Demonstration of SPARQL^{ML}: An Interfacing Language for Supporting Graph Machine Learning for RDF Graphs

Hussein Abdallah
Concordia University
hussein.abdallah@mail.concordia.ca

Waleed Afandi
Concordia University
waleed.afandi@mail.concordia.ca

Essam Mansour
Concordia University
essam.mansour@concordia.ca

ABSTRACT

This demo paper presents KGNet, a graph machine learning-enabled RDF engine. KGNet integrates graph machine learning (GML) models with existing RDF engines as query operators to support node classification and link prediction tasks. For easy integration, KGNet extends the SPARQL language with user-defined predicates to support the GML operators. We refer to this extension as SPARQL^{ML} query. Our SPARQL^{ML} query optimizer is in charge of optimizing the selection of the near-optimal GML models. The development of KGNet poses research opportunities in various areas spanning KG management. In the paper, we demonstrate the ease of integration between the RDF engines and GML models through the SPARQL^{ML} inference query language. We present several real use cases of different GML tasks on real KGs. Using KGNet, users do not need to learn a new scripting language or have a deep understanding of GML methods. The audience will experience KGNet with different KGs and GML models, as shown in our demo video and Colab notebook.

PVLDB Reference Format:
Hussein Abdallah, Waleed Afandi, and Essam Mansour. Demonstration of SPARQL^{ML}: An Interfacing Language for Supporting Graph Machine Learning for RDF Graphs. PVLDB, 16(12): 3974 - 3977, 2023.
doi:10.14778/3611540.3611599

PVLDB Artifact Availability:
The source code, data, and/or other artifacts have been made available at https://github.com/CoDS-GCS/KGNET/.

1 INTRODUCTION

Knowledge graphs (KGs) are heterogeneous directed graphs that store the semantics of relationships of real-world entities in different domains[4]. Graph machine learning (GML) methods, especially graph neural networks (GNNs), have been used recently to apply machine learning on top of graph data and achieved outstanding results in different tasks such as recommendations [9], drug discovery [7] and fraud detection [2]. KGs are built to express the relational graph data structure semantics. However, existing RDF engines lack the support of integrating directly with GML-trained models to allow ad-hoc GML-based queries over the stored KGs.

Data scientists often develop custom ML pipelines for their tasks. The automated SQL-ML pipelines get attention recently on ML-databases [3, 6, 10], such as MindsDB [1]. Likewise, GML pipelines now commonly train ML models on graph data using popular GML frameworks, such as PYG, DGL, and others. For the average user, this responsibility is overwhelming and time-consuming. Furthermore, the trained models are isolated from the RDF engine, where the KG is stored. Therefore, automating the training and inference pipelines via a SPARQL-like query is essential. We refer to this query as a SPARQL$^{\mathbf{ML}}$ query.

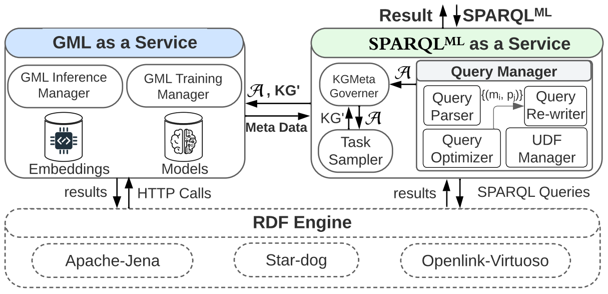

Figure 1: The KGNet architecture, which enables data scientists to automatically train GML models on KGs for querying and inferencing using the SPARQL$^{\mathbf{ML}}$ language interface.

