KG Explorer: a Customisable Exploration Tool for Knowledge Graphs

Thibault Ehrhart, Pasquale Lisena, Raphaël Troncy

KG Explorer: a Customisable Exploration Tool for Knowledge Graphs

Thibault Ehrhart, Pasquale Lisena, and Raphaël Troncy

EURECOM, Sophia Antipolis, France
{ehrhart,lisena,troncy}@eurecom.fr

Abstract. The growing adoption of Knowledge Graphs demands new applications which enable users to search and browse structured data in a suitable way depending on the domain. In this paper, we introduce KG Explorer, a web-based exploratory search engine for RDF-based Knowledge Graphs. The software can be configured in order to adapt to different information domains, customising both the UI components and the queries made for retrieving the information. It also includes features such as full-text search, facet-based advanced search, and the possibility to create lists of favourites items modelled in the knowledge graph.

Keywords: knowledge graphs, data exploration, data access, search interface

1 Introduction

Knowledge Graphs (KG) are more and more adopted for representing the information: today we can find several graphs, small and large, which may represent encyclopedic-general or domain-specific information. Their still growing popularity is due to an interesting set of characteristics, such as explicit semantic, interlinking with external resources, and a great expressiveness coming from Semantic Web technologies. KGs offer structured data that empower semantic search and QA systems among many other possible applications.

More recently, we see the interest in creating beautiful visualisation of KG-powered search results. An example is the use of Knowledge Panels on Search Engines, which are currently moving from simply displaying key-value tuples to integrate images and text for presenting the information nicely (Fig. 1).

Knoweldge Graphs can be stored in dedicated triple store. Those, generally offer – next to the essential SPARQL endpoint – a browsing user interface (UI), which allows an end-user to see the loaded data on a web page. For example, the *facet browser* of Virtuoso[1] shows all incoming and outgoing predicates for a given resource with the respective values[2]. When the value represents a picture, the image is retrieved and displayed in the page. All entity nodes and edges are clickable, so that the user can navigate

[1] http://vos.openlinksw.com/

[2] Example from DBpedia: https://bit.ly/3z5nPLG

Fig. 1. The Knowledge Panel for the search keyword "Goat" in Google (left) and Bing (right). Screenshot taken on 19/03/2021

through the graph in a *follow-your-nose* approach. Other common features are plain text search (`/fct` on Virtuoso), query helper (YASGUI[3]), dereferencing service for linked data URI, or rich visualisation of results (like in Wikidata[4]).

However, these systems fall short when it is necessary to go beyond the simple visualisation of text and images and:

- embrace different media objects, such as video, audio, 3D graphics;
- propose new navigation paradigms, such as related items or recommendations for the next element;
- improve the search and exploration experience based on the domain peculiarities, filtering the results based on time ranges, geographic areas, or values form hierarchical thesauri. Moreover, the connection of the searched object and the value to filter can consist of a single direct property, a property path, or even a more complex query.

To provide a generic solution to these limitations, we introduce KG Explorer, a fully-customisable web application which serves as exploratory search engine [16] for Knowledge Graphs. KG Explorer offers alternative ways to browse a graph, to search and to follow links, to discover new information by exploiting the semantic proximity of entities. Respect to other works in literature, KG Explorer gives the possibility to configure the application components, that can be included or excluded and customised in the functionality or in the visualisation. In addition, a *save in the list* feature is included, bringing to information discovery applications a pattern coming from shopping exploratory search engine.

[3] `https://triply.cc/docs/yasgui-api`
[4] See for example `https://www.wikidata.org/wiki/Q2934`

This paper is organised as follows. After discussing some related work in Section 2, we detail some shared needs for the application in Section 3. The functionalities and architecture of KG Explorer are described respectively in Section 4 and 5. We present a preliminary evaluation in Section 6 and some conclusion in Section 7.

2 Related Work

An extensive survey of facet search has been published in [21]. This work has the merit of defining the basic concepts for the exploratory approach, namely the *extension* (the displayed results), the *intension* (the satisfied query) and the *transition markers*, clickable elements for triggering a transition (a new query). In addition, the work points out the possible kind of configuration of the tool, from the absence of any configuration requirement to the exact content to be displayed (view-based configuration).

