A scalable platform to build the data layer of knowledge graph AI

Lucas Vittor*
Department of Biomedical Informatics
Harvard Medical School
Boston, MA 02115
lvvittor@gmail.com

Iñaki Arango*
Department of Biomedical Informatics
Harvard Medical School
Boston, MA 02115
inakiarango@college.harvard.edu

Ayush Noori*
Department of Biomedical Informatics
Harvard Medical School
Boston, MA 02115
anoori@college.harvard.edu

Joaquin Polonuer*
Department of Biomedical Informatics
Harvard Medical School
Boston, MA 02115
jtpolonuer@gmail.com

Marinka Zitnik†
Department of Biomedical Informatics
Harvard Medical School
Boston, MA 02115
marinka@hms.harvard.edu

Abstract

Knowledge graphs (KGs) underpin modern graph AI, from retrieval-augmented generation to large graph-language models. However, pipelines to construct and maintain KGs remain irreproducible and challenging to scale. We introduce OPTIMUS, an opinionated platform for building large-scale KGs with an emphasis on reproducibility and extensibility. OPTIMUS adopts a data lake-inspired medallion architecture; enforces schema contracts and identifier harmonization; and produces machine learning-ready KG exports. In benchmarking experiments, OPTIMUS constructed a biomedical KG with 192,307 nodes, 21.5M edges, and 88.6M properties from 47 heterogeneous datasets. Parallelized execution reduced wall clock build time by 56.5% compared to sequential execution (143.6 s vs. 62.4 s), while throughput per edge improved as the graph scaled. These results demonstrate that OPTIMUS enables efficient, reproducible, and scalable KG construction, strengthening the data layer of knowledge-grounded AI.

1 Introduction

Knowledge graphs (KGs) are relational databases that use a graph-based data model to encode knowledge-informed interactions between different objects [1, 2]. Formally, a KG is defined by a set of nodes as well as a set of edges that describe relationships between the nodes. In modern heterogeneous KGs, nodes and edges have different types, and may also contain extra properties or information [3]. Since 2012, KGs have been utilized across healthcare [4], networking [5], finance

*Equal contribution
†Correspondence: marinka@hms.harvard.edu

39th Conference on Neural Information Processing Systems (NeurIPS 2025) Workshop: New Perspectives in Advancing Graph Machine Learning.

[6], media [7], search [8], cybersecurity [9], and many other applications. In addition to Google, KGs have been developed by companies including Airbnb, Amazon, eBay, Facebook, IBM, LinkedIn, Microsoft, and Uber [2], highlighting the academic and commercial value of the KG data model to organize human knowledge. Over two decades of research in network science and graph artificial intelligence (AI) has developed algorithms that ingest knowledge from KGs [10], enabling advances in recommendation and search [11], knowledge-intensive question answering [12], science [13], medicine [14, 15], cybersecurity, fraud and anomaly detection [16], materials science [17], code intelligence, weather forecasting [18], and many other domains. Importantly, neural scaling laws suggest that the performance of graph AI models improves as the numbers of nodes and edges in the KG training data increase [19–21]. Therefore, methods to construct KGs and thereby create the data layer of graph AI are critical.

Unfortunately, the ecosystem of developer tools and applications surrounding KGs is still in its infancy. Many of the tools that we expect to work with code or tabular data, such as construction frameworks, standard exchange formats, version control, compression methods, collaboration platforms, and programming libraries, are not available for graph data or are outdated and unmaintained. In particular, methods to construct KGs face the following challenges: **(1) heterogeneity:** heterogeneous and siloed data must be integrated across inconsistent formats, semantics, and licenses into a common namespace; **(2) evolution:** identifiers, ontologies, and releases change frequently; and **(3) reproducibility:** traditional construction pipelines rely on custom scripts or manual steps, leading to poor reproducibility and maintenance. There is a need for principled software engineering solutions for KG construction and maintenance that enforce consistency and enable reuse across projects.

To address this gap, we introduce OPTIMUS, an opinionated platform for building large-scale KGs with an emphasis on reproducibility and extensibility. To demonstrate the functionality of OPTIMUS, we focus on biomedical KGs; however, we emphasize that OPTIMUS is domain-agnostic. By providing a modular, configurable approach to assemble KGs, OPTIMUS aims to accelerate graph machine learning research by establishing better data foundations.

2 Related work

We position OPTIMUS relative to prior work on knowledge extraction, end-to-end pipelines for KG construction, and modern AI systems. For a comprehensive treatment of these subjects, we kindly refer the reader to several reviews on the state of KG construction [22–26].

Knowledge extraction. The first step in building a KG is to extract structured knowledge from one or multiple source databases. Early KGs like DBPedia [27], Freebase [28], and Wikidata [29] were often built manually by human experts, via crowdsourced community contributions, or via rule-based structured information extraction from text. These approaches achieved high precision but required significant human effort, were limited in coverage by contributor expertise or the rule set, and proved difficult to scale. Machine learning (ML) methods soon replaced handcrafted rules with statistical models. For example, named entity recognition (NER) methods can convert natural language text into structured triples [30, 31]. More recently, deep learning systems, including graph neural networks [32, 33] and large language models (LLMs) [34–36], have been used to predict new edges and populate KGs automatically. These methods expand coverage but are computationally intensive and can be nondeterministic or error prone. Importantly, knowledge extraction is only one stage of KG construction; KG developers must still address schema mapping, provenance, identifier harmonization, and releases. Both rule-based and ML-based extraction methods are thus complementary to our work, serving as interchangeable processing nodes inside OPTIMUS, which focuses on orchestrating reproducible end-to-end KG construction.

Pipelines for KG construction. After knowledge extraction, the central challenge is to create a reproducible end-to-end pipeline that integrates extracted knowledge from heterogeneous datasets into consistent, queryable graphs. Many domain-specific KGs have been constructed in areas such as social networks [37], e-commerce [38], biomedicine [4, 39], and genomics [40]. These efforts, however, rely on custom scripts and ad hoc transformations, making them difficult to reproduce or extend to new data sources. Some open-source ecosystems attempt to provide reusable infrastructure for KG construction, including, for example, KGTK [41], `metaphactory` [42], BioCypher [43], and KG-Hub [44]. However, these systems often suffer from several drawbacks, including poor ontology mapping, entity resolution, or release management; complex workflows, requiring substantial re-

engineering to add new datasets; and limited scalability, altogether leading to poor reproducibility and extensibility. For example, of 250 KGs surveyed, only eight (3.2%) had publicly available code for KG construction, and only one (0.4%) construction pipeline could be successfully executed by independent evaluators [45]. This indicates a startling lack of reproducible KG construction methods, motivating the need for OPTIMUS.