The KGNet architecture [1] shown in Figure 1 extends existing RDF engines with two main components GML-as-a-service (GMLaaS) and SPARQL$^{\mathbf{ML}}$ as a Service. The *GMLaaS* automates the training of GML tasks, such as node classification or link prediction, on KGs and maintains a set of trained models. It also provides a Restful API service to manage the trained models and support ad-hoc inference queries. *SPARQL*$^{\mathbf{ML}}$ *as a Service* collects the metadata of the trained models from GMLaaS. This metadata is stored as an RDF KG, called *KGMeta*. This process is managed by the KGMeta governor. *KGMeta* enables seamless integration between the RDF engine and the GML model zoo [5] to allow ad-hoc inference queries in the form of SPARQL$^{\mathbf{ML}}$. KGMeta allows users to identify the appropriate GML model for their task and then get model predictions through SPARQL$^{\mathbf{ML}}$ inference queries. It allows the SPARQL$^{\mathbf{ML}}$ optimizer to choose the optimal model in terms of accuracy and inference time by automatically mapping the user-defined predicates into the appropriate GML model. Our SPARQL$^{\mathbf{ML}}$ as a Service enable data and ML engineers to focus on analytics and model development. The remainder of this paper is organized as follows. Section 2 presents the KGNet SPARQL^{ML} query management and optimization. Section 3 demos five real GML tasks. Section 4 concludes the paper.

2 THE SPARQL$^{\mathbf{ML}}$ AS A SERVICE

The SPARQL$^{\mathbf{ML}}$ as a Service component in Figure 1 transforms SPARQL$^{\mathbf{ML}}$ queries into SPARQL queries, which are then executed

[1] predicting home rental prices with MindsDB SQL-ML

Proceedings of the VLDB Endowment, Vol. 16, No. 12 ISSN 2150-8097.
doi:10.14778/3611540.3611599

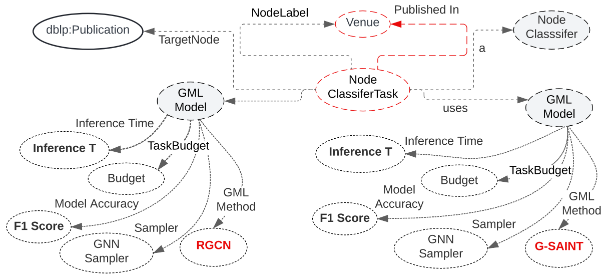

Figure 2: A KGMeta instance of two trained models for the node classification task. The white nodes are nodes from the original data KG. The dashed nodes/edges are metadata collected per trained model. A single task can have multiple trained models, such as RGCN and GraphSAINT.

using existing RDF engines. The SPARQL$^{\mathbf{ML}}$ extension enables users to express SPARQL-like queries for INSERT, DELETE, or SELECT (Inference) queries, such that: (*i*) The trained model metadata, as illustrated in Figure 2, is gathered from a SPARQL$^{\mathbf{ML}}$, *INSERT* query in Figure 3 through a training pipeline for the GML model. The collected metadata is maintained in KGMeta KG in the form of RDF triples, (*ii*) a SPARQL$^{\mathbf{ML}}$ *DELETE* query depicted in Figure 4 deletes the trained model files and their corresponding embeddings from the GMLaaS component, followed by the deletion of their metadata from the KGMeta. (*iii*) a SPARQL$^{\mathbf{ML}}$ *SELECT* query is for querying and inferencing the KG, e.g., the query in Figure 5. When a SPARQL$^{\mathbf{ML}}$ query is received, the query manager parses it. The SELECT query is optimized and rewritten as a SPARQL query. The INSERT or DELETE query is sent to the KGMeta governor for maintaining the KGMeta updates. The UDF manager manages the creation of UDFs in a remote RDF engine. These UDFs enable the communication between the remote RDF engine and KGNet GML inference manager API via Restful HTTP Post/Get requests.

2.1 KGMeta Governor

The KGMeta governor maintains a KGMeta graph, which includes statistics, such as triples and target labels, and metadata collected for trained models for a given task, such as the model accuracy and training time. The INSERT query is a request to train a task on a certain KG. The parsed information includes the task type, i.e., node classification or link prediction, and the task inputs, i.e., the target nodes and the GNN method. This information is encapsulated as a JSON object. The *TrainGML* function shown in Figure 3 takes the JSON object that encapsulates all required information to train a GML model as input and then invokes the GMLaaS training pipeline. Before starting the training pipeline, the KGMeta Governor sends the JSON object to the task sampler to extract a task-oriented subgraph denoted as KG' for the given task. This subgraph KG' is subsequently transformed into a GML dataset and then initiates the model training. Once training is complete, the KGMeta governor receives the trained model's metadata, including trained model score and inference time, to maintain the KGMeta, as illustrated in Figure 2.

