

Enabling Exploratory Search on Manufacturing Knowledge Graphs

Kevin Haller¹, Fajar J. Ekaputra^{1,2}, Marta Sabou^{2,1} and Florina Piroi¹

¹TU Wien, Vienna, Austria

²WU Wien, Vienna, Austria

Abstract

Knowledge graphs have been recognized in manufacturing as a suitable technology for integration of multidisciplinary knowledge from heterogeneous data sources. The effective reuse of this knowledge can better inform stakeholders in their decision making processes and consequently, establish a competitive advantage. In contrast to the utilization of knowledge graphs for autonomous decision making systems, less attention in production research has been given to the creative participation of humans in the exploration of manufacturing knowledge graphs. Exploratory search systems are a promising solution to facilitate this participation. However, most exploratory search systems focus on general knowledge graphs for which common knowledge is sufficient. We argue that within the complex environment of manufacturing, closer attention has to be paid to particular exploratory search features. In this paper, we therefore present a configurable and adaptive exploratory search system, which implements three special features. Firstly, adaptability of the system to multiple (engineering) perspectives. Secondly, visibility of provenance details about statements to simplify investigative work. And finally, a tree view for browsing deep hierarchical structures.

Keywords

knowledge graph, exploratory search, industry 4.0

1. Introduction

The new paradigm shift in manufacturing, which is commonly referred to as the *fourth industrial revolution*, is guided by the fusion of traditional manufacturing technology with modern information and communication technology [1]. This vision is investigated by several strategic initiatives such as *Industrie 4.0* in Germany [2]. Core to all of these initiatives is the digitization of multidisciplinary information about production systems and processes.

Knowledge graphs (as defined by Galkin et al. [3]) are one solution to facilitate this digital transformation. As shown by the survey of Li et al. [4], knowledge graphs have received considerable attention in production research over the last years. In fact, it has been widely recognized as an important component of the next generation of information systems for


VOILA! 2022: 7th International Workshop on Visualization and Interaction for Ontologies and Linked Data, October 23, 2022, Virtual Workshop, Hangzhou, China, Co-located with ISWC 2022.

✉ kevin.haller@tuwien.ac.at (K. Haller); fajar.ekaputra@tuwien.ac.at (F.J. Ekaputra); marta.sabou@wu.ac.at (M. Sabou); florina.piroi@tuwien.ac.at (F. Piroi)

🆔 0000-0001-8949-610X (K. Haller); 0000-0003-4569-2496 (F.J. Ekaputra); 0000-0001-9301-8418 (M. Sabou); 0000-0001-7584-6439 (F. Piroi)



© 2022 Copyright for this paper by its authors. Use permitted under Creative Commons License Attribution 4.0 International (CC BY 4.0).

 CEUR Workshop Proceedings (CEUR-WS.org)

manufacturing [4]. Moreover, industrial enterprises started to construct knowledge graphs related to manufacturing such as Bosch [5] or Siemens [6].

Problem. The effective reuse of manufacturing knowledge can better inform stakeholders in their decision making and consequently, establish a competitive advantage [7]. Much work in production research has been dedicated to the utilization of knowledge graphs for autonomous decision making systems, e.g., Rožanec et al. [8]. However, less attention has been given to the creative participation of human stakeholders in the exploration of manufacturing knowledge graphs. In particular, we present two concrete use cases in which this need arises:

(UC1) *OntoTrans*¹ is an EU project (H2020) which aims to design an environment of tools for translators who are working on innovation challenges to manufacturing processes. Three companies take part in this project with their individual innovation challenge to improve a certain manufactured product in respect to several key performance indicators. An innovation challenge requires the interactive collaboration of stakeholders from different disciplines who have diverging perspectives and conceptualizations of a problem. A translator has to provide a communication-based glue between the involved stakeholders. The innovation challenge as well as the corresponding manufacturing information are converted into an ontological form. A tool to explore this interdisciplinary knowledge graph could assist translators in their tasks.

(UC2) *Aspern pilot factory*² is one of several Austrian "learning and experimentation factories". It currently hosts a series of valuable collaborative and industrial robotic arms as well as a wide range of supporting tools (grippers, 3D cameras, projectors, etc.), which can be used by students, researchers, and companies for their own purposes. However, interested students, researchers, and companies are often unaware of the availability and capabilities of the manufacturing technology in this pilot factory, which contributes to a relatively low usage degree of these expensive, state of the art production equipment. Furthermore, best practises and design patterns in software engineering are a valuable guide for writing robotics software. A knowledge graph was built from this interdisciplinary knowledge and a tool to explore it could help users of the pilot factory to learn about how to realize their individual projects with the available equipment.

