,405
Haskell	8,023	122,788	111,234
Java	50,000	6,989,601	5,197,338
JavaScript	50,000	8,289,901	3,393,747
Julia	2,859	46,284	38,381
Kotlin	21,665	1,467,343	1,045,396
Less	433	17,276	7,389
Lua	42,241	4,605,230	913,898
Mathematica	1,528	164,498	89,853
MATLAB	20,828	1,051,354	665,659
NetLogo	332	900	863
NewLisp	35	5,819	5,148
Nix	1,892	75,093	71,199
Objective-C	7,700	1,899,714	698,137
OCaml	1,961	121,890	69,171
Pascal	5,218	330,832	225,749
Perl	14,673	1,798,520	629,769
PHP	50,000	12,707,727	3,363,040
Processing	2,950	24,723	20,343
Prolog	1,071	38,995	20,279
Python	50,000	2,290,182	1,792,451
R	44,993	589,139	374,812
Raku	158	1,384	1,306
Ruby	13,378	1,579,655	794,364
Rust	42,847	2,496,177	844,258
Scala	5,893	749,370	224,021
Scheme	1,878	106,620	54,226
Scilab	199	4,531	4,084
SQL	130	47,185	41,178
Starlark	146	524	498
Swift	13,924	633,819	439,565
Vue	14,858	457,605	323,672
WebAssembly	68	834	587
Total	733,663	96,990,250	38,681,609

TABLE II
COPYLEFT LICENSES INCLUDED IN THE DATASET.

License	Family	Description[1]
CECILL-1.0 CECILL-1.1 CECILL-2.0 CECILL-2.1 CECILL-C EPL-1.0 EPL-2.0 LGPL-2.1 LGPL-3.0 MS-RL MPL-2.0	Weak Copyleft	Share changes and additions to the licensed software when redistributing.
GPL-2.0 GPL-3.0	Strong Copyleft	Share larger programs built with the licensed software when redistributing. This extends weak copyleft requirements.
AGPL-3.0 EUPL-1.1 EUPL-1.2 OSL-3.0	Network Copyleft	Share larger programs built with the licensed software when redistributing or running it over a network. This extends strong copyleft requirements.

maximize extraction efficiency and avoid GitHub's rate limits, we employ pagination and repository creation date filtering. When the number of repositories within a specified time frame exceeds the rate limit, we narrow the time interval and apply a tumbling window approach to ensure comprehensive coverage. We guide the file extraction based on a list of file extensions from The Stack [5].

D. Cleaning

After collecting the data from online sources, we perform some cleaning steps. First, we exclude files containing fewer than 10 words or exceeding 10 MB in size. We also remove exact duplicates from our own dataset. We use the same approach as the exact deduplication with respect to other datasets described in Section III-A.

III. DEDUPLICATION

An important aspect of fairly evaluating downstream tasks is preventing data leakage [4]. This is often done through a deduplication process. Although there should be no overlap between our non-permissively licensed dataset and permissively licensed datasets due to our selection procedure, it does not completely prevent overlap [10].

Our deduplication strategy consists of exact deduplication and near deduplication. Before each deduplication strategy, we remove all comments (using a regex, based on the programming language) and whitespace from each file. This ensures that small changes to files, such as the removal of a license comment or changes in whitespace characters, still result in the detection of an exact duplicate. The final files included in *The Heap* are the unaltered versions scraped from GitHub.

[1]https://blueoakcouncil.org/copyleft

```
{
    id: 200,
    file_name: "kernel.lisp",
    file_path: "whily_yalo/cc/kernel.lisp",
    content: "REPL: loop (jmp short read-start) ;;
        ...",
    size: 4,099,
    language: "Common Lisp",
    extension: ".lisp",
    total_lines: 125,
    avg_line_length: 27.52,
    max_line_length: 104,
    alphanum_fraction: 0.59,
    repo_name: "whily/yalo",
    repo_stars: 571,
    repo_forks: 32,
    repo_open_issues:1,
    repo_license: "GPL-2.0",
    repo_extraction_date: "9/19/2024, 11:24:32 AM",
    exact_duplicates_stackv1: False,
    exact_duplicates_stackv2: True,
    near_duplicates_stackv1: False,
    near_duplicates_stackv2: True,
    ...
  }
```


