Available online at www.sciencedirect.com

SciVerse ScienceDirect

Metamorphic Data Sources: a User-centric Paradigm to Consume Linked Data in Interactive Workspaces

Giuseppe Desolda[a], Maristella Matera[b], Rosa Lanzilotti[a]

[a]*Dipartimento di Informatica, Universitàdegli Studi di Bari Aldo Moro, Via Orabona, 4, 70125, Bari, Italy*
[b] *Dipartimento di Elettronica, Informazione e Bioingegneria, Politecnico di Milano, Piazza Leonardo da Vinci, 32, 20134, Milano*

Abstract

In the last years, the debate about the success or failure of Linked Data (LD) has been growing. Despite the ever-increasing number of available ontologies and LD datasets, there is still a limited number of applications to let people benefit from using this huge amount of data. Some evident problems relate to the limited opportunities offered to the end users, i.e., people without skills in computer programming, to access, navigate and visualize LD. Tools supporting such tasks typically do not consider the end users' needs; even when they provide abstraction mechanisms to avoid programming, they do not properly hide the complexity of getting oriented into the plethora of available resources. Thus, they end up to be inadequate to real daily scenarios. In this paper, we propose an approach that enables end users to create visually entry points, which we call *Metamorphic Data-Sources* (MDSs), to query and visualize the LD without requiring any prior knowledge of semantic Web or visualization technologies. Through the MDS visual paradigm, end users can tailor ad-hoc data sources to retrieve information on topics they are interested in. The MDS creation process is also driven by a quality model that further helps users select LD elements potentially free of data quality problems. The paper also reports on the results of a user study that we conducted to assess the validity of the MDS paradigm with respect to the user needs.

Keywords: Linked Data visual exploration; Linked Data browsers; Web Mashups; User studies

1. Introduction

Recent advancements in Web technologies have provided a strong potential for gathering data from distributed data sources and applications. We are increasingly dealing with a huge amount of information generated by resources accessible through APIs. However, these resources expose data in different formats referring to diverse data models. To alleviate such heterogeneity, the research community has thus been working on alternative paradigms for data publishing and access. One promising paradigm is the *Linked Data* (LD), which was proposed by Tim Berners-Lee in 2006 as a set of best practices for publishing and connecting structured data on the Web (Berners-Lee, 2006). The most important example of adoption of the LD principles has been the Linking Open Data (LOD) project (Bizer et al., 2009), a powerful blend of LD and Open Data, since it is both linked and available open-source. Before the LD paradigm, the Web was essentially a collection of HTML pages connected by hyperlinks. The goal of LD is to make Web data available in a format that is readable by people and by automatic

agents as well. The main idea is that HTTP URIs, which are used to access Web pages in the traditional vision of WWW, are also exploited to identify real or abstract entities (e.g., a person, a city, an organization). Data publication is grounded on the Resource Description Framework (RDF), a graph data model designed for publishing structured data on the Web, which uses hyperlinks to connect not only Web documents, as in the traditional Web, but every data entity (Hogan et al., 2012). According to the RDF syntax, each entity is coded as a set of triplets in the form *t* = <*subject, predicate, object*>. The *subject* represents the entity that always consists of a URI. The *object* can be a literal or another entity identified by a URI. The *predicate* is represented by a URI; it is defined in ontologies used to provide information about a domain and indicates a semantic relation between the subject and the object.

Today thousands of open-source datasets can be accessed and navigated through the LOD paradigm[*]. The opportunity to access and navigate this wide amount of open, interlinked data has fostered the proliferation of semantic Web browsers that permit to retrieve LOD entities, inspect their properties and navigate through their hyperlinks[†], also supporting complex queries that could not be formulated in traditional search engines.

To tech-savvy users, LD consumption is easy: they have the necessary knowledge of the languages (e.g., SPARQL), as well as the capability, based on their accumulated experience, to retrieve entities and connections between pieces of information. However, there are several users for whom there is a gap between their need to access LD and the skills that this requires, even when they try to use semantic Web browsers (Dadzie et al., 2011). These are the *regular Web users*, who browse Web sites to satisfy their situational information needs without any knowledge of the intrinsic structure of the Web, as well as *domain experts*, who lack technical expertise but have a good understanding of the data available in their domain, and need sophisticated, domain-specific analysis tools to obtain insights for decision making from very large amounts of heterogeneous data (Atzori et al., 2016; Dadzie et al., 2011). Our research focuses especially on domain experts and their frequent need to integrate heterogeneous data from distributed resources into unified, interactive workspaces. This indeed raises important challenges for the definition of alternative paradigms to support the exploration of the plethora of Web resources and the extraction of actionable knowledge.

1.1. Motivations

Despite the encouraging premises and the initial efforts invested in developing semantic Web browsers, there are still no evident results for the adoption of LD-based systems by the mass (Peña et al., 2014). In the last years, this lack has been fueling a debate about the success or failure of LD: at the "AI, Graph Databases and Linked Data Conference" in 2017, different leaders in the LD sector argued that one reason for the failure is "a perceived lack of intuitive and accessible LD infrastructures". Indeed, given the huge number of datasets, entities, ontologies, classes, and attributes, the LD exposes an intrinsic complexity that is even more accentuated by the difficulty for the end users to master ad-hoc query languages.

The current LD browsers mainly address the needs of computer scientists, researchers or members of technical communities, who are acquainted with the RDF data model and are able to extract data using a formal query syntax (Atzori et al., 2016; Dadzie et al., 2011). In other words, such tools do not provide adequate abstractions on top of query languages, nor visual mechanisms that can hide the LD complexity. In order to empower lay users to take advantage of LD data, this paper proposes a visual approach to assist them in creating their own access points on top of LD, so that they can easily retrieve data on topics of interest and represent them according to visual templates that they can also visually configure. The process for creating an access point consists of two main activities, which are facilitated by a visual paradigm: 1) the selection of an ontology class related to a specific topic and 2) the projection of a subset of pertinent class attributes that is operated by mapping attributes to visual elements of a visualization template. The result is a tailored data source coupled with a configured visualization template, which can then be exploited any time by users who need to retrieve and visualize entities related to the chosen topic. We

[*] https://lod-cloud.net/ reports that LOD contains 1,234 datasets with 16,136 links (as of June 2018)

[†] https://www.w3.org/2001/sw/wiki/Category:Semantic_Web_Browser

call this mechanism *Metamorphic*[‡] *Data Source* (MDS), since end users can define access points on LD, whose structure can be adapted to their specific needs. MDSs can be fruitfully used within visual environments for the integration of heterogeneous resources, for example, the different mashup platforms that have been proposed in the last decade (Daniel and Matera, 2014; Hoang et al., 2010; Paredes‐Valverde et al., 2015; Yu et al., 2008). The notion of MDS has been indeed conceived as part of a larger project focusing on the design of mashup platforms that offer visual notations to enable the integration by end users of heterogeneous data sources and Web APIs (Ardito et al., 2014b; Desolda et al., 2016, 2017b).

It is worth remarking that MDSs should not be considered as an alternative to search engines that regular Web users typically adopt for daily and generic queries. Rather, MDSs would better support domain experts or community of users in satisfying domain-specific needs that imply performing queries repeatedly on specific topics. Let us think, for example, to physicians, who often search for illness and drugs on digital handbooks before making prescriptions; or attorneys, who search for laws in civil, criminal or similar codes to prepare their lawsuits; or chemists, which look for elements or formulas to prepare chemical compounds. In these and other similar situations, domain experts daily perform dozens or hundreds of queries on their domain topics. Thus, MDSs built on domain-specific and/or general-purpose datasets can be a valid alternative to traditional search engines to speed up and make more efficient their queries, without getting lost in hyperspace (Otter and Johnson, 2000) and reducing cognitive overload (Kirsh, 2000).