Solution. Exploratory search systems are a promising solution to support human stakeholders in their decision and sense making. Exploratory search is an open-ended and multi-faceted information-seeking activity. It is commonly used in the context of scientific discovery, learning and decision making [9]. Most exploratory search systems focus on general knowledge graphs for which common knowledge is sufficient. We argue that within in the complex environment of manufacturing, closer attention has to be paid to particular exploratory search features.

Contribution. To that end, we provide in Section 2, a list of exploratory search features towards which we want to draw special attention, because they were commonly requested in interviews with stakeholders of our two use cases. Section 3 proposes a configurable and adaptive exploratory search interface, which considers the special search features for manufacturing

¹<https://ontotrans.eu/>

²<https://www.pilotfabrik.at/language/en/>

Common Features (from Marie et al. [10])	Manufacturing Features
(1) Overview and analysis feature (2) Faceted interface (3) Result clustering (4) Facilitator for back and forth navigation (5) Query-suggestions and refinement (6) Serendipitous discovery enforcement (7) Result explanation generator (8) Memorization feature	(9) Multiperspectival exploration (10) Provenance visibility (11) Hierarchical browsing

Table 1
Exploratory search features for the domain of manufacturing.

domain. The system architecture of our exploratory search is briefly introduced in Section 4. Finally, the preliminary evaluation of our exploratory search system is presented in Section 5.

2. Features for Manufacturing

A survey about widespread features in KG-based exploratory search systems was conducted by Marie et al. [10] (see left-side of Table 1). Three additional features that are important to manufacturing professionals were identified from informal interviews (see right-side of Table 1). Two translators to an industry partner in OntoTrans (UC1), two smart manufacturing researchers working in the pilot factory (UC2) with industry and university background respectively, and one simulation expert of production plants were among the interviewees. They were given an introduction to a basic exploratory search system in beforehand, and were then asked to discuss their requirements towards such a search system. The remaining part of this section elaborates on the three extracted features from these collected requirements.

Multiperspectival exploration enables stakeholders to select and switch between different perspectives on a manufacturing knowledge graph. In a manufacturing environment, stakeholders from multiple disciplines work together and they might have different conceptualization of manufacturing entities. Moreover, different disciplines might have a distinct perspective on what information is relevant about a manufacturing entity, and don't want to be overloaded with information that is irrelevant to them. A machinery could for instance be viewed in an impressively detailed manner by mechanical engineers, but only be seen as a black box with certain wiring requirements by electrical engineers.

Provenance visibility is a valuable feature in a manufacturing environment, where interdisciplinary knowledge is integrated from many heterogeneous sources. If a stakeholder is interested in investigating the properties of a physical component, then it might be necessary to know the original data source in which these values for a property have been reported in order to resolve ambiguities for proper decision making. The value of a parameter could have been gathered from promotional material of the manufacturer, or could have been measured by a local engineer. The visibility of provenance information allows stakeholders to quickly reason about the trustworthiness of presented statements.

Hierarchical browsing is a frequent search task over manufacturing knowledge graphs in which the digital twin of production plants or machines might be comprised of deep containment relationships between components. Furthermore, manufacturing processes and bill of materials

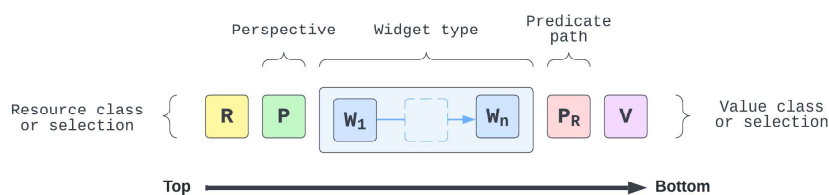


Figure 1: Scope hierarchy for the adaptive UI engine.

of products can have deep hierarchies. Thus, stakeholders must be able to quickly navigate through these hierarchies, and be able to cognitively focus on important parts and not be overwhelmed by the whole hierarchy.

