Semantic Web and Data Science Integration Using Computational Books

Dimitar Mileski, Milos Jovanovik and Dimitar Trajanov
Faculty of Computer Science and Engineering
Ss. Cyril and Methodius University in Skopje, North Macedonia
E-mail: dimitar.mileski@ieee.org, {milos.jovanovik, dimitar.trajanov}@finki.ukim.mk

***Abstract*—This paper presents the architecture for the development of web applications for exploring semantic knowledge graphs through parameterized interactive visualizations. The web interface and the interactive parameterized visualizations, in the form of a computational book, provide a way in which knowledge graphs can be explored. An important part of using this approach for building interactive web visualizations is that we can substitute the knowledge graph entities with other entities within the existing interactive visualizations, execute commands in a web-based environment, and get the same visualization for the new entities. With this architecture, various applications for interactive visualization of knowledge graphs can be developed, which can also stimulate the interest to explore the graph and its entities. We also present a publicly available open source use-case that is built using the concepts discussed in this paper.**

***Index Terms*—Knowledge Graphs, Linked Data, Semantic Web, Visualisation, RDF, SPARQL**

I. Introduction

The Semantic Web, along with the concepts of Linked Data and Knowledge Graphs, are built on mature technologies. However, the development of applications in this domain requires a certain learning curve, for people which are not familiar with the basic concepts. Changing the paradigm and thinking about data as tables to data as graphs, can deter some people from this set of technologies. Many advanced tools developed for knowledge graphs do not have a web interface, which further deters people who are not developers or scientists. In this paper, we present a new way in which a user can explore knowledge graphs and their entities, and can discover new data that can be used for machine learning and data science.

The steps that users need to take are as follows.

1) Select the entities and parameters they want to visualize.
2) Select an existing visualization of the computational book whose data types of parameters will match the entities and parameters they want to visualize.
3) Execute code with the changed parameters, with an in-line interactive web environment.

Such web interactive parameterized visualizations can encourage scientists, engineers, and other users who have no prior knowledge in the field of knowledge graphs, to become familiar with the principles of knowledge graphs, to explore new data, to change entities in the interactive visualization and execute commands, to write and execute SPARQL queries directly on the computational book and update interactive visualizations with data from new entities. After the entities are changed, we have interactive visualizations but with data from the new entities. This can encourage the creation of more such web applications (computational books).

The proposed architecture for building such interactive applications (computational books) uses technologies such as Jupyter Book, and Binder which are open-source, and will allow for the creation of new such applications that will be built with the technologies and principles of this architecture, or can enable modifications of existing applications which are built in the same way. The interactive in-line way of executing the code and the web interface will allow small changes in the code that users make when browsing the computational book to return new data that had not been seen before and to be displayed in the existing interactive visualization. It is also possible to work with a completely different knowledge graph, but in that case the parameters should be changed, or if that part is not parameterized, the users need to change the SPARQL queries. Since the changes are made inline in the interactive web environment, the users have the opportunity to execute the code and get new data and new visualizations. From a developement perspective, one of the most important parts is setting up a reproducible environment.

The proposed architecture uses Jupyter Book [1], which represents a collection of Jupyter Notebooks [2]. Jupyter Book uses Binder [3], which is a code repository that contains code or content that you would like other people to run. This might be a Jupyter Notebook, or an script in a different language, such as R or Julia. Jupyter Book uses Binder to provide a reproducible environment, but also implements an in-line code execution feature directly on the web page.

The rest of the paper is organized as follows: the related work is presented in Section II. In Section III we provide details on these technologies and how they take part in this architecture. Section IV presents a use case that was build with this proposed architecture, where users can see the publicly hosted computational book, and explore new data from knowledge graphs with the interactive visualisations. Section V contains a discussion about the potential of building such computational books for exploring new data from knowledge graphs.

II. Related Work

The uptake and consumption of Linked Data is currently restricted almost entirely to the Semantic Web community.

