`KGSnap!`: query Knowledge Graphs by Snap!

Vincenzo Offertucci[1], Maria Angela Pellegrino[1][0000−0001−8927−5833], and Vittorio Scarano[1][0000−0001−8437−5253]

Dipartimento di Informatica, Università degli Studi di Salerno, Italy
`v.offertucci@studenti.unisa.it`, `{mapellegrino,vitsca}@unisa.it`

Abstract. As the block programming paradigm has been successfully used to teach programming skills, this demo proposes `KGSnap!`, an extension of the block-based programming environment `Snap!`, which allows lay users to build and run queries on a SPARQL endpoint. The proposed approach has the potential to enable lay users to access knowledge graphs without requiring technical skills in query languages.

Keywords: SPARQL · Block-based programming · Snap!

1 Introduction & Related Work

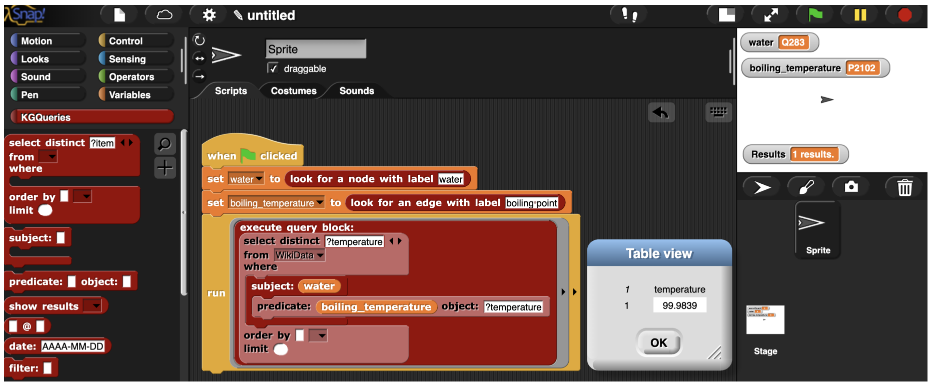
The Semantic Web technologies are increasingly used to model any field of interest, increasing the quantity and diversity of available data modeled by Knowledge Graphs (KGs). However, its potential risks to be left untapped due to the SPARQL complexity [6]. Lay users interested in consuming KGs require tools to mitigate the technical challenges SPARQL poses.

Block programming languages that guide users in dragging and connecting fragments shaped like jigsaw puzzle pieces have successfully introduced programming to non-experts [3]. Just think the vast exploitation of Blockly[1] as part of code.org's Hour of Code, Scratch [4] to create animations and games and MIT App Inventor [8] to build Android Apps. The proposal of letting lay users query data by block-based programming is not new. `SQheLper` [2], `DBLearn` [7], and `DBSnap++` [5] are block-based programming interfaces to query databases. In the Semantic Web community, Bottoni and Ceriani [1] proposed `SPARQL Playground`, introducing KGs querying in Blockly. This demo proposes `KGSnap!`, an extension of Snap! to query KGs that expose SPARQL endpoints.

2 `KGSnap!`: query Knowledge Graphs by Snap!

Snap![2] (formerly Build Your Own Blocks) is a free, open-source, block-based educational graphical programming language and online community allowing learners to explore, create, and remix interactive animations, games, and stories while learning about mathematical and computational ideas.

[1]Blockly, `https://developers.google.com/blockly`
[2]Snap! `https://snap.berkeley.edu`



```
when [flag] clicked
set water to look for a node with label water
set boiling_temperature to look for an edge with label boiling point
run
  execute query block:
    select distinct ?temperature
    from WikiData
    where
      subject: water
        predicate: boiling_temperature object: ?temperature
    order by
    limit
```

Table view

	temperature
1	99.9839

OK

Fig. 1. `KGSnap!` interface.

`KGSnap!` (visible in Fig. 1, freely accessible online[3] and available on GitHub with an Open Source license[4]) extends the Snap! architecture by introducing the possibility to perform SELECT queries on KGs provided with a working SPARQL endpoint. By default, `KGSnap!` is configured to query Wikidata. However, users can easily introduce any SPARQL endpoint of interest by a dedicated block shown in Fig. 2. `KGSnap!` mimics the structure of SPARQL queries to follow the philosophy of block programming to guide learners to experiment gradually with the underlying language and, in the end, to be able to switch to programming in that language. Hence, SPARQL queries are formulated by

