VizKG: A Framework for Visualizing SPARQL Query Results over Knowledge Graphs

Hana Raissya[1], Fariz Darari (✉)[1,2], and Fajar J. Ekaputra[3]

[1] Faculty of Computer Science, Universitas Indonesia, Depok, Indonesia
[2] Tokopedia-UI AI Center of Excellence, Jakarta, Indonesia
{hana.raissya, fariz}@ui.ac.id
[3] Institute of Information Systems Engineering, TU Wien, Vienna, Austria
fajar.ekaputra@tuwien.ac.at

Abstract. Despite the rise of the knowledge graph (KG) popularity, understanding SPARQL query results from a KG can be challenging for users. The use of data visualization tools, e.g., Wikidata Query Service and YASGUI, can help address this challenge. However, existing tools are either focused just on a specific KG or only provided as a web interface. This paper proposes VizKG, a framework that provides a wide range of visualizations for SPARQL query results over KGs. VizKG aims to assist users in extracting patterns and insights from data in KGs, and hence supporting further KG analysis. VizKG features a wrapper that links SPARQL query results and external visualization libraries by mapping query result variables to the required visualization components, currently allowing for 24 types of visualizations. Not only that, VizKG also includes visualization recommendations for arbitrary SPARQL query results as well as extension mechanisms for additional visualization types. In our evaluation, the visualization recommendation feature of VizKG achieves an accuracy of 87.8%. To demonstrate the usefulness of VizKG in practical settings, this paper also reports on use case evaluation over various domains and KGs. A Python-based, Jupyter Notebook friendly implementation of VizKG is openly available at https://pypi.org/project/VizKG/.

Keywords: Visualization · Knowledge Graphs · SPARQL · Insights

1 Introduction

A knowledge graph (KG) mainly describes real-world entities and their interrelations in a graph structure, allowing to cover various domains [3]. The Semantic Web and Linked Data are concrete embodiments of KGs, popularized by Google in 2012 through the Google Knowledge Graph.[4] In the field of data science, the development of the Semantic Web and Linked Data is becoming increasingly important, serving as both primary and contextual data sources [13].

[4] https://blog.google/products/search/introducing-knowledge-graph-things-not/

Visualization is one of the stages in the data science pipeline that plays a key role in data exploration and analysis [4]. Data visualization can help mediate between data scientists and domain experts pertaining to the validation of assumptions and findings. In particular, data visualization might come into handy when little is known about data sources and analytical objectives [5].

Generally, visualizing (tabular) data takes form of charts, such as time-series charts, geographic maps, statistical charts, as well as hierarchies and networks [6]. Visualizations can also be performed over KGs, e.g., Wikidata. Wikidata serves as a (semantic) data hub among all editions of Wikipedia as well as external sources [8]. Wikidata features Wikidata Query Service (WQS), which not only supports SPARQL queries, but also visualizations in the form of image grids, timelines, dimensions, treemaps, and many more. WQS is indeed helpful to provide graphical information about the data stored in Wikidata. Nevertheless, its usage is limited to Wikidata only. On the other hand, visualization services such as LODmilla[5] and YASGUI[6] facilitate visualizing generic KGs though with a limited support of visualization types compared to WQS.

The provision of KG visualization services using a web, standalone tool is however less ideal for the data science community. In terms of data visualization, the Jupyter Notebook[7] is a popular choice and has advantages over other platforms [9], such as real-time interaction with code, inline printing of output, and PDF or HTML exports. Visualizations generated from a Jupyter notebook can be presented in a browser or as a shared Google Colaboratory.[8] With these advantages, Jupyter notebooks enable users to narrate visualizations generated from (Python) libraries, e.g., Matplotlib[9] and Plotly.[10]

Based on the need for exploration and visualization on the increasingly popular KGs, through this study, we propose VizKG (https://pypi.org/project/VizKG/), a Python-based framework with the following functionalities: (i) visualization of SPARQL query results on generic KGs; (ii) automatic recommendation of visualization types; and (iii) extension mechanisms for additional visualization types.