LLMs and AI agents. Recently, LLMs and AI agents have been used to both store and retrieve knowledge. While parametric LLM memory can implicitly store facts, KGs remain indispensable to improve factuality and mitigate hallucinations [46, 47], enable verifiable provenance [48], rapidly integrate new knowledge [49], and enhance interpretability [50]. Constructed KGs can be integrated with LLMs or agents to improve performance at knowledge-intensive tasks [51–55]; therefore, advances in graph AI are complementary to OPTIMUS.

3 System overview

3.1 Desirable features of KG construction platforms

Optimally, methods for KG construction should achieve the following desiderata [25].

1. **Minimizing manual labor.** The pipeline should balance automation with customization, minimizing manual work while still permitting human-in-the-loop input, for example, for source selection or ontology design.
2. **Scalability.** The construction method should be grounded in a uniform data model (*e.g.*, RDF [56] or LPG [3]) and should scale to many large sources.
3. **Customizable knowledge extraction.** Customizable methods for knowledge extraction, or the identification of structured relationships from unstructured or semi-structured data [25], should be supported per source.
4. **Entity resolution.** Based on underlying ontologies, entity resolution and fusion should be performed across sources to unify multiple identifiers that refer to the same concept or entity.
5. **Quality assurance.** The construction pipeline should include quality assurance checks to confirm that the resulting KG is correct, with traceable provenance of each fact; consistent, with canonical identifiers and no contradictions; timely, with the ability to make updates to reflect changes in source datasets; and both complete and succinct [57].
6. **Metadata management and export.** Finally, there should exist tooling for metadata management, and the KG should be exportable in standard formats used for graph AI (*e.g.*, JSON, JSONL, CSV, or Parquet).

To the best of our knowledge, there does not exist a solution that meets all these desiderata. Thus, we designed OPTIMUS to satisfy these design considerations pre-specified by independent reviews of the KG construction literature. Specifically, OPTIMUS was designed with three primary goals:

1. **Extensibility.** A modular design allows for new data sources or processing modules to be incorporated via providers and hooks with minimal changes to the existing workflow.
2. **Reproducibility.** Each KG construction step is deterministic and versioned, with provenance tracking and quality controls.
3. **Machine learning readiness.** Graphs exported from OPTIMUS are ready for use in standard graph AI libraries, including PyTorch Geomtric [58], the Deep Graph Library [59], and NetworkX [60].

3.2 OPTIMUS architecture

OPTIMUS follows a layered medallion architecture adapted from data lake design [61] (Figure 1). First, in the LANDING layer, raw data is downloaded from data providers. In the BRONZE layer, raw data from various sources are ingested in its original form, with minimal cleaning or filtering. The SILVER layer applies validation, normalization, and schema alignment to convert raw inputs into a consistent intermediate representation, resolving identifier mappings and enforcing ontology

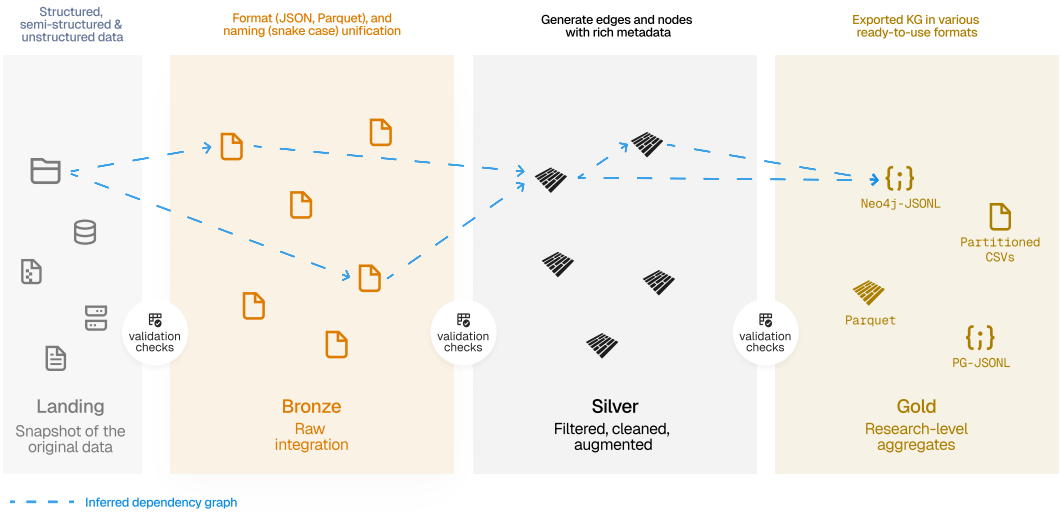

Figure 1: Medallion architecture of OPTIMUS. Snapshots of the original data in LANDING are passed to BRONZE, where naming conventions and file formats are unified while preserving raw semantics; then to SILVER, where records are validated, cleaned, deduplicated, and transformed into node and edge parquet tables with unique identifiers and rich metadata; and finally to GOLD, which materializes versioned KG exports. Circles denote data quality checks that ensure node artifacts are correct before proceeding to downstream nodes. An example analysis workflow is shown for a single dataset; other datasets may traverse different paths within the directed acyclic graph (DAG).

constraints. Finally, the GOLD layer produces of the integrated KG ready for analysis, where all entities and relationships are harmonized into the target schema and augmented with any derived features. This multi-layer pipeline incrementally improves data quality and structure, ensuring that upstream errors are caught early and that the final KG meets standards for consistency.

To orchestrate these steps, OPTIMUS leverages a Kedro workflow engine, enabling parallel processing of independent data sources and caching of intermediate results. The key components of OPTIMUS are as follows (Figure 2).

Catalog. A centralized registry of all datasets and resources. The catalog contains metadata about each data source (name, version, file locations or endpoints, schemas, etc.) and serves as the discovery layer. By querying the catalog, users can identify what data is available and determine its status in the pipeline.