While the utility of Linked Data to non-tech savvy web users is evident, the lack of technical knowledge and an understanding of the intricacies of the semantic technology stack limit such users in their ability to interpret and make use of the Web of Data [4]. A key solution in overcoming this hurdle is to visualize Linked Data in a coherent and legible manner, allowing non-domain and non-technical audiences to obtain a good understanding of its structure, and therefore implicitly compose queries, identify links between resources and intuitively discover new pieces of information [4]. Extensive survey was provided in [4] of current efforts in the Semantic Web community with respect to our requirements, and identify the potential for visual support to lead to more effective, intuitive interaction of the end user with Linked Data.

Casual users find it difficult to explore and use Semantic Web data due to the prevalence of specialized browsers that require complex queries to be formed and intimate knowledge on the structure of datasets [5]. New approach makes it possible to obtain an overview of the dataset being explored using techniques, such as navigation menus, treemaps or sitemaps, which are usually not available in text-based Semantic Web browsers. From there, users can interactively explore the data using facets. Moreover, facets also feature a pivoting operation, motivated during tests with lay users, that removes the main constraint of most faceted browsers, i.e. the inability to combine filters for differently faceted views to build complex queries [5].

In [6], a novel interactive visualization platform is presented, called ALOHA. Their study showed that graph-based interactive visualization is a novel and acceptable approach to end-users who are interested in seeking online health information of various domains. They use user-centered design process to create a user-friendly web-based application, ALOHA, for the general public, to explore knowledge relevant to their needs, through an iterative development process. Moreover, their study showed that graph-based interactive visualization is promising in helping consumers explore complex health concepts quickly and can potentially lead to a new way of finding and consuming health information online [6]. A novel knowledge exploration mechanism such as ALOHA maybe appealing to the general public and can serve as a template for developing consumer-facing, evidence-based (i.e., supported by scientific literature) knowledge graphs in many other health and disease domains [6].

In the paper [7] modules were designed with an emphasis on ease of use, query flexibility, and interactive visualization of results. They presented LDlink, a web-based collection of bioinformatic modules that query single nucleotide polymorphisms (SNPs) in population groups of interest to generate haplotype tables and interactive plots. They have developed different modules, fore example LDhap calculates population specific haplotype frequencies of all haplotypes observed for a list of query SNPs. They use predefined modules, which in our architecture are predefined interactive visualisations. Users can change entities in the interactive visualisations and can explore new data from the knowledge graph.

Two applications were presented in [8], Thinkbase and Thinkpedia, which aim to make web content more accessible and usable by utilizing visualizations of the semantic graph as a means to navigate and explore large knowledge repositories. Both of the applications implement a similar concept: they extract semantically enriched contents from a large knowledge spaces (Freebase and Wikipedia, respectively), create an interactive graph-based representation out of it, and combine them into one interface together with the original text based content.

There is a interactive tool which was presented in [9], for exploring and visualizing large RDF ontologies. Primary goal is to present the information to the user in a simple and intuitive way. It provides an environment where users can select a small set of objects to examine dynamically in real-time, providing better contextual information.

Exploring and visualizing very large datasets has become a major research challenge, of which scalability is a vital requirement. In the survey [10], they describe the major prerequisites and challenges that should be addressed by the modern exploration and visualization systems. Considering these challenges, they present how state-of-the-art approaches from the database and information visualization communities attempt to handle them. Finally, they survey the systems developed by the Semantic Web community in the context of the Web of Linked Data, and discuss to which extent these satisfy the contemporary requirements.