2 Related Work

KG visualization is a research topic that has been around for many years. A stream of research studies aims to support end-users to explore and visualize KGs. One of the earlier studies in this area is SemLens [7], which provides a visual tool allowing end-users to perform robust data analysis on KGs. SemLens, however, focuses only on a single visualization type, i.e., scatter chart. Linked Data Visualization Model (LVDM) aims to provide a formal model for RDF data visualization [2]. LVDM provides two reference implementations: (i) LODVisualization, connecting & analyzing various datasets, and (ii) Payola, providing details on specific parts of KGs. Another approach is LDVizWiz [1],

[5] https://www.dbpedia.org/community/lodmilla/
[6] https://yasgui.triply.cc/
[7] https://jupyter.org/
[8] https://colab.research.google.com/
[9] https://matplotlib.org/
[10] https://plotly.com/

which identifies seven data categories and their associated standard vocabularies for visualizations in existing Linked Data infrastructure and workflow. LinkDaViz [14] is a tool that allows for automatic data visualization. The tool features a recommendation algorithm that automatically binds data properties to visualization options. Nevertheless, the tool does not support direct visualization of arbitrary SPARQL query results. More recently, ProWD [11] is a user-friendly tool developed to visualize knowledge imbalances in Wikidata. The tool supports visualization types like bar charts & area charts, and demonstrates how KG visualizations can be relevant in practice. However, the tool only caters to very specific use cases (i.e., knowledge imbalances) for a specific KG (i.e., Wikidata).

Another stream of research focuses on supporting KG visualizations for more technical, SPARQL-savvy users. In this direction, the most prominent approaches are web-based approaches, such as YASGUI [12] and Wikidata Query Service (WQS). YASGUI is a web-based SPARQL client that can be used to query both remote and local endpoints. It allows visualizing SPARQL query results, albeit with limited visualization types. Unlike a general-purpose SPARQL tool such as YASGUI, the WQS interface has been customized for Wikidata to improve its functionalities. WQS supports a wide variety of result visualizations in addition to the standard tabular view. The WQS interface, however, can only be used on Wikidata. While these tools are helpful for technical users, there are still gaps to support (generic) KG visualizations by the growing role of data scientists, who typically rely on a specific environment (e.g., Jupyter Notebook) for their data science pipeline.

3 VizKG Framework

In this section, we present the general VizKG architecture and workflow (Section 3.1), VizKG visualization recommendation procedure (Section 3.2), and the extension mechanism for new visualization types (Section 3.3).

3.1 Architecture and Workflow

Fig. 1 displays the architecture as well as workflow of VizKG. VizKG consists of four main stages: (i) Preprocessing, (ii) Query Execution, (iii) Visualization Recommendation, and (iv) Visualization Generation.

From the user input, VizKG receives a SPARQL query string, a SPARQL endpoint URL, and (optionally) a preferred chart type (Step 1). Next, the *Preprocessing* stage of VizKG parses the query string and validates the endpoint URL (Step 2). VizKG also checks whether the selected chart type is supported or not. Then, in the *Query Execution*, VizKG invokes a REST-based API call to a remote SPARQL endpoint, to which the SPARQL query is evaluated, and that subsequently the query results (in JSON format) are returned (Step 3). Afterwards, VizKG transforms the query results into a tabular form. VizKG leverages the SPARQLWrapper[11] library to support both the Preprocessing and Query Execution stages.

[11] https://github.com/RDFLib/sparqlwrapper

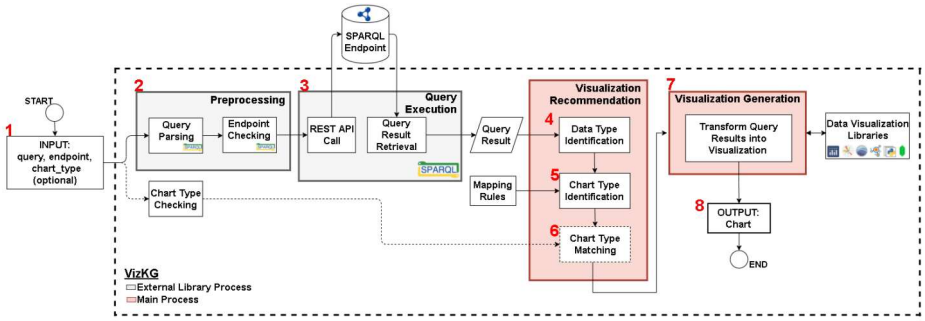

Fig. 1. VizKG Architecture (bit.ly/VizKG-Architecture)