Dataset. An abstraction representing a distinct input to the KG. Datasets encapsulate the logic to fetch or load the raw data (from files, databases, APIs) and define how it should be parsed. In OPTIMUS, each dataset is associated with a schema (expected fields and types), which is validated on load.

Schema. OPTIMUS requires the user to define the node types and edge types that constitute the KG schema (*e.g.*, node types might include `gene`, `disease`, or `drug`; edge types might include `interacts_with`, `treats`, or `is_associated_with`).

Providers. An abstraction that provides versioned, automatic data downloads from different data sources.

Node. A Python transformation function.

Pipeline. A sequence of nodes wired into a directed acyclic graph (DAG)-based workflow, organized by the datasets they consume and produce. Each pipeline is composed of transformations (*e.g.*, cleaning, normalizing, or merging) that convert raw data into graph-ready outputs.

Parameters. Used to define constants for filtering the data across the construction process.

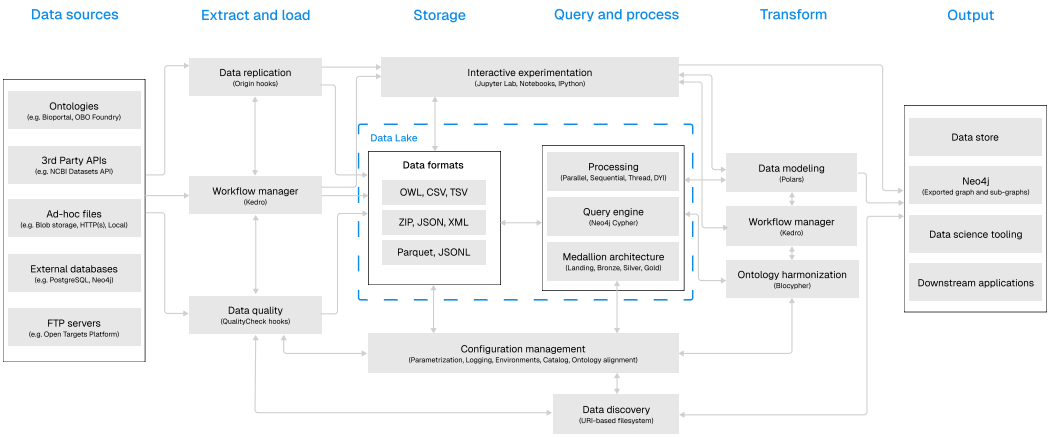

Figure 2: An overview of the components of OPTIMUS. Heterogeneous sources are ingested in the Extract and load stage via data replication (origin hooks) and Kedro-managed workflows, with ingestion gated by data-quality hooks. Data land in a lake that stores interoperable formats and is organized by a medallion architecture to enable incremental, reproducible builds. A central Configuration management layer and a URI-based Data discovery layer prodive governance and traceability. In Query and process, OPTIMUS supports parrallel, sequential, and thread execution and a Neo4j-Cypher query engine, while Interactive experimentation (Jupyter, IPython) attaches directyle to lake snapshots. The Transform stage combines schema-first data modeling with ontology harmonization using Biocypher.

Hook. A mechanism to allow injection of custom behavior into the core execution flow, such as before a node runs (*e.g.*, for checksum checks).

Configuration. A mechanism that separates code from settings, defining the catalog, parameters, and logging configuration, and ontology harmonization across different environments (*e.g.*, base, local, or production). Importantly, configuration management permits users to create KGs that incorporate both private and public data within the same codebase.

These core components work in concert to enable a structured approach to KG construction. In practice, a developer defines datasets in the catalog, writes pipeline definitions that specify how those datasets are transformed across layers, possibly adds hooks for custom steps, and customizes the configuration as necessary. The OPTIMUS runtime then orchestrates the end-to-end build, automatically applying schema validations and recording provenance at each step. We leverage additional features of the Kedro framework, such as namespaces, `kedro-viz`, `kedro-datasets` and catalog injection in Jupyter notebooks. OPTIMUS also includes a command line interface, and the KG can be queried via a Neo4j endpoint to extract subsets of the graph.

3.3 Features

Discovery and provenance. OPTIMUS assigns each dataset a stable URI and organizes entries as `<layer>.<namespace>.<dataset-name>`, recording source endpoints, versions, schemas, and transformation lineage in the catalog. This design enables end-to-end tracing from any node or edge back to the precise inputs and processing steps, supporting auditability and replication.

Reproducibility and lineage visualization. The full OPTIMUS pipeline is rendered via `kedro-viz`, which exposes dependencies across medallion layers and the live execution state, helping authors and reviewers inspect what was computed.

Automated, versioned data download. OPTIMUS fetches missing datasets on demand, validates schemas, and records the acquired version and checksum before ingestion. As a result, subsequent runs recover the same inputs under an identical configuration.

Private datasets with public fallbacks. OPTIMUS permits private sources to participate in the pipeline; when a collaborator lacks access, the system materializes schema-compatible empty place-

holders so the pipeline executes deterministically and results remain comparable. All data sets, public or private, are explicitly versioned in the catalog.

Configuration and parameterization. To separate "what" from "how," configuration files declare catalog entries, parameters, logging, and ontology mappings for different environments (*e.g.*, base, local, production). Common inclusion/exclusion rules are parameterized, allowing researchers to alter scope (filters, sources, schema variants) without code changes.

Interactive experimentation from lake snapshots. OPTIMUS injects the catalog into Jupyter and IPython so scientists can load datasets directly from immutable lake snapshots. This shortens the edit–run–inspect loop while keeping ad-hoc notebooks consistent with pipeline outputs.

Querying and subgraph export. Downstream tasks often need focused subgraphs. OPTIMUS provides a containerized Neo4j service; users run Cypher to probe structure, validate entities, and export task-specific subgraphs for audits or sharing.

Multi-format releases with summary metrics. Each release of OPTIMUS exports PG-JSONL, Parquet, Neo4j-JSONL, and partitioned CSV files, accompanied by `metrics.json` and `metrics.md` summarizing node, edge type counts, degree statistics, sources, and ontologies.

