Data+AI: LLM4Data and Data4LLM

Guoliang Li
Tsinghua University
Beijing, China
liguoliang@tsinghua.edu.cn

Jiayi Wang
Tsinghua University
Beijing, China
jiayi-wa20@mails.tsinghua.edu.cn

Chenyang Zhang
Tsinghua University
Beijing, China
z-cy22@mails.tsinghua.edu.cn

Jiannan Wang
Simon Fraser University
Canada
jnwang@sfu.ca

Abstract

Large language models (LLMs) have revolutionized traditional data management systems by their natural language processing capabilities (e.g., understanding, reasoning, generation, few-shot and zero-shot learning), while data management techniques play a vital role in optimizing AI models (e.g., data preparation, data efficient training and inference). This establishes a two-way relationship where AI enhances data management, and data boosts AI capabilities. This tutorial covers recent advancements and challenges at this intersection, focusing on LLM4Data and Data4LLM over four parts. Initially, we discuss the background and motivations for integrating data and AI, and present the challenges and principles of LLM4Data and Data4LLM. We then explore how LLMs optimize data management by offering practical applications like managing unstructured data. We also examine how data management optimizes AI, particularly in training and fine-tuning LLMs, showcasing techniques for data preparation and inference. Finally, we provide open challenges and future research directions, aiming to drive innovation in both fields.

CCS Concepts

• **Information systems** → **Data management systems**.

Keywords

Data Management, LLM, LLM4Data, Data4LLM

ACM Reference Format:
Guoliang Li, Jiayi Wang, Chenyang Zhang, and Jiannan Wang. 2025. Data+AI: LLM4Data and Data4LLM. In *Companion of the 2025 International Conference on Management of Data (SIGMOD-Companion '25), June 22–27, 2025, Berlin, Germany*. ACM, New York, NY, USA, 7 pages. https://doi.org/10.1145/3722212.3725641

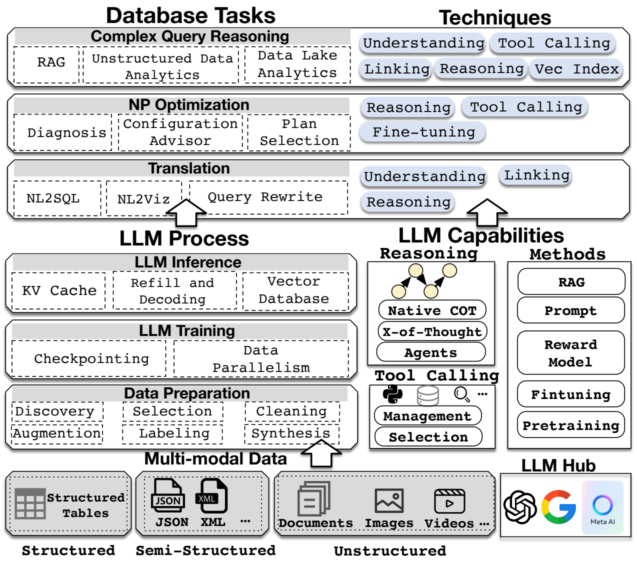

Figure 1: An Architecture for Data4LLM and LLM4Data.

1 Introduction

In the era of big data and artificial intelligence, the synergy between data management systems and AI techniques is becoming increasingly crucial. The advent of large language models (LLMs) and their ability to understand and process natural language has opened new avenues for enhancing traditional database systems. On the other hand, data management techniques are also crucial for training and optimizing AI models, leading to a bidirectional relationship where AI can enhance data management and data can enhance AI capabilities. This tutorial aims to systematically analyze this interaction, focusing on how large language models can optimize data management processes and vice versa.

Tutorial Overview. This tutorial provides a comprehensive overview of the latest advancements and challenges at the intersection of data management and AI, especially LLM4Data and Data4LLM. It is structured into four parts and intended to last for 1.5 hours.

Background and Motivation (15 minutes). The first part, which will last approximately 15 minutes, introduces the background and motivations behind the convergence of data and AI techniques. We will discuss the transformative impact of this integration.

LLM4Data (15 minutes). The second part, spanning 15 minutes, introduces techniques for optimizing data management processes with LLMs. We will first summarize the key challenges and design principles (5 minutes), then examine how LLMs can revolutionize data analytics on unstructured data and data lakes (10 minutes).

Data4LLM (50 minutes). The third part, covering 50 minutes long, discusses how data management techniques can optimize AI, particularly focusing on the role of data management in training, fine-tuning LLMs and LLM inference. We will first introduce the LLM lifecycle and discuss the principles of Data4LLM (10 minutes). We will then present state-of-the-art techniques for data preparation (10 minutes), LLM training (10 minutes), and LLM inference (20 minutes).

Open Challenges (10 minutes). The final part, concluding in 10 minutes, summarizes the open challenges and future research directions in this rapidly evolving field. We will outline several fundamental questions and discuss the potential solutions.

Target Audience. This tutorial is intended for database researchers with a keen interest in the intersection of databases and LLMs. The tutorial does not require prerequisites beyond a basic understanding of databases and LLMs. By the end of this tutorial, participants will have gained insights into the state-of-the-art techniques and systems that harness the power of both data and AI, and will be equipped to explore the cutting-edge research and applications in this domain.

Related Tutorials. A recent tutorial [31] focuses on leveraging LLMs to optimize traditional data management tasks, e.g., database diagnosis and query optimization. In contrast, our tutorial explores the latest advancements in applying LLMs to enhance unstructured data analytics and data lake analytics. Additionally, we cover Data4LLM techniques, which were not addressed in prior work. This broader perspective provides attendees with deeper insights.