The next stage is the *Visualization Recommendation*. The first thing to do in this stage is to identify the datatypes of each query variable (Step 4). The identified datatypes as well as ordering of the variables serve as the main reference to determine recommended charts (or visualizations) of the query results based on the VizKG mapping rules (Step 5). In the case that a preferred chart type is given as an input parameter, VizKG simply checks if the chart type is in the chart recommendations (Step 6). As for the other case when there is no chart type preference given, VizKG bypasses the chart type matching. In the *Visualization Generation* stage, VizKG maps the query result variables to visualization components, and delegates the creation of the visualization to off-the-shelf visualization libraries, such as Matplotlib, Plotly, and Seaborn (Step 7). The final output would be the graphical visualization of the query results (Step 8).

3.2 Visualization Recommendation

The *Visualization Recommendation* stage determines which visualization types are suitable for the returned query results. Here we describe the stage in more detail.

First, the datatypes wrt. the variable values are identified via regex matching. We support the following datatypes: numerical, date, URI, image, coordinate, and label. Then, via the VizKG mapping rules,[12] we check the compatibility of the datatypes of the query variables with the datatypes of the supported charts.

In the checking process, we use the following heuristics: (i) all visualization components (except the optional ones) must have query variables mapped into them; (ii) the datatype of the query variable mapped must conform to the required datatype of the chart; and (iii) whenever there are more than one conforming datatypes, the ordering matters (i.e., it is first-come-first-map).

As an example, if there is a query result with two variables (i.e., V_1 and V_2) of type numerical, then the visualization of scatter chart can be recommended, since there are exactly two non-optional variables required with the same datatypes, and that the variable V_1 is mapped to the X-component and V_2 to the Y-component of the scatter chart. VizKG iterates over all visualization types, checking whether they are compatible or not, and collects the compatible ones as visualization recommendations.

[12] https://bit.ly/VizKG-MappingRules

3.3 Extension Mechanism for New Visualization Types

In the code structure of VizKG,[13] all chart implementations are made modular, in that they have their own classes, inheriting from the `Chart` class. Every class added to VizKG must then make concrete the methods of the datatype compatibility checking, variable-to-component mapping, and visualization generation. These methods are called during the visualization recommendation and generation stages. Furthermore, the added chart implementation must be registered in the `__init__.py` and `chartdict.py` configuration files, listing all the supported VizKG visualizations.

4 Evaluation

In this section, we report on the visualization recommendation evaluation and use case evaluation of VizKG.

4.1 Visualization Recommendation Evaluation

To evaluate the accuracy of VizKG visualization recommendation (as described in Section 3.2), we conduct an experimental evaluation using real-world SPARQL queries taken from the Wikidata query examples page (as of September 16, 2021).[14] We take only the visualization queries from the page, that is, those queries starting with the "#defaultView:[viz-type]" directive where viz-type refers to the preferred visualization type for the query results. We consider 82 out of 92 visualization queries because the remaining 10 queries either give no query results or a time limit error. We run the queries using our VizKG library, and record the visualization recommendations given by VizKG. Then, we check whether the VizKG recommendations include the original, preferred visualization type, as stated in the "#defaultView:[viz-type]" directive of the Wikidata queries.

From the experiment results, we observe that VizKG gives a correct recommendation in 72 out of 82 cases, giving an accuracy of 87.8%. One of the reasons for the incorrect recommendations is that there is a difference in the mapping rules between WQS and VizKG: WQS only requires one numeric variable to generate a scatter chart, whereas VizKG requires two numeric variables. Overall, we believe that the visualization recommendation of VizKG delivers quite a good result considering that the recommendation procedure (as described before) is fairly simple.

4.2 Use Case Evaluation

We highlight use cases of VizKG in several domains (i.e., COVID-19, cultural heritage in Indonesia, and higher education) over a number of knowledge graphs (i.e., Wikidata, DBpedia, BudayaKB, and data.open.ac.uk). The visualization results are discussed below and showcased in Fig. 2.