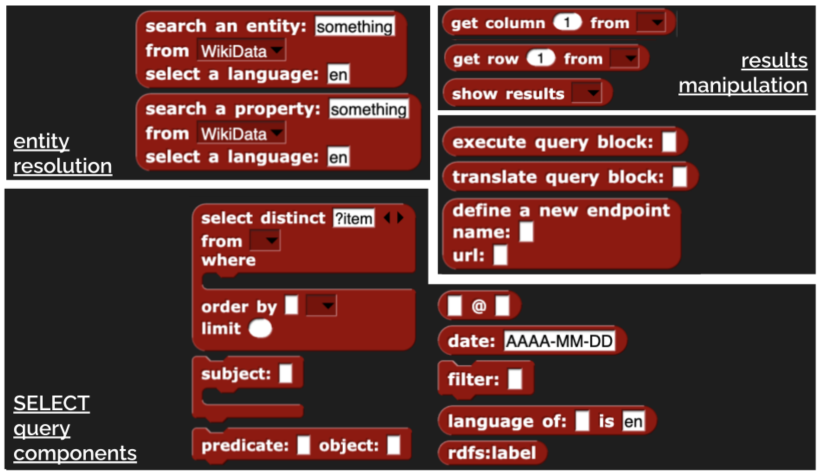
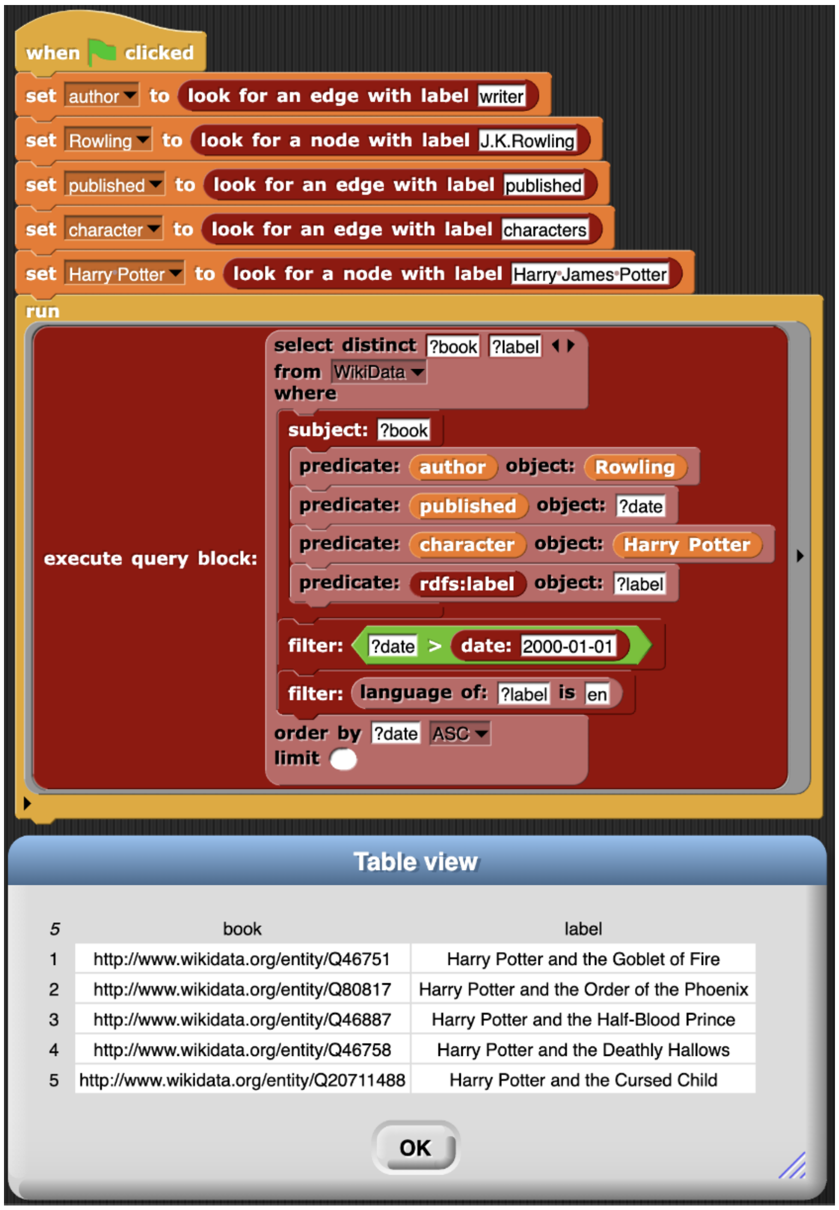


Fig. 2. Blocks implemented in `KGSnap!`.