The SPARQL$^{\mathbf{ML}}_{pv}$ query shown in Figure 5 is an example of an inference query that extends traditional SPARQL query language to apply GML models on top of the DBLP KG. The venue variable

```
prefix dblp:<https://www.dblp.org/>
prefix kgnet:<https://www.kgnet.com/>
Insert into <kgnet>  { ?s ?p ?o }
where {select * from  kgnet.TrainGML(
{Name: 'MAG_Paper-Venue_Classifer',
 GMLTask:{ taskType:kgnet:NodeClassifier,
 targetNode: dblp:publication,
 labelNode: dblp:venue,GNNMethod: RGCN},
 taskBudget:{ maxMemory:50GB, maxTime:1h} })};
```

Figure 3: A SPARQL$^{\mathbf{ML}}$ insert query that trains a paper-venue classifier on DBLP KG. The TrainGML function is implemented inside KGNet to automate a training pipeline for a given task.

```
prefix dblp:<https://www.dblp.org/>
prefix kgnet:<https://www.kgnet.com/>
delete {?NodeClassifier ?p ?o}
where {
?NodeClassifier a kgnet:nodeClassifier.
?NodeClassifier kgnet:targetNode dblp:Publication.
?NodeClassifier kgnet:labelNode dblp:venue.
?NodeClassifier ?p ?o.}
```

Figure 4: A SPARQL$^{\mathbf{ML}}$ delete query that deletes a trained model and its meta-data.

```
1  prefix dblp: <https://www.dblp.org/>
2  prefix kgnet: <https://www.kgnet.com/>
3  select ?title ?venue
4  where {
5  ?paper a dblp:Publication.
6  ?paper dblp:title ?title.
7  ?paper ?NodeClassifier ?venue.
8  ?NodeClassifier a kgnet:nodeClassifier .
9  ?NodeClassifier kgnet:targetNode dblp:Publication.
10 ?NodeClassifier kgnet:labelNode dblp:venue.}
```

Figure 5: SPARQL$^{\mathbf{ML}}_{pv}$: a SPARQL$^{\mathbf{ML}}$ query includes the user defined predicate ?NodeClassifier to invoke a trained node classification (*PV*) model. *PV* predicts a paper's venue by querying and inferencing over the DBLP KG. Lines 8-10 are user-defined triple patterns identifying the candidate node classification models.

is a virtual node that could be predicted using a node classification (NC) model. This query uses a model of type *kgnet:NodeClassifier* to predict a venue for each paper or for a subset of papers that can be identified through a SPARQL filter function. The SPARQL$^{\mathbf{ML}}$ triple patterns in lines 8-10 will retrieve all models of type *kgnet:NodeClassifier* that predict a label of type *dblp:venue* for target node of type *dblp:publication*. In the triple pattern ⟨?paper, ?NodeClassifier, ?venue⟩, we refer to *?NodeClassifier* as a user-defined predicate.

2.2 SPARQL^{ML} query Optimization

The *query manager* optimizes SPARQL$^{\mathbf{ML}}$ queries for model selection and rank-ordering to evaluate user-defined predicates. In the case of SPARQL$^{\mathbf{ML}}_{pv}$ shown in Figure 5, the query parser parse the SPARQL^{ML} query to identify the KG triples patterns in lines 5-6 and the user-defined GML triples patterns in lines

```
prefix dblp: <https://www.dblp.org/>
prefix kgnet: <https://www.kgnet.com/>
select ?title
    sql:UDFS.getNodeClass($m,?paper) as ?venue
where {
    ?paper a dblp:Publication.
    ?paper dblp:title ?title.
}
```

