WASP Software Engineering Assignment

Sonia-Florina Horchidan
sfhor@kth.se
KTH Royal Institute of Technology

August 2023

1 Research Topic

Knowledge graphs currently pose two significant challenges: massive scale and sparsity. Dealing with the former involves addressing slow response times for intricate queries plagued by slow random data accesses, particularly when traversing deep into the graph. The latter arises from missing connections and characteristics in graphs representing real-world information, resulting in incomplete outcomes when analyzing solely explicitly stored data. In my research, I intend to develop an innovative graph database architecture that harnesses the potential of Graph Machine Learning to empower graph queries with predictive capabilities while still providing timely results for complex queries. The visionary graph database architecture is centered around the concept of hybrid operators and optimizations.

The proposed architecture restructures the computation of queries by shifting away from direct data accesses and costly pointer-chasing for traversals. Instead, it adopts the approach of querying learned latent spaces through ML inference. This relies on modern graph representation learning capabilities to approximate intricate graph queries using latent representations, known as embeddings. In this envisioned design, specific database operators are envisioned with two distinct operating modes: (1) DB operators, defined as conventional database operations conducted on explicitly stored data, and (2) ML operators, which replace raw data accesses with Machine Learning inference calls. The latter does not necessitate graph data accesses to conclude a computation; instead, it produces likely results by performing matrix multiplications using a given query input and a trained model. The system is, therefore, responsible for identifying the most suitable query plan, considering the operators' duality, and attempting to minimize the computational costs.

Incorporating approximated outcomes generated by ML inference in the database's result set may present a risk, potentially impeding the solution's acceptance due to skepticism surrounding the query answers. My research explores the viability of implementing uncertainty estimation techniques to address this concern. These techniques aim to establish bounds on the maximum allowable error on a per-query basis, thereby empowering users with control over the predicted outcomes. By doing so, we aim to alleviate doubts and enhance the acceptance of the solution.

2 Course Concepts Related to the Chosen Research Topic

This section discusses principles presented in the course and delves into their relation with my chosen research topic. The concepts which belong to Robert's lectures are marked with (R), while the ones belonging to guest lectures are marked with (G).

Verification versus Validation (R). The distinction between validation and verification holds critical significance in designing a new graph database. Verification ensures the database adheres to the specified requirements and functionalities, confirming the implementation is correct and error-free. On the other hand, validation assesses whether the database meets the actual needs of its users and aligns with their intended use cases. As a researcher, I personally find that verification is, most often, receiving more attention, as research usually involves building a prototype that meets functional and non-functional requirements. However, confirming the need for such a solution in production-grade environments is equally important in my particular research field. I firmly believe that, regardless of design and implementation mastery, efforts to build a tool are futile unless met with equal need and enthusiasm from users. Therefore, the differentiation between validation and verification allows for a more comprehensive evaluation, leading, in my case, to developing a graph database that is not only technically sound but also fits seamlessly into users' workflows, ultimately increasing its effectiveness and adoption in practical applications.

Cognitive biases in AI tools (R). The issue of cognitive biases in AI tools is also of note for my research. As my envisioned graph database relies on ML models to answer queries, one possible design idea is to allow database administrators to train or fine-tune the models using proprietary datasets. However, as users gain more control over the training process, they are susceptible to introducing their own biases unknowingly into the models. These biases can stem from inherent human prejudices, limited perspectives, or skewed datasets used for training. The democratization of AI brings the responsibility of ensuring that users are aware of these potential biases and actively work to mitigate them. Implementing mechanisms for bias detection, model explainability, and clear guidelines for ethical data is paramount to fostering AI tools that are fair, unbiased, and accountable. Nonetheless, a graph database is a universal tool that can be used in a wide range of applications and use cases without many constraints. Therefore, forcing users to seek biases in their data is not feasible, as they need to be aware that such a phenomenon may occur and actively pursue this endeavor. From a database design perspective, I believe the only solutions are ensuring that the database management system provides intuitive functionalities out-of-the-box for bias detection and mitigation and offering training to increase awareness on detecting such biases.

Boundary Value Testing (G). In my research, ML models are used to process and make decisions on data stored in the database. The input data for ML models often come from the database, and the output of the ML model is returned to the user. During the development and testing, boundary value testing can ensure that the ML models and the system as a whole handle various types of input data correctly. If the models are sensitive to certain input ranges or if the database has constraints on the input data, boundary value testing helps identify potential issues. Similarly to the discussion above, when designing such a database, I believe it is essential to include tools for database administrators to perform such tests and be aware of these corner cases.