In usage scenarios like the one described above, the quality of data can strongly influence the user productivity. LD, however, are typically affected by errors, inconsistencies, missing or outdated values and other issues that can limit their usage. The MDS creation uses an LD quality model (Cappiello et al., 2016) to assist the selection of elements (ontology classes and their properties) that can reduce the occurrence of quality problems in the final integrated data set.

1.2. Contribution

The work illustrated in this paper has been developed in the context of a larger research project that aims to promote End-User Development as a solution for letting non-technical users to make sense of the huge availability of Web APIs and online resources (Ardito et al., 2014b; Desolda et al., 2016, 2017b). The leading question of this research relates to *the support that the proposed visual paradigm can offer to users in creating and querying MDSs*. With respect to our previous results, the work described in this paper proposes the following new contributions:

- *A visual paradigm for LD exploration*. We introduce techniques for end users to make sense of LD by means of *i)* a visual paradigm to build and query MDS and *ii)* User Interfaces (UIs) for visualizing the retrieved data. In relation to data visualization, our approach is still in an initial stage. Our aim is indeed to investigate primarily the advantages of providing easy mechanisms to explore LD and build visualizations for selected topics. Extensions and improvements in relation to the quality of the provided visualizations are still possible. In this paper, we already show how the engine for LD querying can be connected to an external API that is able to build adequate and adaptive visualizations. Other, more sophisticated services or software modules can be easily plugged in.
- *LD consumption within interactive information workspaces*. We show how LD can be consumed in a context where the end user is also enabled to access other APIs and might be interested in the integration of all such data within unified interactive workspaces where visualization and data manipulation functions are also offered.
- *A technique for tailoring access to LD*. We propose a technique that helps users identify and filter data that are pertinent in respect to their situational needs and to the current status of the information workspace under composition (i.e., the other data already retrieved and integrated into their workspace). With respect to other systems, we do not focus on providing end users with RDF graphs representing all the available information. Some studies indeed show that this kind of data representation is difficult to understand for non-technical users (Berners-Lee et al., 2008). We rather provide algorithms that help users select LD entities that are deemed useful

[‡] The term metamorphic is due to the analogy with metamorphic rocks, which change their shape under the influence of factors like heat or pressure, in a process called metamorphism - which means "change in form".

for them during a given exploration session, also taking into account the data already integrated into the workspace under composition.

- *A quality-driven process for MDS creation.* Given the quality problems LD generally suffer, we show how the assessment of data quality measures on LD classes and attributes can lead to an incremental process for LD exploration where user choices are driven by indications on the quality of data that can be accessed through the MDS under creation.
- *Lesson learned*: Based on the results of an experimental study involving a sample of users, we present some lessons learned ranging from methodological aspects to more technical features that can drive the design of visual environments for LD consumption.

1.3. Paper Organization

This article is organized as follows. Section 2 discusses the rationale and background behind our research, with reference to the related literature. It summarizes the main characteristics of the Semantic Web browsers so far proposed for LD exploration (Section 2.1); it motivates the need for adopting a quality model for the MDS creation (Section 2.2), and also explains how the mashup paradigm for the visual integration of heterogeneous data sources can be exploited for LD consumption within interactive workspaces. Section 3 illustrates in detail the notion of MDSs. Through a running example (Section 3.1) it describes the different steps for MDS creation; then it introduces the algorithms and the mechanisms that support the creation steps and the MDS execution within interactive workspaces (Sections 3.2). Section 4 shortly illustrates the software architecture of the platform to clarify how different software modules interoperate to enable the functions described in the previous sections and support flexibility in the MDS creation and execution. Section 5 then reports on an experimental study that we performed to compare two different versions of the MDS paradigm (with and without quality measures) and the Google search API. Section 6 finally concludes the paper by reporting on some lesson learned from the study and outlining our future work.

2. Rationale and Background

The LD paradigm ensures the creation of a layer where access to the available resources is facilitated by a standard way of storing and sharing data. However, data access by users who do not possess technical skills is still an issue (Berners-Lee et al., 2008; Herman et al., 2000; Khalili et al., 2018; Namoun et al., 2010; Zang and Rosson, 2008), as it generally requires the formulation of SPARQL queries (Pérez et al., 2009). In the last years, some visual mechanisms have been proposed to facilitate the use of query languages. However, current techniques are still inadequate with respect to lay-user skills. Alternative ways of exploring, integrating and sharing information are needed to improve the usability of access mechanisms so that the huge availability of LD resources can offer advantages to the mass too. Such mechanisms should also consider the quality of LD. Indeed, despite the numerous advantages in terms of data availability, exploring LD can become difficult, as they are not free from quality problems, especially incompleteness (Zaveri et al., 2016).

In the following, we illustrate the most relevant paradigms so far proposed to facilitate LD access – the so-called *visual browsers*. We then discuss the quality of LD and how adequate models can help users select LD resources and LD elements even when they lack the needed technical experience. Finally, we also discuss the benefits that LD can offer when end users are enabled to explore and integrate them within interactive information workspaces.

2.1. Visual Browsers for LD Exploration

In this section, we describe the most popular visual browsers for LD exploration. Table 1 summarizes the main features of the selected systems. Most comprehensive surveys can be found in (Alahmari et al., 2012; Bikakis and Sellis, 2016; Dadzie and Rowe, 2011; Marie and Gandon, 2014). As it can be noted, many of such tools have been dismissed. We discuss them anyway with the aim is to survey pros and cons of the interactive visual paradigms so far proposed.

One of the first LD browsers proposed in literature was *Disco*, a search engine that, given a resource URI, retrieves its data from LD and renders them in a tabular fashion, i.e., not as a simple list of attributes but through a proper HTML page layout (Bizer and Gauß, 2018). If the results include hyperlinks, the user can navigate to the linked resources, as the browser is able to dynamically retrieve the corresponding data by dereferencing URIs. Similar to Disco, also *Zitgist* displays the RDF data reachable through a URI given in input by the user (Bergman and Giasson, 2008). Depending on the retrieved information, the RDF browser then shapes the layout of the result page to optimize the user browsing experience.

The previous browsers assume that users know in advance the URIs to be accessed, which might be a non-trivial activity for those who have a low familiarity with linked datasets (Dadzie et al., 2011). To solve this problem, in addition to URI search, *LD Cloud Cache* provides keyword-based search, as well as a language to specify structured queries for LD data collections hosted by a Virtuoso engine (Virtuoso, 2018). Keyword-based search is also supported by *Sindice* (Tummarello et al., 2007), which then lists the retrieved resources in a format similar to Google search results, and also sorts them according to a ranking algorithm. *Swoogle* also allows users to enter plain text to retrieve ranked, hyperlinked resources from both datasets and ontologies (Ding et al., 2004). It then offers an "advanced" search interface to specify some additional conditions as an SQL query. Similarly, *VisiNav* offers an interactive paradigm designed to easily search and navigate large amounts of Web data (Harth, 2010). In addition to the keyword search and result ranking, it enables three kinds of search tasks: *object focus*, *path traversal*, and *facet specification*. Through these atomic operations, users incrementally assemble complex queries that yield sets of objects as result[§]. More complex queries can then be formulated in *Falcons* (Cheng and Qu, 2009) that, with respect to the previous browsers, is able to process complex queries that go beyond the object names. For example, a query can describe relations between objects (e.g., "AVI 2018 Conference" AND "demo chair"). In addition, when presenting the objects that match the query keywords, Falcons recommends further related objects the user might be interested in.

