KBox: Distributing Ready-to-query RDF Knowledge Graphs

Conference Paper · May 2017

DOI: 10.1007/978-3-319-70407-4_11

4 authors:

Edgard Marx
Hochschule für Technik, Wirtschaft und Kultur Leipzig
44 PUBLICATIONS **221** CITATIONS
SEE PROFILE

Tommaso Soru
University of Leipzig
41 PUBLICATIONS **193** CITATIONS
SEE PROFILE

Ciro Baron Neto
University of Leipzig
17 PUBLICATIONS **172** CITATIONS
SEE PROFILE

Sandro Coelho
Universidade Federal Fluminense
8 PUBLICATIONS **98** CITATIONS
SEE PROFILE

KBox: Distributing Ready-to-query RDF Knowledge Graphs

Edgard Marx[1,2], Tommaso Soru[2], Ciro Baron[2], and Sandro Athaide Coelho[3]

[1] Leipzig University of Applied Sciences (HTWK), Germany
[2] AKSW, University of Leipzig, Germany
[3] Universidade Federal de Juiz de Fora, Brazil
http://kbox.tech

Abstract. The Semantic Web community has successfully contributed to a remarkable number of RDF datasets published on the Web. However, to use and build applications on top of Linked Data is still a cumbersome and time-demanding task. We present KBox, an open-source platform that facilitates the distribution and consumption of RDF data. We show the different APIs implemented by KBox, as well as the processing steps from a SPARQL query to its corresponding result. Additionally, we demonstrate how KBox can be used to share RDF knowledge graphs and to instantiate SPARQL endpoints.

1 Introduction

The advances in the Web of Data lead to an avalanche of open knowledge graphs made available in RDF format, such as DBpedia [3], Freebase [1] and Wikidata [7]. Together, these knowledge graphs encompass millions of facts from a multitude of domains. However, consuming RDF data is still a very cumbersome and time-demanding task.

SPARQL endpoints must handle very complex operations [5] which makes high demand services expensive to host and difficult to maintain [6]. Manifold research efforts have proposed solutions to tackle the reliability of SPARQL endpoints [8,2,6]. However, these methods often imposed limitations [4], as consuming RDF from dump files is a very cumbersome, time-consuming and resource-demanding task.

In this demo, we present KBox [4], an open-source platform that allows users to share and consume ready-to-query RDF knowledge graphs (KGs). We show the functionalities implemented by our software, as well as the different stages from a query to its result. We demonstrate how KBox can be used to facilitate the implementation, integration, and evaluation of Semantic Web systems. KBox source code is available online.[4]

[4] https://github.com/AKSW/KBox

2 Demonstration

The goal of the demonstration will be to show how to share and query RDF knowledge graphs using the KBox platform.

2.1 The Knowledge Box (KBox)

The KBox architecture comprises five components, shown in Figure 1:

1. *Knowledge Graph Name System* (KNS): The KNS is designed to allow users and applications to share and dereference ready-to-query RDF knowledge graphs. With the KNS, a KG can have different names, be distributed by various authorities, as well as stored at several Web Addresses in a decentralized manner. For instance, a version of the DBpedia KG can be distributed not only by DBpedia but also by other authorities. The KNS is composed of different components: The KNS Server, the Knowledge Graph Name (KN), and the target KG.
2. *Knowledge Graph Name (KN)*: The Knowledge Graph Name is the name of the graph represented by an IRI. A KG can have different names. For instance, a KG can be named as *http://example.org* or simply *example*.
3. *KNS Server*: The KNS Server is an `HTTP` server that stores records containing the Knowledge Graph Name and its Web Address (URL) in format `(KN, URL)` that enables the KBox Client to dereference KGs.
4. *Client*: The client is where KBox is running; it interfaces between the User/Application, the Operational System, the Network as well as the KNS.
5. *User/Application (Edge)*: The edge is the KBox Client user, i.e. one or more applications running on a server or a standard user machine.

With KBox, users or applications perform queries targeting Knowledge Graph Names. The KNs are the target ready-to-query RDF KGs that the user or application desire to use.

The KBox client is built upon the KBox `Core`, `Kibe` and `Fusca` libraries. The `Core` library contains the core functions to dereference and uniquely identify resources in the `Network` (Web/Intranet) as well as in the `File System`. The `Kibe` library extends the `Core` and adds the capability to manipulate RDF Knowledge Graphs. Such library contains both the SPARQL and the Resource Description Framework (RDF) layers. The RDF and SPARQL layers allow KBox to deal with RDF data through operations such as reading, serialization and processing. The `Fusca` library enables `User/Application`s to instantiate SPARQL endpoints using any available KG in KBox. The `User/Application` performs operations over the previous layers such as dynamically dereference, aggregate, uniquely identify and execute SPARQL query operations over published KGs.

2.2 Querying

In the KBox architecture, all intelligence is shifted to the client. The process starts when the User/Application provides the KBox Client with a SPARQL query

