KinGVisher – Knowledge Graph Visualizer

Andreas Both[1,2], Aleksandr Perevalov[1], and Aleksandr Gashkov[1]

[1] Leipzig University of Applied Sciences, Leipzig, Germany
[2] DATEV eG, Nuremberg, Germany

Abstract. Knowledge Graphs rely on conceptional knowledge (TBox) and concrete data instances (ABox). Both types of statements are represented by RDF triples, hence, they are located in the same data storage. As this is an advantage from the aspect of generalization, it might become difficult to distinguish between the purpose of the triples when exploring a knowledge graph. Additionally, the success of knowledge graphs leads to larger data sets that might make it also harder to identify and recognize the information of interest, its connections, and patterns. KinGVisher was designed and developed as a visual explorer for knowledge graphs, providing automatic graph layouts and supporting users in creating a clear view of the knowledge graph. We publish this as a full-fledged, open-sourced web application online, which is freely available to all users.

Keywords: Knowledge Graphs · Data Visualization · Graph Visualization · Data Exploration

1 Introduction

Visualizations are useful and necessary to improve the accessibility and understandability of any data. This is particularly the case when dealing with large amounts of data, for instance, RDF knowledge graphs (KGs), in which distribution, differentiation, and stored content are constantly growing. This is not only the case for the general-domain KGs (e.g., DBpedia [1] and Wikidata [7]), but also for the domain-specific ones, as their structures are often difficult to understand. However, the advantage of linked data, which lies in reusing existing vocabularies and creating many instances of the same classes, might become problematic when many triples contain the same resource, and therefore a visualization might become heavily centralized (e.g., in DBpedia, `dbo:Place` is currently part of 804796 triples) or tightly packed. In turn, this may affect resource consumption and efficiency, especially if the visualizations are created in a web browser. Based on this observation, there is a need for a convenient KG visualization tool easily accessible to its users. We set the following requirements for such a tool: (1) The data in a KG must be *filterable* so that the number and both expected resources and unexpected resources can be defined; (2) A visualization of a KG needs to provide access to *layout configurations*, s.t., users can adapt to many different scenarios; (3) A visualization needs to be enabled to work with *large knowledge graphs* (e.g., DBpedia or Wikidata). Additionally, for the sake of applicability and standardization, the tool should use only SPARQL to interact with a KG.

In this work, we introduce KinGVisher – a web application for efficient KG visualizations that supports the aforementioned requirements and enables users to dive into the data like a Kingfisher[3]. We release our tool as a full-fledged web application, which is freely available to all users online[4].

Several implementations exist dedicated to the visualizations of KGs and linked data. For example, work by Kerdjoudj et al. [5], Lohmann et al. [6], Ghosh [3] et al., the non-dynamic UML-styled visualization OWLGrEd [2], the domain-specific SNIK Graph [4], or a KG explorer by the Zazuko company [8]. In contrast to these approaches, we favor a dynamic view of the KG's data that can be filtered and searched interactively (and would not focus on the TBox visualization). Therefore, we define the following features to let the users stay in control of their visualizations: *Whitelist properties* – a list of properties of the knowledge graph can be defined that only should be shown in the visualization, *Blacklist properties* – a list of properties of the knowledge graph can be defined that should not be shown in the visualization, and *Important resources* – a list of resources of the knowledge graph that has to be part of the visualization, i.e., they are used as a starting point of the exploration.

As our approach is using SPARQL only, *the exploration of a KG is to be done iteratively* (in particular, if users have defined important resources), which is done by creating SPARQL queries based on previously discovered resources.

2 Demonstration of KinGVisher

Our demonstrator (cf. Figure 1) provides the option of connecting to any public SPARQL endpoint and optionally defining a specific graph (not required for Wikidata or DBpedia, but common while using triplestores like Stardog[5]) ① of which the data should be evaluated. A core attribute is the user-defined number of maximal edges ② to keep the number of visualized triples within reasonable limits. Several visualization parameters are provided for directly controlling the layout for the current data ③. Moreover, several non-standard visualization configuration parameters are provided ④:

- Show resource labels: in case of being activated, no labels are shown in the visualization to reduce the number of rendered elements.
- Hierarchical layout: might improve the rendering of currently selected subgraphs with chains of connected edges (i.e., high depth).
- Split type nodes: while being activated, for all triples of the form `_:a rdf:type ?type`, the type nodes (`?type`) are intentionally rendered separately as it might lead otherwise to a centralization around types nodes.
- Show visualization options: This will provide an additional control panel (below the visualization) for defining parameters of the used rendering to create customized visualizations. All parameters of the renderer are provided here for individual adjustments by users (e.g., for solving rendering problems).

