A Demonstration of KGLac: A Data Discovery and Enrichment Platform for Data Science

Ahmed Helal
Concordia University
{FN}.{LN}@concordia.ca

Mossad Helali
Concordia University
{FN}.{LN}@concordia.ca

Khaled Ammar
Borealis AI
{FN}.{LN}@borealisAI.com

Essam Mansour
Concordia University
{FN}.{LN}@concordia.ca

ABSTRACT

Data science growing success relies on knowing where a relevant dataset exists, understanding its impact on a specific task, finding ways to enrich a dataset, and leveraging insights derived from it. With the growth of open data initiatives, data scientists need an extensible set of effective discovery operations to find relevant data from their enterprise datasets accessible via data discovery systems or open datasets accessible via data portals. Existing portals and systems suffer from limited discovery support and do not track the use of a dataset and insights derived from it. We will demonstrate KGLac, a system that captures metadata and semantics of datasets to construct a knowledge graph (GLac) interconnecting data items, e.g., tables and columns. KGLac supports various data discovery operations via SPARQL queries for table discovery, unionable and joinable tables, plus annotation with related derived insights. We harness a broad range of Machine Learning (ML) approaches with GLac to enable automatic graph learning for advanced and semantic data discovery. The demo will showcase how KGLac facilitates data discovery and enrichment while developing an ML pipeline to evaluate potential gender salary bias in IT jobs.

PVLDB Reference Format:
Ahmed Helal, Mossad Helali, Khaled Ammar, and Essam Mansour. A Demonstration of KGLac: A Data Discovery and Enrichment Platform for Data Science. PVLDB, 14(12): 2675 - 2678, 2021.
doi:10.14778/3476311.3476317

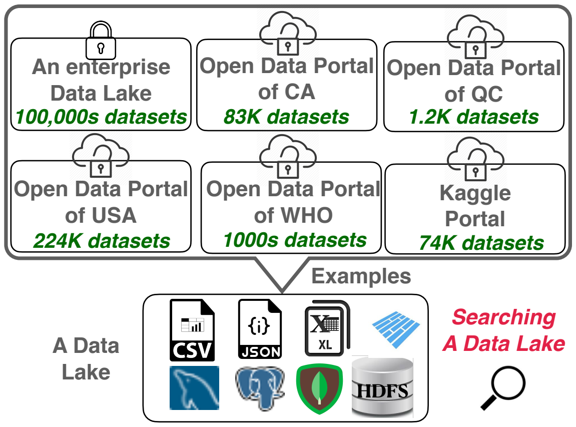

Figure 1: In a data lake, data owners, e.g., countries, provinces, or organizations, collect raw datasets as a results of open data initiatives or for data science projects. Data discovery is a challenging task due to the limited search support of open data portals or search engines, in case of public datasets, and data discovery systems, in case of enterprise datasets. Data owners have no support to track the use of their datasets.

1 INTRODUCTION

Data discovery is an essential task in data science to identify possible datasets relevant to a data science project. Due to the data science growing success and open data initiatives, thousands of machine-readable and structured datasets are collected by data owners in a data storage known as a data lake, as shown in Figure 1. Data owners make these datasets available via data discovery systems in the case of enterprise datasets or via data portals, such as the USA[1]Canada[2], Quebec[3], the World Health Organization[4], and Kaggle[5].

Data portals and search engines, such as Google Dataset Search, provide primitive search capabilities to find and download open datasets in different formats, such as CSV, JSON, and XML. Thus, data integration and enrichment are the primary responsibility of data scientists, who spend most of their time finding, downloading, preparing, and integrating relevant datasets with little or no support from these portals, engines, or existing data science platforms. Moreover, many organizations are encouraged to build a navigational data structure (data catalogue) to support data discovery [2, 4, 8] or to use tools such as Amundsen[6]. Unfortunately, these systems and tools suffer from limited query support and cannot find data items based on learned representations (embeddings).

Furthermore, several methods have been proposed to measure table relatedness [11], support table discovery [3], identify unionable tables [9], and find joinable tables [13]. These methods work in isolation from each other and from data portals and discovery systems. Thus, there is a need for data portals and discovery systems with a flexible query language and an extensible set of discovery operations. Moreover, existing data science platforms, such as MLFlow or Cloud AutoML, and tools, such as Jupyter Notebooks or Google Colab, should be able to communicate easily with these portals and systems.

