GDC Cohort Copilot: An AI Copilot for Curating Cohorts from the Genomic Data Commons

Steven Song [*, 1, 2, 3]   Anirudh Subramanyam [*, 1]   Zhenyu Zhang [1]

Aarti Venkat [1, 4]   Robert L. Grossman [†, 1, 2, 4]

Abstract

Motivation: The Genomic Data Commons (GDC) provides access to high quality, harmonized cancer genomics data through a unified curation and analysis platform centered around patient cohorts. While GDC users can interactively create complex cohorts through the graphical Cohort Builder, users (especially new ones) may struggle to find specific cohort descriptors across hundreds of possible fields and properties. However, users may be better able to describe their desired cohort in free-text natural language.

Results: We introduce GDC Cohort Copilot, an open-source copilot tool for curating cohorts from the GDC. GDC Cohort Copilot automatically generates the GDC cohort filter corresponding to a user-input natural language description of their desired cohort, before exporting the cohort back to the GDC for further analysis. An interactive user interface allows users to further refine the generated cohort. We develop and evaluate multiple large language models (LLMs) for GDC Cohort Copilot and demonstrate that our locally-served, open-source GDC Cohort LLM achieves better results than GPT-4o prompting in generating GDC cohorts.

Availability and implementation: We implement and share GDC Cohort Copilot as a container-ized Gradio app on HuggingFace Spaces, available at `https://huggingface.co/spaces/uc-ctds/GDC-Cohort-Copilot`. GDC Cohort LLM weights are available at `https://huggingface.co/uc-ctds`. All source code is available at `https://github.com/uc-cdis/gdc-cohort-copilot`.

1 Introduction

The National Cancer Institute's (NCI) Genomic Data Commons (GDC) is a highly used resource for cancer research. With over 100,000 unique monthly users, the GDC provides access to high quality, harmonized, multimodal cancer data for over 45,000 patient cases [1]. A typical user workflow using the GDC is to curate a cohort of cases before doing subsequent analysis, either using tools within the GDC Data Portal or through the GDC API [2]. Central to this workflow is the set of filters used to construct the cohort.

The GDC provides the Cohort Builder tool to allow users to interactively select their desired filters. The Cohort Builder is a powerful tool which allows users to select specific values from over 700 filter properties. While the Cohort Builder organizes commonly used filters into user-readable groupings, there are still dozens of properties, each with potentially a hundred or more possible values to filter by. This balance of allowing users to create specific and verbose filters while providing a user-friendly interface is complex. New users of the GDC may especially find it difficult to identify the filters most relevant for their research. However, such users may naturally be able to describe their desired cohort in natural language.

---

[1] Center for Translational Data Science, University of Chicago, Chicago, IL
[2] Department of Computer Science, University of Chicago, Chicago, IL
[3] Medical Scientist Training Program, Pritzker School of Medicine, University of Chicago, Chicago, IL
[4] Section of Biomedical Data Science, Department of Medicine, University of Chicago, Chicago, IL
[*] These authors contributed equally
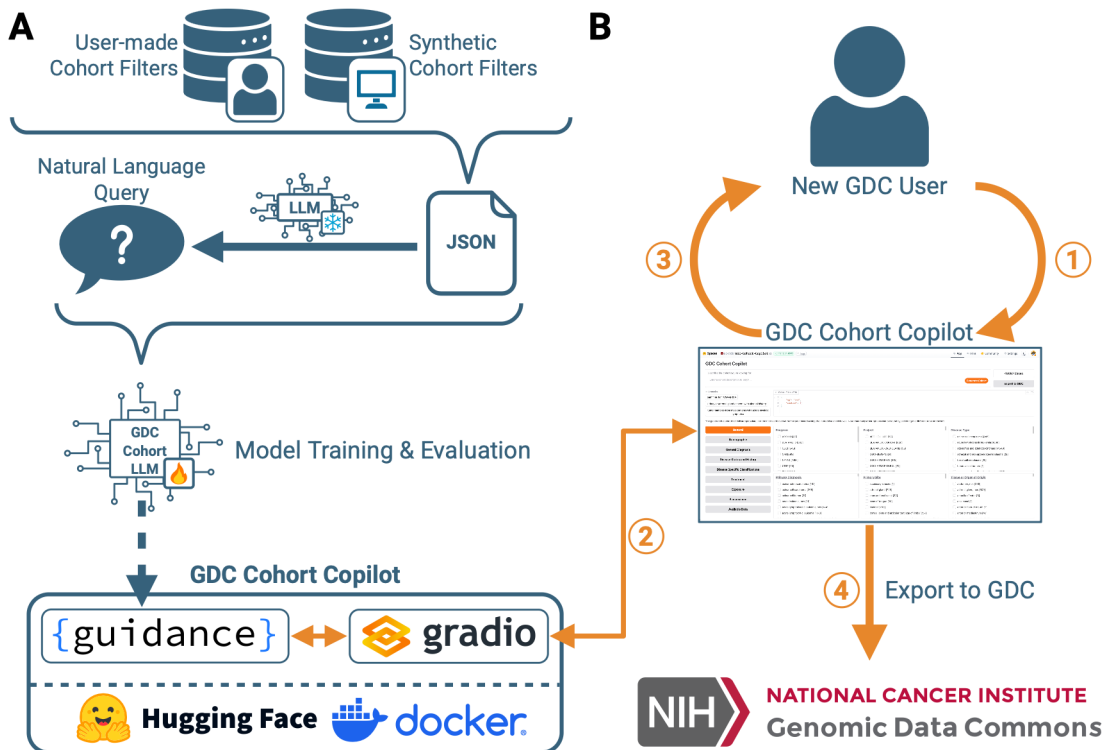[†] Corresponding author: rgrossman1@uchicago.edu