2 Tutorial

2.1 Background and Motivation

The landscape of artificial intelligence has been significantly reshaped with the emergence of Large Language Models (LLMs). These models have revolutionized data management systems due to their semantic understanding abilities and much higher generalization abilities. Similarly, data management optimizations can also directly benefit the effectiveness and efficiency of LLMs. In this section, we explore the compelling opportunities that LLMs present beyond the scope of conventional ML techniques.

Generalizability (Knowledge Coverage). One of the most profound aspects of LLMs is their generalizability across a vast expanse of knowledge. Unlike traditional ML models that are often confined to the specific domains in which they are trained, LLMs boast an extensive knowledge coverage due to their training on diverse datasets. This attribute enables them to understand and process a wide array of queries and tasks, even those that extend beyond their training data, making them an invaluable asset for data management systems that require broad semantic understanding.

Reasoning (Inference). LLMs are not merely passive recipients of data; they possess the capability to reason and infer, a capability that is crucial for complex data management tasks. Their ability to draw logical conclusions from given premises allows for enhanced query processing and optimization, where the model must understand not just the data, but also the relationships and dependencies that underlie the data (and even the tasks).

Semantic-aware Processing. Traditional data management employs a "close-world" model, i.e., it can only get the results that are exactly in the database. However, in many scenarios, we require "open-world", i.e., semantic matching between different representations of the same entity, automatic data transformation, etc. The semantic prowess of LLMs is a key differentiator, where they can discern nuances, context, and subtleties that are typically challenging for traditional ML models. This ability is particularly beneficial in database interactions, where the precision of semantic understanding can lead to more accurate and efficient data retrieval and analysis.

Understanding and Generation. Beyond comprehension, LLMs are capable of generation, which is a significant leap from traditional ML models. They can create human-like text in response to prompts, which can be utilized in data management for generating reports, automating data documentation, and even crafting queries in natural language.

In-context Learning. Perhaps one of the most exciting prospects of LLMs is their ability to perform zero-shot or few-shot learning. This capability allows them to solve tasks without or with very little fine-tuning, respectively, which is a significant advantage over traditional ML models that require extensive amounts of labeled data for training. In the context of data management, this can lead to faster deployment of models and quicker adaptation to new schemas or database structures.

In summary, the opportunities presented by LLMs extend well beyond the traditional boundaries of ML, offering a new frontier for data management systems. Their generalizability, reasoning capabilities, semantic understanding, generative abilities, and aptitude for few-shot learning make them powerful tools for enhancing the efficiency, accuracy, and versatility of data management processes. It is these capabilities that form the foundation of our exploration in this tutorial, as we delve into the practical applications and challenges of integrating LLMs with data management techniques.

2.2 LLM4Data

2.2.1 Challenges and Principles of LLM4Data.

We first discuss the challenges of LLM4Data and then present the design principles and techniques of addressing these challenges.

Challenges of LLM4Data. Using large language models (LLMs) directly for data management tasks presents several challenges, including low accuracy, high computational cost, and hallucination.

Low Accuracy. LLMs generally lack the specialized domain knowledge necessary for accurate data management. Effective data management requires an in-depth understanding of task-specific requirements, e.g., strict equivalence before and after query rewriting, and strict correspondence with actual schema in NL2SQL, which generic LLMs often cannot provide.

High Cost. Data management tasks often involve multiple calls to LLMs, each of which incurs significant computational costs. This becomes particularly problematic for tasks requiring repeated interactions with the LLM.

Hallucination. LLMs are prone to generating outputs that appear plausible but are not grounded in factual data. This introduces the risk of producing misleading results, highlighting the need for verifiable outputs.

Limited Reasoning. LLMs find it challenging to handle complex tasks that involve intricate reasoning (like query rewriting) and multi-step problem-solving (such as analyzing unstructured data).

Effective techniques need to be designed to guide LLMs in reasoning through these tasks.

Principles of LLM4Data. To address these challenges, LLM4Data is guided by the following principles.

Involving Domain Knowledge. To improve accuracy, LLM4Data should integrate domain-specific knowledge and task-specific constraints, ensuring that the model's output aligns with the requirements of the data management task.

Cost-Efficiency Optimization. By optimizing the number of LLM interactions, LLM4Data seeks to reduce the computational cost. This can be achieved through caching and reducing unnecessary model invocations.

Verification and Reliability. To mitigate hallucination, LLM4Data incorporates mechanisms for output verification, ensuring that the results are not only plausible but also grounded in verifiable facts. This may involve referencing external data sources for verification.

Reasoning and Self-Reflection. To handle complex tasks, LLMs must engage in multi-step reasoning, with feedback provided for each decision. Thus, self-reflection is essential for offering precise feedback on task breakdown and analysis.

Technical Solutions of LLM4Data. To tackle these challenges, various techniques can be employed, such as automatic prompt generation, RAG, agents, and fine-tuning.

Prompting. Prompting engineering aims to provide task descriptions, instructions, zero-shot and few-shot learning, in order to instruct LLMs to understand the underlying tasks. The challenges include automatic prompting generation, demonstration examples selection, and prompting compression to reduce the LLMs cost.

RAG. RAG aims to search relevant vertical domain data and feed them into LLMs in order to avoid hallucination. The challenges include semantic document segmentation, embedding strategy selection, embedding indexing and searching, and reranking.

Agent. Agents are used for multi-step reasoning to handle complex tasks. Challenges include understanding the environment, tool invocation, breaking down tasks into multiple steps, reasoning through these steps, and self-reflection.

Fine-tuning. Fine-tuning is used for task alignment and instruction following. Challenges include training data selection and labeling.

2.2.2 LLM4Data Techniques.

We summarize recent studies that utilize LLMs for data analytics over unstructured data and data lakes.