AI for Software Engineering (G). Albeit belonging mostly to the category of Software Engineering for AI, my research project particularly includes techniques for query planning and optimization, a topic that has seen massive efforts and interest from the data management community and which can be improved by AI-assisted techniques. Graph databases often store vast and interconnected datasets, making query planning complex. Given a graph query, the system enumerates all the possible logical execution plans and estimates their cost. The logical plan with the minimum estimated cost is then chosen to be executed. Choosing an unsuitable plan can lead to massive increases in query latency and computational costs. In recent years, AI techniques have shown tremendous potential to analyze query patterns, predict user preferences, and learn from historical query performances, proving valuable to query plans' cost. Moreover, AI techniques can enable adaptive query processing, where the database system can dynamically adjust its query optimization strategies based on changing workloads and data distribution. As a result, AI-enhanced query optimization not only accelerates response times but also improves the overall performance and scalability of graph databases, making them better equipped to handle the demands of modern applications and data-intensive scenarios, therefore showcasing an example where AI for Software Engineering is applied in the field of data management systems.

3 Software Engineering Topics and AI/ML

3.1 Maintenance and Evolution

As my research lies at the intersection of DBMS and AI/ML Engineering, the issue of maintenance and evolution can be analyzed from two different perspectives.

3.1.1 Maintenance of Database Management Systems

Software maintenance is crucial for ensuring the database's long-term viability, as it ensures the continuous process focused on ensuring reliability, efficiency, and security throughout its operational lifespan. Any software deployed in production must meet changing user requirements and adapt to the newest hardware to maintain a competitive edge. However, database software is not different from traditional software and, therefore, involves standard tasks such as regular patching introducing new features, or addressing bugs and security vulnerabilities. Next, we discuss evolution. Traditional relational databases require predefined schemas, which are rarely updated. They allow for evolution mostly in terms of appending new data that comply with the schema, and schema changes usually require manual interventions through expensive table alternations. Unlike such traditional approaches, graph databases also target evolution at the level of the structure of the data, particularly when modeling real concepts and knowledge that evolves naturally over time. Research in dynamic schema evolution techniques, which allow seamless updates to the database structure without losing existing data or disrupting applications, is especially relevant.

3.1.2 Maintenance and Evolution for Pre-trained ML Models

Furthermore, when analyzing my research topic from the angle of powering a database with ML, major challenges arise from deploying ML models. First, deploying ML models in production is rarely a static, one-off process, as it requires frequent evaluations, parameter tuning, monitoring, and updates, alongside a scalable and efficient serving infrastructure [4]. The complexity is especially daunting when the data at hand is in graph format since utilizing models on real-world data can grow problematic when the underlying data distribution changes. Graphs are especially dynamic, and models trained on such data suffer from concept drift. Therefore, they can quickly become obsolete if data is continuously ingested and the models are not updated frequently. In this case, mechanisms such as continuous learning and frequent re-training should be considered, but with the caveat that performing such operations should keep the overhead introduced in the query system to a minimum. Specifically for my design, schema changes, and data dynamism should be treated as first-class citizens. To this end, in my research, I identify some opportunities to ensure that the database can still be queried even if the underlying models take more time to get updated. If the database operates both based on traversals of explicit data and of ML inference calls, and the underlying models are not yet updated to respond to queries that touch fresh data points, the database can potentially fall back to explicit querying only to cater to such queries. In this case, the system will behave like a traditional database, providing only exact but incomplete answers. In this case, the question is whether waiting for the models to train and offer inferred results would still provide an acceleration against explicit querying using DB operators only. However, using such a mechanism, the graph database can ensure that schema and data evolution is accounted for.

3.2 Sustainability

I will now explore the concept of sustainability in software development, with a focus on energy efficiency and environmentally conscious practices.

The field of data management research has made significant progress, with various successful methods and systems deployed in production-like environments. These advancements prioritize computational efficiency, striving for solutions that require minimal computation power. By minimizing computational demands, they also contribute to reducing the carbon footprint. However, there is room for further improvement by embracing fast and cost-effective approximate solutions when exact query answers are not crucial. This approach, which I seek to promote in my research, has been explored in the past but faced skepticism from the industry due to a lack of robust approximation guarantees.

On another note, most of the machine learning (ML) research efforts have been directed toward achieving the best qualitative results without much consideration for their environmental impact. The focus has been on improving the performance of ML models, usually assuming computational power is limitless. This has resulted in a lack of transparency regarding the true energy consumption behind newly published breakthroughs. A study by Patterson et al. in 2021 advocates introducing carbon emissions as a standard metric in industry and academia to address this issue [3].

As my work sits at the intersection of the two research areas, both problems must be addressed and discussed. My research aims to enable energy-efficient graph querying by providing reliable approximate results. Although this goal complies, at first glance, with targets of sustainable software, it is essential to discuss the disadvantages it can bring. Employing Machine Learning models that help answer queries on massive graphs encourages dropping the need for exact results when approximations are enough to promote lower computational costs. However, this project requires training possibly large models on considerable amounts of data, which, depending on the architecture of the chosen models, can be a computationally expensive problem. Therefore, these overheads can potentially negate the advantage of lightweight query approximation, especially if the models are trained frequently. Therefore, special attention must be paid to monitoring the models. Furthermore, ML-oriented accelerators such as TPUs can to further reduce the carbon footprint [3].