Fig. 1. Example of final dataset structure for one entry

a) Exact Deduplication: For exact deduplication, we calculate the SHA-256 hash of each file to identify exact duplicates between *The Heap* and publicly available datasets. We selected this hash function for its low collision probability, which reduces the risk of false positives.

b) Near Deduplication: We also perform near-deduplication between our scraped dataset and the publicly available ones. To achieve this, we utilize the MinHash Locality-Sensitive Hashing (LSH) approach, implemented using the `datasketch`[2] library. We apply the same SHA-256 hashing function as before, with 128 permutations and a precision-recall weight distribution of $40\% - 60\%$. These design choices help mitigate hash collisions while maintaining a balanced trade-off, hence favoring higher recall at the expense of a controlled increase in false positives (removing files that were not duplicates).

We use a shingle size of 7 characters, as code files typically use a smaller set of characters compared to large research articles, where $k = 9$ [12]. This reduces the likelihood of overly common shingles, which could otherwise inflate similarity scores, as would occur with smaller values of k. Files with a Jaccard similarity above 0.7 are flagged as near duplicates, a threshold shown to be effective for duplicate detection [13].

We identify and flag duplicates between our dataset and all publicly available datasets to facilitate a more flexible approach to LLM evaluation, prioritizing both reproducibility and ease of use. This setup minimizes time and computational overhead by removing the burden of duplicate detection from researchers. Users can seamlessly filter data by language or by exact and near-duplicate files, tailoring the dataset to their specific requirements. Table I provides a comprehensive summary of the languages extracted. The third column lists the number of files collected after filtering based on file size and word count. The last column indicates the number of files obtained after removing exact duplicates within our dataset, with exact and near duplicates from other datasets flagged among the remaining files. For more detailed information on the dataset creation process, please refer to the dataset page[3].

[2]https://ekzhu.com/datasketch/lsh.html

TABLE III
LIST OF PUBLICLY-AVAILABLE DATASETS USED FOR DEDUPLICATION

Dataset	Source
The Stack V2 [3]	All permissively licensed and unlicensed files collected in the Software Heritage [14] archive.
The Stack [1]	All permissively licensed repositories collected in the GHArchive [15] and scraped from GitHub.
Red Pajama [2]	Repositories from the GitHub dataset hosted by Google BigQuery [16] licensed under MIT, BSD, or Apache licenses.
GitHub Code [8]	Repositories from the GitHub dataset hosted by Google BigQuery [16].
CodeParrot [7]	All Python files from the GitHub dataset hosted by Google BigQuery [16].

A. Datasets

Our selection of datasets for deduplication is based on previously curated lists [10], with the addition of The Stack V2 [3], which is the only new dataset that has been released since the publication of previous works. We give an overview of all potential datasets in Table III. Due to the comment removal being based on the programming languages of the files, we are not able to infer the correct language for two datasets. The Pile [6], which has been removed and re-uploaded, has lost information about the programming language of a file. Furthermore, due to a known issue with the curation of CodeClippy[4], the languages and names of files are misaligned in the dataset. We also exclude this dataset from deduplication. Although we could predict the languages used in the files in these datasets, the tools that provide this functionality do return incorrect predictions, which could result in a duplicate not being removed. As we aim to provide a guarantee that there is no data contamination in our dataset, we remove these two datasets from consideration.

IV. LAYOUT

The Heap is organized into multiple subsets, each of them corresponding to one programming language. In each subset, the entries included in the dataset can be summarized into 3 groups: file content and metadata, quality indicators, and duplicates. We give an example of one entry in Figure 1.