Proceedings of the VLDB Endowment, Vol. 14, No. 12 ISSN 2150-8097.
doi:10.14778/3476311.3476317

[1]https://www.data.gov/
[2]https://open.canada.ca/
[3]https://www.donneesquebec.ca/fr/
[4]https://www.who.int/data/gho
[5]https://www.kaggle.com/datasets
[6]https://www.amundsen.io/

In this demo, we present KGLac, a data discovery and enrichment platform empowered by knowledge graph technologies and a broad range of ML approaches, including Graph Neural Networks (GNNs) [10, 12]. KGLac is supported by different methods for data profiling and representation learning (embedding) to capture metadata and semantics of datasets to construct a knowledge graph (GLac). KGLac provides a set of data discovery operations implemented using SPARQL queries. This set is extensible as KGLac supports ad-hoc queries. KGLac enables automatic graph learning to advance functionalities, such as classification of similar data items, finding unionable and joinable tables, predicting shortest paths between tables, and inferring new relationships.

We designed KGLac to be deployed on top of a data owner's local data lake to enable efficient and extensible data discovery operations for data scientists, who have access to the data lake. The data owner can track the use and insights derived from the datasets. This tracking support will enable discoveries beyond the original intent of a dataset. In this paper, Section 2 highlights KGLac architecture. Section 3 describes a demonstration scenario for using KGLac with Jupyter Notebooks to develop an ML pipeline analyzing gender salary bias in IT jobs. Section 4 concludes.

2 OVERVIEW OF KGLAC

The KGLac architecture is illustrated in Figure 2. KGLac consists of two main components: (i) GLac Construction which constructs a GLac based on learned representations (embeddings) generated for data items using our Data Profiler, and (ii) Interface Services which provide an extensible set of data discovery operations based on SPARQL queries and enable searching data items based on embedding similarity.

2.1 GLac Construction

KGLac empowers the data discovery and enrichment process using knowledge graph (KG) technologies to construct a navigational data structure, which we call GLac, as a knowledge graph based on RDF-star [6]. KGLac leverages KG benefits such as (i) semantic formalization using a controlled vocabulary for ensuring interoperability, (ii) schema-agnostic allowing the platform to support reasoning and semantic manipulation, e.g., adding new labelled edges between related tables or columns, (iii) recent graph model (RDF-star [6]), which extends the RDF model to annotate nodes painlessly, (vi) powerful query language (SPAQRL) and its extension SPAQRL-star [6].

Data Profiler: KGLac's Data Profiler breaks down available datasets into tables and columns to identify similarities and relationships at different granularities. We used state-of-the-art data profiling techniques [1] and a deep learning model [7] to profile datasets and generate fixed-size, dense column representations (embeddings). Our profiler analyzes datasets at the level of individual columns and utilizes PySpark DataFrame to better leverage distributed systems and scale to vast datasets. PySpark DataFrame supports different data formats, such as CVS, JSON, and Parquet, and connects to various database systems. KGLac stores the profiling information per column in a document database. This information is used by the `GLac Builder`. In KGLac, the embedding of a table or a dataset is an aggregation of its columns or tables' embeddings, respectively.

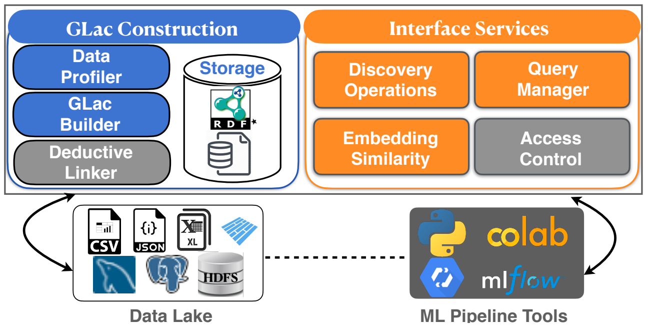

Figure 2: The KGLac architecture, where KGLac gets access to a local data lake, e.g., sets of files or databases, to construct GLac. Then, different ML pipeline tools can communicate with KGLac to facilitate data discovery. KGLac tracks the use of datasets. The gray boxes are not part of this demo.

KGLac could be configured to store the raw data of each column in the document database. In such a case, KGLac will support discovery based on keyword search on a column content.

