

Dual View Knowledge Graph Explorer

Wolfgang Fahl¹ 

BITPlan GmbH, Willich, Germany

Abstract. Exploring Knowledge Graphs with Graph Walks by following edges from a start node of interest is a natural approach. Unfortunately there are often frustrating obstacles. Users might not find what they look for or a step in the walk is cumbersome or not possible. Links might have been rotten, information may be outdated or incomplete. For the Semantification of Usecases Graph Walks are a valuable source for gathering requirements in a “Specification by Example” style. This work introduces a Dual View Knowledge Graph Explorer. It is a software written in python with a demo as a Jupyter notebook.

Keywords: Knowledge Graph · Graph Walk

License: Apache 2.0

URL: <https://github.com/WolfgangFahl/DualViewKnowledgeGraphExplorer>

1 Introduction

User friendly exploring of knowledge graphs is a core selling point of the Semantic Web. There are plenty of cool presentations showing off what a knowledge graph might look like. Unfortunately this state of affairs is usually the result of hard work and the effort of a larger community. Wikidata [2] and OpenStreetMap [1] are examples of projects that managed to have a high impact and visibility.

2 Background

The Wikidata Walkabout (Q99677798)¹ won the Coolest Tool Award 2024. Starting from a class such as City users can select an instance of their choice e.g. London (Q84)² and end up at the corresponding Wikidata page. The shortcomings of this approach have motivated this work. Users might get frustrated when they end up in a presentation style that is technology focused. They might not be familiar with the properties being used and not be able to continue their exploration as intended.

3 Dual View Knowledge Graph explorer

The idea behind the Dual View Knowledge Graph explorer is to present views of a graph walk side by side. One view shows the Knowledge Graph representation of the

¹ <https://www.wikidata.org/wiki/Q99677798>

² <https://www.wikidata.org/wiki/Q84>

current node and one the “natural” view. A common pair is a Wikidata Q-Item shown side by side with the official website of the item.

The prototype of the explorer is implemented as a jupyter notebook. It uses a YAML File declaring

- entity types
- node types
- entities
- nodes
- a walk with steps

in a straightforward way. The python code to render the YAML content in dual view style was generated using Claude Sonnet 4.6 with some manual fixes necessary.

3.1 SEEK Commons Query Objects Use Case

The approach was first tried out at a SEEKCommons (Q118147033)³ Workshop in November 2025. The Graph Walk to be shown was quickly generated with a set of LLM Tools (GPT-5, Grok4, Gemini3, Claude Sonnet 4.6) accessed via OpenRouter to create a yaml file describing the walk see Listing 1.1

Listing 1.1. YAML description of the SEEK Commons Graph Walk

```

1 # Dual-View Knowledge Graph Explorer - Navigation Steps
2 # Author: Wolfgang Fahl / bitplan.com
3 # 2025-11-16 for SeekCommons Workshop
4
5 node_types:
6   organization:
7     label: "Organization"
8     wikidata_id: "Q43229"
9     icon: ""
10 ...
11
12 edge_types:
13   funded_by:
14     label: "funded_by"
15     wikidata_id: "P8324"
16     icon: ""
17 ...
18 nodes:
19   seekcommons:
20     label: "SeekCommons"
21     type: project
22     wikidata_id: "Q118147033"
23     web_url: "https://seekcommons.org/"
24 ...
25

```

³ <https://www.wikidata.org/wiki/Q118147033>

```

26
27 edges:
28   seekcommons_funded_by_nsf:
29     from: seekcommons
30     to: nsf_grant
31     type: funded_by
32
33
34 walks:
35   seekcommons_grant_walk:
36     label: "SeekCommons_Grant_Exploration"
37     start_node: seekcommons
38     sequence:
39       - edge: seekcommons_funded_by_nsf
40       - edge: nsf_operated_by_nd
41       - edge: nsf_has_pi_luis
42       - edge: luis_employed_by_nd

```

This walk is a typical “Happy Path” walk. Figure 1 shows the starting node of the walk displayed as a side by side view by the run kgwebwalker.py jupyter cell command.

4 Evaluation

The Feedback by the SEEKCommons workshop participants was encouraging. It seems feasible to deduct Query Objects from given Knowledge Graphs and discuss the parameterization based on the specification by example walks.