[3] cf. https://en.wikipedia.org/wiki/Kingfisher
[4] https://wse-research.org/kingvisher/
[5] https://www.stardog.com/

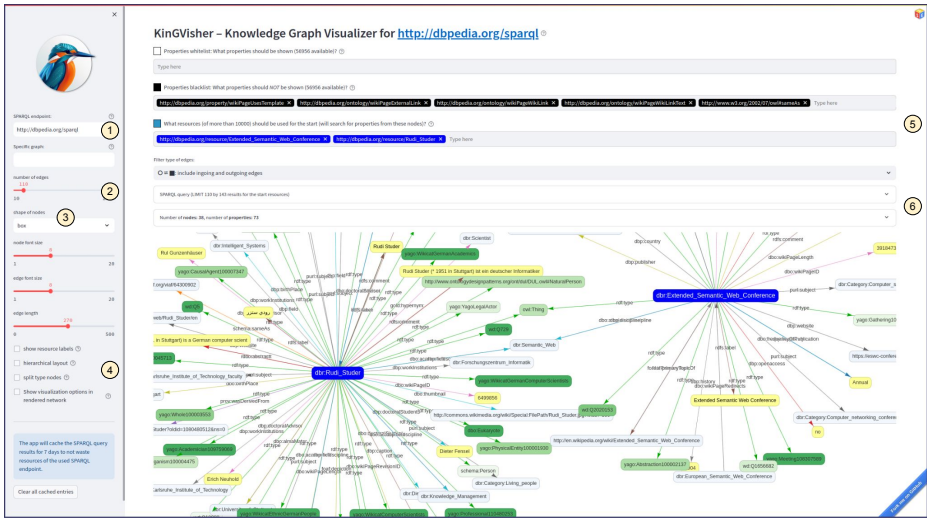

Fig. 1: Screenshot of the KinGVisher UI using the DBpedia SPARQL endpoint.

Additionally, the definition of blacklist (s.t., users can remove unwanted properties that might pollute the graph visualization) and whitelist properties (s.t., users can focus on just a limited set of properties) is possible (5). It is worth mentioning here that it is also possible to define specific, "important" resources that should always be included in the visualization. In this case, the knowledge graph is searched iteratively (using SPARQL) so that connections between these resources can be determined and displayed. These nodes of the visualization are colorized using solid blue. All other nodes are colorized by a shade of green depending on the maximum calculated from the input degree and output degree of the node, i.e., the more important nodes are displayed in a darker color (s.t., users can identify the well-connected nodes easily).

The internal process is made transparent by providing the used SPARQL queries (6), a short statistics of the found properties is shown (that can be fold-out to see the ranking of the properties by appearance in the currently shown sub-graph), as well as the ingoing and outgoing edges of a (clicked) node.

Note, that the performance is highly dependent on the responsiveness of the SPARQL endpoint as – for the sake of general applicability – only SPARQL queries are used to retrieve data. A rudimentary cache has also been integrated to minimize the triggered workload on the SPARQL endpoint which is particularly speeding up the runtime regarding very large knowledge graphs.

Other useful functionalities include the ability to export the image of the rendered Knowledge Graph as a PNG file and to share the current configuration via a URI.

The implementation was done using the Python library Streamlit[6] that provides an integrated technology stack for frontend and backend while serving as a bridge between Python and the React[7] library (JavaScript/TypeScript) to generate a

The application is available as an online demo[8] and pre-bundled as Docker images published on Dockerhub[9] for free usage. The source code is available[10] on GitHub and is published under the MIT open-source license.