Figure 1: Overview of GDC Cohort Copilot implementation and user workflow. (A) Implementation of GDC Cohort Copilot involves training the GDC Cohort LLM to translate from a natural language query of a cohort to the cohort filter JSON. The cohort JSONs are derived from datasets of real user-made cohorts or synthetically generated cohorts. The paired natural language queries are generated by a frozen LLM using the cohort JSONs. The final trained GDC Cohort LLM model is served in a containerized web app that exposes a GDC Cohort Builder-like interface running on HuggingFace Spaces. (B) A user curates their desired cohort using the GDC Cohort Copilot by: (1) inputting a natural language description of a desired cohort (2) which is automatically passed to GDC Cohort LLM. The model is served using Guidance inside a Gradio app. (3) The resulting generated cohort filter is automatically populated back into the interface, allowing the user to manually refine their cohort before (4) exporting the curated cohort to the NCI GDC.

Here, we present GDC Cohort Copilot, an open-source AI copilot that enables users to curate GDC cohorts using natural language. Following the recent success of large language models (LLMs) in generating structured code[3] and database query languages[4,5], the GDC Cohort Copilot is powered by GDC Cohort LLM, an LLM trained to generate structured GDC cohort filters from free-text user input. We demonstrate that our locally-served, open-source model outperforms GPT-4o prompting in cohort construction accuracy. Once generated by the model, the tool automatically populates the cohort filter into a GDC Cohort Builder-like interface that allows the user to further refine their desired cohort. We provide a mechanism for exporting the curated cohort back to the GDC for further analysis. We release GDC Cohort Copilot as the overall framework presented in Figure 1, the GDC Cohort LLM, and the containerized web app.

2   Materials and methods

GDC Cohort Copilot is comprised of both the generative GDC Cohort LLM and the containerized web app interface. The overview of its implementation and user workflow is presented in Figure 1.

Model	TPR	IoU	Exact	BERT
GDC Cohort LLM	**0.855**	**0.832**	**0.702**	**0.919**
GPT-4o	0.720	0.698	0.558	0.894

Table 1: GDC Cohort LLM is significantly better at generating GDC cohorts across all reported metrics compared to GPT-4o ($p < 0.05$). TPR: true positive rate; IoU: intersection over union; Exact: exact match; BERT: F1 BERTScore using SciBERT.

2.1 Inputs and Outputs

The primary input to the GDC Cohort Copilot is a natural language description of a GDC cohort, for example: "cases with gene expression data derived from RNA sequencing for lung adenocarcinoma". Upon submitting the query, the app uses GDC Cohort LLM model to generate and return the corresponding cohort filter JSON. The interface automatically populates the corresponding checkboxes for filter properties specified by the generated JSON. A user can interactively refine the cohort selections, before ultimately exporting and outputting a text file of GDC case identifiers. These case identifiers can be imported by the GDC for further analysis.