Faceted Wikipedia Search [7] is a facet search tool based on DBpedia. The transition markers are sorted and displayed based on their frequency with respect to the number of results, in order to help the user in refining her/his query in successive iterations. Other provided features are free text search and range selection for datatype values. A similar interaction is implemented in *GraFa* [15], which refines the facet list after selecting the text keyword to search or the desired entity type. The involved schemas are indexed in order to have quick response, applying in addition a materialisation for the query returning the bigger number of results. These solutions are, however, based on statistics computed on properties, and do not take into account the domain specificity. In fact, the chosen facets are not always relevant nor useful for the search experience. The Metaphacts ecosystem[5] includes an extension for building customisable apps on top of Linked Data.

FERASAT [9] shows the results obtained through combination of facet values in different visualisation components (maps, charts, etc.), in order to make evident the surprising results. This application targets a public of data experts but it would be quite complex for a broader audience. *LDVizWiz* [1] provides aggregate visualisations for entities of specific types in a KG, such as events which can be displayed on maps, timelines and tables. *Loupe* [14] displays the ontology classes and properties frequently used in tabular format, allowing the user to see how they are normally combined in the triples. These works show exclusively aggregate results, without enabling any customisation depending on the investigated domain.

In *Overture* [11], the visualisation of entity data is extended with custom components, showing a timeline of relevant events and the most similar entities from on a knowledge-based recommender system. In the *WarSampo portal* [10] (about Finnish history in World War II), different tabs allows to switch between a tabular visualisation of data, a timeline and a map, and a photo gallery[6]. The resource page of *Genesis* [5] includes entity textual data, images and videos, as well as a selection of similar and related entities with their own depictions. These examples are ad-hoc developed tools, hard to adapt to new domains.

[5] `https://metaphacts.com/`

[6] Example: `https://www.sotasampo.fi/en/persons/person_61`

The Fresnel vocabulary [17] has been proposed for closing the gap between data and presentation, enabling to define content subsets and formats matched with CSS classes. Similarly, custom views are used for driving the visualisation in [2,20,4]. However, these approaches do not propose solutions to data search. Works like *PepeSearch* revealed good search capabilities, but the results are only shown in tabular form [22]. Other works combines exploratory search with facets [13,23]; however, these works do not focus on customising the user interface.

3 Different Scenarios But Shared Needs

Different users may take advantage from data inside specialised Knowledge Graphs, each one with their own needs and goals. We identified the following shared needs:

- to understand what is in the dataset, and in particular the main resource types (classes) and how they are connected to each other;
- to search for specific resource which satisfy some domain-relevant criteria;
- to obtain detailed information about a particular resource, including multimedia data and smart aggregations using timelines, maps and plots.

These need are highly impacted by the kind of user, which can fall in one of the following scenarios:

- *domain experts* have great interest in the subject, are used to the domain vocabulary and know what they search with precision. They need advanced search capabilities, allowing them to filter the results by several dimensions. The information needs to be complete.
- the *wide public* is rather moved by the curiosity of discovering something new, sometimes having only general or null knowledge about the domain. They need to easily browse the data collection and possibly reach relevant information already after the first click. Some strategies are needed to make them continue the exploration, for example follow-your-nose approaches or the recommendation of similar or related items. The engagement is crucial for their experience.
- *external stakeholders* need to know which relevant information is possible to find in the data and how to easily access it.

We argue that an exploratory search engine [16] enables to fulfil the described needs while being flexible enough to targeting the different personas. In addition, the application should have a proper user interface (UI), which reflects the domain specificity and the institution identity. In the same time, this can improve the final user engagement. Further desired elements are the selection of the language for KGs including multi-lingual contents and an authentication method for data that are not public.

4 KG Explorer Functionalities

Having defined the final goals, we are going to detail in this section the features implemented in KG Explorer: a facet-based advanced search engine, dedicated editorial pages for controlled vocabularies represented in SKOS and generally used in the

knowledge graph, a customised detailed page for the main entities represented in the knowledge graph, the possibility for users to log in and to create personalised lists of favourites or saved items. The software can be configured to adapt to different information domains, changing not only its aspect but also the queries for retrieving the data to display. KG Explorer is open source under Apache License 2.0 at `https://github.com/D2KLab/explorer`. In order to explain the software capabilities, we will refer to three in-use applications of KG Explorer. These examples use data coming from different domains (cultural heritage, television and news), each of them with proper customisation. The links to the applications and the source code are collected in Table 1.