Unstructured Data Analytics. The emergence of LLMs make it possible to conduct analytics over unstructured data through the semantic understanding abilities of LLMs. The data analytics queries can be divided into two types: (1) point queries that only require point look-up of relevant data (e.g., RAG); and (2) aggregation queries that require aggregating a large amount of data and information to make more complex aggregated inferences.

RAG. To answer questions that exceed the knowledge contained in the training data of LLMs, RAG has been proposed as a means to augment LLMs' capabilities by retrieving semantically relevant information from an external text corpus [8, 21, 65]. Typically, retrieval is performed using dense retrieval methods, where the query and documents are converted into embedding vectors, followed by a nearest neighbor search to identify relevant items. In multi-hop reasoning scenarios, this retrieval process is often iterative [65].

Unstructured Document Analytics. Recent advancements in LLMs have significantly enhanced the ability to extract valuable insights from large-scale unstructured datasets [5, 7, 34–36, 43, 58]. Several systems, such as PALIMPZEST [35], LOTUS [43], ZENDB [34], and Unify [58] have been developed to address the challenges in unstructured data analytics.

Schema Extraction. Except retrieving relevant information from the data collections during inference, extracting structured information from unstructured data can also be conducted in a pre-processing step. Some works [7] automatically extract structured schemas from unstructured data with the help of LLMs. The extracted schemas can then be leveraged for answering structured queries in SQL format or natural language queries via table QA methods or transforming the NL query into SQL by NL2SQL methods. Complete reliance on LLMs for extraction inevitably results in huge and unaffordable costs. Evaporate [7] leverages weak supervision methods by extracting a collection of individual rule-based attribute extraction functions and combining their results with weak supervision.

Data Lake Analytics. Compared with data analytics over specified unstructured data, data lake analytics requires an additional schema linking step to associate diverse data relevant to the query and identifies their relationships.

Schema Linking. Answering analytics queries over large-scale data lakes requires linking relationships among the data and identifying those relevant to the query. To address this challenge in hybrid multi-modal data lakes, AOP [59] leverages the observation that all data types possess literal descriptions in varying formats: structured data uses defined schemas with named attributes, semi-structured data has flexible key paths, and unstructured data inherent has textual content. AOP converts these descriptions into a unified semantic embedding space, enabling data linking through similarity measurements between embeddings. Notably, this method can be combined with the previously introduced structural data extraction technique to enhance accuracy, as the two methods are complementary.

Planning. Effective planning is crucial for answering data lake analytics queries, as accurate results depend on a processing path with correct reasoning logic. SYMPHONY [15] decomposes queries into sequences of sub-queries by prompting large language models LLMs with carefully designed prompts, then evaluates each sub-query with additional LLM-generated prompts. CAESURA [53] similarly employs LLMs for planning but integrates tools such as VisualQA, TextQA, and Python UDFs to support multi-modal data processing. ELEET [54] introduces a specialized encoder model, a lightweight language model with 140 million parameters, to enhance efficiency. It expands traditional relational algebra to express multi-modal operators and executes queries using an extended algebra. ELEET's encoder generates embeddings from inputs, which are processed by lightweight decoder heads to perform tasks such as value extraction and deduplication, transforming unstructured data into structured formats. iDataLake [60] orchestrates plans with predefined semantic operators. While CAESURA, SYMPHONY, and iDataLake leverage LLMs to plan and execute queries via NL interfaces, ELEET emphasizes efficiency by relying on manually specified execution plans in its extended algebra framework.

Current methods predominantly focus on planning and suppose target data are specified, but the joint optimization of schema linking and planning remains an open problem. This integration is crucial for realizing efficient and accurate query execution in multi-modal data lakes and needs further research.

2.3 Data4LLM

2.3.1 Challenges and Principles of Data4LLM. We discuss the challenges of Data4LLM and the principles for addressing them. The LLM life-cycle includes pretraining (and incremental pretraining), fine-tuning (SFT and RLHF), prompting, RAG, Agent. The first challenge is to prepare high quality data for pretraining and fine-tuning. The second challenge is to improve the performance of pretraining and fine-tuning. The third challenge is to enhance the LLM inference performance. The techniques include LLM-in-the-loop data preparation, data parallel and checkpointing for pretraining, and prefill-decode disaggregation and KV-cache for fast inference.

