Synergies between Graph Data Management and Machine Learning in Graph Data Pipeline

Arijit Khan
Aalborg University
Aalborg, Denmark
arijitk@cs.aau.dk

***Abstract*—Graphs are popular mathematical tools to model data with relations, such as the Web, social and biological networks, financial transactions, and knowledge bases. Machine learning and recently, deep learning over graphs becomes prevalent. In modern data science applications, complex data move through various processes involved in machine learning to generate the final predictive output, thereby creating a data pipeline consisting of graph data extraction, acquisition, and cleaning, graph embedding, machine learning training and inference, downstream tasks, explainability, and adding human in-the-loop, as depicted in Figure 1. We investigate how graph data management, which deals with effective, efficient, scalable, and user-friendly systems and algorithms for storing, processing, and analyzing large volumes of heterogeneous and complex graphs, could benefit from graph machine learning and vice versa, over the end-to-end graph data pipeline. We shall emphasize on (1) how graph data management helps in graph machine learning, e.g., in scalable graph embedding and designing user-friendly explainability methods; and (2) how graph machine learning helps in graph data management, e.g., in question answering over knowledge graphs.**

***Index Terms*—graph neural networks, graph embedding, explainable AI, knowledge graphs, question answering**

I. Graph Data Management for Graph Machine Learning

Scalable Graph Embedding and Graph Neural Networks Learning. Effective data management algorithms and systems facilitate in large-scale graph embedding and scalable training of graph neural networks (GNNs). Embedding of billion-scale graphs, e.g., in Alibaba, Facebook, Microsoft Academic, Pinterest at scale has recently become ubiquitous. Graph embedding generates low-dimensional vector representations of graph nodes, edges, and entire graphs for downstream machine learning (ML) tasks. Graph embedding techniques can be categorized as matrix-factorization based, random-walk methods, and neural approaches. Representative works include PANE that enables scalable and attributed networks embedding by measuring node-attribute affinity with random walks, embedding computation via joint matrix factorization, and using multi-core parallelization [1]. SketchNE designs a fast, memory-efficient, and scalable graph embedding method via sparse-sign matrix, single-pass singular value decomposition (SVD), fast eigen-decomposition, and shared-memory architecture for a single, CPU-only machine [2]. DistGER employs information-oriented distributed random walks and distributed Skip-Gram learning for scalable graph embedding

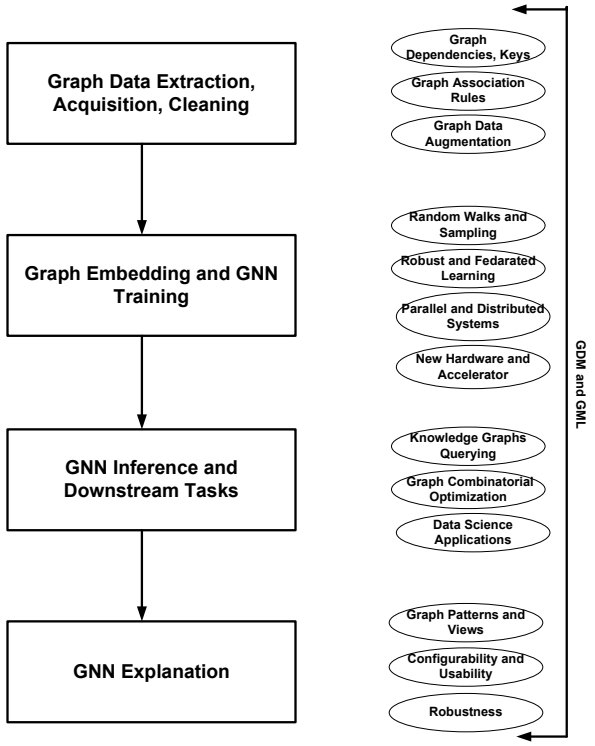

Fig. 1. Graph data pipeline consisting of four stages: (a) graph data extraction, acquisition, and cleaning, (b) graph embedding and graph neural network (GNN) training, (c) GNN inference for downstream tasks, and (d) GNN explanation. To depict the synergy between graph machine learning (GML) and graph data management (GDM) at each stage of the graph data pipeline, we specify critical modules on the right-hand side.