5 Discussion

This paper presents a Jupyter notebook prototype which seems to be just the tip of an iceberg. There is huge potential in this approach since the properties of nodes are not explicitly discussed. The implicitness of the approach is what seems fitting for modern applications. E.g. if you reference a person the persistent identifier is enough to find the persons details as a knowledge graph node and the entry which might be a webpage or a record in a database. There is no need to discuss the details of the webpage or the record - these are readily available for inspection by humans and computers.

6 Future Work

A Browser extension that allows to track the intentions of a user by tracing relevant user interface events such as link clicks and entering of data into input elements would be useful. For finding out the intents of a user their could be an interactive dialog asking questions about the role, background and needs. There might also be an option to let the user point to areas of interest on the screen. The Research question would be how state of the art LLM support would allow to navigate existing knowledge graph nodes

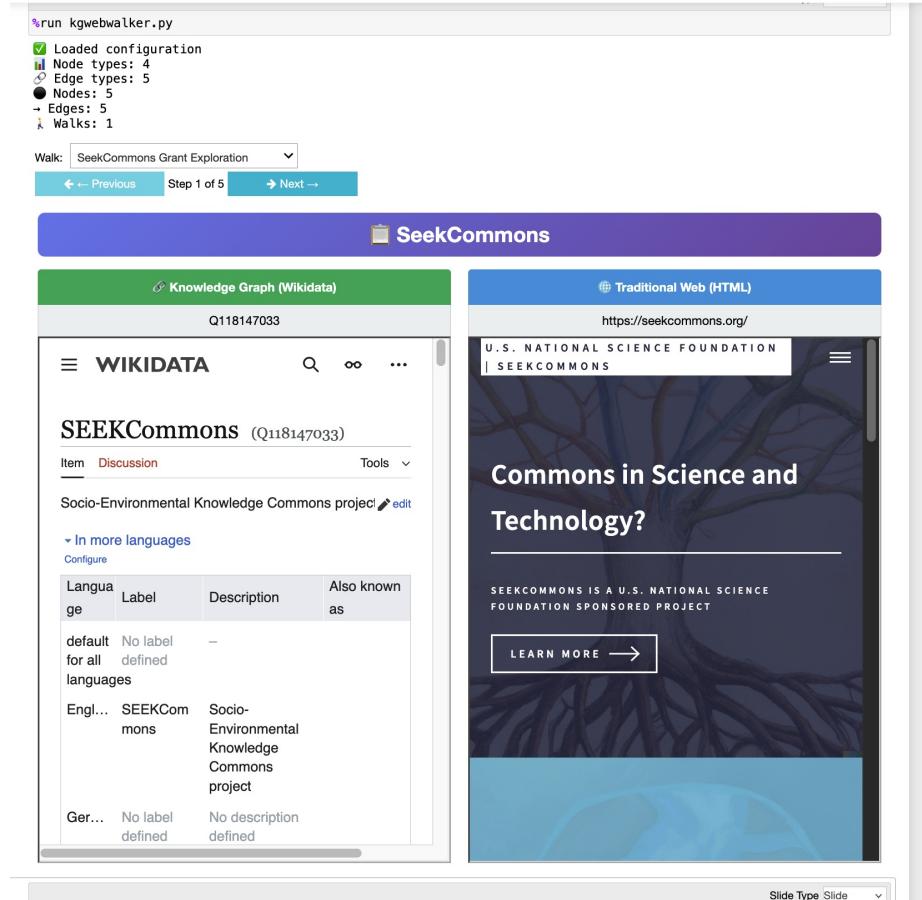


Fig. 1. Step 1 of the sample graph walk

in parallel and ideally suggest additions essentially leading to a full-fledge knowledge graph curation support tool.

References

1. Haklay, M., Weber, P.: Openstreetmap: User-generated street maps. *IEEE Pervasive Computing* **7**(4), 12–18 (2008). <https://doi.org/10.1109/MPRV.2008.80>
2. Vrandečić, D., Krötzsch, M.: Wikidata: a free collaborative knowledgebase. *Communications of the ACM* **57**(10), 78–85 (Sep 2014). <https://doi.org/10.1145/2629489>, <http://dx.doi.org/10.1145/2629489>

Fig. 2. Step 3 of the walk

Fig. 3. Step 4 of the walk

Fig. 4. Further progression in the SEEK Commons Graph Walk