III. Architecture

Figure 1 presents the system architecture. Starting from the left side of the figure, we will store the code in a repository with a code versioning system. Hosting the entire computational book will be achieved with the help of the next component of the architecture. In our repository we will have the files needed to configure and create an executable, reproducible environment. These configuration files are used by Repo2Docker, which is a tool to build, run, and push Docker images from source code repositories. Repo2Docker will create a Docker container which will then be used by Binder to host Jupyter Notebooks. The configuration files that Repo2Docker uses to create a container can be: `requirements.txt`, `Dockerfile`, `apt.txt`, `environment.yaml`, `setup.py`, etc. So far we have a code repository, in which we have a configuration file or a combination of multiple files, with which Binder creates a reproducible environment to run and host Jupyter Notebooks. That environment will include all dependencies that we need to execute the code in the Jupyter Notebook cells. In the configuration files, we will set the dependencies for SPARQL, RDF, RDFLib, visualization libraries and any other dependencies that can enable executable environment. With this, we have a reproducible environment where users can run Jupyter Notebook cells without installing dependencies and we have a hosted Jupyter Notebook, i.e. users will not run Jupyter Notebook locally. The next component of the architecture is Jupyter Book, which is a collection of Jupyter Notebooks that

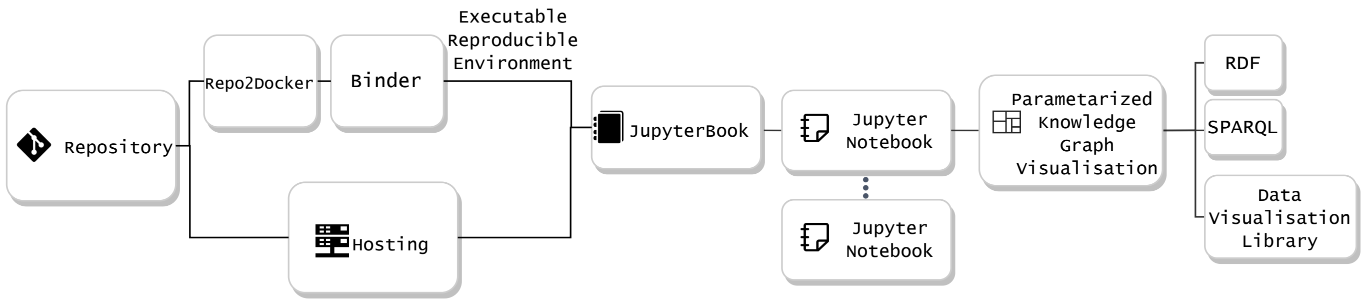

Executable Reproducible Environment
Repository
Repo2Docker
Binder
Hosting
JupyterBook
Jupyter Notebook
Jupyter Notebook
Parametarized Knowledge Graph Visualisation
RDF
SPARQL
Data Visualisation Library

Fig. 1. Architecture overview.

are displayed in the form of a web site. For code execution you have two options in each Jupyter Notebooks. The first option is to create a Binder environment and then directly execute the cells, while changing the visualization parameters with new entities so that we can discover new data from the knowledge graphs, and the second option is to create a Binder session that will be opened in a new browser tab and we will have standard Jupyter Notebook environment or JupyterLab. When building such computational books, with this architecture on each of the pages (Jupyter Notebooks) we will have predefined and parameterized visualizations, for example TreeMap, where users will be able to change the entities and populate the visualizations with new data from the knowledge graph. Libraries and data visualization tools are used to create parameterized visualizations. The data from the knowledge graphs are retrieved and manipulated with SPARQL, the RDF query language.

IV. Use Case: Exploring Medical Data from DBpedia with a Computational Book

With the help of the architecture proposed in this paper, a use-case was developed for exploring DBpedia data related to medicine. The code is hosted on GitHub. In addition to the code versioning system, GitHub provides hosting to static sites through Github Pages. The advantage of hosting Jupyter Book with GitHub Pages is that you only need to enable GitHub Pages in repository settings. In the repository there is a `Dockerfile` file which is one of the configuration file options, which will be used by Repo2Docker to create a reproducible environment in Binder, which will allow users to execute the code in Jupyter Book. Jupyter Book is a collection of Jupyter Notebooks. Each of the Jupyter Notebooks has Plotly [11] visualizations. Figure 3 presents only one of the ready-made parameterized visualizations, and that is the TreeMap of the entities whose activity sector is medicine: `?Thing dbp:activitySector dbr:Medicine`.