3. Adaptive Exploratory Search Interface

Based on our interviews with stakeholders from the two manufacturing use cases, exploratory search interfaces in manufacturing ought to be designed with the requirements from Section 2 in mind. Preferences about the presentation method of entities and arrangement of interface elements might differ from use case to use case. Bootstrapping a new exploratory search system is however a time-consuming and costly endeavour. Thus, we propose in this section a configurable and adaptive search interface, which aims to be flexible and reusable. The architecture of the whole system is outlined in Section 4.

This adaptive search interface is based on the concept of *scopes* and *configurations*, which was introduced by the Linked Data Reactor (LD-R) [11]. A scope is in LD-R a hierarchical permutation of *dataset*, *resource*, *property* and *value*, where *dataset* is at the top and *value* at the bottom of this hierarchy. A presentation template (i.e. a *configuration*) can then be written in JSON format for a particular scope, which tells the web application how to render entities in the knowledge graph that match this scope. Each scope has a specificity, and if entities belong to multiple scopes, then the configuration of the most specific scope overwrites the others. While we adapt this mechanism for our adaptive search interface, the scope components and their hierarchy were changed to integrate multiperspectival exploration.

Scope components are organized in the hierarchy that is depicted in Figure 1.

- At the top of this hierarchy is the *resource class or selection* **R**. This can either be the IRI of a class of resources or an IRI matching one single specific resource. The class `rdfs:Resource` can be used as a wildcard to match all resources in a knowledge graph.
- The following scope component is the *perspective* **P**, which allows to adapt metrics powering the search interface as well as the presentation of widgets and entities to the

currently selected perspective. The underscore (i.e. '_') is a wildcard and represents all perspectives.

- A list of *widgets* $\mathbf{W}_1, \dots, \mathbf{W}_n$ is the next component in the hierarchy. Widget refers here to an interface element with an unique name. A list always starts with the outermost widget and ends with the innermost. The distinct presentation of entities in different widgets (e.g. bookmarks and result overview) is made possible with this scope component.
- *Predicate path* \mathbf{P}_R is one level below in the hierarchy. A predicate path is a subset of a property path in SPARQL, only allowing sequences (i.e. '/'), alternatives (i.e. '|') and inversion (i.e. '^'). A predicate path makes it possible to configure how a matching property itself is presented and how a collection of it's values should be visualized.
- At the bottom is the *value class or selection* \mathbf{V} . This scope component has the purpose to configure the presentation of matching values. In order to facilitate the design of UI components for visualizing provenance details, metadata about relevant named graphs is passed to values. In our solution, it is assumed that the *single-triple named graphs* approach is used to state provenance information in favor of *RDF reification*, *singleton properties* [12] and *RDF-star* [13].

Templates are aggregated in files written in HCL³, which is easy to edit and read by humans. Listing 1 shows a snippet of such a configuration file for the class `RobotType`. UI components are assigned with 'handler' to widgets, properties, and values. These UI components can moreover be customized by passing properties with a 'config' object. Line 29-39 in Listing 1 states that every value for the property "reach" of a `RobotType` shall be rendered as ordinary text literal as long as it isn't a quantity value from the QUDT ontology, which needs some additional parsing. Listing 2 shows how the recommendation section for instances of `RobotType` is configured to use LDS [14] in general, but is limited to specific classes for the two mentioned perspectives. Similarly, distinct ranking metrics could be chosen.

³Hashicorp configuration language - <https://github.com/hashicorp/hcl>

Listing 1: Configuration of RobotType.

```

(10) class = "cobot:RobotType"
(11)
(12) perspective _ widget infobox {
(13)   handler = "GeneralInfoBox"
(14)   config {
(15)     sections = ["prop_table",
(16)                 "recommendations"]
(17)   }
(18) }
(19) perspective _ widget infobox section prop_table {
(20)   handler = "ProvenanceTableSection",
(21)   config {
(22)     neighbourhood {
(23)       include = ["cobot:degreesOfFreedom",
(24)                 "cobot:handlingPayload", "cobot:reach",
(25)                 "cobot:skills"],
(26)     }
(27)   }
(28) }
(29) perspective _ widget infobox {
(30)   property "cobot:reach" {
(31)     handler = "LinkedProperty"
(32)     value _ {
(33)       handler = "TextValue"
(34)     }
(35)     value "qudt:Quantity" {
(36)       handler = "QudtQuantityValue"
(37)     }
(38)   }
(39) }