(a) Small Cardinality Query Form

```
prefix dblp: <https://www.dblp.org/>
prefix kgnet: <https://www.kgnet.com/>
select ?title
  sql:getKeyValue(?venues_dic,?paper) as ?venue
where {
    ?paper a dblp:Publication.
    ?paper dblp:title ?title.
{select sql:UDFS.getNodeClass($m,dblp:Publication)
  as ?venues_dic where { } }}
```

(b) Large Cardinality Query Form

Figure 6: Candidate SPARQL Queries for SPARQL$^{\mathbf{ML}}_{pv}$

7-10. The latter allows the query optimizer to fetch the IRI of the models satisfying the conditions associated with the user-defined predicate *?NodeClassifier* from KGMeta. The *SPARQL*$^{\mathbf{ML}}$ *Query Re-writer* uses the near-optimal GML model with IRI m to generate a candidate SPARQL query. The candidate SPARQL query passes through two optimization phases.

(***Phase 1***) Optimizing for the GML model selection: The KGMeta stores training task metadata, such as model accuracy, inference time, and model cardinality. The query optimizer uses the KGMeta to select the near-optimal GML model with high accuracy and low inference time. We define this problem as an integer programming (IP) optimization problem [5, 8] that minimizes total execution time and maximizes the inference accuracy for a set of candidate-trained models as in equation 1.

$$\begin{aligned} \max \quad & w_1 \cdot Score(m) - w_2 \cdot InferTime(m) \qquad \text{for m } \in Q_M \\ \text{s.t.} \quad & Score(m) > c_1, InferTime(m) < c_2, \\ & w_1 = 0.7, w_2 = 0.3, c_1 > 0, c_2 > 0 \end{aligned} \tag{1}$$

where Q is The GML inference query that is to be mapped into a single model m from a set of trained modes Q_M , c_1 and c_2 are constraints according to user preferences.

For large size $|Q_M|$, constrained multi-objective optimization is essential for optimal model selection during the inference time to enable accurate and fast *SPARQL*$^{\mathbf{ML}}$ query execution pipeline.

(***Phase 2***) : Optimizing for user-defined predicates ordering: Figures 6a, 6b shows two candidate SPARQL query for the SPARQL^{ML} in Figure 5. KGNet currently supports two possible execution plans. The core idea is to map a user-defined predicate into a user-defined function (UDF), such as *sql:UDFS.getNodeClass*. The RDF engine sends HTTP calls during the execution time to the GML inference manager to get the predictions based on the chosen model m in optimization phase 1. The number of HTTP calls may significantly influence the query execution time, corresponding to both the query and model cardinality. For example, SPARQL$^{\mathbf{ML}}_{pv}$ predicts the venue of all papers, whose size is |?papers|. The query template shown in Figure 6a will generate |?papers| HTTP calls. However, the query template shown in Figure 6b reduces the number of HTTP calls to a single call by enforcing an inner select query constructing a dictionary of all papers and their predicted venues. Then, *sql:UDFS.getKeyValue* is used to select the venue of each paper from an intermediate dictionary of all papers and their predicted venues. In a small-cardinality query, the cardinality of the query $|q_c| \ll |m_c|$, where $|m_c|$ is the model cardinality. In a large-cardinality query, where $|q_c| \approx |m_c|$, the single HTTP call form is more efficient.

```
select ?s from <http://kgnet>
where {?s a <kgnet:type/GMLTask>.}
```

(a) SPARQL query that Lists KGNet GML tasks.

```
select ?m from <http://kgnet>
where {?t a <kgnet:type/GMLTask>.
?t <kgnet:GMLTask/targetNode> dblp:Publication .
?t <kgnet:GMLTask/labelNode> dblp:venue .
?t <kgnet:GMLTask/modelID> ?m .}
```

(b) SPARQL query that lists the GML models of DBLP-PV Task.

```
select ?m ?p ?o from <http://kgnet>
where{ ?t a <kgnet:type/GMLTask>.
?t <kgnet:GMLTask/targetNode> dblp:Publication .
?t <kgnet:GMLTask/labelNode> dblp:venue .
?t <kgnet:GMLTask/modelID> ?m
?m <kgnet:GMLModel/GNNMethod> "G-SAINT".
?m ?p ?o. }
```

(c) SPARQL query that lists the attributes of trained Graph SAINT GML models for DBLP-PV task.