The visualizations are parameterized and they use data from the DBPedia knowledge graph. In this use case the data are taken from the DBPedia SPARQL endpoint. In the specific visualization with TreeMap, the parameters can be other entities instead of `?Thing dbp:activitySector dbr:Medicine`. You can completely change the SPARQL endpoint to have a new knowledge graph and new entities, which will be displayed in the existing visualization. That change of entities or SPARQL endpoints can be done in the cells that are directly in the computational book. Figure 2 shows the cells where the code can be changed, and the parameters of the visualization, Figure 3 shows the TreeMap which is one of the visualisations in this use case. This way, users can discover new data from knowledge graphs. The code from this use case is open source[1], and the hosted version is available online[2].

V. Discussion

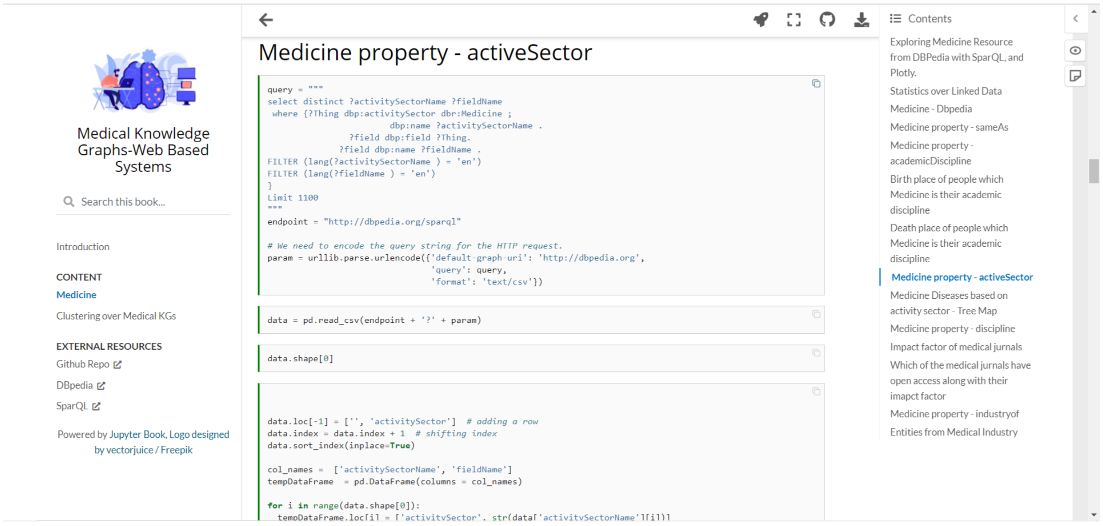
Discovering new data is important for creating and developing new applications for machine learning, data science or creating Semantic Web applications where data will be retrieved from knowledge graphs. The combination of technologies used in the proposed architecture provides ways to discover new data and provide view of the data from a particular perspective through predefined visualizations. First, technologies that allow us to look at data from a different angle rather than tables or spreadsheet are Knowledge Graphs which are part of the Semantic Web and Linked Data technologies. The graph structure allows us to discover new data, over and over again, so that each subsequent execution of the code will potentially return different data. As a second point, the technologies that allow us to have a reproducible environment and executable code in a web environment in the form of a website are Jupyter Book, Jupyter Notebook and Binder. Last, the various visualization libraries will enable us to present data in a way that gives additional context to the data. Parameterization of these visualizations will provide a generic definition that can be reused with other entities or other knowledge graphs.

This architecture aims to encourage users to instantly interact with data and discover new data. We want to encourage the development of applications with this proposed architecture, so users can explore new data from knowledge graphs thought interactive parameterized visualisations.

References

[1] E. B. Community, "Jupyter book," Feb 2020.