2.2 Core Set of Filter Properties

In this initial release of the GDC Cohort Copilot, we simplify the development of the tool by considering only a subset of 68 filter properties from the GDC Cohort Builder. These are the default and most commonly used filter properties exposed by the GDC Data Portal v2.4.0 and additionally have predefined lists of possible values or value ranges (e.g. disease type or age at diagnosis). We refer to this subset of filters as the "core set". We create a JSON schema (from a Pydantic data model) to validate possible filters comprised of the core set.

2.3 GDC Cohort LLM

The GDC Cohort Copilot is powered by a generative LLM, GDC Cohort LLM, which translates natural language queries of cohorts into cohort filter JSONs. We describe here the development and evaluation of GDC Cohort LLM.

2.3.1 Data

GDC Cohort LLM is trained over paired natural language queries and cohort filter JSONs. This data is derived from real user-generated and synthetic cohort filters. 68,209 user-generated cohort filters were supplied by the GDC User Services team from their database of GDC user-saved cohorts. Removing duplicates, null filters, filters with properties outside of the core set, and filters which fail schema validation results in 16,235 usable cohort filters. We additionally experiment with augmenting our dataset by randomly sampling synthetic cohorts filters. Specifically, we randomly sample fields and values from the core set of filter properties. Further details on our random sampling procedure are provided in Section S1.1.1. We experiment with augmenting our training data using 100,000 and 1,000,000 synthetic cohort filters.

One of the primary limitations of our cohort filter dataset is that it does not contain any user-generated natural language descriptions of the cohorts. To address this challenge, we prompt Mistral-7B-Instruct-v0.3[6] to generate a corresponding natural language query for a given cohort filter JSON. Our precise procedure for this reverse translation, including the prompt we use, is provided in Section S1.1.2. We apply this method to all real and synthetic cohort filters.

We finally split our paired samples derived from real user-generated data into 14,235 for training and 2,000 for evaluation. For the evaluation split, we ensure that the model-specific token length of the natural language query and cohort filter for all samples fit within the minimum context length of the different models we experiment with. Additionally, we require that the evaluation samples do not result in empty cohorts (cohorts with 0 cases). This allows us to directly compare each experiment's results which were derived over precisely the same set of data samples.

2.3.2 Model

We experiment with three pretrained LLMs of different architectures and scales: GPT-2[7], BART[8], and Mistral-7B-Instruct-v0.3[6]. We train each of these models using a causal language modeling (CLM - autoregressive) objective. For BART, the input to the encoder is the natural language query while the output of the decoder is the cohort filter JSON. For GPT-2 and Mistral, we concatenate the natural language query with the cohort filter JSON. Additionally, for Mistral, we use low rank adaptation (LoRA)[9] to efficiently train the model for our translation task. We load model weights from HuggingFace and use HuggingFace utilities for training our models. Further training details are described in Section S1.1.3.

At evaluation time, for efficient batched inferencing, we serve the trained models using vLLM[10] with structured decoding using Outlines[11] to ensure that our generated outputs are valid cohort filter JSON. One limitation of the JSON schema we develop (Section 2.2) is that it does not strictly enforce field and property strings; rather, our schema enforces the structure of the filter.

2.3.3 Evaluation

After training the variations of GDC Cohort LLM, differing either in model type or data mixture, we evaluate the generated cohort filters. As we aim to enable accurate retrieval of cohorts, we do not directly evaluate the cohort filter; instead we compare the cases retrieved by the generated cohort filter to the cases retrieved by the true cohort filter. This allows flexibility in the actual content of the cohort filter as many filters may result in the same set of cases, for example selecting the TCGA program is equivalent to selecting all of the individual TCGA projects together. We thus compute three metrics for each filter's set of cases: sensitivity (true positive rate - TPR), Jaccard index (intersection over union - IoU), and a binary indicator for if the predicted and actual cases precisely match (Exact). If a generated filter is not valid (either due to context length truncation or imprecise generated field or value names) and cannot be used to retrieve cases using the GDC API, we use the empty set. TPR, IoU, and Exact are guaranteed to be finite and equal 0 if there are no predicted cases, as we ensure that the actual cases are never null (Section 2.3.1).