```

Listing 2: Different recommendation configurations per perspective.

```

(40) perspective _ widget infobox {
(41)   section recommendations {
(42)     handler = "SimilaritySection"
(43)     config {
(44)       number = 4
(45)       ranking = {
(46)         ldsd {
(47)           step = "esm.exploit.sim.ldsd"
(48)           weight = -1.0
(49)         }
(50)       }
(51)     }
(52)   }
(53) }
(54) perspective RoboticsEngineer widget infobox {
(55)   section recommendations {
(56)     config {
(57)       classes = ["cobot:RobotType"]
(58)     }
(59)   }
(60) }
(61) perspective SoftwareEngineer widget infobox {
(62)   section recommendations {
(63)     config {
(64)       classes = ["cobot:HandlingFunction",
(65)                 "star:ArchitecturalElement"]
(66)     }
(67)   }
(68) }

```

Search interface is provided by a rendering engine that interprets these templates. The interface is intended to be similar to major search engines on the Web in order to lower the learning curve. The main entry point for starting an exploration is a keyword search as depicted in Figure 2. The result set of a keyword search is listed vertically, and an info box is shown for the first entry of the result list. Nonetheless, a user can switch to a different search paradigm, e.g. a tree view.

4. System Architecture

Many KG-based exploratory search systems report to only rely on a SPARQL interface to the target knowledge graph, which has the big advantage that these systems can directly be applied to virtually every RDF-based knowledge graph. An integral part of our system is the computation of centrality metrics (e.g. PageRank) for the ordering of resources according to their "importance" and similarity metrics (e.g. LDSD [14]) for recommendations. A reasonable responsiveness is however an important non-functional requirement towards search interfaces, and we claim that this is hard to achieve without precomputing metrics or sophisticated indexing techniques. Thus, we introduce a *middleware* application which sits on top of the storage solution for the target knowledge graph. The conceptual overview of our whole exploratory search system is depicted in Fig. 3.

The adaptive search interface presented in Section 3 is provided by our **web application**, which is designed to be a thin single-page application. ReactJS⁴ is used to implement the UI

⁴JavaScript library for building user interfaces - <https://reactjs.org>

The screenshot shows a search interface for 'universal robot'. The results are categorized by 'ROBOT TYPE'. The main results list includes:

- UNIVERSAL ROBOTS UR10**: While the largest robot arm in the UR family and the one with the most muscle power, the UR10 does not compromise on precision. The collaborative robot arm will automate heavier-weight process tasks with payload requirements of up to 10 kg.
- UNIVERSAL ROBOTS UR3**: The UR3 collaborative robot is a smaller collaborative table-top robot, perfect for light assembly tasks and automated workbench scenarios. The compact table-top cobot weighs only 24.3 lbs (11 kg), but has a payload of 6.6 lbs (3 kg), ±360-degree rotation ...
- UNIVERSAL ROBOTS UR5**: The slightly bigger UR5 is ideal for automating low-weight processing tasks like picking, placing and testing. The medium-sized robot arm is easy to program, fast to set up and, just like the other collaborative members of the UR family, offers one of the ...
- Neobotix MP-400**: The mobile robot MP-400 was designed for daily use in industrial applications. Its construction is based on the best elements of our well proven robots but also includes many innovative ideas. The MP-400's primary task is the flexible material transport ...
- KUKA LBR IIWA 7 R800**: The LBR iiwa is the world's first series-produced sensitive, and therefore HRC-compatible, robot. LBR stands for "Leichtbauroboter" (German for lightweight robot), iiwa for "intelligent industrial work assistant". This signals the beginning of a new era in ...

The detailed view for the UR10 includes a table of properties:

Property Table (Provenance)	
	Pilot factory interviews Cobot List from Robotics World
Handling Function	Move, Approach, Depart, Retract
Skill	Wait, Retrieve, Store
Reach	1300 mm
Handling Payload	10 kg
Degrees Of Freedom	6

Related Robot Types include:

- UNIVERSAL ROBOTS UR5**: The slightly bigger UR5 is ideal for automating low-weight proce...
- UNIVERSAL ROBOTS UR3**: The UR3 collaborative robot is a smaller collaborative table-top ...
- FANUC CR7IA**: I'm small, flexible and can work right by your side. I take care ...
- KUKA LBR IIWA 7 R800**: The LBR iiwa is the world's first series-produced sensitive, and ...