Developer experience. OPTIMUS uses modern tooling (`uv`, `docker`) and Kedro's engine for parallel, sequential, and threaded execution. A CLI exposes common tasks (benchmarking, checksums, metrics generation), making local runs consistent with continuous integration (CI).

Multiple runtimes. OPTIMUS provides multiple runtimes to take advantage of the performance characteristics of the host machine. `SequentialRunner` executes nodes sequentially in topological order; `ParallelRunner` executes independent nodes concurrently via multiprocessing (preferred for CPU-bound workloads); `ThreadRunner` executes independent nodes as threads within a single process (shared memory), effective for I/O-bound workloads with lower overhead than multiprocessing.

4 Performance evaluation

To evaluate the efficiency and scalability of OPTIMUS, we assessed **(1) end-to-end execution time for each runtime** and **(2) throughput as the KG grows**. All experiments were conducted on OPTIMUSKG, a biomedical KG constructed with OPTIMUS from 47 datasets (Appendix A.1). OPTIMUSKG contains 192,307 nodes, 21,499,963 edges, and 88,647,077 properties (Tables 1 and 2).

Setup. To ensure reproducibility, experiments were run with all files from the LANDING layer pre-downloaded and present on disk, eliminating variation caused by differences in network speeds. All benchmarks were run offline on a single machine with an AMD Ryzen 7 5700G CPU, 64 GB DDR4 RAM, and a 500 GB NVMe SSD.

Execution time as a function of runtime. We measured the wall time required by OPTIMUS to produce the final Parquet-formatted KG across the three available runtimes: `SequentialRunner`, `ThreadRunner`, and `ParallelRunner`. Enabling parallelism, both thread-wise and process-wise, substantially improved performance. Basic sequential execution generated the graph in 143.6 s on average, while the thread-parallelized version averaged 81.3 s (43.4% faster) and the process-parallelized version averaged 62.4 s (56.5% faster). `ParallelRunner` was consistently the fastest implementation for OPTIMUSKG (Figure 3a) as it exploits parallel paths in the graph construction DAG.

Throughtput as a function of edge count. To assess scalability, we incrementally enabled more datasets in the OPTIMUSKG Parquet-formatted construction pipeline and measured the impact on performance, as assessed by wall time normalized by total edge count (Figure 3b). Edge count was increased by adding edge types in ascending order of their cardinality (Supplementary Table 3). All runs used the `ParallelRunner` runtime with asynchronous I/O enabled. As reflected by the pronounced downward slope in Figure 3b, average time per edge generally decreased as the graph grew. We attribute this to fixed startup costs and file I/O overhead, which were progressively absorbed with increased edge count. However, certain additions induced significant dataset overhead, increasing throughput. For example, the addition of `exposure-protein` edges required the ingestion and normalization of OpenTargets, a large external resource, which involved costly entity resolution joins. The addition of `drug-drug` edges required parsing and normalizing DrugBank's large XML dataset and deriving information from deeply nested records, which necessitated expensive joins.

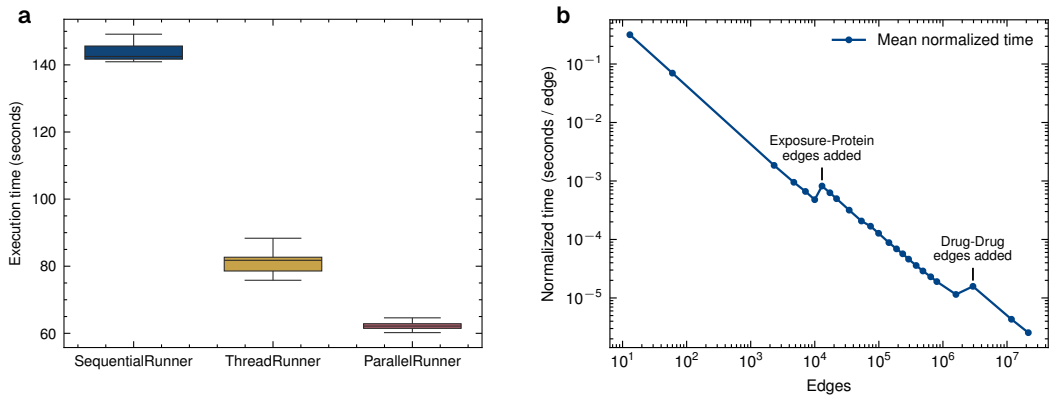

Figure 3: OPTIMUS is efficient and scalable. OPTIMUS's execution time was benchmarked as a function of the runtime, edge count, and runner type. **(a)** Execution time distributions (seconds) for 100 independent runs of the Bronze to Silver pipeline under three runners. Asynchronous I/O was enabled where supported (*i.e.*, ParallelRunner and SequentialRunner). **(b)** Mean normalized construction time per edge versus cumulative edge cardinality for an incremental, edge-type expansion (log-log). Each marker is the arithmetic mean of 100 independent runs of the construction pipeline; plotted values are the wall-clock mean time (seconds) divided by the cumulative number of edges in that benchmark (seconds per edge). Error bars are omitted because standard deviations are small relative to the axis resolution (largest SD $= 0.83$ s at the largest edge type by edge count; mean $= 54.81$ s).

These cases illustrate that heterogeneous upstream sources and complex transformations can have a significant impact on runtime, even with modest edge counts.

5 Discussion

We introduced OPTIMUS, a reproducible and extensible platform for large-scale KG construction. By combining a medallion architecture with modern workflow orchestration, OPTIMUS addresses long-standing challenges in KG construction and maintenance, including data heterogeneity, poor reproducibility, and limited scalability. Benchmarking shows that OPTIMUS achieves efficient execution and improved throughput as graphs grow. Although demonstrated in biomedicine with the construction of OPTIMUSKG, the OPTIMUS platform is domain-agnostic and readily extensible to other fields.

There are limitations to our work. The quality of the KG that OPTIMUS generates ultimately depends on upstream data sources, which may contain errors, incomplete annotations, or licensing constraints. Versioning in OPTIMUS depends on whether upstream sources themselves provide versioned releases. Addressing these challenges will require the generation of higher-quality datasets and improved knowledge extraction algorithms. Nonetheless, we envision that OPTIMUS will enable the generation of KGs to serve as the data foundation for the next generation of graph AI systems, including retrieval-augmented generation, graph LLMs, and autonomous agents, advancing the integration of structured, verifiable, and traceable knowledge into AI.