Table 1. List of the most important tools for LD visual exploration and classification dimensions

System	Client Type	Access	Starting point	Available	Visualization
DBpedia mobile	Mobile App	Read	GPS position	Yes	Geo Map
Disco	Browser	Read	URI	Dismissed	Tabular
Falcons	Browser	Read	keywords	Dismissed	List
FenFire	Browser	Read	URI	Dismissed	Graph
FERASAT	Browser	Read/Write	Keywords	Yes	UI Components
graphVizdb	Browser	Read	Keywords	Yes	Graph
Horus	Desktop/App	Read	keywords	Dismissed	List/Graph
iLD	Mobile	Read	URI	Dismissed	Tabular
IsaViz	Desktop/App	Read/Write	keywords/URI	Yes	Graph
LENA browser	Desktop/App	Read	keywords	Dismissed	Tabular
LD Cloud Cache	Browser Plugin	Read	keywords/URI	Yes	List
Lod Live	Browser	Read	keywords/URI	Yes	Graph
LODWheel	Browser	Read	keywords	Yes	Graph
Marbles	Browser	Read	URI	Dismissed	Tabular
ObjectViewer	Browser	Read/Write	URI	Dismissed	Graph
OpenLink DataExplorer	Browser	Read	keywords/URI	Yes	List/Tabular/Graph
Piggy Bank	Browser	Read	keywords	Dismissed	Tabular
RelFinder	Browser	Read	keywords	Yes	Graph
Sig.ma	Browser	Read	keywords	Dismissed	Tabular
Sindice	Browser	Read	keywords	Yes	Tabular
SWiPE	Browser	Read	Search by example	Dismissed	Wikipedia page
Swoogle	Browser	Read	keywords	Yes	List
Tabulator	Browser Plugin	Read	URI/SPARQL	Dismissed	Tabular
VisiNav	Browser	Read	keywords	Dismissed	List
Watson	Browser	Read	keywords	Dismissed	Tabular/Graph
Zitgist	Desktop/App	Read	URI	Dismissed	Tabular

[§] https://youtu.be/7FaSDpMIXXo

Tabulator is an open-source Firefox plugin (Berners-Lee et al., 2008) originally written as a linked-data browser (Berners-Lee et al., 2006). It provides visual mechanisms to navigate the LD, without requiring any domain-specific programming skill. It supports two kinds of operations: *exploration*, to see what information is available, and *querying*, to gather similar subgraph patterns. These operations are performed on tables similar to spreadsheets. Exploration is supported by presenting results in a table of predicate/object pairs. If the object is a URI, the users may recursively open a nested view showing the property objects in turn. Querying is activated by starting a search for a subset of fields retrieved in the exploration mode; the LD subgraphs matching the given fields are still displayed in a tabular format.

While the previous browsers present the query results in a tabular format, other approaches exploit graph-based visualizations that strongly reflect the graph data model characterizing RDF. *LODWheel* (Stuhr et al., 2010) is a prototype for visualizing RDF data through graphs and charts; its development implied an extensive evaluation of open-source JavaScript libraries for visualizing data, to identify the ones with the highest score based on a number of evaluation criteria adopted to rank the offered visualizations. In (Heim et al., 2010), the authors focus on the interactive discovery of relationships between selected elements via the Semantic Web. In line with our research goals, they recognize the importance of providing users with interactive tools to efficiently get an overview on relationships found in the considered datasets, to interactively explore them and to easily spot relationships that are relevant in a certain situation. They, therefore, propose *RelFinder*, a tool that supports: i) an overview that aggregates relationships according to different dimensions (e.g., statistical, topological, or semantic ones), then ii) an exploration of the found relationships by means of interactive features (e.g., dynamic filtering) and visual clues (e.g., highlighting). *LodLive* (Camarda et al., 20122012) then proposes a nice, graph-based visual metaphor with big circles representing a class, also identified by a color and a label, and small circles placed around the big ones to represent object properties, such as direct and inverse relations and *owl:sameAs* properties. The selection of small circles progressively opens new big ones, thus facilitating the graph navigation. *Fenfire* (Hastrup et al., 2009). makes the graph visualizations scalable in the number of nodes by allowing the users to focus on one node at a time, which is displayed together with its adjacent nodes. In (Dadzie et al., 2011) the authors discuss the usability of mechanisms for LD consumption. They propose a nice mechanism to filter out irrelevant elements from a graph-based representation of LD datasets. *ObjectViewer* (Lerner and Self, 2018) displays RDF data through an interactive graph, with labels acting as links to other graphs. The graph display works well for smaller data sets, but quickly becomes very large to scroll around when the RDF graph becomes more complex. To solve this problem and facilitate graph exploration, *graphVizdb* (Bikakis et al., 2016) proposes an interactive visualization of very large graphs, which can be easily applied to LOD exploration. The platform allows the user to interact with a visualized graph in a way that is very similar to the exploration of maps at multiple levels. The user navigates on the graph by moving a viewing window ("horizontal" navigation); based on the new coordinates reached on the client-side canvas, a spatial range query (i.e., a window query) is identified and computed. This query retrieves all elements of the graph (nodes and edges) that overlap with the current window. A system that combines many of the features available in the above-described browsers is *Watson* (d'Aquin and Motta, 2011). In addition to its keyword-based search function and the capability to present results both in a tabular and a graph-based format, it offers APIs exposing high-level functions for finding, exploring and querying semantic data and ontologies available online. Programmers can execute functions for ontology construction, matching, sense disambiguation and question answering. Unfortunately, no visual mechanism is provided to open these opportunities also to non-technical users.

Despite the accurateness of graph-based visualizations, due to their exact match with the LD data model, it is acknowledged that the graph-based visual metaphor does not fit the mental model of non-technical users (Namoun et al., 2010; Zang and Rosson, 2008) and is not always the best way to represent large amounts of data items (Herman et al., 2000).

Marbles offers a different kind of visualization (Becker and Bizer, 2018). It is a server-side application that shows Semantic Web content in a human-readable way by using Fresnel *lenses* and *formats*. Lenses specify which properties of an RDF resource have to be displayed and how these properties have to be ordered. Fresnel formats determine how the selected properties are rendered by specifying RDF-specific formatting attributes and by

providing hooks to CSS, which are used to specify fonts, colors, margins, borders, and other decorative elements[**]. For example, in Marbles, colored dots are used to relate the origin of displayed data with a list of data sources. The Fresnel visualization technique has been exploited by other RDF browsers like *Piggy Bank* (Huynh et al., 2007), *LENA browser* (Franz et al., 2010), *Horus* (Napoleoni et al., 2014), *IsaViz* (Pietriga, 2018).

Another interesting LD visualization is available in *Sig.ma* (Tummarello et al., 2010). Starting from a textual search, this application enables the creation of *Entity Profiles*, which usually include information aggregated from multiple Web sources and presented to the user in an interactive visual interface. The most relevant properties and their values are listed, also specifying the data sources of provenance. Users are able to interactively explore the results and the related data sources. The Entity Profile is created by combining large-scale semantic web indexing, logic reasoning, data aggregation heuristics, ad-hoc ontology consolidation.

Based on a completely different paradigm, *SWiPE* (Atzori and Zaniolo, 2012) supports users in expressing complex queries on DBpedia such as: "Who are the U.S. presidents who took office when they were 55-years old or younger, during the last 60 years". It offers a search-by-example paradigm that allows users to enter the query conditions directly in the Infobox of a Wikipedia page. SWiPE then returns a description of the Wikipedia pages that satisfy the query conditions entered by the user.