Figure 2: Entry point for exploration (keyword search).

components, and the state is managed with Redux⁵. In order to be able to render the UI, the *web application* needs to load the UI configuration (see Section 3). The required data for rendering the UI components is then fetched by assembling corresponding *exploration flows*, and sending them to the *middleware*. The *web application* is only responsible for the correct rendering and isn't handling any RDF data itself.

⁵State container for JavaScript applications - <https://redux.js.org>

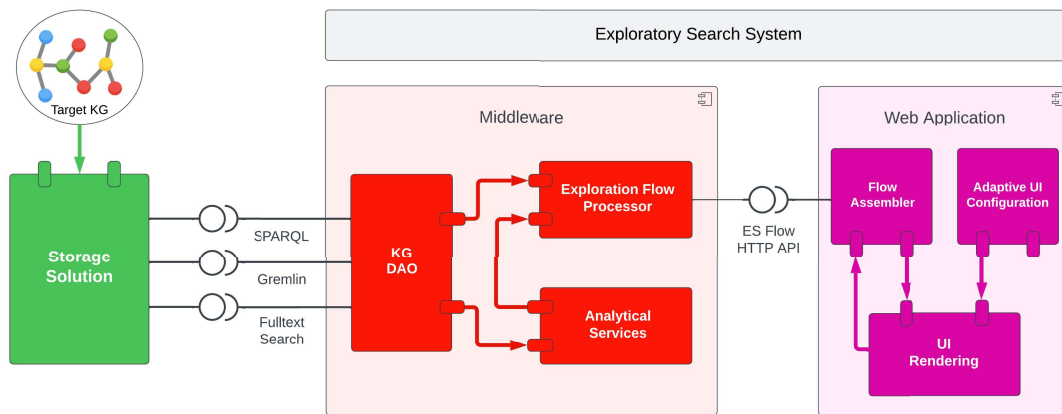


Figure 3: Conceptual overview of the exploratory search system.

An *exploration flow* is a sequence of steps which all execute a single operation. Listing 3 shows a flow that computes the weighted sum of LDSD [14] and a general peer pressure metric for all pairs between a particular gripper and all other resources in the knowledge graph. Then, the collection of pairs is ordered by this weighted sum, and only the top 10 pairs are eventually returned. The goal of an *exploration flow* is to abstract the more complex part of the Semantic Web technologies, and only expose the basic concepts of RDF to the *web application*.

Listing 3: Top 10 recommendations for a certain gripper (Python).

```
(1) f = Single(resource='ex:gripper04') \
(2)   >> PairWith(flow=All()) >> (LSDS() | PeerPressure()) \
(3)   >> WeightedSum('sum', {Sim.lsd: -1, Sim.peerpressure: 1}) \
(4)   >> OrderBy('sum', strategy=Order.DESC) >> Limit(n=10)
(5) resp = FlowAPI("http://localhost:8080").execute(f)
```

The *middleware* is a Spring Boot application written in Java and it has two main tasks. Firstly, it provides an interpreter for *exploration flows*, which parses flows and executes their steps one by one in the correct order. Secondly, it orchestrates the computation of analytical services such as centrality and similarity metrics among others. However, we can't solely rely on SPARQL for these tasks, because:

- (a) Graph traversal is limited in the specification of SPARQL 1.1 [15]. While property paths add the ability to check whether a route of arbitrary length exists between two nodes in the knowledge graph, a wider range of path operations aren't supported such as computing the shortest distance between two nodes. Query languages for property graphs usually don't have this limitation, and RDF-based knowledge graphs can easily be represented in property graphs. However, at the moment of writing no query language is accepted as a standard for property graphs. GQL is one of the emerging attempts to establish such a standard [16]. Cypher and Gremlin are nevertheless two prominent choices.
- (b) Full-text searches are a feature of many triplestores, but it is not included in the speci-

TASK A	TASK B
<p>Imagine that you are a member of a team which is working on a manufacturing project with collaborative robots and you have access to the equipment located at the pilot factory in Aspern. In this project, you have to move a 4 kg heavy and cubic object with a length of 20cm from a conveyor belt to a manufacturing cell that is 60cm away. Goal You have been asked to design a hardware setup for this project with equipment that is available at the pilot factory. Your team would be glad about a brief presentation of your findings.</p>	<p>Imagine that you are a member of a team which is working on a collaborative robot picking up a small transistor from a storage box and placing it on a certain position on a circuit board. The robot works in proximity of human workers and it might have to interrupt it's task, due to them coming too close. However, we don't want the robot to remain in this state, and proceed with the task as soon as possible. Goal: You have been asked to explore design patterns that could be lend from software engineering for this kind of error handling and eventually present promising design patterns to your team.</p>

Table 2
Exploratory search tasks for evaluating the system.

cation of SPARQL 1.1 [15]. The SPARQL syntax for issuing a full-text search is vendor-specific, and the configurability of search indices varies from vendor to vendor. Alternatively, one might want to use an external solution for creating full-text indices.