2.3.2 Data4LLM Techniques. Data4LLM includes automatic data preparation, efficient LLM pre-training, and fast LLM inference.
Data Preparation. Data preparation includes data discovery, selection, cleaning, augmentation, labeling and synthesis.
Data Discovery. Training data for LLMs often contain mixtures of data from diverse sources and domains. Establishing an appropriate domain mixture ratio is crucial for effective pretraining [13]. Determining the ratio has been approached through experimental heuristics and intuitions [16, 20], importance resampling [64], and gradient-based methods that assess each domain's contribution [18]. Systems like Data-Juicer [13] provide an automated evaluation framework integrated with LLM training and evaluation pipelines to identify optimal domain mixture ratios effectively.
Data Selection. Training or fine-tuning LLMs is both costly and time-intensive, primarily due to the extensive parameter updates required for models with billions of parameters. These costs are influenced by two key factors: the size of the model and the volume of training data. To mitigate these challenges, recent research has emphasized the importance of data selection as a preprocessing step before training [9]. The goal of data selection is to identify a small yet representative subset of the full dataset—referred to as a coreset [12, 57]—that achieves comparable model performance to training on the entire dataset. While coreset selection has been extensively studied for tabular data [11, 12, 57], similar principles are being adapted for LLMs [67]. For LLMs, data selection techniques often rely on specific importance metrics, such as perplexity [14] and influence functions [63], to identify high-value training examples.
Data Cleaning. Effective data cleaning is a cornerstone of building high-quality datasets for training LLMs, which generally consists of two aspects: data filtering to remove low-quality or toxic data and data deduplication to remove redundant data that may harm model performance [23, 29]. Data filtering aims to remove low-quality, irrelevant or toxic data that may hinder model training. This is accomplished through heuristic rules [41, 46], classifiers [10, 62], metric-based thresholds [39] or their combination. These methods are applied to identify and exclude noisy or redundant text from large corpora. Similarly, toxic data filtering can employ heuristic rule-based methods [30] or n-gram hashing techniques [46]. For deduplication, n-gram hashing techniques [24, 46, 52] at both the line [52] and document levels [24] are employed.
Data Augmentation. Data augmentation is used to increase the diversity of training dataset by applying various transformations to the existing data. This can help improve the performance and generalization of machine learning models. The techniques include data linking, synonym replacement, etc.
Data Labeling. Data labeling is the process of annotating data with tags or labels that define its characteristics, making it suitable for use in supervised machine learning models. The techniques include crowdsourced labelling, weak supervision, model-based labelling, transfer learning, active learning, etc.
Data Synthesis. Data synthesis aims to generate artificial data that mimics the properties and distribution of real-world data. The techniques include statistical methods, generative Models (e.g., GANs, VAEs), rule-based methods, etc.
LLM Training. We summarize checkpointing for LLM training recovery and data parallelism for accelerating LLM training.
Checkpointing. During the pretraining and fine-tuning phases of LLMs, the training state needs to be saved and persisted using checkpointing techniques. These saved checkpoints can later be retrieved for continued training or analysis. Additionally, the parallel configuration may change during training, necessitating checkpoint resharding to redistribute the saved distributed checkpoints. Systems such as PyTorch Distributed Checkpoint (DCP) [51], DeepSpeed Universal Checkpointing (UCP) [33], and ByteCheckpoint [56] support checkpoint resharding. Checkpoints can be stored in several formats, including array-based [1, 2, 50], file-based [49, 56], and disaggregated formats [51]. Various optimization techniques for checkpointing have been proposed, such as pipeline checkpointing with computation [27, 38, 56], differential checkpointing [17], quantization [17], determining checkpointing frequency [38], and asynchronous checkpointing [27, 37, 38, 61].
Data Parallelism. Training large models typically involves various parallelism techniques, including data parallelism, pipeline parallelism, tensor parallelism, and sequence parallelism, which are often used in combination [26, 40, 48]. We briefly introduce several optimized data parallelism methods. DeepSpeed ZeRO [6, 47] is a memory optimization technique designed to eliminate memory redundancies during large language model (LLM) training using data parallelism. PyTorch's Fully Sharded Data Parallel (FSDP) [68] technique shards model parameters and applies several optimization strategies to enable efficient training. Colossal-AI [32] improves the tensor sharding and offloading mechanism.
LLM Inference. LLMs are primarily based on the Transformer architecture [55], with notable examples including OpenAI's GPT-3 and GPT-4, Google's BERT, T5, and BLOOM. The inference process in LLMs built on the Transformer generates text based on a given input prompt. This process involves key steps. First, the input text is converted to tokens. Each token is then mapped to a high-dimensional vector through embedding layers. Following this, the model enters the prefill stage of the attention mechanism, which computes three vectors for each token: the query, key, and value vectors. These vectors are used to calculate the attention scores. Next, the model uses these attention scores to produce a context-sensitive

representation of each token passed through multiple layers. The model then generates the next token. This process continues sequentially during the decoding stage, where the model predicts one token at a time based on the previous tokens. The LLM uses key and value vectors of all previous tokens to compute the attention scores and generates the next token. The KV cache mechanism is proposed to store these vectors to avoid repeated calculation of key and value vectors. KV cache enables faster and more efficient inference. The inference process is typically measured by Service Level Objectives (SLOs), typically including Time to First Token (TTFT) and Time Between Tokens (TBT). TTFT demonstrates the performance of the prefill stage, while TBT reflects the performance of the decoding stage. Next, we introduce prefill and decoding design, KV cache data management, and vector databases for LLM inference.

Prefill and decoding design. Batching is a common technique in machine learning inference that processes multiple requests at once to improve computational efficiency and throughput. Continuous batching [66] is an optimization technique designed to address LLMs' dynamic input and ouput. Instead of waiting for the completion of a batch, continuous batching starts a new request once an old request is finished. The prefill stage handles the input prompt, leading to a long computing time. In contrast, a decoding step generates one token at a time, which consumes much less time. Batching a prefill stage with a decoding step stalls the decoding. The chunked-prefill [4, 25] technique is proposed to solve the problem. It splits the prefill stage into chunks and batches a chunk of prefill with a decoding step. Some research works find that processing prefill and decoding of a request on one GPU use excessive computing resources to meet SLOs. They propose two-stage disaggregation [25, 44, 45, 69], which splits prefill and decoding computation on different GPUs.

KV cache Data Management. Traditional KV cache storage, a fixed amount of space is pre-allocated for each request [66], corresponding to the maximum capacity of the KV cache. This strategy, however, leads to several issues: it limits the system's ability to handle very long inputs, and it also wastes a significant amount of memory for shorter inputs. To address these challenges, vLLM [28] proposes a shared prefix, an optimization technique to reduce redundant computation. It reserves a set of memory blocks for predefined shared prefixes, and the shared prefix will be computed only once. Prompt Cache [22] precomputes and stores the attention states of frequently text segments and reuses the attention states across different requests. TensorRT-LLM [3, 42] also involves the KV cache reuse technique and allows developers to determine the block size of the KV cache between 64 to 2 tokens. This fine-grained method optimizes memory usage and reuse rates. AttentionStore [19] develops a hierarchical KV cache storage system. KV cache eviction and transmission are also important. Cache eviction algorithms such as LRU (Least Recently Used) and LFU (Least Frequently Used) are commonly used. vLLM [28] employs an all-or-nothing eviction policy that evicts all or none of the blocks of a sequence. Mooncake [45] proposes a heuristic-based automated KV cache migration method. AttentionStore [19] proposes scheduler-aware fetching and eviction schemes to place the KV cache for faster inference. TensorRT-LLM [3] includes an intelligent eviction algorithm that forms a tree-like structure of dependencies for KV cache blocks. This algorithm evicts dependent nodes first, even if they have more recent reuse counters. To reduce KV cache transmission overheads, overlapping the transmission with the GPU computation is a common method [19, 45].