Some mobile applications to navigate LD have also been proposed. One of the most famous is *DBpedia Mobile*, an Android app (also available for desktop computers). Based on the current GPS position of a mobile device, this app creates a map depicting information about nearby locations extracted from the DBpedia dataset. Starting from the map, users can explore information about locations and then can navigate into DBpedia and other interlinked datasets such as GeoNames, Revyu, EuroStat, and Flickr. Another mobile app was *iLD*, a linked-data browser for the iPhone that, taking in input a resource URI, retrieves and display in tabular fashion the results.

In comparison with the tools described above, the goal of MDSs is not only facilitating the LD exploration and visualization but also defining customized entry points that the users can reuse to retrieve data within interactive workspaces in different usage situations. MDSs have intrinsic visualization capabilities that derive from the construction process itself: the user building an MDS can decide which visualization template (which we call *UI components*) to use to render the MDS data. Each type of UI component also offers a set of manipulation functions which make the displayed data actionable. The inclusion of data into UI component also facilitates the integration of MDSs with other data sources, through an event-driven synchronization at the UI level that adds interactivity to the whole workspace (Cappiello et al., 2015; Desolda et al., 2017a). This aspect further increases the MDS added value, as Linked Data can also be consumed in a larger context where different data can be integrated and compared.

A similar component-based paradigm for LD consumption is adopted by FERASAT (Khalili et al., 2018), a faceted LD browser that proposes a set of structured UI components, each one offering a different type of visualization, that can be dynamically adopted to present some selected properties on the fly, i.e., while the user interacts with the system. The component-based organization of the FERASAT interactive workspace is very similar to the one proposed by our approach. However, while our platform enables the integration of different components, in FERASAT the different UI components provide separate (i.e., not synchronized) visualizations of properties of the same entity. Also, while MDSs enable accessing data till reaching the instance level, FERASAT focuses on browsing class properties, offering visualizations of aggregate values for selected properties of a class. Finally, another distinguishing feature of MDS, which is discussed in Section 2.2., is that its creation is driven by data quality dimensions. This aspect is not addressed by any of the visual tools described above.

2.2. Quality Issues

It is indubitable that LD offers a powerful paradigm to store, link and share data on the Web in a structured fashion, which represents an opportunity to enrich interactive information workspaces with information that would not be found easily by invoking Web APIs. However, principles governing LD do not guarantee the quality of the published data. Different studies demonstrated that often LD datasets are affected by quality problems that reduce their adoption in real contexts (Hogan et al., 2012; Zaveri et al., 2016). To consume data from this global data space

[**] https://www.w3.org/2005/04/fresnel-info

in an integrated fashion, data quality is an important challenge to be addressed. Indeed, the LD cloud is growing very fast and it contains data originating from hundreds of data sources, whose quality is very diverse, as values may be incomplete or incorrect. Even though gathering and publishing such huge amounts of data open new opportunities, data is only as useful as its quality.

In literature, there are different models, frameworks, and tools for assessing data quality, in particular for data stored in a relational database (Lee et al., 2002; Pipino et al., 2002; Wang, 1998). Nevertheless, quality of LD introduces new problems like coherence via links to external datasets, data representation quality or consistency with regard to implicit information. A comprehensive survey on LD quality is reported in (Zaveri et al., 2016). Starting from dimensions typically used to evaluate traditional data quality (Batini et al., 2009), the article surveys 21 approaches and extracts 23 data quality dimensions along with their definitions and corresponding metrics, which can be applied to assess the quality of LD. The resulting model is valuable since it considers several quality aspects, but it can be cumbersome to apply, especially if an automatic, on-the-fly computation of metrics is needed. This is an important requirement for our framework, where quality dimensions are used, at composition time, to suggest to users' classes and properties that might have in the end a reduced number of quality problems.

In (Cappiello et al., 2016), a simplified version of the previous model is proposed. The authors identified a minimal set of dimensions, and related metrics, built by considering and redefining all the 23 original dimensions, and adding some new concepts. The resulting model is composed of the following dimensions:

- *Amount of data*: the extent to which the quantity or volume of data is appropriate for a particular task (Wang and Strong, 1996). It can be measured through the average number of properties that characterize an entity in the considered dataset.
- *Conciseness*: the degree to which a resource is characterized by a set of properties that is free of redundancy (Wang and Strong, 1996). It can be measured through the ratio of the number of non-redundant properties and the total number of properties. For the whole dataset, conciseness is then the average of the conciseness measures for all its entities.
- *Completeness*: the extent to which data are of sufficient depth, breadth and scope for the task at hand (Wang and Strong, 1996). It can be measured through the ratio of the number of retrieved properties of an entity and the number of required properties for a specific resource; the required properties are the ones included in an ideal list of properties considered complete.
- *Navigability*: the degree to which different entities in the same dataset are linked. Starting from a specific entity and all the triples in which it is involved as subject, the measure considers the percentage of objects that are URIs with respect to the total number of objects.
- *Interlinking*: the degree to which resources in the dataset are linked with the same resource of an external dataset. It corresponds to the total number of *owl:sameAs* links of a whole dataset.

As explained in Section 3.2.6, we have adopted this model for assisting users in the selection of classes and their properties. In the context of the MDS creation, this model represents a good alternative to the original one (Zaveri et al., 2016), as it focuses on dimensions that can be automatically computed; thus it can be used to evaluate dynamically the user's choices.

2.3. The Mashup Paradigm for Linked Data Consumption

The notion of MDS has been conceived as part of a larger project focusing on the design of EFESTO (Ardito et al., 2014b; Desolda et al., 2019; Desolda et al., 2016, 2017b), a mashup platform that offers visual notations to enable the integration by end users of heterogeneous data sources and Web APIs. The experience that we gained in the last years and different field studies (Ardito et al., 2014a; Ardito et al., 2014b) led us to observe that, despite the wide availability of APIs, very often it is difficult for users to find mashup components that satisfy their diverse and very specific information needs. The data sources available today describe a portion of a domain and in several cases they do not provide details that are deemed relevant for a given application or user. Even worse, it is not possible to tailor the access to their data set, which is constrained by the exposed APIs. This limitation could be overcome by letting users integrate multiple data sources, for example by means of lightweight platforms and composition paradigms based on *mashup techniques* (Daniel and Matera, 2014).

Still, the *components* that can be integrated within mashup platforms could not be enough to satisfy specific information needs, and importing new components within interactive workspaces implies the configuration and wrapping of external services and data sources - an activity that cannot be performed by lay users and requires the intervention of platform administrators. This aspect represents an obstacle to the diffusion and acceptance of mashup platforms in real contexts and everyday use. We, therefore, investigated how LD, thanks to its huge amount of entities and attributes, and its standard paradigm for data publishing and access, would improve the openness of mashup platforms.

Leveraging the LD paradigm for data access, *MDSs* provide tailored access points over LD datasets that are not pre-packaged and are shaped by the users themselves. MDSs can be therefore considered a mechanism for the interactive exploration of LD by end users, as well as an improvement towards the openness of mashup platforms. This last aspect gives more flexibility to mashup composition, thus increasing its effectiveness as a paradigm that allows end users to manage the integration of heterogeneous data (Casati, 2011; Daniel and Matera, 2014).