[3]https://huggingface.co/datasets/WizzF/Heap-Forge
[4]https://github.com/CodedotAI/gpt-code-clippy/issues/71

a) File Content and Metadata: For the file content and metadata, we list the actual content of the file, which is the main information to be used in downstream tasks. We also include information about filename and path, as this has been included in the pre-training procedure of some LLMs [3], [11], [17].

b) Quality Indicators: To facilitate the selection of files for downstream use, we incorporated several quality indicators previously utilized in related works, ensuring the dataset can be easily filtered and selected. We included numerical statistics about the file such as the *total_lines*, *avg_line_length*, *max_line_length* and *alphanumeric fraction*, as well as repository-wide statistics such as *repo stars*, *repo forks*, *open issues* and the *extraction date of the repo*. The repository star count will be artificially inflated for languages where more than $50,000$ repositories exist, due to the ordering of the repositories in the collection steps.

c) Duplicates: As we deduplicate *The Heap* with respect to a number of other publicly available datasets, we incorporate two columns for every dataset. One column contains a Boolean value, whether there is an exact duplicate of the given file in the dataset, and the other column contains a Boolean value describing whether there is a near duplicate of the given file in the dataset. We choose not to remove files but to use a Boolean mask in order to maximize the amount of available data for each available dataset.

V. Future Improvements

In future iterations of this dataset, several potential improvements could be made. These include enhancing the deduplication process, releases of new training datasets, providing detailed information about the natural languages represented in the dataset, and tracking the evolution of codebases.

a) New Datasets: The main goal of this dataset is to reduce the burden of deduplicating a dataset used for downstream tasks for future research. This is only effective if the dataset is deduplicated against all available datasets. As new datasets are released we intend to pass them through the same pipeline to ensure *The Heap* remains relevant for the future.

b) Deduplication: We addressed the deduplication of datasets using two widely adopted methods: exact deduplication based on hashing and near-deduplication leveraging locality-sensitive hashing. However, there is limited research on what constitutes an effective deduplication strategy. There could be issues with duplicates at a lower granularity level than file-based deduplications, as well as possible issues with the provenance of code fragments. Once studies are conducted on the impact of various deduplication approaches, we plan to incorporate these strategies as a new entry in the dataset.

c) Cleaning: We include all files that we scraped that were not duplicates, while this gives us a dataset of deduplicated files, there is still the question of file "quality". In NLP research, keywords have been used for filtering websites, such as `lorem ipsum` or `TODOs` [18], and code datasets have been cleaned of autogenerated files using a similar approach [3]. We believe that this may also affect the quality of code datasets. Specifically, languages that rely heavily on boiler plating, such as Java, may benefit from removing certain common phrases from their corpus. This will be included as a further filtering step in a future release of the dataset.

d) Topic Modeling: While languages can be used to loosely select an area that is being analyzed (Mathematica for mathematics, or JavaScript for web-based projects), many languages can be used in multiple specializations/areas. Adopting the FineWeb topic modeling approach for code datasets would create interesting annotations for the code files, as well as show any form of topic-based imbalances in the dataset.

e) Natural Language: An under-explored research area involves the presence of multiple natural languages within code. As natural languages are often mixed within one file [19], we plan to adopt a Parts of Speech-like tagging [20] system for the natural languages present in each file. This can give information about the performance of code LLMs when the code is not in English. This will both help the development of non-English code LLMs, as well as aid English-focused LLMs, as they can be evaluated on only English.

VI. Limitations/Challenges

The limitations and challenges faced by this dataset are two-fold. First, other actors may decide to train their models on this data, removing the benefits, and second, developers may object to their code being present in this dataset. We address these problems as follows.

a) Training: In order to use *The Heap* for a fair evaluation of an LLM, the researcher must be sure that the target LLM has not been trained on *The Heap*. Aside from our deduplication ensuring this fact for current existing LLMs, our collection process also adds a layer of protection from the inclusion of *The Heap* in the training procedure. The trend of training LLMs has shifted to only training on permissively licensed data, which would exclude *The Heap*. Furthermore, the restriction of *The Heap* to research only, alleviates the problems with author attribution in LLM generations as trained models are not intended to be used by end-users [10], [21].