[3]. GraphVite [4] proposes a high-performance CPU-GPU hybrid architecture by co-optimizing the algorithm and the system, simultaneously performing graph random walks on CPUs and embedding training on GPUs.

Scaling GNN training has received more attention with emphasis on smart sampling strategies, parallel and distributed systems, new hardware and accelerator, computation and communication paradigms [5]. Various sampling strategies, e.g., node-wise sampling, layer-wise sampling, and graph-wise sampling are developed. XGNN [6] designs a multi-GPU GNN training system to fully utilize GPU and CPU memory and high-speed interconnects. ReGNN develops a ReRAM-based architecture for GNN acceleration [7]. DUCATI introduces a dual-cache system to better utilize spare GPU memory and accelerate mini-batch generation during GNN training [8]. Facebook has designed the distributed multi-relations based

graph embedding system PyTorch-BigGraph [9], and Amazon has developed the distributed graph neural networks-based system DistDGL [10]. ByteGNN adopts mini-batch sampling, two-level scheduling, and effective graph partitioning for high parallelism and better resource utilization [11].

Usable Graph Neural Network Explainability Methods. Explainability methods for GNNs discover important nodes, edges, subgraphs, and their features that are influential for GNN outcomes, and are gaining increasing attention [12]. Explainability is crucial in developing and deploying "black-box" deep neural models for interdisciplinary applications, while ensuring transparency and reliability.

Existing GNN explainability approaches derive explanations for an individual instance or a certain class label. These methods do not provide targeted, user-friendly, interactive, and configurable explanations for multiple class labels of interest [14]. Such approaches may also generate large-size or redundant explanations, hence are not easily comprehensible. Moreover, these explanations are not easily queryable, making it difficult for domain experts to understand a GNN's outcome by bridging domain knowledge with the GNN's decision making process. Finally, small changes to input graphs can drastically update GNN results, as well as its explainability; hindering the deployment of neural models and explainability methods in safety-critical applications.

We demonstrate how graph data management (GDM) can facilitate graph machine learning (GML) in regards to GNN explanations. **(1)** We discuss the usage of graph views and graph patterns to generate user-friendly explanations by bridging domain knowledge with GNN results. Furthermore, we present solutions that extract explanations for GNNs in a concise and configurable manner, tuned for multiple class labels of interest [15]. **(2)** We discuss algorithms to make GNN explanations robust to small changes in graphs, thus generating critical, invariant structures across a set of similar graphs [16].

II. Graph Machine Learning for Graph Data Management

Effective Knowledge Graphs Question Answering. Query processing is the bread-and-butter for the data management community. Knowledge graph (KG) is a graph-based data model to store facts as ⟨subject, predicate, object⟩ triples, or as a large-scale graph having nodes (subjects and objects) and edges (predicates) [17]. Querying KGs is difficult due to their massive volume, heterogeneity, and incompleteness [18]. Due to schema-flexibility, the same kind of information can be stored in many diverse ways in a KG. A user seeking relevant information must formulate the query in different ways so to cover all possible schemas. Hence, the user requires to have full knowledge of the vocabulary and the underlying schemas defined in the KG, which is challenging [19].

Machine learning-based query answering can identify missing relations from incomplete KGs. Natural language queries (NLQs) are either semantically parsed to structured SPARQL queries over KGs using neural approaches, or are processed in an end-to-end manner using sequential models. KG embedding methods can identify approximate and semantically relevant answers w.r.t. users' queries [20], [21]. Recently, large language models (LLMs) are becoming mainstream to query knowledge graphs through retrieval augmented methods.

III. Acknowledgement

This talk is based on a sequence of recent works [3], [12], [14]–[16], [18]–[21]. Arijit Khan acknowledges support from the Novo Nordisk Foundation grant NNF22OC0072415.

References

[1] R. Yang, J. Shi, X. Xiao, Y. Yang, J. Liu, and S. S. Bhowmick, "Scaling attributed network embedding to massive graphs," PVLDB 14, 1 (2020), 37–49.

[2] Y. Xie, Y. Dong, J. Qiu, W. Yu, X. Feng, and J. Tang, "SketchNE: Embedding billion-scale networks accurately in one hour," IEEE Trans. Knowl. Data Eng. 35, 10 (2023), 10666–10680.