We devised the MDS paradigm as an extension of the EFESTO platform, where users can visually define how to integrate data extracted from Web APIs, and select and visually configure templates for data visualization (e.g., by means of a list, a map, a graph) (Desolda et al., 2015). EFESTO also provides a set of tools, which we call *Actionable UI components*, that exploit functions local to the platform or exposed by remote APIs to allow the user to "act" on the extracted contents, for example to *collect&save favourites*, to *compare items*, to *plot data items on a map*, to *inspect full content details*, or to *arrange items in a mind map to highlight relationships* (Desolda et al., 2017a). This enables a kind of active sense-making, i.e., the information can be *elastically* transformed towards the actual accomplishment of task goals. Through these tools, and their particular task semantics, users are empowered to interact with the displayed information in a contextual manner, thus raising information in mashups to the level of task objects the user can act upon (Ardito et al., 2015). In Section 3.2.5, we will show in more detail how the notion of Actionable UI components can be applied to MDSs.

3. Metamorhpic Data Sources

In (Dadzie et al., 2011), the authors highlight some challenges to face in order to encourage the consumption of Linked Data by a larger audience of lay users:

- *Exploration starting point*: several LD browsers require the user to start browsing from a specific, valid URI. How can the users identify their starting point when they do not know what an LD URI is?
- *Combating information overload*: LD resources are characterized by a high information density, e.g., a high number of entities, entity properties and links to other entities, which lead to information overload and make it difficult for the users to identify the information they are looking for.
- *Returning something useful*: LD are represented in RDF, which is not easily understandable by users. Adequate, i.e., easy to understand, abstractions are needed on top of RDF, so that to improve the *usability of LD*. The quality of the retrieved data can also influence the usability of data.
- *Enabling interaction*: lay users are familiar with interactive mechanisms to explore and manipulate data on the Web. A similar experience should be offered also for LD exploration and manipulation.

In the rest of this section, we show how our approach based on MDS tries to give solutions to the previous aspects. Before illustrating in detail the algorithms and the techniques that enforce the elements listed above, we introduce a motivating scenario that explains how an MDS can be created and used within the interactive, mashup-based composition environment offered by the EFESTO platform.

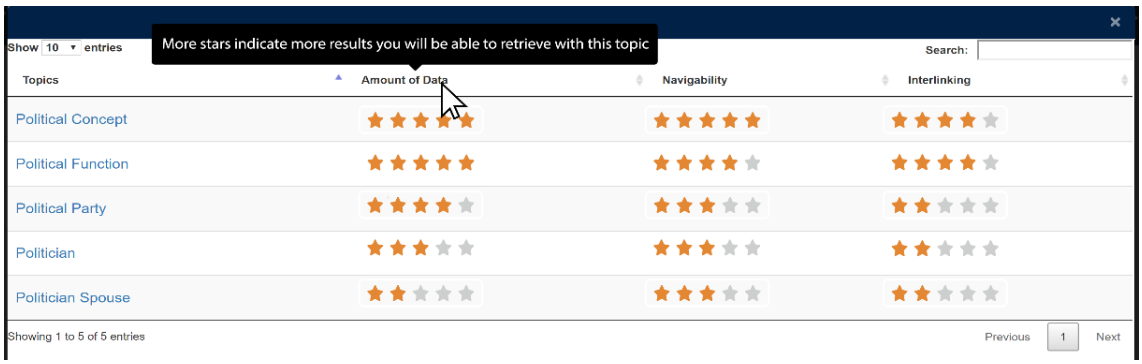
3.1. Motivating Scenario

Leonard is a journalist. He has to write a newspaper article about some US Presidents, thus he decides to use the EFESTO mashup platform to retrieve the information needed to argument what he wants to discuss in the article. To this aim, after logging into the platform and choosing the MDS creation, he types the keyword "Politic", which refers to the topic he is interested in (Figure 1a). EFESTO thus shows a list of topics (Figure 1b) and Leonard can choose one of them.

The topic selection is assisted by three quality indexes (see Section 3.2.6 for more details), whose values are normalized on a five-stars rating[††]. Leonard can access an explanation of these indexes by moving the mouse pointer on the index names, as shown in Figure 1b. Given his information need and the quality indexes, Leonard selects "Politician", then the *UI template* to visualize the final MDS properties (Figure 1c). He can now define how to visualize the Politician properties inside the selected UI template. Starting from the properties visualized on the left (Figure 1d, circle #1), Leonard drags & drops *birthPlace*, *birthDate* and *party* into the fields of the selected UI template (Figure 1d, circle #2). During this mapping activity, a box under the template schema shows a preview of how the data will be visualized at runtime, which also includes some representative values (Figure 1d, circle #3). A WYSIWYG paradigm is indeed provided to facilitate the understanding of what Leonard is creating. As in the topic selection step, also in this phase three quality indexes drive the user's selection of properties. After filling in all the fields of the UI template, Leonard saves the MDS configuration.

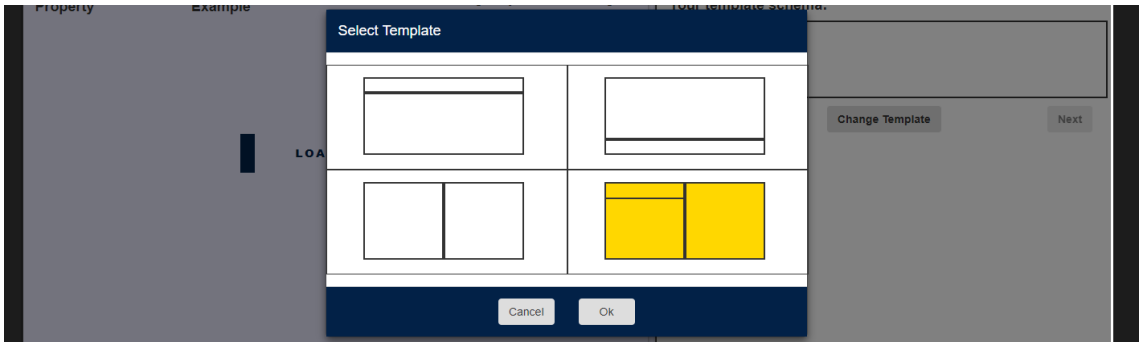
From now on, Leonard will be able to use the created "Politicians" MDS to retrieve details about a specific US President. For example, in Figure 2 he has typed "Barack Obama" in the search field, and the retrieved data are visualized according to the visual template previously configured. The entity properties can then be inspected by clicking on the "+" button. A pop-up window appears with a wider set of attributes that the users can, in turn, navigate (see Figure 3).

a

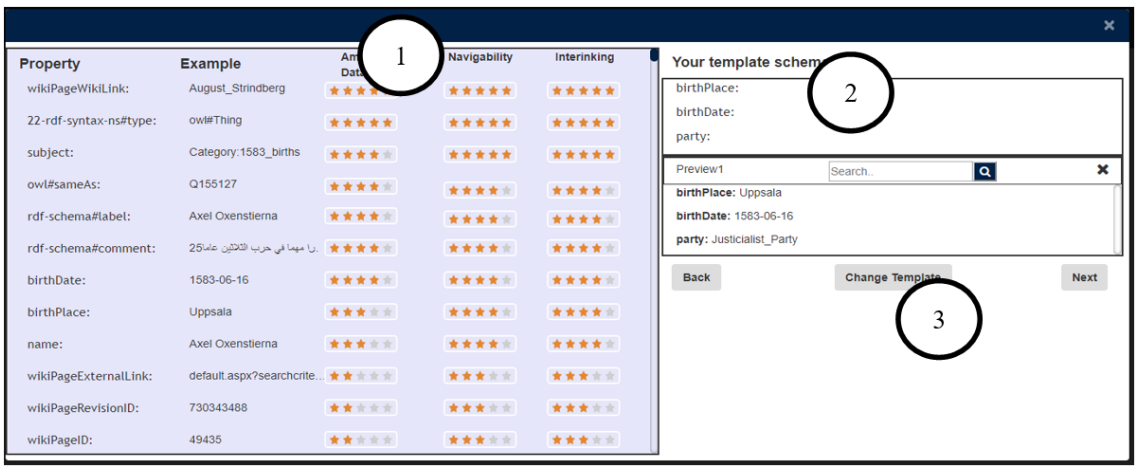


b

[††] A min-max normalization is performed on the resulting index values: min and max are used as the lowest and highest values among the values resulting for all the retrieved ontology classes.



c



d

Figure 1. MDS creation process. The users have to: a) type one or more keywords to indicate the topics they are interested in; b) select a topic from the list of topics semantically related to the user keyword; c) choose a UI template to visualize the MDS results; d) map topic properties into the chosen template.

