Demonstration

Galaxy: A Platform for Explorative Analysis of Open Data Sources

Seyed-Mehdi-Reza Beheshti [#], Boualem Benatallah [#], Hamid Reza Motahari-Nezhad [#,*]

[#] *University of New South Wales, Sydney, Australia*
{sbeheshti,boualem,hamidm}@cse.unsw.edu.au

[*] *IBM, Almaden Research Center, San Jose, USA*
motahari@us.ibm.com

ABSTRACT

A large volume of Open Data is being generated on a continuous basis. Examples of this are the case of social, natural, and information systems such as World Wide Web and social networks. Most entities and objects in the Open Data are interconnected, forming a complex, semi-structured, and information-rich networks. In this sense, Linked Open Data has the potential to be similar to a federated database. Since Linked Open Data is based on W3C standards, it is possible to implement a federation infrastructure, however, the current SPARQL standard makes it challenging to analyze the Open Data in an explorative manner. Consequently, it will be hard to discover the hidden knowledge in the relationships among entities in Open Data sources. In this paper, we present Galaxy, a platform for explorative analysis of Open Data Sources. Galaxy facilitates the analysis of Open Data graphs based on simple abstractions, i.e. folders and paths, which enable an analyst to group related entities in the graph or find paths among entities. Galaxy uses Hadoop data processing platforms to store and retrieve large numbers of RDF triples and to support cost-effective and Web-scale processing of Semantic Web data through a Folder-Path enabled extension of SPARQL.

Keywords

Linked Data, Open Data Analytics, Querying Graphs

1. INTRODUCTION

Open Data sources may include any information that can be obtained without a privileged position. Examples include electronic and print media (e.g. RSS feeds from newspapers), social media (Twitter, Facebook, Instagram, YouTube), and blog sites (e.g. Tumblr, Wordpress). The production of knowledge from Open Data is seen by many organizations as an increasingly important capability that can complement the traditional intelligence sources. In particular, most entities and objects in the Open Data are interconnected, forming a complex, semi-structured, and information-rich networks which can be modeled using graphs. In this sense, Linked Open Data has the potential to be similar to a federated database: combining these data sources offer a rich information resource for enterprise analysis.

Since Linked Open Data is based on W3C standards (e.g. RDF format and the SPARQL query language), it is possible to implement a federation infrastructure, however, the current SPARQL standard makes it challenging to analyze the Open Data in an explorative manner. Consequently, it will be hard to discover the hidden knowledge in the relationships among entities in Open Data sources. For example it is important to quickly form an intelligence picture from the Open Data sources around a topic of interest (such as country, person, organization or event), group related entities around that topic, find paths among entities, and use all these information for the follow-on analysis. There is a need for graph representation models and efficient approaches for expressing and executing these types of queries. In particular, manipulating, querying, and analyzing Linked Open Data graphs to discover new knowledge is of high interest.

In this paper, we present Galaxy, a platform for explorative analysis of Open Data Sources. Galaxy helps in facilitating the analysis of Open Data graphs based on simple abstractions, i.e. folders and paths (introduced in our earlier work [4]), which enables an analyst to group related entities in the graph or find paths among entities. A folder node contains a set of entities that are related to each other, i.e., the set of entities in a folder node is the result of a given query that requires grouping graph entities in a certain way. We define a path node for each query that results in a set of paths (i.e. transitive relationship between two entities which can be codified using regular expressions). Folder and Path nodes, can represent a network snapshot, i.e. a subgraph, from multiple perspectives and granularities. Folder and Path nodes can be timed [3]: Timed folder/path nodes can show their evolution for the time period that they represent. Galaxy uses Hadoop data processing platforms to store and retrieve large numbers of RDF triples and to support cost-effective and Web-scale processing of Semantic Web data through a Folder-Path enabled extension of SPARQL.

The rest of the paper is organized as follows. In Section 2, we present some key components of our system, while in Section 3 we describe our demonstration scenario.

©2016, Copyright is with the authors. Published in Proc. 19th International Conference on Extending Database Technology (EDBT), March 15-18, 2016 - Bordeaux, France: ISBN 978-3-89318-070-7, on OpenProceedings.org. Distribution of this paper is permitted under the terms of the Creative Commons license CC-by-nc-nd 4.0

2. SYSTEM OVERVIEW

Figure 1 represents the architecture of the Galaxy. The main components of the system include the Extracted Data Folder and the Graph Query Engine.