While we do not require the generated cohort filter precisely match the actual filter, we do evaluate whether they are semantically similar. To do so, we reverse translate the predicted cohort filters into natural language queries and compute the F1 BERTScore[12] (BERT) between the original and derived queries. We specifically use SciBERT[13] in the computation of the BERTScore. Additionally, we compare all of our trained models against a prompting-based alternative using OpenAI's GPT-4o[14] that requires an expensive, long context window of approximately 15,000 tokens per prompt. Further details of this comparison implementation are provided in Section S1.1.4. Finally, we report the average of all metrics across all 2,000 evaluation filters and apply paired t-tests (for TPR, IoU, BERT metrics) or McNemar's test (for Exact metric) with Bonferroni correction to evaluate statistical significance.

2.4 Web App, Containerization, and Deployment

We develop the web app for GDC Cohort Copilot as a Gradio[15] app deployed in a HuggingFace Space. HuggingFace Spaces provides out-of-the-box containerization with Gradio apps, enabling users to download and run GDC Cohort Copilot locally with docker. We package the GPT-2 variant of GDC Cohort LLM, trained over real and 1 million synthetic data samples, with GDC Cohort Copilot; in addition to its strong evaluation metrics, its architecture as a decoder-only, small-scale LLM enables it to be efficiently served. Specifically, we serve GDC Cohort LLM using Guidance[a] for structured generation. While we utilize GPU acceleration in our HuggingFace Space for serving GDC Cohort LLM, the model only requires approximately 1 GB of GPU VRAM and can even run efficiently on CPU. Our implementation of GDC Cohort Copilot allows it to be accessible to a wide variety of biomedical research users.

3 Results

We first evaluate the adaptability of various model types to our filter generation task as GDC Cohort LLM (Table S6). Training over user-derived data, we find that GPT-2 (TPR=0.365; IoU=0.331; Exact=0.221)

---

[a] https://github.com/guidance-ai/guidance

Comparison	GDC Cohort LLM	GPT-4o
Achieves SOTA	✓	✗
Open source	✓	✗
Deploy locally	✓	✗
Structured outputs	✓	✓
No training	✗	✓
Required tokens	≤ 1024	$> 15k$

Table 2: Conceptual comparison of GDC Cohort LLM and GPT-4o as LLMs to power GDC Cohort Copilot.

significantly outperforms BART (TPR=0.117, p=8.56e-89; IoU=0.078, p=7.73e-114; Exact=0.028, p=3.69e-94) and Mistral (TPR=0.124, p=2.33e-90; IoU=0.117, p=3.62e-80; Exact=0.092, p=9.35e-39) models over case-retrieval metrics. Over query-based metrics, GPT-2 (BERT=0.819) outperforms BART (BERT=0.735, p=4.03e-106). While GPT-2 is statistically worse than Mistral (BERT=0.835, p=6.47e-5), the difference is relatively small and not meaningful in the context of poor case-retrieval capabilities.

Given GPT-2's strong adaptability to GDC Cohort LLM, we next explore how to improve its performance by training over synthetic data mixtures (Table S7). We find that, compared to a baseline using only user-derived data (TPR=0.365; IoU=0.331; Exact=0.221; BERT=0.819), incorporating 100 thousand synthetically generated records with our real user data (TPR=0.783, p=9.59e-217; IoU=0.748, p=7.19e-227; Exact=0.607, p=1.00e-188; BERT=0.902, p=7.67e-145) significantly improves all metrics. We further train over a mixture of 1 million synthetic records with user records and find that this provides significantly stronger results (TPR=0.855, p=6.03e-18; IoU=0.832, p=1.85e-23; Exact=0.702, p=1.20e-23; BERT=0.919, p=3.74e-16) than using only 100 thousand synthetic samples.