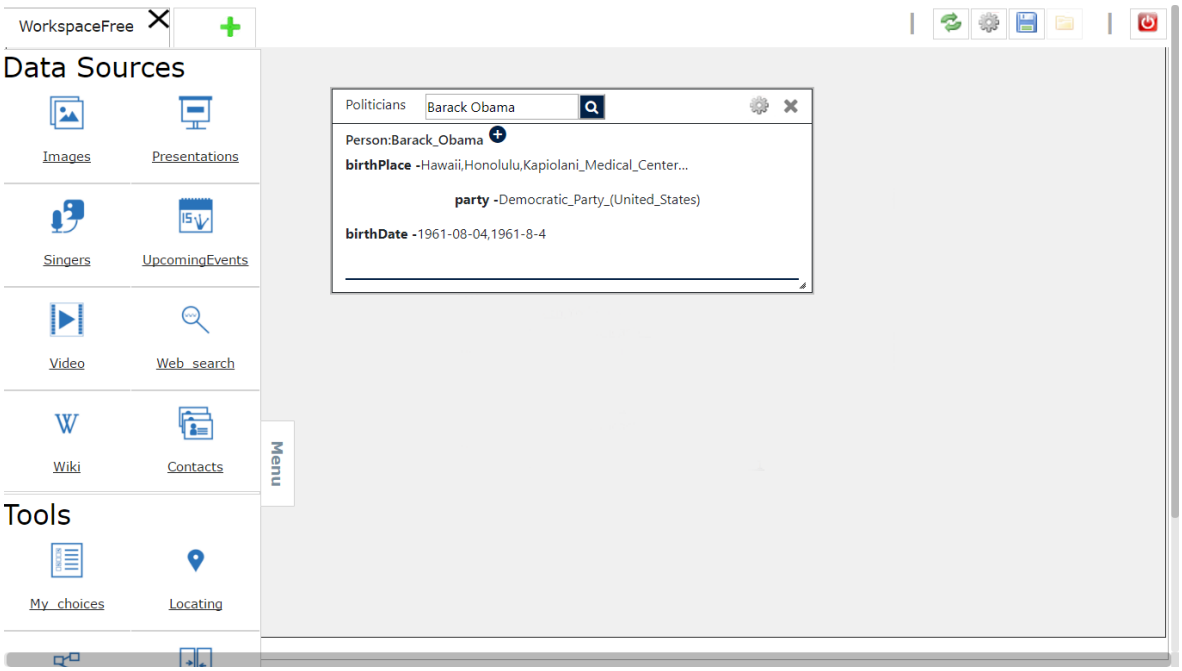


Figure 2. Results of a query to a metamorphic data source created from scratch.

Figure 3. Detailed view of the entity Barack Obama: the users can inspect each attribute by clicking on the related link.

3.2. Metamorphic Data Source: Behind the Scene

Looking behind the scene, an MDS is a view on the LD space, tailored to all the entities belonging to an ontology class, i.e., the topic selected by the user. The resulting MDS can be queried through visual mechanisms that help users filter the relevant attributes. In the following sections, we show all the steps for creating an MDS and exploiting it within an interactive information workspace where it can be composed with other data sources. The current prototype supports the creation of MDSs on top of DBpedia. This choice is due to two main reasons: first, DBpedia is one of the widest general-purpose datasets in the LOD cloud, and almost all the remaining LD datasets are linked with it. Second, operating on all the LOD datasets requires the development of an endpoint able to perform federated queries, as well as to merge their results (Görlitz and Staab, 2011). This aspect is an important challenge for LD (Hartig et al., 2009), however, it is out of the scope of this research. Thanks to the richness of DBpedia, we can safely assume it is a valid starting point to build significant MDSs. Anyway, the MDS paradigm

was devised so that it can be easily plugged on top of any LD dataset/ontology – this will be better explained in the next paragraphs.

3.2.1. Retrieving the most Relevant Topics

As explained in our scenario, the first step for configuring an MDS is the selection of a topic of interest. This step technically consists in selecting an ontology class. However, LD ontologies are often characterized by hundreds of classes, thus a manual exploration would be inefficient if not impossible. In order to facilitate the topic selection, we developed an algorithm (see Algorithm 1) that, starting from a user keyword(s), retrieves from any LD ontology (DBpedia ontology in the current implementation) a set of classes that are semantically related to the user typed keyword(s). The algorithm takes in input *i)* the set *K* of user-typed keywords, *ii)* the set *W* of words available in the WordNet lexical ontology and *iii)* the set *C* of classes available in the LD ontology. It then acts according to two main phases: 1) it finds all the WordNet words semantically linked to the typed keywords, and 2) it retrieves all the ontology classes whose names are syntactically similar to the WordNet words previously found.

ALGORITHM 1. Discovering ontology classes semantically related to the user-typed keyword(s)

Input: Set *K* of user keywords, Set *W* of all words in WordNet, Set *C* of ontology classes
Output: Set *O* of ontology classes semantically related to keywords in *K*

```
Declare a set T of temporary WordNet words;
for each keyword k ∈ K do //step 1 starts
    ki = stemming(k)
    for each word w ∈ W do
        wi=stemming(w)
        syntax_similarity = Soundex_distance(ki,wi);
        if (syntax_similarity > threshold)
        then
                T = T + wi;
                T = T + Synonyms(wi);
                T = T + Hyperonyms(wi);
                T = T + Hyponyms(wi);
                T = T + Holonymys(wi);
                T = T + Meronyms(wi);
                T = T + Related nouns(wi);
        end
    end
end //step 1 ends
for each t ∈ T do //step 2 starts
    ti=stemming(t)
    for each c ∈ C do
        ci = stemming(c)
        syntax_similarity = Soundex_distance(ci,t);
        if (syntax_similarity > threshold)
        then
            O = O + c;
        end
    end
end  //step 2 ends
```

The goal of the first step is to extend the user keywords with other words semantically linked. For this purpose, some WordNet semantic relationships about nouns are exploited (Miller, 1995): *synonymy*, which holds between words that denote the same concept and are interchangeable in many contexts; *hyperonymy*, which links more general words, e.g., “furniture”, to increasingly specific ones, e.g., “bed”; *hyponymy*, which is the opposite of the previous one; *holonymy*, which associates a term denoting the whole with a term denoting a part of (e.g., “tree” is a holonym of ”leaf”); *Meronymy*, which is the opposite of holonymy. For adverbs/adjective, we also considered *Related nouns.* At the current stage, we do not use other semantic relationships. Indeed, a preliminary assessment of

the algorithm showed that if further relationships for verbs (entailment) and adverbs/adjective (verb participles, derivational information) are included, often the retrieved classes are not relevant.