ontologies	#entities	links
ADASilk - *domain*: silk heritage		
CIDOC-CRM CRMsci	675,112	Source code: `https://git.io/adasilk` Application: `https://ada.silknow.org/`
MeMAD Explorer - *domain:* TV and Radio programmes		
EBUcore	1,079,969	Source code: `https://git.io/memad-explorer` Application: `https://explorer.memad.eu/`
ASRAEL Search Engine - *domain:* news and events		
OpenAnnotation rNews schema.org	968,602	Source code: `http://bit.ly/asrael-se` Application: `http://asrael.eurecom.fr/search-engine`

Table 1. In-use instances of KG Explorer (including ASRAEL Search Engine which is a fork of the main tool).

4.1 A standardised experience

KG Explorer offers a user experience based on four different kinds of pages. The **landing page** contains a search box which allows the user to perform a free text search on entities modeled in the Knowledge Graph. When the user enters a search term, the exploratory search engine executes a SPARQL query with a `REGEX` filter in order to select all items that have a label or a title that partially matches the search terms. The search query algorithm can also be changed in the configuration file to cover all datatype properties of the graph. The results are shown in an auto-complete box.

The **browse page** (Fig. 2) contains a faceted search engine which allows users to perform an advanced search for the main entities of the Knowledge Graph. The sidebar on the left side contains facets (or filters). Each facet generates an extra condition to the main SPARQL query used for searching.

In addition to a textual search box, the exploratory search engine provides shortcuts to so-called **vocabulary pages**, which show all terms belonging to a particular thesaurus – e.g. a *ConceptScheme* in the SKOS namespace. These vocabularies are defined in the configuration file, and are usually materialised as concepts in the KG. Clicking on a vocabulary term will bring the user to a pre-filtered browse page, in order to see the related items in the graph.

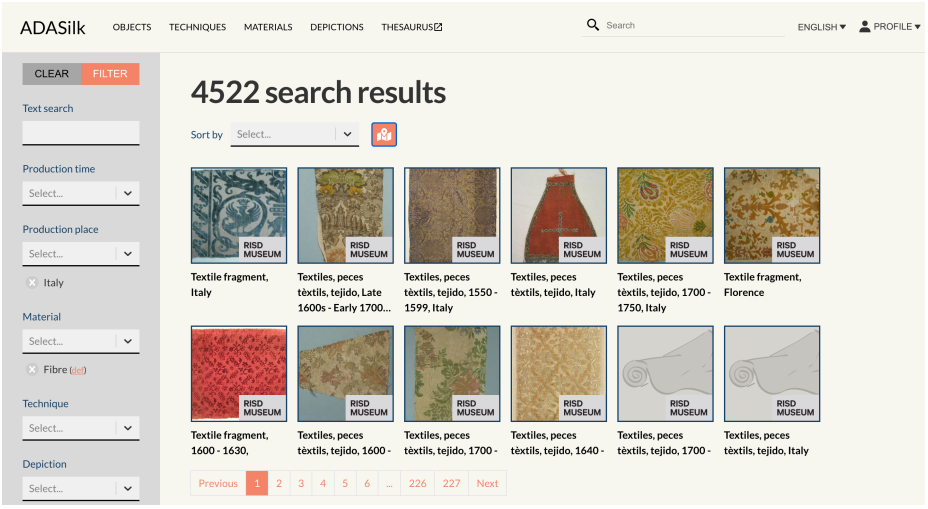

Fig. 2. The browse page in ADASilk.

Finally, the **detail page** shows all the information related to a single entity. There are currently 3 layouts available for detail pages: collection (grid-based list of items), gallery (carousel of images), and video (media player). Custom pages can be added by creating new JavaScript files in the `pages/` directory, and exporting the class as a React component. Each page is automatically included in the build and associated with a route based on its file name. New layouts can also be added to the project, by creating a new file in the `pages/details/` directory, and referring to its name in the `view` property in the configuration file. This is being used for developing the video player view in the MeMAD Explorer, handling also authentication to the media server.

4.2 User profiles

KG Explorer includes an authentication system which allows users to create an account, log in and have access to additional features. The OAuth authentication method is used for creating a new profile and for any successive login, relying on signing-in via Google, Facebook, and Twitter. Once logged in, users have the possibility to create named lists for storing searched items. A “save” button is present on each detail page, allowing to add the current page to an existing list or to create a new one. Lists can be retrieved in the profile page of the user, from where they can also be made public and shared with anyone using a permalink. Moreover, from the profile page it is possible to link or unlink additional OAuth accounts, as well as manage the existing lists or even delete the user profile.