References

1. Ehrlinger, L. & Wöß, W. Towards a definition of knowledge graphs. *SEMANTiCS (Posters, Demos, SuCCESS)* **48,** 2 (2016).
2. Hogan, A. *et al.* Knowledge Graphs. *ACM Comput. Surv.* **54,** 71:1–71:37. doi:10.1145/3447772 (2021).
3. Angles, R., Arenas, M., Barceló, P., Hogan, A., Reutter, J. & Vrgoč, D. Foundations of Modern Query Languages for Graph Databases. *ACM Comput. Surv.* **50,** 68:1–68:40. doi:10.1145/3104031 (2017).

4. Chandak, P., Huang, K. & Zitnik, M. Building a knowledge graph to enable precision medicine. *Scientific Data* **10,** 67. doi:10.1038/s41597-023-01960-3 (2023).
5. Le-Phuoc, D., Nguyen Mau Quoc, H., Ngo Quoc, H., Tran Nhat, T. & Hauswirth, M. The Graph of Things: A step towards the Live Knowledge Graph of connected things. *Journal of Web Semantics* **37-38,** 25–35. doi:10.1016/j.websem.2016.02.003 (2016).
6. Cheng, D., Yang, F., Wang, X., Zhang, Y. & Zhang, L. *Knowledge Graph-based Event Embedding Framework for Financial Quantitative Investments* in *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval* (Association for Computing Machinery, New York, NY, USA, 2020), 2221–2230. doi:10.1145/3397271.3401427.
7. Dye, M., Ekanadham, C., Saluja, A. & Rastogi, A. *Supporting content decision makers with machine learning* 2021.
8. Xiong, C., Power, R. & Callan, J. *Explicit Semantic Ranking for Academic Search via Knowledge Graph Embedding* in *Proceedings of the 26th International Conference on World Wide Web* (International World Wide Web Conferences Steering Committee, Republic and Canton of Geneva, CHE, 2017), 1271–1279. doi:10.1145/3038912.3052558.
9. Jia, Y., Qi, Y., Shang, H., Jiang, R. & Li, A. A Practical Approach to Constructing a Knowledge Graph for Cybersecurity. *Engineering. Cybersecurity* **4,** 53–60. doi:10.1016/j.eng.2018.01.004 (2018).
10. Xia, F. *et al.* Graph Learning: A Survey. *IEEE Transactions on Artificial Intelligence* **2,** 109–127. doi:10.1109/TAI.2021.3076021 (2021).
11. Ying, R., He, R., Chen, K., Eksombatchai, P., Hamilton, W. L. & Leskovec, J. *Graph Convolutional Neural Networks for Web-Scale Recommender Systems* in *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining* (Association for Computing Machinery, New York, NY, USA, 2018), 974–983. doi:10.1145/3219819.3219890.
12. Yasunaga, M., Ren, H., Bosselut, A., Liang, P. & Leskovec, J. *QA-GNN: Reasoning with Language Models and Knowledge Graphs for Question Answering* 2022. doi:10.48550/arXiv.2104.06378.
13. Townshend, R. J. L. *et al.* Geometric deep learning of RNA structure. *Science* **373,** 1047–1051. doi:10.1126/science.abe5650 (2021).
14. Johnson, R., Li, M. M., Noori, A., Queen, O. & Zitnik, M. Graph Artificial Intelligence in Medicine. *Annual Review of Biomedical Data Science* **7,** 345–368. doi:10.1146/annurev-biodatasci-110723-024625 (2024).
15. Zitnik, M., Agrawal, M. & Leskovec, J. Modeling polypharmacy side effects with graph convolutional networks. *Bioinformatics* **34,** i457–i466. doi:10.1093/bioinformatics/bty294 (2018).
16. Akoglu, L., Tong, H. & Koutra, D. Graph based anomaly detection and description: a survey. *Data Mining and Knowledge Discovery* **29,** 626–688. doi:10.1007/s10618-014-0365-y (2015).
17. Xie, T. & Grossman, J. C. Crystal Graph Convolutional Neural Networks for an Accurate and Interpretable Prediction of Material Properties. *Physical Review Letters* **120,** 145301. doi:10.1103/PhysRevLett.120.145301 (2018).
18. Lam, R. *et al.* Learning skillful medium-range global weather forecasting. *Science* **382,** 1416–1421. doi:10.1126/science.adi2336 (2023).
19. Liu, J., Mao, H., Chen, Z., Zhao, T., Shah, N. & Tang, J. *Neural Scaling Laws on Graphs* 2024. doi:10.48550/arXiv.2402.02054.
20. Sypetkowski, M. *et al. On the Scalability of GNNs for Molecular Graphs* 2024. doi:10.48550/ARXIV.2404.11568.
21. Ji, X. *et al. Uni-Mol2: Exploring Molecular Pretraining Model at Scale* 2024. doi:10.48550/arXiv.2406.14969.
22. Ryen, V., Soylu, A. & Roman, D. Building Semantic Knowledge Graphs from (Semi-)Structured Data: A Review. *Future Internet* **14,** 129. doi:10.3390/fi14050129 (2022).
23. Tamašauskaitė, G. & Groth, P. Defining a Knowledge Graph Development Process Through a Systematic Review. *ACM Trans. Softw. Eng. Methodol.* **32,** 27:1–27:40. doi:10.1145/3522586 (2023).