2.4 Open Challenges

LLM Inference. Leveraging LLM inference is crucial for unlocking the potential of data, especially in areas like RAG and analytics on unstructured data. However, conventional LLM inference techniques often fall short in utilizing data-level optimization strategies. Techniques such as query batching and scheduling, frequent token caching, managing data placement among GPU and CPU memory, and separating prefill from decoding are not fully exploited.

LLM-in-the-loop Data Preparation System. Current data preparation systems rely on several independent tools for tasks like data discovery, selection, cleaning, integration, and generation. However, there's a need for a comprehensive, end-to-end solution that can tackle these processes with a unified approach. Incorporating LLMs into the data preparation process is essential for enhancing accuracy while maintaining manageable overhead.

Data Flywheel. The data flywheel is a self-reinforcing cycle where data collection, analysis, and application continuously enhance model accuracy and serving quality, while in turn driving further data generation and business growth. There are some challenges including ensuring data quality, integrating diverse systems, scaling infrastructure, maintaining privacy and security, generating actionable insights, and designing effective feedback loops.

3 Presenters

Guoliang Li is a full professor in the Department of Computer Science, Tsinghua University. His research interests mainly include machine learning for databases and databases for machine learning. He got VLDB 2017 early research contribution award, TCDE 2014 early career award, VLDB 2023 Industry Best paper runner-up, Best of SIGMOD 2023, SIGMOD 2023 research highlight award, DASFAA 2023 Best Paper Award, CIKM 2017 best paper award. He will present the background and motivation, and open challenges.

Jiayi Wang is a PhD student in the Department of Computer Science, Tsinghua University, advised by Guoliang Li. His research interest is mainly optimizing databases with machine learning techniques. He will present LLM4Data.

Chenyang Zhang is a PhD student in the Department of Computer Science, Tsinghua University, advised by Guoliang Li. Her research interest is LLM inference optimizations. She will present LLM training and inference.

Jiannan Wang is Chief Data Scientist at Huawei Cloud and Associate Professor in the School of Computing Science at Simon Fraser University. His research focuses on data agents and cloud-native data systems. He will present on data preparation for LLMs.

4 Acknowledgment

Guoliang Li is the corresponding author. This paper was supported by National Key R&D Program of China (2023YFB4503600), Shenzhen Project (CJGJZD20230724093403007), NSF of China (62232009), Zhongguancun Lab, and Huawei.

References

[1] Tensorstore. https://google.github.io/tensorstore/, 2024.

[2] Zarr. https://zarr.dev/, 2024.

[3] A. E. abd Nick Comly and T. Johnsen. 5x faster time to first token with nvidia tensorrt-llm kv cache early reuse. https://developer.nvidia.com/blog/5x-faster-time-to-first-token-with-nvidia-tensorrt-llm-kv-cache-early-reuse/, 2024.

[4] A. Agrawal, N. Kedia, A. Panwar, J. Mohan, N. Kwatra, B. Gulavani, A. Tumanov, and R. Ramjee. Taming Throughput-Latency tradeoff in LLM inference with Sarathi-Serve. In *18th USENIX Symposium on Operating Systems Design and Implementation (OSDI 24)*, pages 117–134, Santa Clara, CA, July 2024. USENIX Association.

[5] E. Anderson, J. Fritz, A. Lee, and et al. The design of an llm-powered unstructured analytics system. *arXiv:2409.00847*, 2024.

[6] Anonymous. ZeRO++: Extremely efficient collective communication for large model training. In *Machine Learning for Systems 2023*, 2023.

[7] S. Arora, B. Yang, S. Eyuboglu, A. Narayan, A. Hojel, I. Trummer, and C. Ré. Language models enable simple systems for generating structured views of heterogeneous data lakes. *Proc. VLDB Endow.*, 17(2):92–105, 2023.

[8] A. Asai, Z. Wu, Y. Wang, A. Sil, and H. Hajishirzi. Self-rag: Learning to retrieve, generate, and critique through self-reflection. *ICLR 2024*, 2024.

[9] M. Böther, T. Robroek, V. Gsteiger, R. Holzinger, X. Ma, P. Tözün, and A. Klimovic. Modyn: Data-centric machine learning pipeline orchestration. *Proc. ACM Manag. Data*, 3(1):55:1–55:30, 2025.

[10] T. B. Brown, B. Mann, N. Ryder, M. Subbiah, J. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell, S. Agarwal, A. Herbert-Voss, G. Krueger, T. Henighan, R. Child, A. Ramesh, D. M. Ziegler, J. Wu, C. Winter, C. Hesse, M. Chen, E. Sigler, M. Litwin, S. Gray, B. Chess, J. Clark, C. Berner, S. McCandlish, A. Radford, I. Sutskever, and D. Amodei. Language models are few-shot learners. In H. Larochelle, M. Ranzato, R. Hadsell, M. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*, 2020.

[11] C. Chai, J. Liu, N. Tang, J. Fan, D. Miao, J. Wang, Y. Luo, and G. Li. Goodcore: Data-effective and data-efficient machine learning through coreset selection over incomplete data. *Proc. ACM Manag. Data*, 1(2):157:1–157:27, 2023.