Due to these limitations of SPARQL, the *middleware* expects three interfaces to the stored knowledge graph: (1) SPARQL 1.1, (2) Gremlin as query language for property graphs, and (3) simple full-text search API. The *middleware* includes plugins, which implement these three interfaces for a number of popular triplestores (Blazegraph, GraphDB, Stardog and Virtuoso). And given that barely any triplestore supports the Gremlin query language, the *middleware* provides also a mechanism to clone the knowledge graph over the SPARQL interface into an embedded JanusGraph⁶ instance.

5. Preliminary Evaluation

Our exploratory search system⁷ was evaluated on a small scale with five participants for the pilot factory use case (UC2), which is why we want to set a heavy focus on qualitative analysis. All five participants were non-experts in smart manufacturing and robotics, but they were knowledgeable in software engineering. Participants took in our experiments the role of an information seeker and explored the domain of collaborative robotics. The evaluation aimed to assess the ability of our system to allow a user to interactively investigate, learn and make sense of a topic from this manufacturing domain.

The two tasks listed in Table 2 were designed to elicit exploratory search behaviour from participants. The experiment started with a brief survey to assess participant's a-priori knowledge about robotics and software engineering. Then, they were given both exploratory search tasks one-by-one, and had 20 minutes time with our search system for each task. Afterwards, their gained knowledge was assessed by a new survey. Moreover, we asked the participants

⁶Open source distributed graph database - <https://janusgraph.org/>

⁷Online deployment - <https://data.ifs.tuwien.ac.at/cobot>

Source code for local deployment - <https://gitlab.tuwien.ac.at/kevin.haller/cobot-playground>

to rate the "usefulness" of single features for completing their tasks, and conducted informal interviews to gather feedback. All the sessions were recorded for later video analysis.

Overall, all participants gained some additional knowledge and were able to present a reasonable solution for both tasks to their fictional team. Regarding the exploratory search features, the participants thought that info boxes were the most useful, which isn't surprising given that it plays a prominent role in our search system. On the other hand, the tree view for browsing deep hierarchies wasn't rated highly. A video analysis of the recorded sessions showed that the participants were overburdened by the current design of the tree view and the click rate was low. Moreover, it showed that they preferred to use browser tabs as well as back and forth navigation over our in-built memorization feature.

6. Related Work

Little work has been published for exploratory search in manufacturing and production. Metaphactory describes in [17] their platform for knowledge graph management and briefly outlines use cases in the engineering and manufacturing domain at Siemens. While exploratory search is not discussed, it touches on faceted navigation, query building assistance as well as customizable search experience and result visualization. Sabou et al. [18] show an approach to transform a central repository of architectural knowledge at Siemens into a knowledge graph, and furthermore, present a search system with an user interface similarly to major search engines on the Web. The search interface includes info boxes with faceted navigation and a KG-based recommendation engine. Yet, these systems don't address the manufacturing features discussed in this paper.

Much work has been published about utilizing the semantics of knowledge graphs to render configurable presentations of resources. Fresnel [19] is an ontology that provides a vocabulary to annotate resources, classes and properties with presentation knowledge. A browser understanding this vocabulary can then visualize these entities accordingly. Rutledge et al. [20] propose an extension of this Fresnel ontology that allows to define presentation knowledge for provenance information as well. LESS [21] introduces on the other side a new template language based on Smarty⁸ to define the presentation of resources. A LESS processor is then taking the designed template and applies it either to a specified RDF document or the result set of a specified SPARQL query. Uduvudu [22] utilizes templates in a similar manner, but additionally proposes algorithms for the automatic selection of templates based on the input data. These solutions focus however exclusively on the visualization of entities in a knowledge graph and thus, not all aspects of configuring an adaptive search interface are covered.