GLac Builder: In GLac, vertices represent data nodes, such as a node of type dataset, table, or column, while edges represent relationships between these nodes. A data node is identified by a URI and does not contain any raw data. KGLac detects similarity, i.e., edges, between a pair of nodes of the same type based on embedding similarity between the pair. KGLac generates two types of embeddings. The first type is based on a column's raw content and is used to identify content similarity relationships. For the first type, we developed a model inspired by [7]. The second type of embeddings is based on column or table names to identify semantic or schema similarity relationships. For the second type, we utilized a method based on Word Embeddings [5]. In KGLac, other kinds of edges connect a column to its table or a table to its dataset.

Lac Ontology: We developed the Lac [7] ontology, which conceptualizes relationships and entities in data lakes. For example, suppose column A has a contentSimilarity with column B. In this case, we represent it as a triple: ⟨`lac:A`, `lac:contentSimilarity`, `lac:B`⟩ where lac:contentSimilarity accepts lac:Column as domain and range. To annotate the relationships, we represent triples in RDF-star, e.g., if we are 75% certain that column A has a content similarity with column B, we add ⟨⟨`lac:A lac:contentSimilarity lac:B`⟩, `lac:certainty`, `0.75`⟩. Figure 3 illustrates part of a GLac graph. Lac could be used and extended by different data discovery systems to model their navigational data structure.

Deductive Linker: In KGLac, the local datasets newly added to the data lake are profiled individually to construct a sub-graph for them. Then, KGLac identifies links between nodes in the sub-graph and nodes in the main GLac. Thus, our GLac is a deductive graph that utilizes inference rules and automatic graph learning methods to incrementally introduce and enhance the relationships among

[7]Lac means lake in French

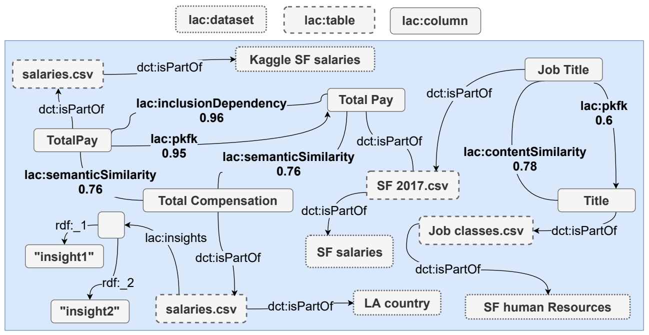

Figure 3: A GLac graph has three classes: Column, Table, and Dataset. KGLac detects different content-based relations, such as content similarity, primary key foreign key (PKFK) and inclusion dependency, and schema-based relations, such as semantic similarity based on the column names.

the different nodes in the graph. We solve this problem as a link prediction problem using GNN-based models.

2.2 Interface Services

GLac is hosted on an RDF-star engine, such as Blazegraph or Apache Jena. To interact with GLac, KGLac supports various data discovery operations via both predefined and ad hoc SPARQL queries. KGLac formulates the query results as a Pandas Dataframe, which is widely used in data science pipelines. Thus, different data science platforms can easily interact with KGLac. Examples of the KGLac discovery operations are finding: a) tables or columns based on schema similarity, b) a common schema to union two tables, c) columns to be used in joining two tables, d) paths between two tables. In addition to these operations, KGLac enables users to annotate data nodes with insights and descriptions.

KGLac also supports embedding similarity search, i.e., finding a table or column based on the stored embeddings. For example, KGLac can generate embeddings for a given Dataframe and search the document database for similar embeddings, i.e., discover datasets based on content similarity. On the one hand, this functionality enables an enterprise to allow data scientists to search all the enterprise datasets without exposing the raw data. On the other hand, we have a fixed size vector regardless of the raw data's actual size. Thus, this method can scale easily to large datasets. In KGLac, GLac could be annotated using SPARQL update queries to interlink triples or data nodes with insights derived from data science projects benefiting from these datasets.

2.3 KGLac in Use

After deploying KGLac, a data owner needs first to profile the local datasets and construct a GLac. The next step is to enable data scientists to access KGLac via its interface library to utilize KGLac's data discovery operations and query the GLac or the generated embeddings. KGLac is not a static platform. As more datasets are added, KGLac continuously and incrementally maintains the GLac.

Table 1: A schema similarity evaluation.

Systems	Precision	Recall	Macro F1
KGLac	0.89	0.69	0.78
Aurum	0.79	0.26	0.39

The KGLac portal might have access restrictions to prevent unauthorized users or could be public for anyone. Authorized users have access to query GLac or embeddings. However, accessing the actual data files in the data lake may need another level of authorization. KGLac can provide a `get` operator to extract raw data of a data node, i.e., dataset or table, in case of open datasets, e.g., open data portals of Quebec or Kaggle.