Figure 7: SPARQL queries to explore the KGMeta KG.

3 DEMONSTRATION OF GML USECASES.

3.1 Exploring KGMeta GML Models.

For ease of use and seamless integration, KGMeta KG allows users to explore KGNet GML models by writing simple SPARQL queries as in Figure 7. The user can explore the catalogue of GML tasks using a select query shown in Figure 7a and then explore the list of GML models trained for a GML task using a query in Figure 7b. The metadata of a trained GML model can be retrieved using a query like the one in Figure 7c.

3.2 GML Tasks

Table 1 shows five GML tasks for node classification (NC) and link prediction (LP). For every task the inference query is written in form of a SPARQL^{ML} query where the query engine is in charge of parsing, optimizing and re-writing that query into a SPARQL query. The SPARQL^{ML} optimizer selects the best model for each task by solving the IP maximization problem in Eq:1 for the available set of models by selecting them from the KGMeta KG. After model selection, The user-defined predicate ordering optimization is used to generate the final SPARQL query. The KGs data are loaded into Openlink-Virtuoso (V.07.20.32) RDF engine. The GML inference API is accessible through HTTP Restful *doPost* calls. The SPARQL^{ML}

Table 1: **Node classification (NC) and link prediction (LP) GML tasks. The SPARQL^{ML} query is optimized for the highest score, the lowest inference time, or both as integer programming (IP) optimization problems. IP is the default strategy in KGNet.**

Task	Type	KG	GML Method	Model Score(%)	Inference Time (sec)	SPARQL ^{ML} Optimization		
						Highest Score	Lowest Infer. Time	IP
Paper-Venue #Papers=1.2M	NC	DBLP	GraphSAINT	89.88	49			√
			RGCN	80.07	28		√	
			ShadowSAINT	90.89	64	√		
			SeHGNN	86.2	72			
Paper-Venue #Papers=703K	NC	MAG	GraphSAINT	81.08	31			√
			RGCN	69.49	19		√	
			ShadowSAINT	82.48	43	√		
			SeHGNN	76.4	45			
Fraud Detection #Transaction=590K	NC	IEEE-CIS	GraphSAINT	96.72	30			
			RGCN	96.62	15		√	
			ShadowSAINT	96.57	26			√
			SeHGNN	97.05	37	√		
Film Language #Films=26K	NC	Linked-MDB	GraphSAINT	83.9	15			
			RGCN	86.5	12	√	√	√
			ShadowSAINT	84.5	19			
			SeHGNN	85.4	26			
Connected Airports #Links=32.3K	LP	YAGO3-10	MorsE	85.27	51			√
			RGCN	29.12	23		√	
			LHGNN	99	79	√		
Author Affiliation #Links=108K	LP	DBLP	MorsE	89.07	113	√		√
			RGCN	10.05	62		√	
Person Occupation #Links=1.7M	LP	ogbl-wikikg2	MorsE	77	328	√		√
			RGCN	6	109		√	

query optimizer selects the model with the highest accuracy and reasonable inference time compared with the highest score and lowest inference time rule-based optimization shown in Table 1.

Node classification tasks: We demonstrate 3 node classification tasks where the inference query is written in SPARQL^{ML} syntax. For each task, we trained 4 GNN models using SOTA methods (RGCN, GraphSAINT, ShadowSAINT, and SeHGNN) in [1]. The evaluation metric is classification accuracy. The first task uses the DBLP[2] KG and predicts a publication venue for each paper. The second task similarly uses MAG[3] KG and predicts a publication venue for each paper. The third task uses the IEEE-CIS[4] dataset to predict whether an e-Commerce transaction is fraudulent or not. The fourth task uses the Linked-MDB[5] dataset to predict the language of a film.