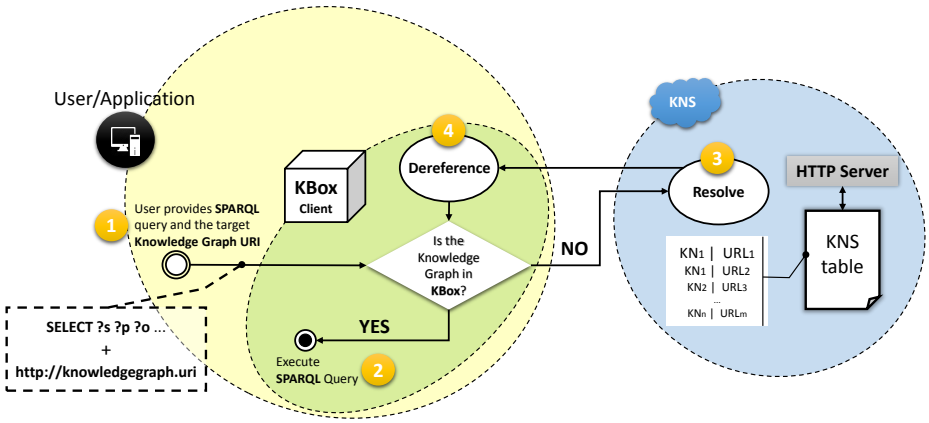

Fig. 1. KBox

along with a KG Name (see (1)). The KBox client first checks the availability of the KN. If the KG is locally available, the Client executes the SPARQL query (see (2)). Otherwise, if the KG is not available locally, the Client uses the KNS table to resolve the KN (see (3)). That is, it locates the ready-to-query KG in one of the available entries in the KNS table. If the KG is located, the KBox Client dereferences it (see (4)) and finally executes the SPARQL query (see (2)).

KBox can also be used to query multiple knowledge graphs (Listing 1.2). This functionality enables users to execute federated SPARQL queries, better define the target data as well as perform more efficient queries.

In this demo, we are going to show how to query single and multiple knowledge graphs. Furthermore, we are going to demonstrate how to use and build applications using KBox API natively (Listing 1.3) or from the command line interface (Listing 1.1).

```
$ java -jar kbox-v0.0.1-alpha3.jar -sparql "Select (count(
    distinct ?s) as ?n) where {?s ?p ?o}" -kb "https://www.w3
    .org/2000/01/rdf-schema" -install
```

Listing 1.1. Querying RDF knowledge graphs in command line interface using KBox runnable JAR file.

```
$ java -jar kbox-v0.0.1-alpha3.jar -sparql "Select (count(
    distinct ?s) as ?n) where {?s ?p ?o}" -kb "https://www.w3
    .org/2000/01/rdf-schema,http://xmlns.com/foaf/0.1" -
    install
```

Listing 1.2. Federated Querying RDF knowledge graphs in command line interface using KBox runnable JAR file.

```
KBox.query("Select (count(distinct ?s) as ?n) where {?s ?p ?o
    }", true, "https://www.w3.org/2000/01/rdf-schema");
```

Listing 1.3. Querying RDF knowledge graphs using KBox Java library.

2.3 Instantiating

KBox also facilitates the instatiation of SPARQL endpoints thanks to the `Fusca` library. KBox is distributed as executable JAR file (Listing 1.1), library (Listing 1.3), and `Docker` container (Listing 1.4). In this demo, we are going to show how to instatiate a SPARQL endpoint using each of these interfaces.

```
docker pull aksw/kbox
docker run aksw/kbox -server -kb "https://www.w3.org/2000/01/
    rdf-schema" -install
Loading Model...
Publishing service on http://localhost:8080/kbox/query
Service running ;-) ...
```

Listing 1.4. Instantiating a SPARQL endpoint using KBox Docker container.

3 Conclusion

During the demo, we will present KBox, an open-source platform that allows users to share and consume ready-to-query RDF knowledge graphs. We will explain the steps between an SPARQL query and its result, the available APIs, as well as how to instantiate a SPARQL endpoint using different interfaces. Finally, we will show pratical examples of applications using KBox and queries using the latest version of large datasets such as DBpedia 2015-10.[5]

4 Acknowledgments

This work was partly supported by a grant from the German Research Foundation (DFG) for the project *Professorial Career Patterns of the Early Modern History: Development of a scientific method for research on online available and distributed research databases of academic history* under the grant agreement No GL 225/9-1, by CNPq under the program *Ciências Sem Fronteiras* process 200527/2012-6.

[5] Available at the Knowledge Graph Name `http://dbpedia.org/2015-10/en`.

References

1. K. Bollacker, C. Evans, P. Paritosh, T. Sturge, and J. Taylor. Freebase: a collaboratively created graph database for structuring human knowledge. In *SIGMOD*, pages 1247–1250. ACM, 2008.
2. J. D. Fernández, M. A. Martínez-Prieto, C. Gutiérrez, A. Polleres, and M. Arias. Binary RDF Representation for Publication and Exchange (HDT). *Web Semantics: Science, Services and Agents on the World Wide Web*, 19:22–41, 2013.
3. J. Lehmann, R. Isele, M. Jakob, A. Jentzsch, D. Kontokostas, P. N. Mendes, S. Hellmann, M. Morsey, P. van Kleef, S. Auer, and C. Bizer. DBpedia - A Large-scale, Multilingual Knowledge Base Extracted from Wikipedia. *Semantic Web Journal*, 6(2):167–195, 2015.
4. E. Marx, C. Baron, T. Soru, and S. Auer. KBox: Transparently Shifting Query Execution on Knowledge Graphs to the Edge. In *ICSC*, 2017.
5. J. Pérez, M. Arenas, and C. Gutierrez. Semantics and Complexity of SPARQL. *ACM Trans. Database Syst.*, 34(3):16:1–16:45, Sept. 2009.
6. R. Verborgh, O. Hartig, B. De Meester, G. Haesendonck, L. De Vocht, M. Vander Sande, R. Cyganiak, P. Colpaert, E. Mannens, and R. Van de Walle. Querying datasets on the web with high availability. In *International Semantic Web Conference*, pages 180–196. Springer, 2014.
7. D. Vrandečić and M. Krötzsch. Wikidata: a free collaborative knowledgebase. *Communications of the ACM*, 57(10):78–85, 2014.
8. P. Yuan, P. Liu, B. Wu, H. Jin, W. Zhang, and L. Liu. Triplebit: A Fast and Compact System for Large Scale RDF Data. *Proc. VLDB Endow.*, 6(7):517–528, May 2013.