Extracted Data Folders. Open data are complex, unstructured and generated at a high rate, resulting in many challenges to ingest, store, index, and analyze such data efficiently. The notion of *extracted data folders* serves to enable the ingestion of data (from open data sources), and the persistence of this data in accordance with a particular defined schema. Machine learning techniques can be used to construct the schema for an open data source [5]. We assume that the expert analysts will construct the schema for each folder. Figure 4 illustrates the Twitter schema where the main entities include users, tweets, links, domains, and hashTags. Folders provide a federated data access infrastructure upon which the federated analysis will operate. We also envisage folders to support multiple layers of granularity (e.g., split or merge existing folders). Folders can also be combined to create higher-level virtual folders, called federated folders, using filter, project and join operators.

Graph Query Engine (SPARQL extension). Due to space restrictions, in this paper we highlight the main component of the query engine. However, we refer to our paper [2] for algorithmic and other details. Figure 2 presents the graph processing architecture which consists of the following components: graph loader, data mapping layer, query mapping layer, regular expression processor, time-aware controller, and OLAP (on-line analytical processing) controller.

Graph Loader. Input graph (e.g. Twitter extracted folder) can be in the form of RDF, N3, or XML. We developed a workload physical design by developing a loader algorithm. This algorithm is responsible for: (i) validating the input graph; and (ii) generating the triples, where two types of triples are recognized: attribute-edges (e.g., "Bob @age 35") and relationship-edges (e.g., "Bob knows Fred"). We use the '@' symbol for representing attribute edges and distinguishing them from the relationship edges.

Data Mapping Layer. This layer is responsible for creating: (i) object-store, which contains all objects in the input graph uniquely identified by an identifier. Each object contains an arbitrary list of attribute-edges describing its features; (ii) link-store, which contains all directed links between pairs of objects represented as relationship-edges; and (iii) data element mappings between semantic web technology (i.e. Resource Description Framework) and Hadoop file system. As a result, objects-store and link-store will be stored in Hadoop cluster.

Query Mapping Layer. This layer is consist of a parser for parsing SPARQL like queries (based upon the syntax of Folder-Path extension [2, 3, 4] of SPARQL) and a SPARQL-to-PigLatin translation algorithm. In order to translate the SPARQL queries into Pig-Latin we follow a specific format in which data is read from the HDFS, a number of Pig-Latin operations (e.g., LOAD, SPLIT, JOIN, FILTER, GROUP, and STORE) are performed on the data, and then the resulting relation is written back to the file system. In particular, SPARQL graph pattern matching is dominated by join operations, and is unlikely to be efficiently processed. We use existing query optimization techniques [7, 8, 9] to generate the optimal query plan by reinterpreting certain join tree structures as grouping operations, i.e., to enable a greater degree of parallelism in join processing. In the

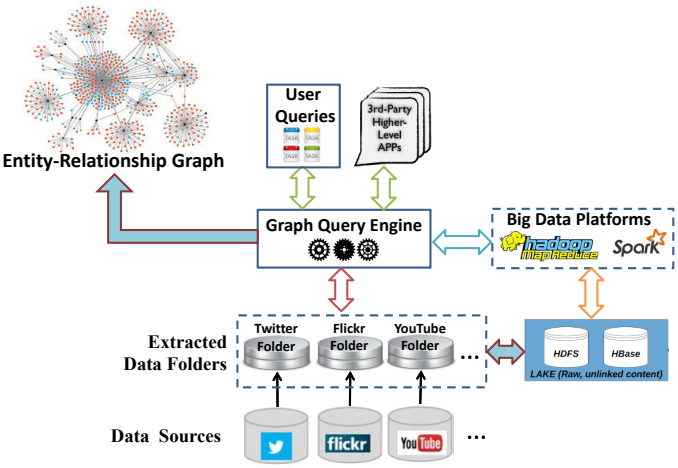

Figure 1: The Galaxy Architecture.

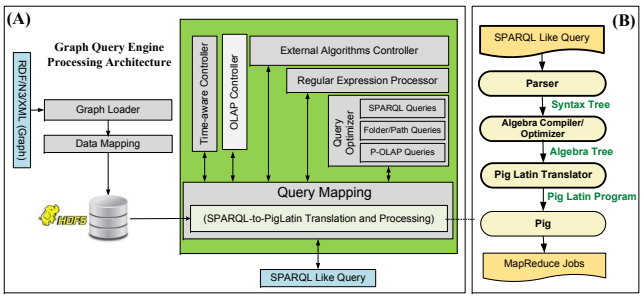

Figure 2: The Query Engine Architecture.

following, we illustrate an example for a sample mapping between SPARQL and Pig-Latin.