24. Zhong, L., Wu, J., Li, Q., Peng, H. & Wu, X. A Comprehensive Survey on Automatic Knowledge Graph Construction. *ACM Comput. Surv.* **56,** 94:1–94:62. doi:10.1145/3618295 (2023).
25. Hofer, M., Obraczka, D., Saeedi, A., Köpcke, H. & Rahm, E. Construction of Knowledge Graphs: Current State and Challenges. *Information* **15,** 509. doi:10.3390/info15080509 (2024).
26. Choi, S. & Jung, Y. Knowledge Graph Construction: Extraction, Learning, and Evaluation. *Applied Sciences* **15,** 3727. doi:10.3390/app15073727 (2025).
27. Auer, S., Bizer, C., Kobilarov, G., Lehmann, J., Cyganiak, R. & Ives, Z. *DBpedia: A Nucleus for a Web of Open Data* in *The Semantic Web: 6th International Semantic Web Conference, 2nd Asian Semantic Web Conference, ISWC 2007 + ASWC 2007, Busan, Korea, November 11-15, 2007. Proceedings* (Springer-Verlag, Berlin, Heidelberg, 2007), 722–735. doi:10.1007/978-3-540-76298-0_52.
28. Bollacker, K., Evans, C., Paritosh, P., Sturge, T. & Taylor, J. *Freebase: a collaboratively created graph database for structuring human knowledge* in *Proceedings of the 2008 ACM SIGMOD international conference on Management of data* (Association for Computing Machinery, New York, NY, USA, 2008), 1247–1250. doi:10.1145/1376616.1376746.
29. Vrandečić, D. & Krötzsch, M. Wikidata: a free collaborative knowledgebase. *Commun. ACM* **57,** 78–85. doi:10.1145/2629489 (2014).
30. Etzioni, O. *et al.* Unsupervised named-entity extraction from the Web: An experimental study. *Artificial Intelligence* **165,** 91–134. doi:10.1016/j.artint.2005.03.001 (2005).
31. Al-Moslmi, T., Gallofré Ocaña, M., L. Opdahl, A. & Veres, C. Named Entity Extraction for Knowledge Graphs: A Literature Overview. *IEEE Access* **8,** 32862–32881. doi:10.1109/ACCESS.2020.2973928 (2020).
32. Liu, S., Grau, B., Horrocks, I. & Kostylev, E. *INDIGO: GNN-Based Inductive Knowledge Graph Completion Using Pair-Wise Encoding* in *Advances in Neural Information Processing Systems* **34** (Curran Associates, Inc., 2021), 2034–2045.
33. Dai, G., Wang, X., Zou, X., Liu, C. & Cen, S. MRGAT: Multi-Relational Graph Attention Network for knowledge graph completion. *Neural Networks* **154,** 234–245. doi:10.1016/j.neunet.2022.07.014 (2022).
34. Wei, Y., Huang, Q., Kwok, J. T. & Zhang, Y. *KICGPT: Large Language Model with Knowledge in Context for Knowledge Graph Completion* in *Findings of the Association for Computational Linguistics: EMNLP 2023* (2023), 8667–8683. doi:10.18653/v1/2023.findings-emnlp.580.
35. Zhang, Y., Chen, Z., Guo, L., Xu, Y., Zhang, W. & Chen, H. *Making Large Language Models Perform Better in Knowledge Graph Completion* in *Proceedings of the 32nd ACM International Conference on Multimedia* (Association for Computing Machinery, New York, NY, USA, 2024), 233–242. doi:10.1145/3664647.3681327.
36. Yao, L., Peng, J., Mao, C. & Luo, Y. *Exploring Large Language Models for Knowledge Graph Completion* in *ICASSP 2025 - 2025 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)* (2025), 1–5. doi:10.1109/ICASSP49660.2025.10889242.
37. Ugander, J., Karrer, B., Backstrom, L. & Marlow, C. *The Anatomy of the Facebook Social Graph* 2011. doi:10.48550/arXiv.1111.4503.
38. Dong, X. L. *Challenges and Innovations in Building a Product Knowledge Graph* in *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining* (Association for Computing Machinery, New York, NY, USA, 2018), 2869. doi:10.1145/3219819.3219938.
39. Himmelstein, D. S. *et al.* Systematic integration of biomedical knowledge prioritizes drugs for repurposing. *eLife* **6** (ed Valencia, A.) e26726. doi:10.7554/eLife.26726 (2017).
40. Feng, F. *et al.* GenomicKB: a knowledge graph for the human genome. *Nucleic Acids Research* **51,** D950–D956. doi:10.1093/nar/gkac957 (2023).
41. Ilievski, F. *et al. KGTK: A Toolkit for Large Knowledge Graph Manipulation and Analysis* in *The Semantic Web – ISWC 2020* (eds Pan, J. Z. *et al.*) (Springer International Publishing, Cham, 2020), 278–293. doi:10.1007/978-3-030-62466-8_18.
42. Haase, P., Herzig, D. M., Kozlov, A., Nikolov, A. & Trame, J. metaphactory: A platform for knowledge graph management. *Semantic Web* **10,** 1109–1125. doi:10.3233/SW-190360 (2019).