[12] C. Chai, J. Wang, N. Tang, Y. Yuan, J. Liu, Y. Deng, and G. Wang. Efficient coreset selection with cluster-based methods. In *Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining, KDD 2023, Long Beach, CA, USA, August 6-10, 2023*, pages 167–178. ACM, 2023.

[13] D. Chen, Y. Huang, Z. Ma, H. Chen, X. Pan, C. Ge, D. Gao, Y. Xie, Z. Liu, J. Gao, Y. Li, B. Ding, and J. Zhou. Data-juicer: A one-stop data processing system for large language models. In *Companion of the 2024 International Conference on Management of Data, SIGMOD/PODS 2024, Santiago AA, Chile, June 9-15, 2024*, pages 120–134. ACM, 2024.

[14] J. Chen, Z. Chen, J. Wang, K. Zhou, Y. Zhu, J. Jiang, Y. Min, W. X. Zhao, Z. Dou, J. Mao, Y. Lin, R. Song, J. Xu, X. Chen, R. Yan, Z. Wei, D. Hu, W. Huang, and J. Wen. Towards effective and efficient continual pre-training of large language models. *CoRR*, abs/2407.18743, 2024.

[15] Z. Chen, Z. Gu, L. Cao, J. Fan, S. Madden, and N. Tang. Symphony: Towards natural language query answering over multi-modal data lakes. In *13th Conference on Innovative Data Systems Research, CIDR 2023, Amsterdam, The Netherlands, January 8-11, 2023*. www.cidrdb.org, 2023.

[16] N. Du, Y. Huang, A. M. Dai, S. Tong, D. Lepikhin, Y. Xu, M. Krikun, Y. Zhou, A. W. Yu, O. Firat, B. Zoph, L. Fedus, M. P. Bosma, Z. Zhou, T. Wang, Y. E. Wang, K. Webster, M. Pellat, K. Robinson, K. S. Meier-Hellstern, T. Duke, L. Dixon, K. Zhang, Q. V. Le, Y. Wu, Z. Chen, and C. Cui. Glam: Efficient scaling of language models with mixture-of-experts. In K. Chaudhuri, S. Jegelka, L. Song, C. Szepesvári, G. Niu, and S. Sabato, editors, *International Conference on Machine Learning, ICML 2022, 17-23 July 2022, Baltimore, Maryland, USA*, volume 162 of *Proceedings of Machine Learning Research*, pages 5547–5569. PMLR, 2022.

[17] A. Eisenman, K. K. Matam, S. Ingram, D. Mudigere, R. Krishnamoorthi, K. Nair, M. Smelyanskiy, and M. Annavaram. Check-N-Run: a checkpointing system for training deep learning recommendation models. In *19th USENIX Symposium on Networked Systems Design and Implementation (NSDI 22)*, pages 929–943, Renton, WA, Apr. 2022. USENIX Association.

[18] S. Fan, M. Pagliardini, and M. Jaggi. DOGE: domain reweighting with generalization estimation. In *Forty-first International Conference on Machine Learning, ICML 2024, Vienna, Austria, July 21-27, 2024*. OpenReview.net, 2024.

[19] B. Gao, Z. He, P. Sharma, Q. Kang, D. Jevdjic, J. Deng, X. Yang, Z. Yu, and P. Zuo. Cost-efficient large language model serving for multi-turn conversations with cachedattention, 2024.

[20] L. Gao, S. Biderman, S. Black, L. Golding, T. Hoppe, C. Foster, J. Phang, H. He, A. Thite, N. Nabeshima, S. Presser, and C. Leahy. The pile: An 800gb dataset of diverse text for language modeling. *CoRR*, abs/2101.00027, 2021.

[21] Y. Gao, Y. Xiong, X. Gao, K. Jia, and et al. Retrieval-augmented generation for large language models: A survey. *CoRR*, abs/2312.10997, 2023.

[22] I. Gim, G. Chen, S.-s. Lee, N. Sarda, A. Khandelwal, and L. Zhong. Prompt cache: Modular attention reuse for low-latency inference. In P. Gibbons, G. Pekhimenko, and C. D. Sa, editors, *Proceedings of Machine Learning and Systems*, volume 6, pages 325–338, 2024.

[23] D. Hernandez, T. B. Brown, T. Conerly, N. DasSarma, D. Drain, S. E. Showk, N. Elhage, Z. Hatfield-Dodds, T. Henighan, T. Hume, S. Johnston, B. Mann, C. Olah, C. Olsson, D. Amodei, N. Joseph, J. Kaplan, and S. McCandlish. Scaling laws and interpretability of learning from repeated data. *CoRR*, abs/2205.10487, 2022.

[24] J. Hoffmann, S. Borgeaud, A. Mensch, E. Buchatskaya, T. Cai, E. Rutherford, D. de Las Casas, and et al. An empirical analysis of compute-optimal large language model training. In S. Koyejo, S. Mohamed, A. Agarwal, D. Belgrave, K. Cho, and A. Oh, editors, *NeurIPS 2022*, 2022.

[25] C. Hu, H. Huang, L. Xu, X. Chen, J. Xu, S. Chen, H. Feng, C. Wang, S. Wang, Y. Bao, N. Sun, and Y. Shan. Inference without interference: Disaggregate llm inference for mixed downstream workloads, 2024.

[26] Y. Huang, Y. Cheng, A. Bapna, O. Firat, M. X. Chen, D. Chen, H. Lee, J. Ngiam, Q. V. Le, Y. Wu, and Z. Chen. *GPipe: efficient training of giant neural networks using pipeline parallelism*. Curran Associates Inc., Red Hook, NY, USA, 2019.