Example 1. DBLP[1] is an open source data for computer science bibliographical network. Adam, an OLAP analyst, is interested in partitioning the DBLP graph into a set of authors having same interests. Then he plans to apply a set of OLAP style operations (e.g., calculating authors ranking and contribution degree) on constructed partitions. Details about this example, including the SPARQL query can be found in [2]. Processing this query using Pig Latin's query algebra, results in the query plan shown in Figure 3. The logical plan can be described as follows: (1) load the input dataset using the LOAD operator in Pig-Latin; (2) split the dataset, based on the partitioning condition, and create triple tables for related predicates. Next step is to filter the dataset into related authors, where the 'interest' triplestore will be needed for the partitioning phase and 'publications' and 'citations' triplestores will be needed to apply OLAP style operations on partitions; (3) filter the graph using the result of previous step, i.e., to support the triple syntax and weave the predicated to related partitions. Notice that, in the case of using JOIN operator in this step, the triple syntax will be no longer available; (4) group by the interest table on the object column to remove redundant values, e.g., cases where two or more authors, different subjects, having same interests; (5) evaluation of OLAP operations on graphs independently for each partition, providing a natu-

[1]http://dblp.uni-trier.de/db/

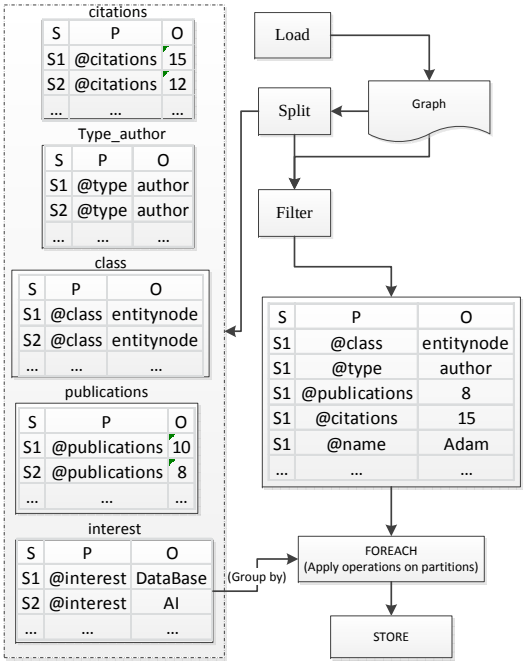

Figure 3: Query plan for the Example 1.

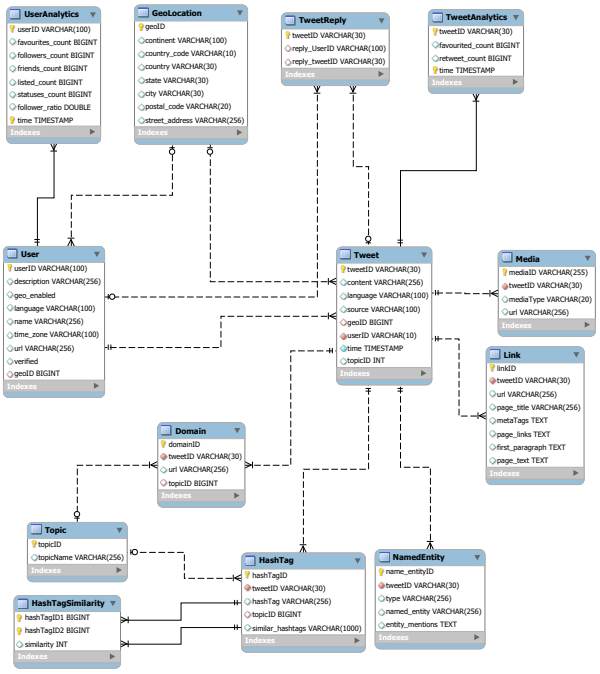

Figure 4: Twitter Extraction Folder Schema.

ral parallelization of execution; and (6) store the final result on Hadoop cluster using the STORE operator in Pig-Latin.

Regular Expression Processor. This component is responsible for parsing graph patterns. In particular, graph analysts can codify their knowledge into regular expressions that describe paths through the nodes and edges in the graph. The regular expression processor supports optional elements (?), loops (+,*), alternation (|), and grouping ((...)).