43. Lobentanzer, S. *et al.* Democratizing knowledge representation with BioCypher. *Nature Biotechnology* **41,** 1056–1059. doi:10.1038/s41587-023-01848-y (2023).
44. Caufield, J. H. *et al.* KG-Hub—building and exchanging biological knowledge graphs. *Bioinformatics* **39,** btad418. doi:10.1093/bioinformatics/btad418 (2023).
45. Babalou, S., Samuel, S. & König-Ries, B. *Reproducible Domain-Specific Knowledge Graphs in the Life Sciences: a Systematic Literature Review* 2023. doi:10.48550/arXiv.2309.08754.
46. Lavrinovics, E., Biswas, R., Bjerva, J. & Hose, K. Knowledge Graphs, Large Language Models, and Hallucinations: An NLP Perspective. *Journal of Web Semantics* **85,** 100844. doi:10.1016/j.websem.2024.100844 (2025).
47. Sansford, H., Richardson, N., Maretic, H. P. & Saada, J. N. *GraphEval: A Knowledge-Graph Based LLM Hallucination Evaluation Framework* 2024. doi:10.48550/arXiv.2407.10793.
48. Wang, Y., Lipka, N., Rossi, R. A., Siu, A., Zhang, R. & Derr, T. Knowledge Graph Prompting for Multi-Document Question Answering. *Proceedings of the AAAI Conference on Artificial Intelligence* **38,** 19206–19214. doi:10.1609/aaai.v38i17.29889 (2024).
49. Mitchell, T. *et al.* Never-ending learning. *Commun. ACM* **61,** 103–115. doi:10.1145/3191513 (2018).
50. Ying, Z., Bourgeois, D., You, J., Zitnik, M. & Leskovec, J. *GNNExplainer: Generating Explanations for Graph Neural Networks* in *Advances in Neural Information Processing Systems* **32** (Curran Associates, Inc., 2019).
51. Pan, S., Luo, L., Wang, Y., Chen, C., Wang, J. & Wu, X. Unifying Large Language Models and Knowledge Graphs: A Roadmap. *IEEE Transactions on Knowledge and Data Engineering* **36,** 3580–3599. doi:10.1109/TKDE.2024.3352100 (2024).
52. Sanmartin, D. *KG-RAG: Bridging the Gap Between Knowledge and Creativity* 2024. doi:10.48550/arXiv.2405.12035.
53. Matsumoto, N. *et al.* KRAGEN: a knowledge graph-enhanced RAG framework for biomedical problem solving using large language models. *Bioinformatics* **40,** btae353. doi:10.1093/bioinformatics/btae353 (2024).
54. Edge, D. *et al. From Local to Global: A Graph RAG Approach to Query-Focused Summarization* 2025. doi:10.48550/arXiv.2404.16130.
55. Mavromatis, C. & Karypis, G. *GNN-RAG: Graph Neural Retrieval for Large Language Model Reasoning* 2024. doi:10.48550/arXiv.2405.20139.
56. Wood, D., Lanthaler, M. & Cyganiak, R. RDF 1.1 concepts and abstract syntax. *W3C Recommendation, W3C* (2014).
57. Madnick, S. E., Wang, R. Y., Lee, Y. W. & Zhu, H. Overview and Framework for Data and Information Quality Research. *J. Data and Information Quality* **1,** 2:1–2:22. doi:10.1145/1515693.1516680 (2009).
58. Fey, M. & Lenssen, J. E. *Fast Graph Representation Learning with PyTorch Geometric* 2019. doi:10.48550/arXiv.1903.02428.
59. Wang, M. *et al. Deep Graph Library: A Graph-Centric, Highly-Performant Package for Graph Neural Networks* 2019.
60. Hagberg, A., Swart, P. J. & Schult, D. A. *Exploring network structure, dynamics, and function using NetworkX* tech. rep. LA-UR-08-05495; LA-UR-08-5495 (Los Alamos National Laboratory (LANL), Los Alamos, United States, 2007).
61. Armbrust, M., Ghodsi, A., Xin, R. & Zaharia, M. *Lakehouse: a new generation of open platforms that unify data warehousing and advanced analytics* in *Proceedings of CIDR* **8** (2021), 28.

A Technical Appendices and Supplementary Material

A.1 Data sources used to build OPTIMUSKG

OPTIMUSKG was constructed from 47 biomedical data sources, including the Anatomical Therapeutic Chemical Classification System (ATC), Bgee Gene Expression Database (Bgee), British National Formulary (BNF), Clinical Genomic Information (CGI), Clinical Genome Resource (ClinGen), ClinicalTrials.gov, Comparative Toxicogenomics Database (CTD), DailyMed, Disease Ontology (DOID), DrugBank, European Medicines Agency (EMA), Experimental Factor Ontology (EFO), Genomics England PanelApp (Genomics England), Gene Ontology (GO), Human Phenotype Ontology (HPO), Kyoto Encyclopedia of Genes and Genomes (KEGG), Mammalian Phenotype Ontology (MP), Medical Dictionary for Regulatory Activities (MedDRA), Medical Subject Headings (MeSH), Monarch Disease Ontology (MONDO), NCI Thesaurus (NCIt), Ontology for Biomedical Investigations (OBI), Ontology for General Medical Science (OGMS), Ontology of Biological Attributes (OBA), OnSIDES Adverse Drug Events Resource (OnSIDES), Open Targets Platform, Open Targets Rare Disease (OTAR), Orphanet Rare Disease Ontology (Orphanet), Phenotype And Trait Ontology (PATO), PsyGeNET, PubChem, PubMed, PubMed Central (PMC), Reactome, Uber-anatomy ontology (UBERON), Universal Protein Resource (UniProt), U.S. Food and Drug Administration (FDA), and Wikipedia, among other datasets.

Label	Count	Percentage	Avg. Degree	Avg. Properties
Gene	61,106	31.78%	323.65 ± 443.99	17.86 ± 3.78
Disease	36,992	19.24%	279.49 ± 1278.84	15.82 ± 0.92
BiologicalProcess	25,754	13.39%	11.15 ± 110.19	8.67 ± 0.47
Phenotype	19,341	10.06%	51.94 ± 430.65	8.96 ± 3.55
Drug	16,591	8.63%	167.02 ± 388.27	9.32 ± 4.25
Anatomy	14,624	7.60%	603.53 ± 4876.54	8.27 ± 0.85
MolecularFunction	10,161	5.28%	11.43 ± 150.56	8.71 ± 0.46
CellularComponent	4,052	2.11%	28.28 ± 267.97	8.55 ± 0.50
Pathway	2,805	1.46%	18.76 ± 29.49	3.00 ± 0.05
Exposure	881	0.46%	14.29 ± 43.85	3.75 ± 0.43

Table 1: Summary statistics for node types in OPTIMUSKG, including counts, relative proportions, average node degrees, and average number of properties per node type.