[1]https://github.com/dimitarmileski/Medical-KGs-WBS
[2]https://dimitarmileski.github.io/Medical-KGs-WBS/intro.html


Medical Knowledge Graphs-Web Based Systems
Search this book...
Introduction
CONTENT
Medicine
Clustering over Medical KGs
EXTERNAL RESOURCES
Github Repo
DBpedia
SparQL
Powered by Jupyter Book, Logo designed by vectorjuice / Freepik
Contents
Exploring Medicine Resource from DBPedia with SparQL, and Plotly.
Statistics over Linked Data
Medicine - Dbpedia
Medicine property - sameAs
Medicine property - academicDiscipline
Birth place of people which Medicine is their academic discipline
Death place of people which Medicine is their academic discipline
Medicine property - activeSector
Medicine Diseases based on activity sector - Tree Map
Medicine property - discipline
Impact factor of medical jurnals
Which of the medical jurnals have open access along with their imapct factor
Medicine property - industryof
Entities from Medical Industry

Medicine property - activeSector

```
query = """
select distinct ?activitySectorName ?fieldName
 where {?Thing dbp:activitySector dbr:Medicine ;
                        dbp:name ?activitySectorName .
                ?field dbp:field ?Thing.
              ?field dbp:name ?fieldName .
FILTER (lang(?activitySectorName ) = 'en')
FILTER (lang(?fieldName ) = 'en')
}
Limit 1100
"""
endpoint = "http://dbpedia.org/sparql"

# We need to encode the query string for the HTTP request.
param = urllib.parse.urlencode({'default-graph-uri': 'http://dbpedia.org',
                                'query': query,
                                'format': 'text/csv'})
```

```
data = pd.read_csv(endpoint + '?' + param)
```

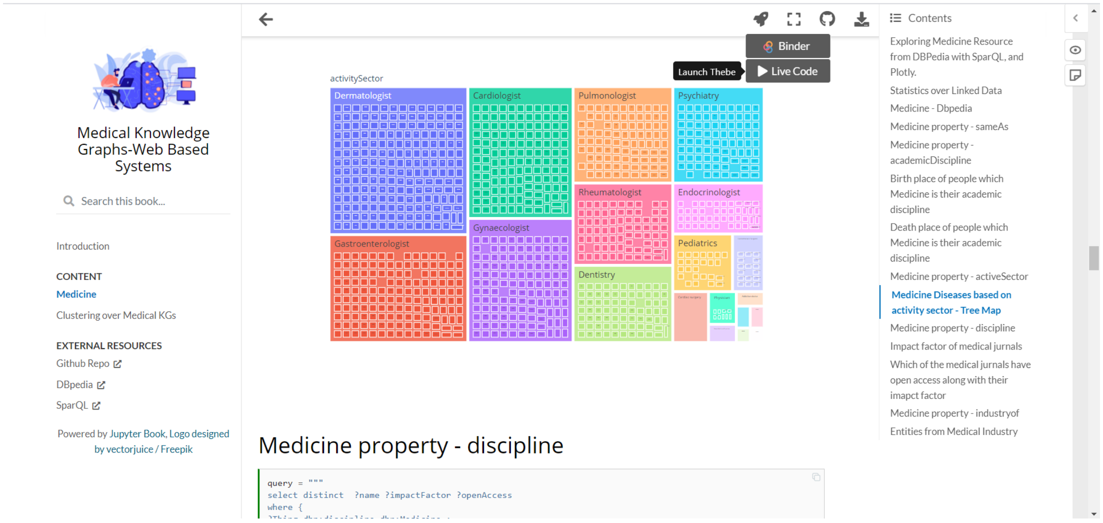
```
data.shape[0]
```

```
data.loc[-1] = ['', 'activitySector']  # adding a row
data.index = data.index + 1  # shifting index
data.sort_index(inplace=True)

col_names =  ['activitySectorName', 'fieldName']
tempDataFrame  = pd.DataFrame(columns = col_names)

for i in range(data.shape[0]):
  tempDataFrame.loc[i] = ['activitySector', str(data['activitySectorName'][i])]
```

Fig. 2. SPARQL query cell execution.