4.3 Generic tool, custom configuration

Each domain and KG has its own characteristic. KG explorer is capable of working on top of any RDF-based Knowledge Graph, by configuring an instance of it using a

JavaScript file (`config.js`). The configuration allows to define a wide set of options, such as the chosen SPARQL endpoint, the supported language for internationalisation, and some layout-related settings – i.e. which images to use, which components to show or hide, etc.

Of particular interest is the possibility of defining the pages that compose the application, through the `route` field of the configuration file. The example in Listing 1[7] shows the available options, which include the choice between *browse* or *vocabulary* page, the page URI, the applied JSON query for listing the results (following the SPARQL Transformer syntax, as described in Section 5).

In *browse* pages, the `filters` property can contain a list of available fields for the advanced search, detailing also which changes are applied to the query when filters are applied. The list of available values can be loaded with a query (defined or made globally available as *vocabulary*). The main query condition is defined with the *baseWhere* property, with the minimal amount of triples required in order to improve performances. Once the list of results has been fetched, a second query is made to get the details of each result. This query is defined within the *query* property. The labels for the **internationalisation** are collected in specific JSON files to include in the project directory.

The front-end also supports **custom styles** which can be defined in a `theme.js` file. This allows to further customise the appearance of the user interface. It is possible to choose the global font set and a custom colour palette. Moreover, specific components can also be customised, by using the name of component and defining CSS rules following the *styled-components* syntax. Finally, adding custom pages and view (Section 4.1) enable the developer to include new **visualisation components**. Examples are maps and 3D visualisation in ADASilk.

5 Architecture

Fig. 3 shows an overview of the architecture and the technologies used in KG Explorer. KG Explorer is developed in a **containerised approach**, implemented within the Docker framework[8]: thanks to the use of independent and self-sufficient containers, Docker enables the deployment of this architecture on any machine, automatically installing and running the required software. This approach also allows to easily extend and deploy new instances of the application from the base image, including custom configuration and assets, as has been done in the instances in Table 1.

The **web application** is composed of several web technologies. The front-end is produced using *React*[9]. It uses encapsulated components that manage their own state to help maximise code re-usability. *Next.js*[10] is used for server-side rendering and page-based routing. It relies on a file-based structure for routing, where each page has its own file, stored in the `src/pages` directory. Special routes are dedicated to serve

[7] The code is extracted from the ADASilk configuration and is fully available at `https://github.com/silknow/adasilk/blob/main/config/routes/object.js`

[8] `https://www.docker.com/`

[9] `https://reactjs.org/`

[10] `https://nextjs.org/`

```
{
    objects: {
        view: 'browse', // type of view ('browse' or 'vocabulary')
        showInNavbar: true,
        rdfType: 'http://erlangen-crm.org/current/E22_Man-Made_Object',
        uriBase: 'http://data.silknow.org/object',
        details: { view: 'gallery' },
        filters: [{  // set of filters to appear in the advanced search
          id: 'material', // material filter
          isMulti: true, // 1 or more values can be selected
          isSortable: true,
          vocabulary: 'material', // values taken from a vocabulary
          whereFunc: () => [ // added to the base query when filtering
            '?production ecrm:P126_employed ?material',
            `OPTIONAL {
               ?broaderMat (skos:member|skos:narrower)* ?material }`
          ],
          filterFunc: (values) => { // add to base query when filtering
            return [values.map((val) =>
            `?material = <${val}> || ?broaderMaterial = <${val}>`)
                .join(' || ')];}
        }],
        baseWhere: [
            'GRAPH ?g { ?id a ecrm:E22_Man-Made_Object }',
            '?production ecrm:P108_has_produced ?id',
        ],
        query: { // base query
          '@graph': [{
            '@type': 'http://erlangen-crm.org/.../E22_Man-Made_Object',
            '@id': '?id',
            '@graph': '?g',
            label: '$rdfs:label',
            identifier: '$dc:identifier',
            description: '$ecrm:P3_has_note',
          }],
          $where: ['GRAPH ?g { ?id a ecrm:E22_Man-Made_Object }']
        }
    }
}
```

Listing 1: Partial definition of the ‘Objects’ route in ADASilk, with the optional filter by material