Linked Data Reactor (LD-R) [11] is a faceted explorer with a configurable and adaptive user interface. It proposes the concept of *scopes* and *configurations* as outlined in more detail in Section 3. LD-R enables the configuration of facets following these newly introduced concepts. Moreover, it implements a mechanism to change its UI components based on the persona of a user. Nonetheless, LD-R exclusively focuses on faceted exploration. Furthermore, we argue that simple perspectives are a more accessible mechanism for stakeholders in manufacturing than user personas in LD-R. KG-Explorer [23] proposes similarly a configurable and adaptive

⁸PHP template engine - <https://www.smarty.net>

search interface that mainly focuses on faceted exploration, but not exclusively. Some aspects of the overall search interface can be adapted in a configuration file as well as templates can be created for editorial pages about entities. However, the search behaviour and interface elements can't be customized for different user perspectives. Moreover, these systems don't address the other two manufacturing features discussed in this paper.

7. Conclusion and Future Work

KG-based exploratory search systems share a common set of features. In this paper, we identified and reported three particular features from interviews that are additionally relevant for enabling exploratory search on manufacturing knowledge graphs. The adaptability of the search system to *multiple (engineering) perspectives* is one of these features. Moreover, the *visibility of provenance* information supports stakeholders in manufacturing in their investigative work. Finally, *browsing deep hierarchical structures* of containment relationships is a frequent task on manufacturing knowledge graphs.

We presented an adaptive and configurable exploratory search system, which implements solutions to these features among others. A new scope component representing *perspectives* was introduced to allow the *multiperspectival* configuration of visualizations as well as the underlying search system. The *visibility of provenance* details was addressed by passing information about the named graphs for each statement to the corresponding UI components and hence, enabling the implementation of UI components that visualize this knowledge. The requirement of *browsing deep hierarchies* was addressed with the implementation of a simple tree view (as known from Protegé for example) in combination with info boxes.

A small-scale evaluation of our system showed the usefulness of the implemented features for participants for completing their search tasks, with the exception for the tree view and memorization feature. Participants were overburdened by our current design of the tree view. Moreover, our implemented memorization feature was mainly ignored by participants in favor of maintaining multiple browser tabs. Overall, the participants were able to present a reasonable solution for their search tasks.

Hence, as future work we want to improve on the tree view for *browsing deep hierarchies*. Furthermore, we want to make our rendering engine compatible with RDF-star [13], due to its increasing popularity. Eventually, we aim to evaluate the improved system on the OntoTrans use case (UC1) with a larger number of participants.

Acknowledgments

This work is supported by European Union's Horizon 2020 research project OntoTrans under Grant Agreement No 862136.

References

- [1] S. Haag, R. Anderl, Digital twin – Proof of concept, Manufacturing Letters 15 (2018) 64–66. doi:10.1016/j.mfglet.2018.02.006, Industry 4.0 and Smart Manufacturing.