Our final GDC Cohort LLM model is thus trained from a GPT-2 foundation over a mixture of 1 million synthetic and real user data. Importantly, GDC Cohort LLM (TPR=0.855; IoU=0.832; Exact=0.702; BERT=0.919) significantly outperforms a prompting-based implementation of cohort filter generation using GPT-4o (TPR=0.720, p=8.01e-37; IoU=0.748, p=2.08e-36; Exact=0.607, p=2.12e-37; BERT=0.894, p=3.57e-26) across all metrics (Table 1). Because GDC Cohort LLM is specifically trained for this task, to provide a more fair comparison, we prompt GPT-4o with a list of all possible field-value pairs which consume 15K tokens. This reduces the potential for GPT-4o to hallucinate invalid field or value names. Despite this, we find that our open-source, small-scale GDC Cohort LLM model achieves better results than GPT-4o. We conceptually compare GDC Cohort LLM to GPT-4o in Table 2.

Finally, we package GDC Cohort LLM with our GDC Cohort Copilot tool as a containerized Gradio app running on HuggingFace Spaces. GDC Cohort Copilot can additionally be downloaded and run locally using docker. We serve a GDC Cohort Builder-like interface to allow users to interactively curate cohorts using both natural language based descriptions and graphical checkboxes. We integrate with NCI GDC by providing utilities to export curated cohorts back to the GDC for further user-specific analysis.

4   Conclusion

GDC Cohort Copilot is a novel copilot tool to use natural language to assist in the curation of cohorts from the NCI GDC. Users can interactively use the copilot and a graphical interface to discover and refine their cohort. We experiment with various model types and data mixtures to develop GDC Cohort LLM, and demonstrate that our open-source, small-scale model is better able to accurately translate natural language descriptions of cohorts into their corresponding GDC cohort filter. We share GDC Cohort Copilot as a containerized Gradio app deployed on HuggingFace Spaces, ultimately providing an accessible tool to aid biomedical cancer researchers in their data curation efforts.

Acknowledgements

S1   Supplemental information

S1.1   Supplemental Methods

S1.1.1   Random Cohort Generation

We augment our existing user-derived cohort data by creating synthetic cohort filter JSONs through a two-step stochastic process. To create a new JSON filter, we first select the number of fields, n, to include by sampling a nonzero integer value from a Chi-Square distribution with 6 degrees of freedom, using rejection-sampling to ensure i.i.d. samples. Next, we randomly select n distinct fields from the core set of filter properties. For fields with numerical ranges, we randomly select a filter operator (one of $\leq, <, \geq, >$) and a corresponding numerical value within the field's allowable range. Conversely, all non-numerical fields use an "in" operator with a list of values. To create this list of filtering values for a given field, we first randomly select the number of values, m, with a minimum of 1 and a maximum of 5. The m values are then randomly selected from the list of possible values for the given field. We construct each filter as a JSON object and validate it against the GDC cohort schema to ensure correctness. Synthetic filter uniqueness is strictly enforced by computing an MD5 hash digest of the sorted JSON representation and rejecting all duplicates. We iteratively generate filters until the target number of unique, valid, synthetic cohort filters are generated.

S1.1.2   Reverse Translation

In our dataset of real user-derived cohort filters (originally, assembled using the GDC Cohort Builder), the cohort filters are JSON strings. As we aim to train an LLM to generate these filters using natural language, we develop a reverse-translation method to prompt an LLM to generate a JSON filter's corresponding natural language description. Specifically, we use `mistralai/Mistral-7B-Instruct-v0.3` for this task without any additional fine-tuning. We serve the model using vLLM on a single NVIDIA A100 80GB GPU and generate natural language queries using a set seed, temperature 0, and a maximum of 4,096 completion tokens. We use in-context learning to prompt the model with 2 examples of our task; our precise prompt for reverse-translation is provided in Table S3. We use this same process to generate natural language queries for our synthetic data samples (Section S1.1.1). Additionally, we use this method in our evaluation of semantic similarity between two cohort filters (Section 2.3.3).

S1.1.3   Model Training Details

We describe here our model training details for our experiments in developing GDC Cohort LLM. All models were trained using a single NVIDIA A100 80GB GPU. For all experiments, we use a linear learning rate schedule with an initial learning rate of 5e-5, 8 gradient accumulation steps, and a maximum sequence length of 1,024. For training the LoRA adapter for Mistral, we target the following layers: `q_proj`, `k_proj`, `v_proj`, `o_proj`, `gate_proj`, `down_proj`, `up_proj`. Specific hyperparameters for each experiment are further detailed in Table S4.