Time-aware Controller. RDF databases are not static and changes may apply to graph entities (i.e. nodes, edges, and folder/path nodes) over time. Time-aware controller is responsible for data changes and incremental graph loading. Moreover, it creates a monitoring code snippet and allocate it to a folder/path node in order to monitor its evolution and update its content over time.

GOLAP controller. This component is responsible for supporting on-line analytical processing on graphs, through partitioning graphs (using folder and path nodes) and allows evaluation of OLAP operations on graphs independently for each partition, providing a natural parallelization of execution, details can be found in [2].

External Algorithms Controller. This component is responsible to support applying existing graph mining algorithms (e.g. graph reachability and shortest path) to the open data graph, and store the result in a folder/path node for the follow on analysis. For example we developed interfaces to support various graph mining algorithms [1] such as Transitive Closure, GRIPP, Tree Cover, Chain Cover, Path-Tree Cover, and Shortest-Paths.

3. DEMONSTRATION SCENARIO

The demonstration scenario consists of three parts. First, we would like that the attendee appreciates the difficulties that one can encounter when dealing with open data sources. We start with a Twitter dump[2] and illustrate how we generate the Twitter extraction folder according to the generated Twitter schema (Figure 4). Next, we illustrate how we use the query language to construct content-based relationships among related entities. When talking about content we mainly deal with entity attributes, where we consider content-based relationships as correlation condition-based relationships. For example, a correlation condition in Twitter may enable grouping entities in different ways, e.g. Tweets coming from the same location or users from the same timezone, and store them in folder nodes. Figure 5 presents the set of related tweets whose location country is the same as Australia.

Next, we present to the attendee an interactive scenario where she would be able to generate the 'influence graph' among users in the Twitter open data through the following steps: (i) Using Folder Nodes, to form an intelligence picture from the Twitter data around a topic of interest (i.e. Twitter User) by grouping related entities around that topic and store them in folder nodes. Examples are, folders for users who: (a) belong to the same location, (b) tweet the same topics (we assume that we have a topic discovery algorithm), (c) use similar hashTags/links in their tweets; and (d) retweet similar tweets; (ii) Using Path Nodes to construct relationships among twitter constructed folders. For example a reachability algorithm will be used (in path node queries) to see if two different twitter users are reachable through a shard friend or a retweet path. As the result set of related patterns can be stored in path nodes for further analysis; (iii) Constructing relationships among open/federated data sources. We will provide the attendee with a set of folders constructed from other open data sources such as Wikidata and DBPedia. The attendee will use query templates

[2]https://archive.org/details/archiveteam-twitter-stream-2012-02

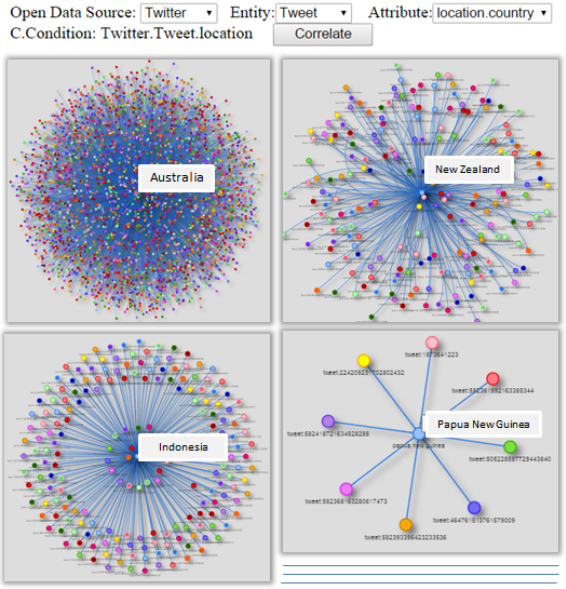

Figure 5: An example of partitioning the graph for the follow on analysis.