Furthermore, existing works such as membership inference attacks, have been extended to the scale of entire datasets [22]. This should make it possible in the near future to retroactively test for the inclusion of *The Heap* in the training procedures of a model.

b) Ethics: With the rapid rise of public repositories being used to train code language models, many authors of older repositories were unaware that their code could be utilized for such purposes, leaving them unable to opt-out. Moreover, there is currently no consensus on how developers can opt in or out of having their code included in datasets. We acknowledge these ethical concerns regarding the use of code in deep learning practices and offer the ability for repository owners to opt out of having their code included in our dataset. Although this approach is not ideal, as it places the burden of exclusion on the authors, it aligns with the current best practices [3].

VII. Conclusion

We present *The Heap*, a multilingual dataset of source code that we deduplicated against datasets commonly used in the (pre-)training of large language models. *The Heap* enables researchers to conduct investigations into the behavior and performance of code large language models without the need to perform extensive deduplication with other datasets. This addresses the shortcomings of LLM investigations not testing for data leakage in 90% of all investigations [4] allowing for more robust conclusions to be made.

We release the dataset (only for research purposes) and outline a road map for future features such as natural language annotation, topic annotations, and further cleaning procedures to be incorporated into the dataset, to make higher-quality evaluations easier and more available for all researchers.

References

[1] Denis Kocetkov, Raymond Li, Loubna Ben Allal, Jia Li, Chenghao Mou, Carlos Muñoz Ferrandis, Yacine Jernite, Margaret Mitchell, Sean Hughes, Thomas Wolf, Dzmitry Bahdanau, Leandro von Werra, and Harm de Vries. The stack: 3 tb of permissively licensed source code, 2022.

[2] Together Computer. Redpajama: An open source recipe to reproduce llama training dataset, 2023.

[3] Anton Lozhkov, Raymond Li, Loubna Ben Allal, Federico Cassano, Joel Lamy-Poirier, Nouamane Tazi, Ao Tang, Dmytro Pykhtar, Jiawei Liu, Yuxiang Wei, et al. Starcoder 2 and the stack v2: The next generation. *arXiv preprint arXiv:2402.19173*, 2024.

[4] Antonio Vitale, Rocco Oliveto, and Simone Scalabrino. A catalog of data smells for coding tasks. *ACM Trans. Softw. Eng. Methodol.*, December 2024. Just Accepted.

[5] Denis Kocetkov, Raymond Li, Loubna Ben allal, Jia LI, Chenghao Mou, Yacine Jernite, Margaret Mitchell, Carlos Muñoz Ferrandis, Sean Hughes, Thomas Wolf, Dzmitry Bahdanau, Leandro Von Werra, and Harm de Vries. The stack: 3 TB of permissively licensed source code. *Transactions on Machine Learning Research*, 2023.

[6] Leo Gao, Stella Biderman, Sid Black, Laurence Golding, Travis Hoppe, Charles Foster, Jason Phang, Horace He, Anish Thite, Noa Nabeshima, et al. The pile: An 800gb dataset of diverse text for language modeling. *arXiv preprint arXiv:2101.00027*, 2020.

[7] Thomas Wolf, Leandro von Werra, and Lewis Tunstall. Codeparrot dataset. *https://huggingface.co/datasets/transformersbook/codeparrot*.

[8] github-code. *https://huggingface.co/datasets/codeparrot/github-code*.

[9] Tolga Bolukbasi, Adam Pearce, Ann Yuan, Andy Coenen, Emily Reif, Fernanda Viégas, and Martin Wattenberg. An interpretability illusion for bert. *arXiv preprint arXiv:2104.07143*, 2021.

[10] Jonathan Katzy, Razvan Popescu, Arie Van Deursen, and Maliheh Izadi. An exploratory investigation into code license infringements in large language model training datasets. In *Proceedings of the 2024 IEEE/ACM First International Conference on AI Foundation Models and Software Engineering*, FORGE '24, page 74–85, New York, NY, USA, 2024. Association for Computing Machinery.