S1.1.4   Comparison to GPT-4o

To compare our trained GDC Cohort LLM against an expensive, closed source alternative using OpenAI's GPT-4o, we adopt the following prompting strategy. We use the OpenAI python library v1.84.0 to send requests to the OpenAI API. We use OpenAI's Structured Outputs by specifying our Pydantic data model (Section 2.2) as the response format to chat completions. This same data model and corresponding JSON

schema are also used by our locally served model. We pin the model version to `gpt-4o-2024-08-06`, set temperature to 0, set a fixed generation seed, and specify a maximum of 1,024 completion tokens. We use the prompt presented in Table S5. To provide GPT-4o with the set of all possible filter fields and their values without model tuning, we input in the prompt the entire field-value mapping for the core set of filter properties. This mapping is approximately 15,000 tokens long and thus, regardless of performance metrics, this method is infeasible for scalable inference as the token cost rapidly compounds. At the time of our experiments, the cost of our prompt to GPT-4o was approximately \$0.04.

```
Given the following examples of dict and sentence pairs, generate the
sentence that describes a new dict between <<>>.
Use the 'field' and it's corresponding 'value' information to correctly
identify the different categories.
Examples:

Example 1:
Dict :
{'op': 'and',
 'content': [{'op': 'in',
   'content': {'field': 'cases.project.program.name', 'value': ['TARGET']}},
  {'op': 'in',
   'content': {'field': 'cases.project.project_id',
     'value': ['TARGET-ALL-P1',
      'TARGET-ALL-P2',
      'TARGET-ALL-P3',
      'TARGET-AML']}},
  {'op': 'in',
   'content': {'field': 'cases.diagnoses.site_of_resection_or_biopsy',
     'value': ['bone marrow']}},
  {'op': 'in',
   'content': {'field': 'cases.samples.tissue_type', 'value': ['tumor']}},
  {'op': 'in',
   'content': {'field': 'cases.samples.tumor_code',
     'value': ['acute lymphoblastic leukemia (all)']}}]}
Sentence :
acute lymphoblastic leukemia tumor code for bone marrow tumors that belong to
TARGET-ALL-P1, TARGET-ALL-P2, TARGET-ALL-P3, TARGET-AML projects. |<eos>|

Example 2:
Dict:
{'op': 'and',
 'content': [{'op': 'in',
   'content': {'field': 'cases.project.program.name', 'value': ['CGCI']}},
  {'op': 'in',
   'content': {'field': 'cases.project.project_id', 'value': ['CGCI-BLGSP']}},
  {'op': 'in',
   'content': {'field': 'cases.diagnoses.tissue_or_organ_of_origin',
     'value': ['hematopoietic system, nos']}},
  {'op': 'in',
   'content': {'field': 'cases.samples.preservation_method',
     'value': ['ffpe']}}]}
Sentence:
ffpe samples for hematopoietic system, nos that belong to the CGCI-BLGSP
project. |<eos>|


<<{Filter}>>


Sentence:
```

Table S3: Prompt to reverse-translate natural language descriptions of input cohort filters using an LLM.

Model	Trainable Params	Train Epochs	Batch Size	Precision	Specifics
Mistral	11M	1	64	bf16	LoRA(r=4, α=16, dropout=0.1)
BART	139M	3	32	bf16	warmup steps=0
GPT-2	124M	10	32	fp16	warmup steps=500
GPT-2-100K	124M	10	32	fp16	warmup steps=500
GPT-2-1M	124M	10	32	fp16	warmup steps=500

Table S4: Model training hyperparameters for GDC Cohort LLM experiments.

Line	Role	Message
1	System	Construct NCI GDC cohort filters based on the input cohort description using the given list of possible fields and values.
2	User	Here is the list of possible fields and values: [Field-Value List] Use the above properties to construct a NCI GDC cohort filter for the following cohort description: [Query]

Table S5: Prompt to derive cohort filters using OpenAI's GPT-4o. The field-value list is populated using a predefined mapping for all core set filter properties. This list consumes approximately 15,000 tokens. Cohort filters are generated using structured decoding.