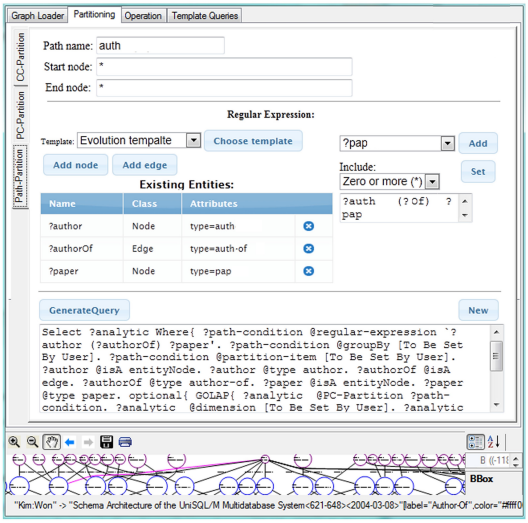

Figure 6: Screenshots of the front-end tool: assisting user to generate regular expressions.

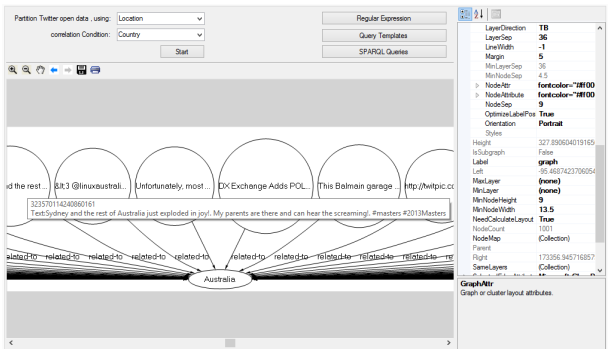

Figure 7: Screenshots of the front-end tool: assisting user to generate correlation conditions.

to discover similarity among different folders, e.g. a tweet in Twitter folder can be related to a topic in Wikidata folder; and (iv) Generating the 'influence graph' among users in Twitter. The attendee will use SPARQL like queries to further analyze the folder and path nodes and use the front-end tool to visualize the influence graph. In order to facilitate creating SPARQL queries, we provide a front-end tool for assisting users to create SPARQL queries in an easy way. Figures 6 and 7 illustrates screenshots of the front-end tool,

4. CONCLUSION AND FUTURE WORK

In this paper, we presented Galaxy, a platform for explorative analysis of Open Data Sources. Galaxy assists the analysts to quickly form an intelligence picture from the Open Data sources around a topic of interest, group related entities around that topic (folder nodes), find paths among entities (path nodes), and use all these information for the follow on analysis. Galaxy uses Hadoop data processing platforms to store and retrieve large numbers of RDF triples in Hadoop file system. As future work, we plan to make use of interactive graph exploration and visualization techniques which can help users to quickly identify the interesting parts of a graph.

5. ACKNOWLEDGEMENTS

We Acknowledge the Data to Decisions CRC (D2D CRC) and the Cooperative Research Centres Programme for funding part of this research.

6. REFERENCES

[1] C. C. Aggarwal and H. Wang. *Managing and Mining Graph Data*. Springer Publishing Company, Incorporated, 2010.

[2] S.-M.-R. Beheshti, B. Benatallah, and H. Motahari-Nezhad. Scalable graph-based olap analytics over process execution data. *Distributed and Parallel Databases*, pages 1–45, 2015.

[3] S.-M.-R. Beheshti, B. Benatallah, and H. R. M. Nezhad. Enabling the analysis of cross-cutting aspects in ad-hoc processes. In *CAiSE*, pages 51–67, 2013.

[4] S.-M.-R. Beheshti, B. Benatallah, H. R. M. Nezhad, and S. Sakr. A query language for analyzing business processes execution. In *BPM*, pages 281–297, 2011.

[5] A. Doan, P. Domingos, and A. Y. Halevy. Reconciling schemas of disparate data sources: A machine-learning approach. In *ACM Sigmod Record*, volume 30, pages 509–520. ACM, 2001.

[6] M. Fayzullin, V. Subrahmanian, M. Albanese, C. Cesarano, and A. Picariello. Story creation from heterogeneous data sources. *Multimedia Tools and Applications*, 33(3):351–377, 2007.

[7] M. F. Husain, L. Khan, M. Kantarcioglu, and B. M. Thuraisingham. Data intensive query processing for large rdf graphs using cloud computing tools. In *IEEE CLOUD*, pages 1–10, 2010.

[8] H. Kim, P. Ravindra, and K. Anyanwu. From sparql to mapreduce: The journey using a nested triplegroup algebra. *PVLDB*, 4(12):1426–1429, 2011.

[9] P. Ravindra, H. Kim, and K. Anyanwu. An intermediate algebra for optimizing rdf graph pattern matching on mapreduce. In *ESWC*, pages 46–61, 2011.