[11] Raymond Li, Loubna Ben allal, Yangtian Zi, Niklas Muennighoff, Denis Kocetkov, Chenghao Mou, Marc Marone, Christopher Akiki, Jia LI, Jenny Chim, Qian Liu, Evgenii Zheltonozhskii, Terry Yue Zhuo, Thomas Wang, Olivier Dehaene, Joel Lamy-Poirier, Joao Monteiro, Nicolas Gontier, Ming-Ho Yee, Logesh Kumar Umapathi, Jian Zhu, Ben Lipkin, Muhtasham Oblokulov, Zhiruo Wang, Rudra Murthy, Jason T Stillerman, Siva Sankalp Patel, Dmitry Abulkhanov, Marco Zocca, Manan Dey, Zhihan Zhang, Urvashi Bhattacharyya, Wenhao Yu, Sasha Luccioni, Paulo Villegas, Fedor Zhdanov, Tony Lee, Nadav Timor, Jennifer Ding, Claire S Schlesinger, Hailey Schoelkopf, Jan Ebert, Tri Dao, Mayank Mishra, Alex Gu, Carolyn Jane Anderson, Brendan Dolan-Gavitt, Danish Contractor, Siva Reddy, Daniel Fried, Dzmitry Bahdanau, Yacine Jernite, Carlos Muñoz Ferrandis, Sean Hughes, Thomas Wolf, Arjun Guha, Leandro Von Werra, and Harm de Vries. Starcoder: may the source be with you! *Transactions on Machine Learning Research*, 2023. Reproducibility Certification.

[12] Jure Leskovec, Anand Rajaraman, and Jeffrey David Ullman. *Mining of Massive Datasets*. Cambridge University Press, USA, 2nd edition, 2014.

[13] Miltiadis Allamanis. The adverse effects of code duplication in machine learning models of code. In *Proceedings of the 2019 ACM SIGPLAN International Symposium on New Ideas, New Paradigms, and Reflections on Programming and Software*, Onward! 2019, page 143–153, New York, NY, USA, 2019. Association for Computing Machinery.

[14] software heritage. *https://docs.softwareheritage.org/index.html*.

[15] gharchive. *https://www.gharchive.org/*.

[16] google bigquery. *https://cloud.google.com/bigquery/public-data*.

[17] CodeGemma Team, Heri Zhao, Jeffrey Hui, Joshua Howland, Nam Nguyen, Siqi Zuo, Andrea Hu, Christopher A Choquette-Choo, Jingyue Shen, Joe Kelley, et al. Codegemma: Open code models based on gemma. *arXiv preprint arXiv:2406.11409*, 2024.

[18] Jesse Dodge, Maarten Sap, Ana Marasović, William Agnew, Gabriel Ilharco, Dirk Groeneveld, Margaret Mitchell, and Matt Gardner. Documenting large webtext corpora: A case study on the colossal clean crawled corpus. *arXiv preprint arXiv:2104.08758*, 2021.

[19] Timo Pawelka and Elmar Juergens. Is this code written in english? a study of the natural language of comments and identifiers in practice. In *2015 IEEE International Conference on Software Maintenance and Evolution (ICSME)*, pages 401–410. IEEE, 2015.

[20] Alebachew Chiche and Betselot Yitagesu. Part of speech tagging: a systematic review of deep learning and machine learning approaches. *Journal of Big Data*, 9(1):10, 2022.

[21] Shayne Longpre, Robert Mahari, Anthony Chen, Naana Obeng-Marnu, Damien Sileo, William Brannon, Niklas Muennighoff, Nathan Khazam, Jad Kabbara, Kartik Perisetla, et al. The data provenance initiative: A large scale audit of dataset licensing & attribution in ai. *arXiv preprint arXiv:2310.16787*, 2023.

[22] Pratyush Maini, Hengrui Jia, Nicolas Papernot, and Adam Dziedzic. Llm dataset inference: Did you train on my dataset? *arXiv preprint arXiv:2406.06443*, 2024.