S1.2 Supplemental Results

Model Type	TPR	IoU	Exact	BERT
GPT-2	**0.365**	**0.331**	**0.221**	0.819
BART	0.117	0.078	0.028	0.735
Mistral	0.124	0.117	0.092	**0.835**

Table S6: GPT-2 significantly outperforms ($p < 0.05$), or is not meaningfully worse than, BART and Mistral as foundations for GDC Cohort LLM, when trained over user-derived cohort filters.

Data Mixture	TPR	IoU	Exact	BERT
Real	0.365	0.331	0.221	0.819
Real + 100K Synthetic	0.783	0.748	0.607	0.902
Real + 1M Synthetic	**0.855**	**0.832**	**0.702**	**0.919**

Table S7: Training over synthetic data mixtures significantly improves GDC Cohort LLM performance ($p < 0.05$).

References

[1] Allison P Heath, Vincent Ferretti, Stuti Agrawal, Maksim An, James C Angelakos, Renuka Arya, Rosita Bajari, Bilal Baqar, Justin HB Barnowski, Jeffrey Burt, et al. The NCI genomic data commons. *Nature genetics*, 53(3):257–262, 2021.

[2] Mark A Jensen, Vincent Ferretti, Robert L Grossman, and Louis M Staudt. The NCI genomic data commons as an engine for precision medicine. *Blood, The Journal of the American Society of Hematology*, 130(4):453–459, 2017.

[3] Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde De Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, et al. Evaluating large language models trained on code. *arXiv preprint arXiv:2107.03374*, 2021.

[4] Mohammadreza Pourreza, Hailong Li, Ruoxi Sun, Yeounoh Chung, Shayan Talaei, Gaurav Tarlok Kakkar, Yu Gan, Amin Saberi, Fatma Ozcan, and Sercan Ö. Arik. CHASE-SQL: multi-path reasoning and preference optimized candidate selection in text-to-sql. In *The Thirteenth International Conference on Learning Representations*, 2025.

[5] Balaji Ganesan, Sambit Ghosh, Nitin Gupta, Manish Kesarwani, Sameep Mehta, and Renuka Sindhgatta. LLM-powered GraphQL generator for data retrieval. In *Proceedings of the Thirty-Third International Joint Conference on Artificial Intelligence*, pages 8657–8660, 2024.

[6] Albert Q. Jiang, Alexandre Sablayrolles, Arthur Mensch, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Florian Bressand, Gianna Lengyel, Guillaume Lample, Lucile Saulnier, Lélio Renard Lavaud, Marie-Anne Lachaux, Pierre Stock, Teven Le Scao, Thibaut Lavril, Thomas Wang, Timothée Lacroix, and William El Sayed. Mistral 7B. *arXiv preprint arXiv:2310.06825*, 2023.

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

[8] Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Veselin Stoyanov, and Luke Zettlemoyer. BART: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7871–7880. Association for Computational Linguistics, July 2020.

[9] Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, Weizhu Chen, et al. LoRA: Low-rank adaptation of large language models. *ICLR*, 1(2):3, 2022.

[10] Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph Gonzalez, Hao Zhang, and Ion Stoica. Efficient memory management for large language model serving with PagedAttention. In *Proceedings of the 29th Symposium on Operating Systems Principles*, pages 611–626, 2023.

[11] Brandon T Willard and Rémi Louf. Efficient guided generation for large language models. *arXiv preprint arXiv:2307.09702*, 2023.

[12] Tianyi Zhang, Varsha Kishore, Felix Wu, Kilian Q. Weinberger, and Yoav Artzi. BERTScore: Evaluating text generation with BERT. In *8th International Conference on Learning Representations*, 2020.

[13] Iz Beltagy, Kyle Lo, and Arman Cohan. Scibert: A pretrained language model for scientific text. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 3615–3620, 2019.

[14] Aaron Hurst, Adam Lerer, Adam P Goucher, Adam Perelman, Aditya Ramesh, Aidan Clark, AJ Ostrow, Akila Welihinda, Alan Hayes, Alec Radford, et al. GPT-4o system card. *arXiv preprint arXiv:2410.21276*, 2024.

[15] Abubakar Abid, Ali Abdalla, Ali Abid, Dawood Khan, Abdulrahman Alfozan, and James Zou. Gradio: Hassle-free sharing and testing of ml models in the wild. *arXiv preprint arXiv:1906.02569*, 2019.