[13] https://github.com/fadirra/vizkg
[14] https://bit.ly/WD-queries-examples

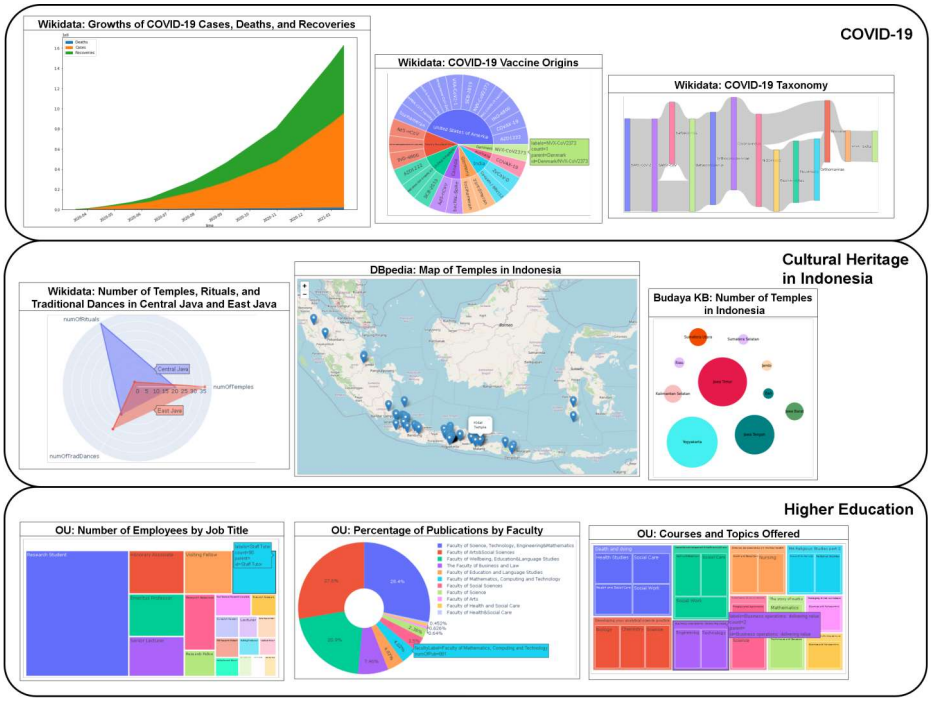

Fig. 2. VizKG Use Case Visualization (bit.ly/VizKG-Usecase)

COVID-19. This domain is relevant at the moment, particularly in regard to the latest updates of the COVID-19 pandemic. The data source used is Wikidata, accessed through its SPARQL endpoint.[15] The visualizations made are about: (i) the growth of COVID-19 cases, deaths, and recoveries using the stacked area chart type; (ii) origins of COVID-19 vaccines using the sunburst type; and (iii) COVID-19 taxonomy using the dimensions visualization type.

Cultural Heritage in Indonesia. In this domain, VizKG is evaluated over three different KGs: Wikidata, DBpedia, and BudayaKB. As a context, BudayaKB is a KG specialized for Indonesian cultural heritage, ranging from traditional music to folklores [10]. The left figure uses a radar chart to compare the number of traditional dances, rituals, and temples in the province of Central Java and East Java. The middle figure visualizes how Indonesian temples distribute on the map. The right figure illustrates the different number of temples by Indonesian provinces using the bubble chart type.

The middle figure, which is based on DBpedia,[16] relies on the following SPARQL query that retrieves Indonesian temples (= "candi" in Indonesian) including their geolocations and labels.

[15] https://query.wikidata.org/sparql
[16] https://dbpedia.org/

```
SELECT * WHERE {
     ?item dbo:wikiPageWikiLink dbr:Candi_of_Indonesia;
           geo:geometry ?geo .
     ?item rdfs:label ?itemLabel.
     FILTER(LANG(?itemLabel) = "en")
}
```

In the query, there are three variables (in the order they appear): `?item`, `?geo`, and `?itemLabel`. Via the VizKG mapping rules (as mentioned in Section 3.2), the query results can be visualized into a map, by mapping the `?geo` variable to the coordinate component and the `?itemLabel` variable to the popup component of the visualization. Note that `?item` need not be mapped in this case. A video demonstrating how VizKG can be used to visualize a map of Indonesian temples in DBpedia is available.[17]

Higher Education. Here, the data is obtained from the Open University (OU).[18] The topics are about employee profiles, courses, and publications of the Open University. We showcase the number of employees based on the job title with the treemap visualization, the number of publications by faculty using the donut chart type, and offered topics and courses using the treemap.