3 Conclusions

In this paper, we presented KinGVisher – a web-based Knowledge Graph Visualizer – that can interact with any available SPARQL endpoint. Our demonstrator serves to help users interactively visualize and explore knowledge graphs. In contrast to other visualization tools, our implementation is by design capable of working with data (iteratively) retrieved from very large knowledge graphs. Due to the challenges of creating visualizations for (large) knowledge graphs, our tool offers many configuration options to put users in control (e.g., while allowing them to manipulate the complete configuration parameters). Many of these options were never introduced in other tools. Hence, we assume several benefits for the research community, e.g., while working with large knowledge graphs where filtering resources and properties is helpful, or for better understanding the patterns and data available in the context of particular data artifacts of a given knowledge graph.

Acknowledgments This work has been supported in part by the German ministry BMWi under grant 16DTM107B (*ASAGuR*) and a research project of ITZBund (Germany): "Entwicklung und Erforschung von IT-basierten Lösungen im Rahmen des ChatBot-Frameworks des Bundes (Question-Answering-Komponenten zur Erweiterung des ChatBot-Frameworks)".

References

1. Auer, S., Bizer, C., Kobilarov, G., Lehmann, J., Cyganiak, R., Ives, Z.: DBpedia: A nucleus for a web of open data. In: International Semantic Web Conference. pp. 722–735. Springer (2007). https://doi.org/10.1007/978-3-540-76298-0_52
2. Čerāns, K., Ovčiņņikova, J., Liepiņš, R., Grasmanis, M.: Extensible visualizations of ontologies in OWLGrEd. In: Hitzler, P., Kirrane, S., Hartig, O., de Boer, V., Vidal, M.E., Maleshkova, M., Schlobach, S., Hammar, K., Lasierra, N., Stadtmüller, S., Hose, K., Verborgh, R. (eds.) The Semantic Web: ESWC 2019 Satellite Events. pp. 191–196. Springer International Publishing, Cham (2019). https://doi.org/10.1007/978-3-030-32327-1_38

[6] https://streamlit.io/
[7] https://react.dev/
[8] https://wse-research.org/kingvisher-knowledge-graph-visualizer/
[9] https://hub.docker.com/r/wseresearch/knowledge-graph-visualizer
[10] https://github.com/WSE-research/KinGVisher-Knowledge-Graph-Visualizer

3. Ghosh, D., Rajabi, E.: KG-visual: a tool for visualizing RDF knowledge graphs. In: Research Conference on Metadata and Semantics Research. pp. 126–136. Springer (2021). https://doi.org/10.1007/978-3-030-98876-0_11
4. Jahn, F., Höffner, K., Schneider, B., Lörke, A., Pause, T., Ammenwerth, E., Winter, A.: The SnIK graph: visualization of a medical informatics ontology. In: MEDINFO 2019: Health and Wellbeing e-Networks for All, pp. 1941–1942. IOS Press (2019). https://doi.org/10.3233/SHTI190724
5. Kerdjoudj, F., Curé, O.: RDF knowledge graph visualization from a knowledge extraction system. In: Joint Proceedings of the 1st International Workshop on Summarizing and Presenting Entities and Ontologies and the 3rd International Workshop on Human Semantic Web Interfaces (SumPre 2015, HSWI 2015) co-located with the 12th ESWC, 2015. CEUR Workshop Proceedings, vol. 1556. CEUR-WS.org (2015)
6. Lohmann, S., Link, V., Marbach, E., Negru, S.: WebVOWL: Web-based visualization of ontologies. In: Lambrix, P., Hyvönen, E., Blomqvist, E., Presutti, V., Qi, G., Sattler, U., Ding, Y., Ghidini, C. (eds.) Knowledge Engineering and Knowledge Management. pp. 154–158. Springer International Publishing, Cham (2015). https://doi.org/10.1007/978-3-319-17966-7_21
7. Vrandečić, D., Krötzsch, M.: Wikidata: a free collaborative knowledgebase. Communications of the ACM **57**(10), 78–85 (2014). https://doi.org/10.1145/2629489
8. Zazuko: Graph Explorer (2024), `https://github.com/zazuko/graph-explorer`, accessed: 2024-03-01