Our platform will open a new business model for companies selling datasets, i.e., the cost of datasets could be per access instead of selling an entire dataset. Moreover, KGLac will allow data owners to track the use of their data nodes and enable better utilization of them to maximize data science potentials.

3 DEMONSTRATION SCENARIO

We will demonstrate that KGLac could be seamlessly integrated into data science pipelines to facilitate data discovery and enrichment. During the demonstration, the audience will experience several aspects of KGLac using real datasets of topics most of them are familiar with, such as income, job classifications, and gender. We collected more than 125 real datasets from sources, such as Kaggle and governmental open data portals. The collected datasets contain thousands of columns of different data types. In our main scenario, a data scientist has access to this data lake and wants to examine salary-related issues, such as potential gender pay gap or salary bias. The audience may find it easy to adjust the main scenario to explore different issues.

As a preprocessing step, KGLac profiled the data lake and stored the profiling information including the generated embeddings into Elasticsearch, i.e., our document store. Using the collected datasets, Table 1 illustrates the outstanding performance of KGLac w.r.t Aurum due to KGLac's data cleaning and context-based semantic affinity. KGLac, then, constructed the GLac, a knowledge graph representing a global schema for the data lake. The constructed GLac is stored in Blazegraph, i.e., our RDF-star engine. We developed our data profiling sub-system on top of Apache Spark to easily scale to large datasets. The data scientist uses KGLac interface services in Jupyter Notebook for the data discovery and enrichment phase. KGLac enables embedding similarity search to discover tables or columns that have similar representations without revealing the raw data. The data scientist can also annotate datasets or tables with derived insights.

3.1 Find Salaries Datasets

KGLac enables keyword search to find datasets, tables, or columns of similar names. KGLac returns the results encapsulated in a Pandas Dataframe that has information, such as origin, number of rows and columns, and paths to the physical CSV files. To start, the data scientist looks for tables with *column names* similar to `pay` or `salary`. This could be expressed using KGLac's discovery operation as the following:

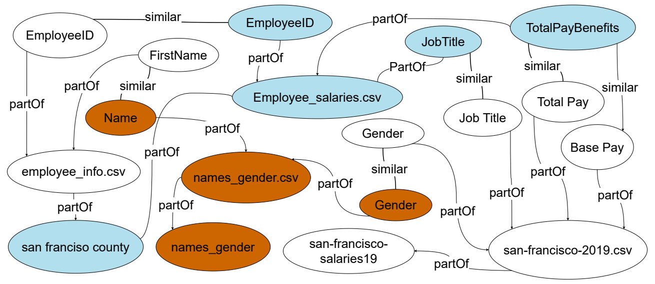

Figure 4: There are two paths between the selected tables.

```
stables = KGLac.search_tables_on([['salary', 'pay']])
```

With this successful start, the user discovered tables related to salaries. Using native Pandas APIs, the user can examine the table with the largest number of rows that include employee IDs, job titles, and salaries. The discovered salary tables do not have information regarding gender. Thus, the next step is to discover tables containing information regarding the person name and gender using a similar KGLac operation to find tables with information about `gender` or `sex`. The user finds a dataset that includes names and gender and then uses KGLac to discover how this table could be merged with the salaries dataset chosen before.

```
gtables = KGLac.search_tables_on([['gender', 'sex']])
```

3.2 Join Path Discovery

KGLac leverages its knowledge graph to find potential join paths between any two tables. The data scientist, in this scenario, uses the discovery of intermediate joinable tables feature. KGLac discovers intermediate tables, e.g., number of hops, and shows the connection between them. This could be expressed as the following:

```
KGLac.get_path_between_tables(
          stables.iloc[0],gtables.iloc[3], hops=2)
```

KGLac discovers join paths between two tables using a SPARQL query. We also support more challenging queries, such as the shortest path between nodes, e.g., tables. Figure 4 shows the intermediate tables connecting the salary and gender tables. There is a path that maps employee IDs to their names. The user can easily use Pandas DataFrame to join the three tables to infer the employees' genders.

3.3 Discover Unionable Columns

While the user finding a path between the salary and gender table, she finds an opportunity to union the selected salary table with another table that contains information related to salary (`Total Pay`) and gender, as shown in Figure 4. KGLac provides support to find a common schema, i.e., columns that could be used to union two tables. This is very useful as tables may have a large number of columns. As Pandas DataFrame, the user can union the tables using the recommended common schema by KGLac.