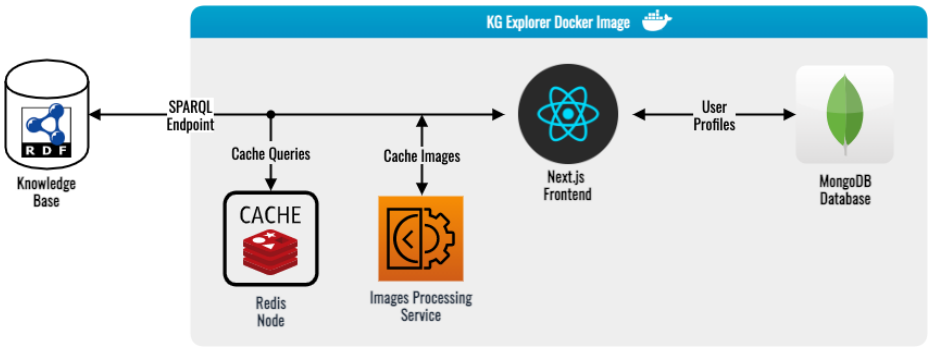

Fig. 3. Architecture of KG Explorer

APIs, used on the server-side for handling authentication, fetching profile data, and searching. The library *styled-components*[11] is used for styling components using scoped CSS. Other used frameworks are *i18next*[12] for the internationalisation and *next-auth*[13] for OAuth authentication.

KG Explorer makes requests to a Knowledge Graph through its exposed SPARQL endpoint. In order to easily include and manipulate queries in JavaScript, those are written in the JSON query syntax proposed by **SPARQL Transformer** [12]. The SPARQL Transformer library makes it easy to define queries using JavaScript objects (called *JSON queries*) which can be edited and merged to create the final query. For instance, each filter from the faceted search appends its own conditions to the base query, as seen in Section 4.3. Looking again at Listing 1, when a filter is applied, the base query is modified applying new `WHERE` and `FILTER` expressions, respectively defined in `whereFunc` and `filterFunc`. The use of JSON queries makes it possible to simply append this expression in the `$where` and `$filter` properties of SPARQL Transformer, and avoids a much more complex manipulation of text which the use of plain SPARQL queries would require. SPARQL Transformer also rewrites the output of SPARQL queries in a more suitable format for web development. In particular, SPARQL results composed of bindings between variables and solutions are transformed into self-contained JSON objects, including all the information about the entities, getting rid of some verbosity of the standard notation. Queries results are processed and cached into a *Redis* database[14] in order to improve performances. The results are stored as a JSON string, and the original query is used as the key for retrieving the cached result. User profiles and lists are saved in a *MongoDB* database[15].

[11] `https://styled-components.com/`
[12] `https://www.i18next.com/`
[13] `https://next-auth.js.org/`
[14] `https://redis.io/`
[15] `https://www.mongodb.com/`

6 Preliminary Evaluation

Preliminary evaluations of KG Explorer were conducted as part of the SILKNOW project [18]. The application has been used by 216 users, reflecting different audience, domain and technical skills (Table 2). The users were asked to perform some search activities and to comment on the results reflecting both the intrinsic quality of the knowledge graph which is hard to isolate and the ability of searching for specific items and of browsing and discovering new items.

Domain	English	French	Spanish	Italian	**Total**
Cultural Heritage	0	0	14	14	**28**
Education related to social science	1	0	6	4	**10**
Information and communication technology	1	17	42	67	**126**
Textile or creative industry	0	1	1	1	**3**
Tourism	0	1	0	2	**3**
Media	0	2	2	3	**7**
Other	0	2	15	22	**39**
					216

Table 2. Target audience used during the evaluation of ADASilk.

During the evaluation, each user session has been recorded, after consent, for successive analysis. To do this, the rrweb[16] library is implemented into the UI in order to record and then replay each interaction with the interface. The recorded sessions are saved as JSON objects in a database. At the end of the evaluation, the sessions were exported as MP4 videos using rrvideo.[17] We report below the most common issues and what users perceive as anomalous behaviour.