5 Conclusions and Future Work

This paper proposes VizKG as a Python-based framework for visualizing SPARQL query results over KGs. The framework facilitates the creation of visualization for generic KGs, automatic visualization recommendation, and extension mechanism for new visualization types. The VizKG visualization recommendation has been evaluated over real-world queries from Wikidata and achieves an accuracy of 87.8%. The use case evaluation of the VizKG framework is done over three domains (i.e., COVID-19, cultural heritage, and higher education) and four KGs (i.e., Wikidata, DBpedia, BudayaKB, and Open University). All of our use case examples are available as a Google Colab notebook.[19]

For future work, we plan to enhance the visualization recommendation with compatibility ranking so that not only binary recommendation is given. We also envision that more rigorous user testing can be performed to get a better understanding of the features and limitations of the VizKG framework. Furthermore, integrating the VizKG framework to kglab,[20] a hub package for graph-based data science libraries, can be a viable future direction to support wider audience.

[17] https://bit.ly/VizKGDemoTemples
[18] https://data.open.ac.uk/
[19] https://bit.ly/VizKGColab
[20] https://derwen.ai/docs/kgl/

Acknowledgements

We thank the anonymous reviewers for their careful feedback. The dissemination of this work is funded by a grant from Program Kompetisi Kampus Merdeka (PK-KM) 2021 of Faculty of Computer Science, Universitas Indonesia. Furthermore, this work was sponsored by the Austrian Research Promotion Agency FFG under grant 877389 (OBARIS) and the Vienna Business Agency (VasQua project).

References

1. Atemezing, G.A., Troncy, R.: Towards a Linked-Data based Visualization Wizard. In: COLD (2014)
2. Brunetti, J.M., Auer, S., García, R., Klímek, J., Nečaskỳ, M.: Formal Linked Data Visualization Model. In: IIWAS (2013)
3. Ehrlinger, L., Wöß, W.: Towards a Definition of Knowledge Graphs. In: SEMANTiCS Posters and Demos (2016)
4. Fayyad, U.M., Grinstein, G.G., Wierse, A. (eds.): Information Visualization in Data Mining and Knowledge Discovery. Morgan Kaufmann (2001)
5. Gómez-Romero, J., Molina-Solana, M., Oehmichen, A., Guo, Y.: Visualizing Large Knowledge Graphs: A Performance Analysis. Future Gener. Comput. Syst. **89**, 224–238 (2018)
6. Heer, J., Bostock, M., Ogievetsky, V.: A Tour through the Visualization Zoo. CACM **53**(6), 59–67 (2010)
7. Heim, P., Lohmann, S., Tsendragchaa, D., Ertl, T.: SemLens: Visual Analysis of Semantic Data with Scatter Plots and Semantic Lenses. In: I-Semantics (2011)
8. Malyshev, S., Krötzsch, M., González, L., Gonsior, J., Bielefeldt, A.: Getting the Most Out of Wikidata: Semantic Technology Usage in Wikipedia's Knowledge Graph. In: ISWC (2018)
9. Perkel, J.M.: Why Jupyter is Data Scientists' Computational Notebook of Choice. Nature **563**(7729), 145–146 (Oct 2018)
10. Putra, H.S., Mahendra, R., Darari, F.: BudayaKB: Extraction of Cultural Heritage Entities from Heterogeneous Formats. In: WIMS. pp. 6:1–6:9 (2019)
11. Ramadhana, N.H., Darari, F., Putra, P.O.H., Nutt, W., Razniewski, S., Akbar, R.I.: User-Centered Design for Knowledge Imbalance Analysis: A Case Study of ProWD. In: VOILA (2020)
12. Rietveld, L., Hoekstra, R.: YASGUI: Not Just Another SPARQL Client. In: SALAD (2013)
13. Ristoski, P., Paulheim, H.: Semantic Web in Data Mining and Knowledge Discovery: A Comprehensive Survey. J. Web Semant. **36**, 1–22 (2016)
14. Thellmann, K., Galkin, M., Orlandi, F., Auer, S.: LinkDaViz – Automatic Binding of Linked Data to Visualizations. In: ISWC (2015)