3.4 Embedding Similarity Search

The remaining step is to enrich the dataset with job fields, which requires a table containing a mapping from job titles to job categories such as IT, health, and sales. After the union of the two tables, the user introduces a DataFarme containing data that are not seen together for a table by KGLac. KGLac enables a user to generate column embeddings on the fly for columns in a given DataFrame. The data scientists using embedding similarity search operation will find a table of job titles and classifications to join with the current table. After joining the tables based on the job title column, the data scientist manages to get a DataFrame with the required information namely, salary, gender, and job field and can filter records belonging only to IT jobs. Finally, the data scientist optionally leaves a review of the datasets used to build their pipeline, which might help other practitioners work on similar problems.

4 CONCLUSION

In this paper, we demonstrate KGLac, a holistic system that helps organizations leverage the rich data they have in their data lake by profiling them and creating a knowledge graph for data integration. KGLac allows organizations to increase the efficiency of their data science team. KGLac can play a pivotal role in allowing different teams to check data across different departments based on the column embeddings without having access to the data. In KGLac, data discovery based on embeddings similarity enables a new horizon towards responsible data science, allowing enterprises to unleash their teams' innovation potential without sacrificing privacy.

KGLac offers a set of discovery operations that interact with the constructed knowledge graph and query the embedding generated to the datasets. KGLac could be easily integrated into existing data science platforms. We built KGLac to be a platform with state-of-the-art algorithms assisting data scientists in data discovery, data integration, data exploration, and data enrichment. KGLac empowers data discovery on data lakes using knowledge graph technologies. We are developing different methods to enable automatic graph learning for advanced and semantic data discovery.

REFERENCES

[1] Ziawasch Abedjan, Lukasz Golab, Felix Naumann, and Thorsten Papenbrock. 2018. *Data Profiling*.
[2] Alex Bogatu, Alvaro A. A. Fernandes, Norman W. Paton, and Nikolaos Konstantinou. 2020. Dataset Discovery in Data Lakes. In *ICDE*.
[3] Christina Christodoulakis, Eric Munson, Moshe Gabel, Angela Demke Brown, and Renée J. Miller. 2020. Pytheas: Pattern-based Table Discovery in CSV Files. *PVLDB* 13, 11 (2020).
[4] Raul Castro Fernandez, Ziawasch Abedjan, Famien Koko, Gina Yuan, Samuel Madden, and Michael Stonebraker. 2018. Aurum: A Data Discovery System. In *ICDE*.
[5] Josu Goikoetxea, Eneko Agirre, and Aitor Soroa. 2016. Single or Multiple? Combining Word Representations Independently Learned from Text and WordNet. In *AAAI*.
[6] Olaf Hartig. 2019. Foundations to Query Labeled Property Graphs using SPARQL. In *SEM4TRA-AMAR@SEMANTICS*, Vol. 2447.
[7] Jonas Mueller and Alex Smola. 2019. Recognizing Variables from Their Data via Deep Embeddings of Distributions. In *ICDM*.
[8] Fatemeh Nargesian, Ken Q. Pu, Erkang Zhu, Bahar Ghadiri Bashardoost, and Renée J. Miller. 2020. Organizing Data Lakes for Navigation. In *SIGMOD*.
[9] Fatemeh Nargesian, Erkang Zhu, Ken Q. Pu, and Renée J. Miller. 2018. Table Union Search on Open Data. *PVLDB* 11, 7 (2018).
[10] Z. Wu, S. Pan, F. Chen, G. Long, C. Zhang, and P. S. Yu. 2020. A Comprehensive Survey on Graph Neural Networks. *TNNLS*.
[11] Yi Zhang and Zachary G. Ives. 2020. Finding Related Tables in Data Lakes for Interactive Data Science. In *SIGMOD*.
[12] Z. Zhang, P. Cui, and W. Zhu. 2020. Deep Learning on Graphs: A Survey. *The IEEE Transactions on Knowledge and Data Engineering (TKDE)* (2020). https://doi.org/10.1109/TKDE.2020.2981333
[13] Erkang Zhu, Dong Deng, Fatemeh Nargesian, and Renée J. Miller. 2019. JOSIE: Overlap Set Similarity Search for Finding Joinable Tables in Data Lakes. In *SIGMOD*.