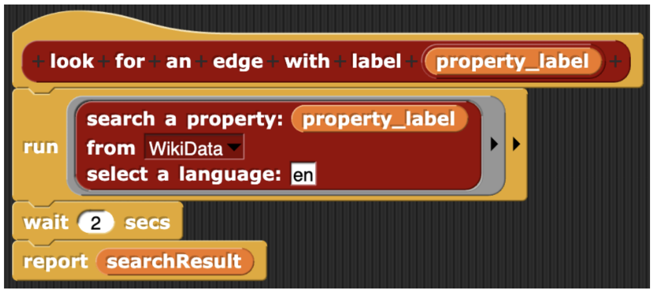
[3]KGSnap!: `https://isislab-unisa.github.io/Snap/snap.html`
[4]KGSnap! `https://github.com/isislab-unisa/KnowledgeGraphsAndSnap`

specifying triples (subject, predicate, object). The complete set of supported features is visible in Fig. 2 and concerns entity resolution from user-defined labels, components to assemble SELECT queries covering Basic Graph Patterns, such


when clicked
set author to look for an edge with label writer
set Rowling to look for a node with label J.K.Rowling
set published to look for an edge with label published
set character to look for an edge with label characters
set Harry Potter to look for a node with label Harry James Potter
run
execute query block:
select distinct ?book ?label
from WikiData
where
subject: ?book
predicate: author object: Rowling
predicate: published object: ?date
predicate: character object: Harry Potter
predicate: rdfs:label object: ?label
filter: ?date > date: 2000-01-01
filter: language of: ?label is en
order by ?date ASC
limit
Table view
OK

5	book	label
1	http://www.wikidata.org/entity/Q46751	Harry Potter and the Goblet of Fire
2	http://www.wikidata.org/entity/Q80817	Harry Potter and the Order of the Phoenix
3	http://www.wikidata.org/entity/Q46887	Harry Potter and the Half-Blood Prince
4	http://www.wikidata.org/entity/Q46758	Harry Potter and the Deathly Hallows
5	http://www.wikidata.org/entity/Q20711488	Harry Potter and the Cursed Child

Fig. 3. Use case of KGSnap!. We query Wikidata to retrieve books authored by J.K.Rowling with Harry Potter as character.



```
look for an edge with label (property_label)
run [search a property: (property_label)
     from WikiData
     select a language: en]
wait 2 secs
report (searchResult)
```

Fig. 4. Entity resolution performed by `KGSnap!` to solve user-defined labels.

as path traversal, filters, sorting, and other supporting features to manipulate results and introducing new endpoints. Once the SPARQL query is completed, users can visualize them as data tables and store specific results in variables to refine queries iteratively. Moreover, query results can be downloaded as JSON or CSV files, while the query can be downloaded as a TXT file.

Demonstration. During the demo, we will show `KGSnap!` in practice. Supposing we are interested in retrieving all the books authored by J.K. Rowling, having Harry Potter as a character and published in this century. The resulting query is visible in Fig. 3, which relies on the definitions of custom functions, wrapping blocks for performing entity resolution visible in Fig. 4.

References

1. Bottoni, P., Ceriani, M.: Sparql playground: A block programming tool to experiment with sparql. In: VOILA@ISWC. p. 103 (2015)
2. Jacobs, S., Jaschke, S.: Sqhelper: A block-based syntax support for sql. In: Global Engineering Education Conference (EDUCON). pp. 478–481. IEEE (2021)
3. Moon, H., Cheon, J., Lee, J.: Teaching block-based programming: A systematic review of current approaches and outcomes. In: Society for Information Technology & Teacher Education International Conference. pp. 73–78. Association for the Advancement of Computing in Education (AACE) (2020)
4. Resnick, M., Maloney, J., Monroy-Hernández, A., Rusk, N., Eastmond, E., Brennan, K., Millner, A., Rosenbaum, E., Silver, J., Silverman, B., et al.: Scratch: programming for all. Communications **52**(11), 60–67 (2009)
5. Silva, Y.N., Nieuwenhuyse, A., Schenk, T.G., Symons, A.: Dbsnap++: creating data-driven programs by snapping blocks. In: the 23rd Annual Conference on Innovation and Technology in Computer Science Education. pp. 170–175. ACM (2018)
6. Vargas, H., Aranda, C.B., Hogan, A., López, C.: RDF explorer: A visual SPARQL query builder. In: ISWC. pp. 647–663. Springer (2019)
7. Vinayakumar, R., Soman, K., Menon, P.: DB-learn: studying relational algebra concepts by snapping blocks. In: 9th International Conference on Computing, Communication and Networking Technologies (ICCCNT). pp. 1–6. IEEE (2018)
8. Wolber, D., Abelson, H., Spertus, E., Looney, L.: App inventor. O'Reilly Media, Inc. (2011)