The algorithm first finds, for each keywords *k*, if a similar word *w* exists in WordNet. Stemming is applied to both *k* and each *w* retrieved in WordNet before their comparison[‡‡]. The stemming phase reduces inflected (or derived) words to their word stem, base or root form. If the similarity score, i.e., the *Soundex distance* (Zobel and Dart, 1996), between the stemmed *k* and *w* is major than a given threshold (0.8/1 in our algorithm according to a tuning phase performed on dozens of trials), the word *w* is added into the set T.

As second step, the algorithm finds all the ontology classes whose stemmed name are syntactically similar to the stemmed words in T. The found classes are added into the set *O* that is the final output of the algorithm. Since the resulting list can include irrelevant classes, the algorithm exploits a relevance measure to first rank and then reduce the result list, cutting out results whose relevance is below a threshold. The relevance is computed on the basis of the *composition context*, as explained in the following.

3.2.2. Exploiting the Composition Context to Rank the Retrieved Classes

Algorithm 1 helps the users filter the LD ontologies to find all the classes semantically related to the given keyword(s). However, the resulting list can still include irrelevant classes or a large number of classes, due to the multiple semantic relationships that are considered in WordNet, and because of the large number of classes in the LD ontologies. For this reason, we consider the *composition context:* we take advantage of the inclusion of MDS within an interactive information workspace and thus exploit a characterization of the content of the workspace under creation in order to 1) rank the list of resulting classes based on their relevance with the content already in place, and 2) reduce the list by removing the classes whose relevance is under a given threshold.

This is a problem typical of information retrieval and different systems exploit additional contextual information to improve retrieval accuracy. There are several types of information that can be exploited as context. For example, *relevance feedback* (Rocchio, 1971) is a technique that asks users to provide context details about an information-search task. This technique is widely recognized to be effective for improving retrieval accuracy; however, it requires explicit feedback by users who have, for example, to express the topic they are interested in or, in some cases, indicate explicitly which of the retrieved results are relevant. Since these additional actions might introduce cognitive effort and time overload, this technique is typically not adopted in real contexts. A widely diffused technique is instead the *implicit feedback*, whose main advantage is that it does not require any user effort (Joachims, 2002; Kelly and Teevan, 2003; Sugiyama et al., 2004; White et al., 2004). The context characteristics are in fact derived from the interaction history, past queries, documents the user has chosen to view.

During MDS creation, *implicit feedback* can be introduced by tracking the status of the information workspace under construction. In EFESTO, a workspace indeed includes several *UI components* (Desolda et al., 2017a), each one offering a view over one or more data sources. Each UI component is associated with a set of *terms* that characterize the dataset displayed through it. These terms derive from annotations specified for the related data sources when they are registered into the platform, and describe the main information entities (e.g., musician, video, upcoming events, documents) that can be retrieved. Since MDSs do not follow the same registration process, for the UI components in charge of displaying MDS data the set of terms consists of the topics (e.g., the classes) selected by the users among the ones in the set *O* constructed by Algorithm 1. The union of all the terms that characterize every single UI component in the workspace contributes to determine the implicit feedback.

We thus adapted a well-known algorithm that considers *implicit feedback* (Shen et al., 2005), *click-through information* and *previous queries*. The *KL-divergence retrieval* model (Zhai and Lafferty, 2001) is at the ground of this approach and proposes to treat context-sensitive retrieval as estimating a query-language model based on the current query and any search context information. According to this model, the retrieval task involves computing a query language model θ_Q for a given query and a document language model θ_D for a document, and then computing their KL divergence $D(\theta_Q||\theta_D)$, which can be used as the score assigned to the document. Formally, let $H_Q = (Q_1,...Q_{k-1})$ be the query history and the current query be Q_k. Let $H_C = (C_1,\ldots,C_{k-1})$ be the click-through history, where C_i is the concatenation of all clicked documents' summaries in the *i*-th round of retrieval since we may

[‡‡] In our implementation, a copy of WordNet with all the words already stemmed was cached to avoid the WordNet words stemming for each comparison.

reasonably treat all these summaries equally. The problem is then to estimate a context query model, which we denote by $p(\omega|\theta_k)$, based on the current query Q_k, as well as on the query history H_Q and click-through history H_C. In (Shen et al., 2005) the authors proposed and compared four different models, from which we selected the one with the most promising performance, the *Batch Bayesian updating*, also called *BatchUp*.

Some adaptations were required to use the BatchUp model in our system. In particular, H_Q becomes the history of the topics the users searched for, while H_C contains the history of the MDS topics/properties and the queries performed on the MDSs. According to the BatchUp model, when the users perform a query Q_k to search for topics, we rank and select the most pertinent classes by using $p(\omega|\theta_k)$. In particular, for each class we calculate $p(\omega|\theta_k)$ as relevance value used to rank the resulting list and, if necessary, to remove results below a threshold that in our current implementation is 0.5/1.

3.2.3. Querying the Metamorphic Data Source

At the end of the configuration, an MDS component is ready to retrieve entities related to the selected topic. As depicted in Figure 2, the final representation of an MDS within the interactive workspace is a UI component displaying the query result(s) according to the chosen visual template. A search box is available in the top part of the MDS. Searching MDS results is supported by an algorithm (see Algorithm 2) that finds relevant LOD entities starting from the keywords that the user enters in the search box.

ALGORITHM 2. Querying the MDS to retrieve entities belonging to the MDS topic

Input: Set *K* of user keywords; *c* is the MDS class
Output: Set *R* of entities belonging to *c*

```
Declare a set E of temporary LOD entities;

//step 1: invokes a function implementing two SPARQL queries that retrieve LOD entities containing the user-typed keywords ∈ K
E = retrieveEntities(K);

//step 2: searches for and selects entities belonging to the MDS class
for each entity e ∈ E do
    if (e.class == c) then //if the class of the entity is the same of the MDS class then add the entity to R
        R = R + e;
    end
end

//step 3: if there are no entities belonging to the MDS class, the 'nearest' entity is chosen
if (R is empty) do
    h2 = height of c in its ontology;
    for each e ∈ E do
        h1 = height of e.class in its ontology;
        LCA = LowestCommonAncestor(h1,h2);
        h3 = height of the LCA;
        dist = (h1 + h2) - (2 * h3);
        e = extendEntityWithItsDistance(dist,e);
    end
    R = selectEntitiesWithMinimumDistance(E);
end
return R;
```

The algorithm first executes a SPARQL query that retrieves from the plugged datasets entities whose labels contain the user typed keywords (step 1 in Algorithm 2). An example of SPARQL query referring to "Barack Obama" launched in the Leonard's MDS is the following (we report in bold the keyword used as parameter):

```
SELECT ?uri ?label
WHERE {
   ?uri rdfs:label ?label .
```

```
   FILTER (regex(str(?label), "Barack Obama", "i")  &&
langMatches(lang(?label),"en"))
}
```