Link Prediction tasks: We demonstrate three link prediction tasks. For each task, we trained 3 GNN models using SOTA methods (RGCN, LHGNN, and MorsE) in [1]. The evaluation metric is Hits@10. The first task uses the Yago3-10[6] dataset to predict a connected airport link between two airports. The second task similarly uses DBLP KG to predict an affiliation link between the author and the research institute. The third task uses the ogbl-wikikg2[7] dataset to predict the person's occupation link.

The presented tasks are varying in task type, model cardinality, KG size, used GML methods, and data domain that shows the applicability and ease of use of SPARQL^{ML}. We are enriching the KGNet system by adding more tasks and GML models for benchmarking the SPARQL^{ML} queries for further GML applications and ML-based query optimization for inference and insert queries.

[2]DBLP-03-2022: https://dblp.org/rdf/release/
[3]MAG-2020-05-29: https://makg.org/rdf-dumps/
[4]IEEE-CIS-Fraud Detection: https://www.kaggle.com/c/ieee-fraud-detection
[5]LinkedMDB: https://triplydb.com/Triply/linkedmdb/sparql/linkedmdb
[6]YAGO3-10: https://paperswithcode.com/dataset/yago3-10
[7]ogbl-wikikg2: https://ogb.stanford.edu/docs/linkprop/#ogbl-wikikg2

4 CONCLUSION

In this demo paper, we present our KGNet platform, which provides SPARQL^{ML} as a service to extend existing RDF engines to support querying and inferencing KGs using trained models. Our service maintains all the metadata collected from training GML models as a KG called KGMeta, which is stored alongside the domain-specific KG. This enables seamless integration between trained GML models and existing RDF engines. Executing SPARQL^{ML} query includes two main optimizations: i) selecting the model that achieves the highest accuracy with the lowest inference time, and ii) ranking ordering to optimize the inference query HTTP calls.

REFERENCES

[1] Hussein Abdallah and Essam Mansour. 2023. Towards a GML-Enabled Knowledge Graph Platform. In *2023 IEEE 39th International Conference on Data Engineering (ICDE)*. 2946–2954. https://doi.org/10.1109/ICDE55515.2023.00225

[2] Yingtong Dou, Zhiwei Liu, and et.al. 2020. Enhancing Graph Neural Network-based Fraud Detectors against Camouflaged Fraudsters. In *(CIKM)*. 315–324. https://doi.org/10.1145/3340531.3411903

[3] Joseph M. Hellerstein, Christopher Ré, and et.al. 2012. The MADlib Analytics Library or MAD Skills, the SQL. *ArXiv* abs/1208.4165 (2012).

[4] Aidan Hogan, Eva Blomqvist, and et.al. 2021. Knowledge Graphs. 54, 4, Article 71 (July 2021), 37 pages. https://doi.org/10.1145/3447772

[5] Ziyu Li and et.al. 2023. Optimizing ML Inference Queries Under Constraints. In *Web Engineering*. Springer Nature Switzerland, Cham, 51–66.

[6] Qiuru Lin, Sai Wu, and et.al. 2022. A Comparative Study of in-Database Inference Approaches. In *(ICDE)*. 1794–1807. https://doi.org/10.1109/ICDE53745.00180

[7] Xuan Lin, Zhe Quan, and et.al. 2020. KGNN: Knowledge Graph Neural Network for Drug-Drug Interaction Prediction. In *IJCAI*. https://doi.org/10.24963/380

[8] Immanuel Trummer and Christoph Koch. 2016. Multi-Objective Parametric Query Optimization. *SIGMOD Rec.* 45, 1 (2016), 24–31. https://doi.org/10.1145/2949741.2949748

[9] Shiwen Wu, Fei Sun, Wentao Zhang, Xu Xie, and Bin Cui. 2023. Graph Neural Networks in Recommender Systems: A Survey. *ACM Comput. Surv.* 55, 5 (2023), 97:1–97:37. https://doi.org/10.1145/3535101

[10] Xuanhe Zhou, Chengliang Chai, Guoliang Li, and Ji Sun. 2022. Database Meets Artificial Intelligence: A Survey. *IEEE TKDE* 34, 3 (2022), 1096–1116.