From the analysis of all the tests conducted through ADASilk, a commonly encountered issue is related to the text search functionality. While offering free text search was found to be an essential feature, it also raises some expectations that the search query will be somehow interpreted. Users are familiar with Google which interprets and disambiguates search queries while offering personalized answers. In contrast, KG Explorer offers either a naive *text search* that aims to match resources for which the search terms can be encountered in a datatype property value or a *concept search* which can lookup and auto-complete concepts from controlled vocabularies typically used in facets. Often, users have entered simple search strings expecting that their translations in other languages will bring the same result set.

The relevance of the search results was also pointed out as an issue during the evaluation, in particular, by domain experts. The sole SPARQL query language offers only the possibility of returning a set of exact solutions to a query without natural ways of ranking the resources within this set nor with the possibility to consider partially related resources. The numerous methods enabling to build knowledge graph embeddings are

[16] `https://github.com/rrweb-io/rrweb`
[17] `https://github.com/rrweb-io/rrvideo`

promising to bring this notion of relevance, e.g., in measuring the distance between each document. We observe that some triple stores, such as GraphDB[18], have started to provide native support for semantic similarity searches.

After a software improvement in order to overcome the observed limitations, a second evaluation has been done using the System Usability Score (SUS) questionnaire [3]. The participants has been composed of 125 people representative of the previously defined stakeholders. The participant were speaking English, Spanish or Italian, were mostly of higher education (75%) and in the age range 21-30 (59%)[19]. Even with possibility of improvement – the system obtained a SUS score of 67.03 – over 70% of the participants declared the intention to use it [6].

7 Conclusion and Future Work

KG Explorer provides a domain-specific user experience for exploring the information contained in a Knowledge Graph. The software can be easily customised and adapted in the UI and in the content, defining the queries for retrieving the data, the facets to be used, and the relevant vocabularies. KG Explorer is already used in real-world applications, in particular as wide-public entry-point for Knowledge Graphs of research projects. In this context, a user evaluation is currently being carried out where the goal is to measure the usability of the application in the fulfilment of common tasks, identified by domain experts. The outcome of this evaluation will be used for further improving the application.

Future developments will also involve new functionalities such as having custom facet selectors for datatypes, for example ranges for numbers and dates. Finally, we would like to exploit the vocabularies in order to provide a smart text search field, going beyond the exact match on text: this can be implemented by recognising terms defined in vocabularies and attaching them to the most appropriate property in the generated query, in a query interpretation behaviour. In this field, previous research has proved the suitability of embedding techniques for representing a query, in order to get more relevant results [8,24]. This text search may be also be further combined with structured search.

Acknowledgements

This work has been partially supported by the European Union's Horizon 2020 research and innovation program within the SILKNOW (grant agreement No. 769504) and MeMAD (grant agreement No. 780069) projects, and by the French National Research Agency (ANR) within the ASRAEL project (grant number ANR-15-CE23-0018).

References

1. Atemezing, G.A., Troncy, R.: Towards a Linked-Data Based Visualization Wizard. In: 5^{th} International Conference on Consuming Linked Data (COLD). Riva del Garda, Italy (2014)

[18] `https://graphdb.ontotext.com/`

[19] More detail about distribution of participants is available in [19]