4 Future trends and directions of Software Engineering

As ML models become more sophisticated and accessible, the ”Software Engineering for ML” movement, to which my research contributes mostly, is bound to gain traction. Democratizing the usage of ML models for querying capabilities is essential for ensuring that these powerful technologies are accessible to a broader range of developers rather than limited to ML field experts. Therefore, my research project is, in my opinion, relevant to the current context.

I will now discuss crucial concepts and technologies that will, in my view, be significant in the coming years for ML/AI Engineering. Lastly, I will address a key question related to the emergence of efficient code-generation technologies.

Cloud Computing Driving Acceleration and Accessibility. Cloud computing is poised to play a pivotal role in shaping the future of Software Engineering in the context of AI/ML. The rise of cloud-based infrastructures has already facilitated the accelerated growth of Machine Learning (ML) technologies, making them more accessible to a broader audience. As cloud services continue to evolve, we can expect even more seamless integration of ML capabilities into various software applications. This integration will likely lead to a rapid shift towards real-time predictions and data-driven decision-making, even more than in the past. Consequently, I believe businesses and industries will harness AI/ML algorithms to make faster, more informed decisions based on real-time data streams, and providing tools to facilitate this flow will be of utmost importance in the next 5-10 years.

Regulation and Convergence of SE and ML Practices. The proliferation of AI/ML technologies has introduced new challenges for Software Engineering. Managing massive datasets, alongside the code and algorithms developed, necessitates novel methodologies and regulations. Software Engineering has not encountered such a scale of data-centric development before, and, in my view, it must adapt to this changing landscape, which comes with its own set of difficulties, such as data quality, integrity, privacy, or security. Moreover, as ML models become the norm in various industries, we will probably witness some convergence between traditional Software Engineering methods and ML practices, as it was the case documented in a recent study conducted with software teams at Microsoft, where the teams deploying ML models made use of transferred software engineering processes and standards to integrate ML workflows seamlessly [1]. The study showed a clear movement toward merging the two processes. Furthermore, another large-scale survey conducted at Microsoft showed that more than

60% of the interviewed ML specialists and data scientists were initially hired as software engineers and only later transitioned into ML-centric roles, showing the need in the industry for inter-disciplinary professionals [2]. Therefore, in the future, I foresee that SE professionals will need to develop an understanding of ML concepts, and ML practitioners will need to adopt software engineering principles to ensure robust and scalable ML applications.

Will ML Subsume SE? Lastly, I think we should address the significant question that looms over the future of Software Engineering with the rise of ML techniques - will ML eventually subsume SE, rendering human software engineers obsolete? While tools like Large Language Models and Github Copilot demonstrate considerable capabilities in generating code, they also have limitations, amongst which we can enumerate the lack of context awareness, inability to handle ambiguity, unforeseen bias, or difficulty in supporting new, emerging technologies. It is reasonable to assume that repetitive and already-standardized tasks may be automated using ML techniques, but more complex areas, such as system architecture design or human-centric user experience design, may still require human ingenuity and expertise. I believe software engineers will continue to play a crucial role in conceptualizing, designing, and implementing sophisticated and innovative software solutions that demand human creativity and specialized domain knowledge.

References

[1] Saleema Amershi, Andrew Begel, Christian Bird, Robert DeLine, Harald C. Gall, Ece Kamar, Nachiappan Nagappan, Besmira Nushi, and Thomas Zimmermann. Software engineering for machine learning: a case study. In Helen Sharp and Mike Whalen, editors, *Proceedings of the 41st International Conference on Software Engineering: Software Engineering in Practice, ICSE (SEIP) 2019, Montreal, QC, Canada, May 25-31, 2019*, pages 291–300. IEEE / ACM, 2019.

[2] Miryung Kim, Thomas Zimmermann, Robert DeLine, and Andrew Begel. Data scientists in software teams: State of the art and challenges. *IEEE Trans. Software Eng.*, 44(11):1024–1038, 2018.

[3] David A. Patterson, Joseph Gonzalez, Quoc V. Le, Chen Liang, Lluis-Miquel Munguia, Daniel Rothchild, David R. So, Maud Texier, and Jeff Dean. Carbon emissions and large neural network training. *CoRR*, abs/2104.10350, 2021.

[4] D. Sculley, Gary Holt, Daniel Golovin, Eugene Davydov, Todd Phillips, Dietmar Ebner, Vinay Chaudhary, Michael Young, Jean-François Crespo, and Dan Dennison. Hidden technical debt in machine learning systems. In Corinna Cortes, Neil D. Lawrence, Daniel D. Lee, Masashi Sugiyama, and Roman Garnett, editors, *Advances in Neural Information Processing Systems 28: Annual Conference on Neural Information Processing Systems 2015, December 7-12, 2015, Montreal, Quebec, Canada*, pages 2503–2511, 2015.