Label	Count	Percentage	Directed	Undirected	Avg. Properties
Disease-Protein	9,770,626	45.44%	0	9,770,626	4.02 ± 0.40
Anatomy-Protein	8,787,955	40.87%	0	8,787,955	4.00 ± 0.00
Drug-Drug	1,345,376	6.26%	1,345,376	0	3.00 ± 0.06
Phenotype-Protein	793,279	3.69%	793,279	0	4.01 ± 0.32
BiologicalProcess-Protein	158,410	0.74%	0	158,410	5.00 ± 0.00
Disease-Phenotype	157,144	0.73%	0	157,144	11.00 ± 0.00
CellularComponent-Protein	105,309	0.49%	0	105,309	5.00 ± 0.00
MolecularFunction-Protein	90,933	0.42%	0	90,933	5.00 ± 0.00
Drug-Disease	55,378	0.26%	0	55,378	4.00 ± 0.00
Pathway-Protein	46,977	0.22%	0	46,977	2.00 ± 0.00
Disease-Disease	45,083	0.21%	45,083	0	2.00 ± 0.00
BiologicalProcess-BiologicalProcess	44,494	0.21%	44,494	0	2.00 ± 0.00
Phenotype-Phenotype	24,862	0.12%	24,862	0	2.00 ± 0.00
Drug-Protein	20,694	0.10%	20,694	0	2.42 ± 1.04
Anatomy-Anatomy	19,004	0.09%	19,004	0	2.00 ± 0.00
MolecularFunction-MolecularFunction	12,587	0.06%	12,587	0	2.00 ± 0.00
CellularComponent-CellularComponent	4,639	0.02%	4,639	0	2.00 ± 0.00
Drug-Phenotype	4,251	0.02%	0	4,251	3.73 ± 0.68
Exposure-Protein	2,989	0.01%	2,989	0	32.00 ± 0.00
Pathway-Pathway	2,819	0.01%	2,819	0	2.00 ± 0.00
Exposure-Exposure	2,443	0.01%	2,443	0	32.00 ± 0.00
Exposure-Disease	2,391	0.01%	2,391	0	32.00 ± 0.00
Exposure-BiologicalProcess	2,260	0.01%	0	2,260	32.00 ± 0.00
Exposure-MolecularFunction	47	0.00%	0	47	32.00 ± 0.00
Exposure-CellularComponent	13	0.00%	0	13	32.00 ± 0.00

Table 2: Summary statistics for edge types in OPTIMUSKG, including counts, relative proportions, directionality, and average number of properties per edge type.

Run	Total Edges	New Edge Type	New Edges
1	13	Exposure–CellularComponent	13
2	60	Exposure–MolecularFunction	47
3	2,320	Exposure–BiologicalProcess	2,260
4	4,711	Exposure–Disease	2,391
5	7,154	Exposure–Exposure	2,443
6	9,973	Pathway–Pathway	2,819
7	12,962	Exposure–Protein	2,989
8	17,213	Drug–Phenotype	4,251
9	21,852	CellularComponent–CellularComponent	4,639
10	34,439	MolecularFunction–MolecularFunction	12,587
11	53,443	Anatomy–Anatomy	19,004
12	74,137	Drug–Protein	20,694
13	98,999	Phenotype–Phenotype	24,862
14	143,493	BiologicalProcess–BiologicalProcess	44,494
15	188,576	Disease–Disease	45,083
16	235,553	Pathway–Protein	46,977
17	290,931	Drug–Disease	55,378
18	381,864	MolecularFunction–Protein	90,933
19	487,173	CellularComponent–Protein	105,309
20	644,317	Disease–Phenotype	157,144
21	802,727	BiologicalProcess–Protein	158,410
22	1,596,006	Phenotype–Protein	793,279
23	2,941,382	Drug–Drug	1,345,376
24	11,729,337	Anatomy–Protein	8,787,955
25	21,499,963	Disease–Protein	9,770,626

Table 3: Incremental edge-type addition during scalability benchmarking. Each run adds a new edge type to OPTIMUSKG, reporting both the cumulative edge count and the marginal edges contributed. Edge types are ordered by ascending edge cardinality.

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: We have, to the best of our ability, considered each claim and we believe each to be accurate.

   Guidelines:

   - The answer NA means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A No or NA answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: They are described in the Discussion section (Section 5).

   Guidelines:

   - The answer NA means that the paper has no limitation while the answer No means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [NA]

Justification: [NA]

Guidelines:

- The answer NA means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: To the best of our ability, we have made the details necessary to reproduce the experimental results of the paper available in our methodological description.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - If the paper includes experiments, a No answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
   - If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
   - Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
   - While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
     (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
     (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
     (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
     (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

Answer: [No]

Justification: The code for OPTIMUS will be publicly released via an open-source GitHub repository at a future date.

Guidelines:

- The answer NA means that paper does not include experiments requiring code.
- Please see the NeurIPS code and data submission guidelines (`https://nips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- While we encourage the release of code and data, we understand that this might not be possible, so "No" is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
- The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://nips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
- The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
- At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
- Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer, etc.) necessary to understand the results?

   Answer: [Yes]

   Justification: To the best of our ability, we have made the details necessary to reproduce the experimental results of the paper available in our methodological description.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [Yes]

   Justification: [Yes]

   Guidelines: Figure 3a, error bars are provided representing the interquartile range of the data distribution. In Figure 3b, error bars are omitted because standard deviations are small relative to the axis resolution (largest SD $= 0.83$ s at the largest edge type by edge count; mean $= 54.81$ s). This is explained in the figure caption.

   - The answer NA means that the paper does not include experiments.
   - The authors should answer "Yes" if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
   - The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).

- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g. negative error rates).
- If error bars are reported in tables or plots, The authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: This information is provided in the Performance evaluation section (Section 4).

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: We believe that we are fully compliant with the NeurIPS Code of Ethics.

   Guidelines:

   - The answer NA means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer No, they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [NA]

    Justification: We have not identified direct harms caused by the research process, and negative societal impacts from the intended use of OPTIMUS are unlikely. One possible risk is that automatically constructed graphs may contain factual inaccuracies due to errors in upstream data sources or integration pipelines. However, this risk is not specific to OPTIMUS and is already present in existing KG construction workflows. Moreover, by emphasizing provenance, versioning, and observability, OPTIMUS arguably mitigates rather than exacerbates these risks. Therefore, we assess that the broader societal impact of this work is limited.

    Guidelines:

- The answer NA means that there is no societal impact of the work performed.
- If the authors answer NA or No, they should explain why their work has no societal impact or why the paper does not address societal impact.
- Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pretrained language models, image generators, or scraped datasets)?

    Answer: [NA]

    Justification: [NA]

    Guidelines:

    - The answer NA means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
    - We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [NA]

    Justification: [NA]

    Guidelines:

    - The answer NA means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.
    - For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.

- If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [NA]

    Justification: [NA]

    Guidelines:

    - The answer NA means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [NA]

    Justification: [NA]

    Guidelines:

    - The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [NA]

    Justification: [NA]

    Guidelines:

    - The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
    - Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.

- We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
- For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

    Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does not impact the core methodology, scientific rigorousness, or originality of the research, declaration is not required.

    Answer: [NA]

    Justification: [NA]

    Guidelines:

    - The answer NA means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
    - Please refer to our LLM policy (`https://neurips.cc/Conferences/2025/LLM`) for what should or should not be described.