2. Berners-Lee, T., Chen, Y., Chilton, L., Connolly, D., Dhanaraj, R., Hollenbach, J., Lerer, A., Sheets, D.: Tabulator: Exploring and Analyzing linked data on the Semantic Web. In: 3^{rd} International Semantic Web User Interaction Workshop (SWUI) (2006)
3. Brooke, J.: SUS: a retrospective. Journal of usability studies **8**(2), 29–40 (2013)
4. Chauvat, N., Amarger, F., Wouters, L.: Un navigateur pour le Web des données liées. In: 30es Journées Francophones d'Ingénierie des Connaissances, IC 2019. pp. 167–182. Toulouse, France (2019)
5. Ermilov, T., Moussallem, D., Usbeck, R., Ngonga Ngomo, A.C.: GENESIS: A Generic RDF Data Access Interface. In: International Conference on Web Intelligence (WI). pp. 125—131 (2017)
6. Gaitán, M., León, A.: SILKNOW System Evaluation. project deliverable D7.6, H2020 SILKNOW (2021)
7. Hahn, R., Bizer, C., Sahnwaldt, C., Herta, C., Robinson, S., Bürgle, M., Düwiger, H., Scheel, U.: Faceted Wikipedia Search. In: 13^{th} Conference on Business Information Systems (BIS) (2010)
8. Hamilton, W.L., Bajaj, P., Zitnik, M., Jurafsky, D., Leskovec, J.: Embedding Logical Queries on Knowledge Graphs. In: 32^{nd} International Conference on Neural Information Processing Systems (NIPS). pp. 2030—2041 (2018)
9. Khalili, A., van den Besselaar, P., de Graaf, K.A.: FERASAT: A Serendipity-Fostering Faceted Browser for Linked Data. In: 17^{th} International Semantic Web Conference (ISWC). pp. 351–366 (2018)
10. Koho, M., Ikkala, E., Leskinen, P., Tamper, M., Tuominen, J., Hyvönen, E.: WarSampo knowledge graph: Finland in the Second World War as Linked Open Data. Semantic Web Journal pp. 1–14 (2020)
11. Lisena, P., Achichi, M., Fernandez, E., Todorov, K., Troncy, R.: Exploring Linked Classical Music Catalogs with OVERTURE. In: 15^{th} International Semantic Web Conference (ISWC), Posters & Demos Track. Kobe, Japan (2016)
12. Lisena, P., Meroño-Peñuela, A., Kuhn, T., Troncy, R.: Easy Web API Development with SPARQL Transformer. In: 18^{th} International Semantic Web Conference (ISWC). pp. 454–470. Auckland, New Zealand (2019)
13. Marie, N., Gandon, F., Ribière, M., Rodio, F.: Discovery Hub: On-the-Fly Linked Data Exploratory Search. In: Proceedings of the 9th International Conference on Semantic Systems. p. 17–24. I-SEMANTICS '13, Association for Computing Machinery, New York, NY, USA (2013). https://doi.org/10.1145/2506182.2506185, https://doi.org/10.1145/2506182.2506185
14. Mihindukulasooriya, N., Poveda-Villalón, M., García-Castro, R., Gómez-Pérez, A.: Loupe - An Online Tool for Inspecting Datasets in the Linked Data Cloud. In: 14^{th} International Semantic Web Conference (Posters & Demos) (2015)
15. Moreno-Vega, J., Hogan, A.: Grafa: Scalable faceted browsing for rdf graphs. In: 17^{th} International Semantic Web Conference (ISWC). pp. 301–317 (2018)
16. Palagi, E., Gandon, F., Giboin, A., Troncy, R.: A Survey of Definitions and Models of Exploratory Search. In: ACM Workshop on Exploratory Search and Interactive Data Analytics (ESIDA). Limassol, Cyprus (2017)
17. Pietriga, E., Bizer, C., Karger, D., Lee, R.: Fresnel: A Browser-Independent Presentation Vocabulary for RDF. In: 5^{th} International Semantic Web Conference (ISWC). pp. 158–171 (2006)
18. Seidita, V., Lo Cicero, G., Vitella, M.: Testing Report in a Real Scenario. project deliverable D7.2, H2020 SILKNOW (2021)
19. Seidita, V., Lo Cicero, G., Vitella, M., Mladenic, D., Gaitán, M., Troncy, R., Portales, C.: Usability evaluation by online users of the system. project deliverable D7.3, H2020 SILKNOW (2021)

20. Tummarello, G., Cyganiak, R., Catasta, M., Danielczyk, S., Delbru, R., Decker, S.: Sig.ma: Live views on the Web of Data. Journal of Web Semantics **8**(4), 355–364 (2010)
21. Tzitzikas, Y., Manolis, N., Papadakos, P.: Faceted Exploration of RDF/S Datasets: A Survey. Journal of Intelligent Information Systems **48**(2), 329—-364 (2017)
22. Vega-Gorgojo, G., Giese, M., Heggestøyl, S., Soylu, A., Waaler, A.: Pepe-search: Semantic data for the masses. PLOS ONE **11**(3), 1–12 (03 2016). https://doi.org/10.1371/journal.pone.0151573, https://doi.org/10.1371/journal.pone.0151573
23. Waitelonis, J., Sack, H.: Towards exploratory video search using linked data. Multimedia Tools and Applications **59**(2), 645–672 (Jul 2012). https://doi.org/10.1007/s11042-011-0733-1, https://doi.org/10.1007/s11042-011-0733-1
24. Xiong, C., Power, R., Callan, J.: Explicit Semantic Ranking for Academic Search via Knowledge Graph Embedding. In: 26^{th} International Conference on World Wide Web (WWW). pp. 1271—-1279. Perth, Australia (2017)