[27] Z. Jiang, H. Lin, Y. Zhong, Q. Huang, Y. Chen, Z. Zhang, Y. Peng, X. Li, C. Xie, S. Nong, Y. Jia, S. He, H. Chen, Z. Bai, Q. Hou, S. Yan, D. Zhou, Y. Sheng, Z. Jiang, H. Xu, H. Wei, Z. Zhang, P. Nie, L. Zou, S. Zhao, L. Xiang, Z. Liu, Z. Li, X. Jia, J. Ye, X. Jin, and X. Liu. MegaScale: Scaling large language model training to more than 10,000 GPUs. In *21st USENIX Symposium on Networked Systems Design and Implementation (NSDI 24)*, pages 745–760, Santa Clara, CA, Apr. 2024. USENIX Association.

[28] W. Kwon, Z. Li, S. Zhuang, Y. Sheng, L. Zheng, C. H. Yu, J. Gonzalez, H. Zhang, and I. Stoica. Efficient memory management for large language model serving with pagedattention. In *Proceedings of the 29th Symposium on Operating Systems Principles*, SOSP '23, page 611–626, New York, NY, USA, 2023. Association for Computing Machinery.

[29] K. Lee, D. Ippolito, A. Nystrom, C. Zhang, D. Eck, C. Callison-Burch, and N. Carlini. Deduplicating training data makes language models better. In S. Muresan, P. Nakov, and A. Villavicencio, editors, *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), ACL 2022, Dublin, Ireland, May 22-27, 2022*, pages 8424–8445. Association for Computational Linguistics, 2022.

[30] A. Lees, V. Q. Tran, Y. Tay, J. Sorensen, J. P. Gupta, D. Metzler, and L. Vasserman. A new generation of perspective API: efficient multilingual character-level transformers. In A. Zhang and H. Rangwala, editors, *KDD '22: The 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining, Washington, DC, USA, August 14 - 18, 2022*, pages 3197–3207. ACM, 2022.

[31] G. Li, X. Zhou, and X. Zhao. LLM for data management. *Proc. VLDB Endow.*, 17(12):4213–4216, 2024.

[32] S. Li, H. Liu, Z. Bian, J. Fang, H. Huang, Y. Liu, B. Wang, and Y. You. Colossal-ai: A unified deep learning system for large-scale parallel training. In *Proceedings of the 52nd International Conference on Parallel Processing*, ICPP '23, page 766–775, New York, NY, USA, 2023. Association for Computing Machinery.

[33] X. Lian, S. A. Jacobs, L. Kurilenko, M. Tanaka, S. Bekman, O. Ruwase, and M. Zhang. Universal checkpointing: Efficient and flexible checkpointing for large scale distributed training, 2024.

[34] Y. Lin, M. Hulsebos, R. Ma, S. Shankar, S. Zeighami, A. G. Parameswaran, and E. Wu. Towards accurate and efficient document analytics with large language models. *CoRR*, abs/2405.04674, 2024.

[35] C. Liu, M. Russo, M. J. Cafarella, L. Cao, P. B. Chen, Z. Chen, M. J. Franklin, T. Kraska, S. Madden, and G. Vitagliano. A declarative system for optimizing AI workloads. *CoRR*, abs/2405.14696, 2024.

[36] S. Madden, M. J. Cafarella, M. J. Franklin, and T. Kraska. Databases unbound: Querying all of the world's bytes with AI. *Proc. VLDB Endow.*, 17(12):4546–4554, 2024.

[37] A. Maurya, R. Underwood, M. M. Rafique, F. Cappello, and B. Nicolae. Datastates-llm: Lazy asynchronous checkpointing for large language models. In *Proceedings of the 33rd International Symposium on High-Performance Parallel and Distributed Computing*, HPDC '24, page 227–239, New York, NY, USA, 2024. Association for Computing Machinery.

[38] J. Mohan, A. Phanishayee, and V. Chidambaram. CheckFreq: Frequent, Fine-Grained DNN checkpointing. In *19th USENIX Conference on File and Storage Technologies (FAST 21)*, pages 203–216. USENIX Association, Feb. 2021.

[39] N. Muennighoff, A. M. Rush, B. Barak, T. L. Scao, N. Tazi, A. Piktus, S. Pyysalo, T. Wolf, and C. A. Raffel. Scaling data-constrained language models. In A. Oh, T. Naumann, A. Globerson, K. Saenko, M. Hardt, and S. Levine, editors, *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 - 16, 2023*, 2023.

[40] D. Narayanan, M. Shoeybi, J. Casper, P. LeGresley, M. Patwary, V. Korthikanti, D. Vainbrand, P. Kashinkunti, J. Bernauer, B. Catanzaro, A. Phanishayee, and M. Zaharia. Efficient large-scale language model training on gpu clusters using megatron-lm. In *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*, SC '21, New York, NY, USA, 2021. Association for Computing Machinery.

[41] E. Nijkamp, B. Pang, H. Hayashi, L. Tu, H. Wang, Y. Zhou, S. Savarese, and C. Xiong. Codegen: An open large language model for code with multi-turn program synthesis. In *The Eleventh International Conference on Learning Representations, ICLR 2023, Kigali, Rwanda, May 1-5, 2023*. OpenReview.net, 2023.

[42] NVIDIA. Tensorrt-llm. https://github.com/NVIDIA/TensorRT-LLM, 2024.

[43] L. Patel, S. Jha, C. Guestrin, and M. Zaharia. LOTUS: enabling semantic queries with llms over tables of unstructured and structured data. *CoRR*, abs/2407.11418, 2024.

[44] P. Patel, E. Choukse, C. Zhang, A. Shah, I. Goiri, S. Maleki, and R. Bianchini. Splitwise: Efficient generative llm inference using phase splitting. In *2024 ACM/IEEE 51st Annual International Symposium on Computer Architecture (ISCA)*, pages 118–132, 2024.