- [2] T. Bauernhansl, M. T. Hompel, B. Vogel-Heuser, *Industrie 4.0 in Produktion, Automatisierung und Logistik: Anwendung-Technologien-Migration*, Springer, 2014. doi:10.1007/978-3-658-04682-8.
- [3] M. Galkin, S. Auer, M.-E. Vidal, S. Scerri, *Enterprise Knowledge Graphs: A Semantic Approach for Knowledge Management in the Next Generation of Enterprise Information Systems*, in: ICEIS, 2017, pp. 88–98. doi:10.5220/0006325200880098.
- [4] X. Li, M. Lyu, Z. Wang, C.-H. Chen, P. Zheng, *Exploiting knowledge graphs in industrial products and services: A survey of key aspects, challenges, and future perspectives*, *Computers in Industry* 129 (2021) 103449. doi:10.1016/j.compind.2021.103449.
- [5] J. Strötgen, T.-K. Tran, A. Friedrich, D. Milchevski, F. Tomazic, A. Maruszczyk, H. Adel, D. Stepanova, F. Hildebrand, E. Kharlamov, *Towards the Bosch Materials Science Knowledge Base*, in: ISWC 2019 Satellite Tracks, volume 2456, CEUR Workshop Proceedings, 2019, pp. 323–324.
- [6] T. Hubauer, S. Lamparter, P. Haase, D. M. Herzig, *Use Cases of the Industrial Knowledge Graph at Siemens*, in: ISWC 2018 P&D/Industry/BlueSky Tracks, volume 2180, CEUR Workshop Proceedings, 2018.
- [7] C. Palmer, Z. Usman, O. C. Junior, A. Malucelli, R. I. Young, *Interoperable manufacturing knowledge systems*, *International Journal of Production Research* 56 (2018) 2733–2752. doi:10.1080/00207543.2017.1391416.
- [8] J. M. Rožanec, J. Lu, J. Rupnik, M. Škrjanc, D. Mladenčić, B. Fortuna, X. Zheng, D. Kiritsis, *Actionable cognitive twins for decision making in manufacturing*, *International Journal of Production Research* 60 (2022) 452–478. doi:10.1080/00207543.2021.2002967.
- [9] R. W. White, R. A. Roth, *Exploratory Search: Beyond the Query-Response Paradigm*, *Synthesis Lectures on Information Concepts, Retrieval, and Services* 1 (2009). doi:10.2200/s00174ed1v01y200901icr003.
- [10] N. Marie, F. Gandon, *Survey of Linked Data Based Exploration Systems*, in: 3rd International Workshop on Intelligent Exploration of Semantic Data, volume 1279, CEUR Workshop Proceedings, 2015.
- [11] A. Khalili, A. Loizou, F. van Harmelen, *Adaptive Linked Data-Driven Web Components: Building Flexible and Reusable Semantic Web Interfaces*, in: European Semantic Web Conference, Springer, 2016, pp. 677–692. doi:10.1007/978-3-319-34129-3_41.
- [12] V. Nguyen, O. Bodenreider, A. Sheth, *Don't like RDF reification? Making statements about statements using singleton property*, in: International Conference on World Wide Web (WWW), Association for Computing Machinery, 2014, p. 759–770. doi:10.1145/2566486.2567973.
- [13] O. Hartig, P.-A. Champin, G. Kellog, A. Seaborne, *RDF-star and SPARQL-star*, W3C Community Group Report (2021). URL: <https://w3c.github.io/rdf-star/cg-spec>.
- [14] A. Passant, *Measuring semantic distance on linking data and using it for resources recommendations*, in: AAAI Spring Symposium Series, 2010.
- [15] S. Harris, A. Seaborne, E. Prud'hommeaux, *SPARQL 1.1 query language*, W3C recommendation (2013). URL: <https://www.w3.org/TR/sparql11-query/>.
- [16] ISO/IEC JTC 1/SC 32, *Information Technology — Database Languages — GQL*, Standard ISO/IEC CD 39075, International Organization for Standardization, 2019.
- [17] P. Haase, D. M. Herzig, A. Kozlov, A. Nikolov, J. Trame, *Metaphactory: A platform for*

- knowledge graph management, *Semantic Web 10* (2019). doi:10.3233/SW-190360.
- [18] M. Sabou, F. J. Ekaputra, T. Ionescu, J. Musil, D. Schall, K. Haller, A. Friedl, S. Biffl, Exploring Enterprise Knowledge Graphs: A Use Case in Software Engineering, in: *European Semantic Web Conference, 2018*, pp. 560–575. doi:10.1007/978-3-319-93417-4_36.
- [19] E. Pietriga, C. Bizer, D. Karger, R. Lee, Fresnel: A Browser-Independent Presentation Vocabulary for RDF, in: *International Semantic Web Conference, Springer, 2006*, pp. 158–171. doi:10.1007/11926078_12.
- [20] L. W. Rutledge, P. Mellema, T. Pietersma, S. M. M. Joosten, Displaying triple provenance with extensions to the Fresnel vocabulary for semantic browsers, in: *Workshop on the Visualization and Interaction for Ontologies and Linked Data, volume 3023, CEUR Workshop Proceedings, 2021*, pp. 103–114.
- [21] Raphael, D. S. A. Sören, Doehring, LESS - Template-Based Syndication and Presentation of Linked Data, in: *Extended Semantic Web Conference, Springer, 2010*, pp. 211–224. doi:10.1007/978-3-642-13489-0_15.
- [22] M. Luggen, A. Gschwend, B. Anrig, P. Cudré-Mauroux, Uduvudu: a graph-aware and adaptive UI engine for linked data, in: *Workshop on Linked Data on the Web (LDOW), volume 1409, CEUR Workshop Proceedings, 2015*.
- [23] T. Ehrhart, P. Lisena, R. Troncy, KG Explorer: a Customisable Exploration Tool for Knowledge Graphs, in: *Workshop on the Visualization and Interaction for Ontologies and Linked Data, volume 3023, CEUR Workshop Proceedings, 2021*, pp. 63–75.