[3] P. Fang, A. Khan, S. Luo, F. Wang, D. Feng, Z. Li, W. Yin, and Y. Cao, "Distributed graph embedding with information-oriented random walks," PVLDB 16, 7 (2023), 1643–1656.

[4] Z. Zhu, S. Xu, J. Tang, and M. Qu, "GraphVite: A High-Performance CPU-GPU Hybrid System for Node Embedding," WWW (2019), 2494–2504.

[5] Y. Shao, H. Li, X. Gu, H. Yin, Y. Li, X. Miao, W. Zhang, B. Cui, and L. Chen, "Distributed graph neural network training: A survey," ACM Computing Surveys (2024).

[6] D. Tang, J. Wang, R. Chen, L. Wang, W. Yu, J. Zhou, and K. Li, "XGNN: Boosting multi-GPU GNN training via global GNN memorystore," PVLDB 17, 5 (2024), 1105–1118.

[7] C. Liu, H. Liu, H. Jin, X. Liao, Y. Zhang, Z. Duan, J. Xu, and H. Li, "ReGNN: a ReRAM-based heterogeneous architecture for general graph neural networks," DAC (2022), 469–474.

[8] X. Zhang, Y. Shen, Y. Shao, and L. Chen, "DUCATI: A dual-cache training system for graph neural networks on giant graphs with the GPU," Proc. ACM Manag. Data 1, 2 (2023), 1–24.

[9] A. Lerer, L. Wu, J. Shen, T. Lacroix, L. Wehrstedt, A. Bose, and A. Peysakhovich, "Pytorch-BigGraph: A large scale graph embedding system," MLSys (2019).

[10] D. Zheng, C. Ma, M. Wang, J. Zhou, Q. Su, X. Song, Q. Gan, Z. Zhang, and G. Karypis, "DistDGL: Distributed graph neural network training for billion-scale graphs," IA3 (2020), 36–44.

[11] C. Zheng, H. Chen, Y. Cheng, Z. Song, Y. Wu, C. Li, J. Cheng, H. Yang, and S. Zhang, "ByteGNN: Efficient graph neural network training at large scale," PVLDB 15, 6 (2022), 1228–1242.

[12] A. Khan and E. B. Mobaraki, "Interpretability methods for graph neural networks," DSAA (2023), 1–4.

[13] H. Yuan, H. Yu, S. Gui, and S. Ji, "Explainability in graph neural networks: A taxonomic survey," IEEE Transactions on Pattern Analysis and Machine Intelligence, 45, 5 (2023), 5782–5799.

[14] T. Chen, D. Qiu, Y. Wu, A. Khan, X. Ke, and Y. Gao, "User-friendly, interactive, and configurable explanations for graph neural networks with graph views," SIGMOD (2024).

[15] T. Chen, D. Qiu, Y. Wu, A. Khan, X. Ke, and Y. Gao, "View-based explanations for graph neural networks," SIGMOD (2024).

[16] D. Qiu, M. Wang, A. Khan, and Y. Wu, "Generating robust counterfactual witnesses for graph neural networks," ICDE (2024).

[17] G. Weikum, X. L. Dong, S. Razniewski, and F. M. Suchanek, "Machine knowledge: Creation and curation of comprehensive knowledge bases," Found. Trends Databases 10, 2-4 (2021), 108–490.

[18] A. Khan, "Knowledge graphs querying," SIGMOD Rec. 52, 2 (2023), 18–29.

[19] Y. Wang, A. Khan, X. Xu, S. Ye, S. Pan, and Y. Zhou, "Approximate and interactive processing of aggregate queries on knowledge graphs: A demonstration," CIKM (2022), 5034–5038.

[20] Y. Wang, A. Khan, T. Wu, J. Jin, and H. Yan, "Semantic guided and response times bounded top-k similarity search over knowledge graphs," ICDE (2020), 445–456.

[21] Y. Wang, A. Khan, X. Xu, J. Jin, and Q. Hong, T. Fu, "Aggregate queries on knowledge graphs: Fast approximation with semantic-aware sampling," ICDE (2022), 2914–2927.