Medical Knowledge Graphs-Web Based Systems
Search this book...
Introduction
CONTENT
Medicine
Clustering over Medical KGs
EXTERNAL RESOURCES
Github Repo
DBpedia
SparQL
Powered by Jupyter Book, Logo designed by vectorjuice / Freepik
Binder
Launch Thebe
Live Code
activitySector
Dermatologist
Cardiologist
Pulmonologist
Psychiatry
Rheumatologist
Endocrinologist
Gynaecologist
Gastroenterologist
Pediatrics
Dentistry
Contents
Exploring Medicine Resource from DBPedia with SparQL, and Plotly.
Statistics over Linked Data
Medicine - Dbpedia
Medicine property - sameAs
Medicine property - academicDiscipline
Birth place of people which Medicine is their academic discipline
Death place of people which Medicine is their academic discipline
Medicine property - activeSector
Medicine Diseases based on activity sector - Tree Map
Medicine property - discipline
Impact factor of medical jurnals
Which of the medical jurnals have open access along with their imapct factor
Medicine property - industryof
Entities from Medical Industry

Medicine property - discipline

```
query = """
select distinct  ?name ?impactFactor ?openAccess
where {
```

Fig. 3. Parameterized treemap visualisation.

[2] T. Kluyver, B. Ragan-Kelley, F. Pérez, B. E. Granger, M. Bussonnier, J. Frederic, K. Kelley, J. B. Hamrick, J. Grout, S. Corlay *et al.*, *Jupyter Notebooks-a publishing format for reproducible computational workflows.*, 2016, vol. 2016.

[3] Project Jupyter, Matthias Bussonnier, Jessica Forde, Jeremy Freeman, Brian Granger, Tim Head, Chris Holdgraf, Kyle Kelley, Gladys Nalvarte, Andrew Osheroff, M. Pacer, Yuvi Panda, Fernando Perez, Benjamin Ragan Kelley, and Carol Willing, "Binder 2.0 - Reproducible, interactive, sharable environments for science at scale," in *Proceedings of the 17th Python in Science Conference*, Fatih Akici, David Lippa, Dillon Niederhut, and M. Pacer, Eds., 2018, pp. 113 – 120.

[4] A.-S. Dadzie and M. Rowe, "Approaches to Visualising Linked Data: A Survey," *Semantic Web*, vol. 2, no. 2, pp. 89–124, 2011.

[5] J. M. Brunetti, R. García, and S. Auer, "From Overview to Facets and Pivoting For Interactive Exploration of Semantic Web Data," *International Journal on Semantic Web and Information Systems (IJSWIS)*, vol. 9, no. 1, pp. 1–20, 2013.

[6] X. He, R. Zhang, R. Rizvi, J. Vasilakes, X. Yang, Y. Guo, Z. He, M. Prosperi, J. Huo, J. Alpert *et al.*, "ALOHA: Developing an Interactive Graph-Based Visualization for Dietary Supplement Knowledge Graph Through User-Centered Design," *BMC Medical Informatics and Decision Making*, vol. 19, no. 4, pp. 1–18, 2019.

[7] M. J. Machiela and S. J. Chanock, "LDlink: A Web-Based Application for Exploring Population-Specific Haplotype Structure and Linking Correlated Alleles of Possible Functional Variants," *Bioinformatics*, vol. 31, no. 21, pp. 3555–3557, 2015.

[8] C. Hirsch, J. Hosking, and J. Grundy, "Interactive Visualization Tools for Exploring the Semantic Graph of Large Knowledge Spaces," in *Workshop on Visual Interfaces to the Social and the Semantic Web (VISSW2009)*, vol. 443, 2009, pp. 11–16.

[9] L. Deligiannidis, K. J. Kochut, and A. P. Sheth, "RDF Data Exploration and Visualization," in *Proceedings of the ACM First Workshop on CyberInfrastructure: Information Management in eScience*, 2007, pp. 39–46.

[10] N. Bikakis and T. Sellis, "Exploration and Visualization in the Web of Big Linked Data: A Survey of the State of the Art," *arXiv preprint arXiv:1601.08059*, 2016.

[11] P. T. Inc. (2015) Collaborative data science. Montreal, QC. [Online]. Available: https://plot.ly