[45] R. Qin, Z. Li, W. He, M. Zhang, Y. Wu, W. Zheng, and X. Xu. Mooncake: A kvcache-centric disaggregated architecture for llm serving, 2024.

[46] C. Raffel, N. Shazeer, A. Roberts, K. Lee, S. Narang, M. Matena, Y. Zhou, W. Li, and P. J. Liu. Exploring the limits of transfer learning with a unified text-to-text transformer. *J. Mach. Learn. Res.*, 21:140:1–140:67, 2020.

[47] S. Rajbhandari, J. Rasley, O. Ruwase, and Y. He. Zero: memory optimizations toward training trillion parameter models. In *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*, SC '20. IEEE Press, 2020.

[48] D. Team. Deepspeed. https://github.com/microsoft/DeepSpeed, 2024.

[49] H. F. Team. Safetensors. https://huggingface.co/docs/safetensors/index, 2024.

[50] N. M. Team. dist_checkpointing package. https://docs.nvidia.com/megatron-core/developer-guide/latest/api-guide/dist_checkpointing.html, 2024.

[51] P. Team. Getting started with distributed checkpoint (dcp). https://pytorch.org/tutorials/recipes/distributed_checkpoint_recipe.html, 2023.

[52] H. Touvron, T. Lavril, G. Izacard, X. Martinet, M. Lachaux, T. Lacroix, B. Rozière, N. Goyal, E. Hambro, F. Azhar, A. Rodriguez, A. Joulin, E. Grave, and G. Lample. Llama: Open and efficient foundation language models. *CoRR*, abs/2302.13971, 2023.

[53] M. Urban and C. Binnig. CAESURA: language models as multi-modal query planners. In *14th Conference on Innovative Data Systems Research, CIDR 2024, Chaminade, HI, USA, January 14-17, 2024*. www.cidrdb.org, 2024.

[54] M. Urban and C. Binnig. Efficient learned query execution over text and tables [technical report]. *arXiv preprint arXiv:2410.22522*, 2024.

[55] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. u. Kaiser, and I. Polosukhin. Attention is all you need. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017.

[56] B. Wan, M. Han, Y. Sheng, Y. Peng, H. Lin, M. Zhang, Z. Lai, M. Yu, J. Zhang, Z. Song, X. Liu, and C. Wu. Bytecheckpoint: A unified checkpointing system for large foundation model development, 2024.

[57] J. Wang, C. Chai, N. Tang, J. Liu, and G. Li. Coresets over multiple tables for feature-rich and data-efficient machine learning. *Proc. VLDB Endow.*, 16(1):64–76, 2022.

[58] J. Wang and J. Feng. Unify: An unstructured data analytics system. In *ICDE 2025*, 2025.

[59] J. Wang and G. Li. Aop: Automated and interactive llm pipeline orchestration for answering complex queries. In *CIDR 2025*, 2025.

[60] J. Wang, G. Li, and J. Feng. idatalake: An llm-powered analytics system on data lakes. *Data Engineering*, page 57.

[61] Y. Wang, X. Kang, S. Shi, X. He, Z. Tang, X. Pan, Y. Zheng, X. Wu, A. C. Zhou, B. He, and X. Chu. Fault-tolerant hybrid-parallel training at scale with reliable and efficient in-memory checkpointing, 2024.

[62] A. Wettig, A. Gupta, S. Malik, and D. Chen. Qurating: Selecting high-quality data for training language models. In *Forty-first International Conference on Machine Learning, ICML 2024, Vienna, Austria, July 21-27, 2024*. OpenReview.net, 2024.

[63] M. Xia, S. Malladi, S. Gururangan, S. Arora, and D. Chen. LESS: selecting influential data for targeted instruction tuning. In *Forty-first International Conference on Machine Learning, ICML 2024, Vienna, Austria, July 21-27, 2024*. OpenReview.net, 2024.

[64] S. M. Xie, S. Santurkar, T. Ma, and P. Liang. Data selection for language models via importance resampling. In A. Oh, T. Naumann, A. Globerson, K. Saenko, M. Hardt, and S. Levine, editors, *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 - 16, 2023*, 2023.

[65] S. Yao, J. Zhao, D. Yu, N. Du, and et al. React: Synergizing reasoning and acting in language models. In *ICLR*, 2023.

[66] G.-I. Yu, J. S. Jeong, G.-W. Kim, S. Kim, and B.-G. Chun. Orca: A distributed serving system for Transformer-Based generative models. In *16th USENIX Symposium on Operating Systems Design and Implementation (OSDI 22)*, pages 521–538, Carlsbad, CA, July 2022. USENIX Association.

[67] C. Zhang, H. Zhong, K. Zhang, C. Chai, R. Wang, X. Zhuang, T. Bai, J. Qiu, L. Cao, Y. Yuan, G. Wang, and C. He. Harnessing diversity for important data selection in pretraining large language models. *CoRR*, abs/2409.16986, 2024.

[68] Y. Zhao, A. Gu, R. Varma, L. Luo, C.-C. Huang, M. Xu, L. Wright, H. Shojanazeri, M. Ott, S. Shleifer, A. Desmaison, C. Balioglu, P. Damania, B. Nguyen, G. Chauhan, Y. Hao, A. Mathews, and S. Li. Pytorch fsdp: Experiences on scaling fully sharded data parallel. *Proc. VLDB Endow.*, 16(12):3848–3860, Aug. 2023.

[69] Y. Zhong, S. Liu, J. Chen, J. Hu, Y. Zhu, X. Liu, X. Jin, and H. Zhang. DistServe: Disaggregating prefill and decoding for goodput-optimized large language model serving. In *18th USENIX Symposium on Operating Systems Design and Implementation (OSDI 24)*, pages 193–210, Santa Clara, CA, July 2024. USENIX Association.