The resulting entities can belong to classes slightly or completely different from the one used to build the MDS. LD entities are often labeled in a wrong way due to the automatic or manual translation of HTML pages/databases into the RDF format. Therefore, in order to find relevant results, the query does not limit the search to the only entities belonging to the MDS class, as the risk would be not retrieving relevant entities that belong to a different class. To better understand this problem, let us suppose that Leonard builds the MDS by using the *President* class, and then launches the query on Barack Obama[§§]. The first result in the returned list is the entity that Leonard would expect from the MDS (http://dbpedia.org/page/Barack_Obama). However, the class of this entity is *Politician*, a superclass of *President* (http://mappings.dbpedia.org/server/ontology/classes/). When none of the retrieved entities belongs to the MDS class, the algorithm selects entities belonging to classes that maximize the probability of being relevant with respect to MDS topic. Thus, for each entity found by the previous SPARQL query, the algorithm runs a second query to get entity class and its properties (step 1 in Algorithm 2), i.e.:

```
SELECT ?value
WHERE {
   <http://dbpedia.org/resource/Barack_Obama>   rdf:type   ?value.
}
```

Given the result of the second query, the algorithm first searches for and selects entities belonging to the MDS class (step 2 in Algorithm 2). If not any, then the *nearest* entity is chosen (step 3 in Algorithm 2). For this purpose, the ontology is represented as an n-ary tree. For each entity, a distance *dist* between the entity class and the MDS class is calculated according to the metric proposed in (Djidjev et al., 1991). In particular, the following elements are determined: 1) the height *h1* of the entity class; 2) the height *h2* of the MDS class; 3) the LCA (Lowest Common Ancestor, (Bender et al., 2005)); 4) the height h3 of the LCA. Then, the distance is calculated as *dist = (h1 + h2) - (2 * h3)*. After computing all the *dist* for each entity, the entity whose class has the minimum *dist* is selected and its properties displayed. If several entities with equal minimum *dist* are identified all of them are selected and displayed.

3.2.4. Visualizing the Retrieved Data

After the algorithm selects the entities, their attributes are plotted into the visual template according to the visual mapping performed by the user during the MDS creation. It is worth remarking that if the algorithm retrieves entities not belonging to the MDS class, the risk is to have null values for some of the properties the MDS was configured for. For instance, for the MDS built for the President class, if some properties specific of this sub-class were chosen to be displayed in the MDS (e.g., *prefect*), some null values are rendered in the MDS UI template if the retrieved entities belong to some President super-class. This is for example the case of Barack Obama result belonging to Politician, which is a super-class of President. However, users can still reconfigure the MDS by choosing another class and other more suitable properties.

The visualization of a more detailed set of entity properties (see the pop-up windows in Figure 3) then is managed by invoking an external API for entity summarization, SUMMA (Thalhammer and Stadtmüller, 2015). Starting from an entity URI, SUMMA builds a visual representation of the *n* most relevant properties. As default, SUMMA shows properties whose value is a URI to foster the LD, but this logic can be customized to visualize also properties with literal values. The visualization of entity properties is an aspect that can be improved in our platform. The adoption of an external API, however, shows how the platform privileges separation of concerns, an aspect that can facilitate handling data visualizations through any other local or external service that can be plugged on top of the logic for MDS management.

[§§] The DBpedia API that executes the SPARQL query related to Barack Obama is available at: https://bit.ly/2G8ZaxP.

3.2.5. Visual Manipulation of MDS results

As already described in Section 2.3, we use *Actionable UI components* to elastically manipulate MDS data. Thanks to this paradigm, users can act on entities retrieved through MDSs, for example they can collect and save favorites, plot data on a map, compare different results based on some properties (Desolda et al., 2017a). Going back to our scenario, after creating his MDS, Leonard can introduce in the workspace a *Map* tool, which he calls *Birth place*, to visualize on a map the birthplaces of US Presidents (see Figure 4). Leonard can drag & drop inside this tool the entities found through the MDS, to visualize them as pins on the map depending on the value of the birth-place property. Similarly, he uses another Map tool he calls *Alma Mater*, to visualize the locations of Universities where US Presidents got their degrees, and interestingly he discovers that Barack Obama graduated at the Columbia University in 1983 and also at the Harvard Law School in 1991. A third tool, *Comparing*, is used in this example to compare different Presidents' properties, like birthplace, birthdate and party.

Each of these tools is conceived as a general-purpose instrument that users can instantiate in multiple ways on MDS data. The tools are initially agnostic of how to deal with the data of a given MDS, thus users are in charge of defining their behavior, i.e., based on which properties the tools have to compute their function. For example, a Map tool was used by Leonard in two different ways to add different views on his MDS: *i)* to visualize US Presidents birthplaces and *ii)* to visualize degree Universities of US Presidents. To define how each tool has to operate with the MDS results, we adopt a *programming-by-example* technique: the first time one tool is applied on an MDS entity, the user has to map the MDS class properties onto the *capability* of the tool. For example, as shown in Figure 5, when Leonard moves for the first time an MDS entity inside the *Comparing* container, a pop-up window asks him to map specific Politicians properties onto the container characteristic attributes. As a result, the container will use the associated properties for its visualization. In other words, by means of simple drag & drop actions, the user trains the tool to visualize, for Politicians entities, *Birth place*, *Birth data* and *Party*. From now on, each time a Politicians entity is moved inside the *Comparing* container, the entity is automatically visualized on the basis of this training example.

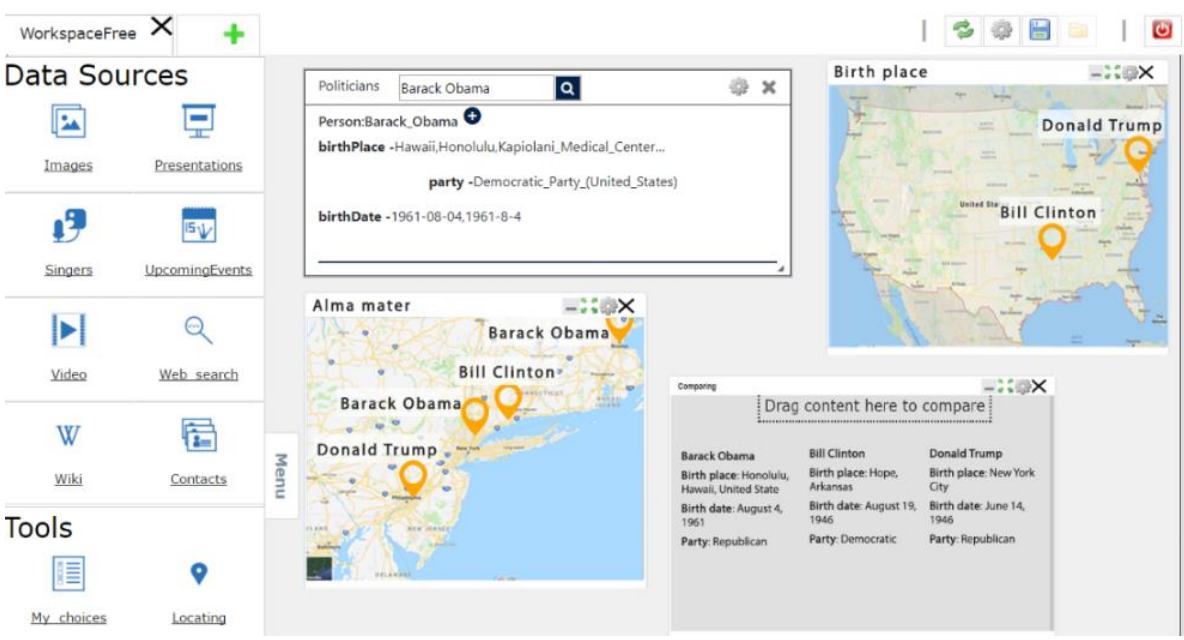


Figure 4. Example of an interactive workspace with an MDS that retrieve US Presidents and three tools (two *Maps* and one *Comparing* component) to manipulate and visualize the MDS results.

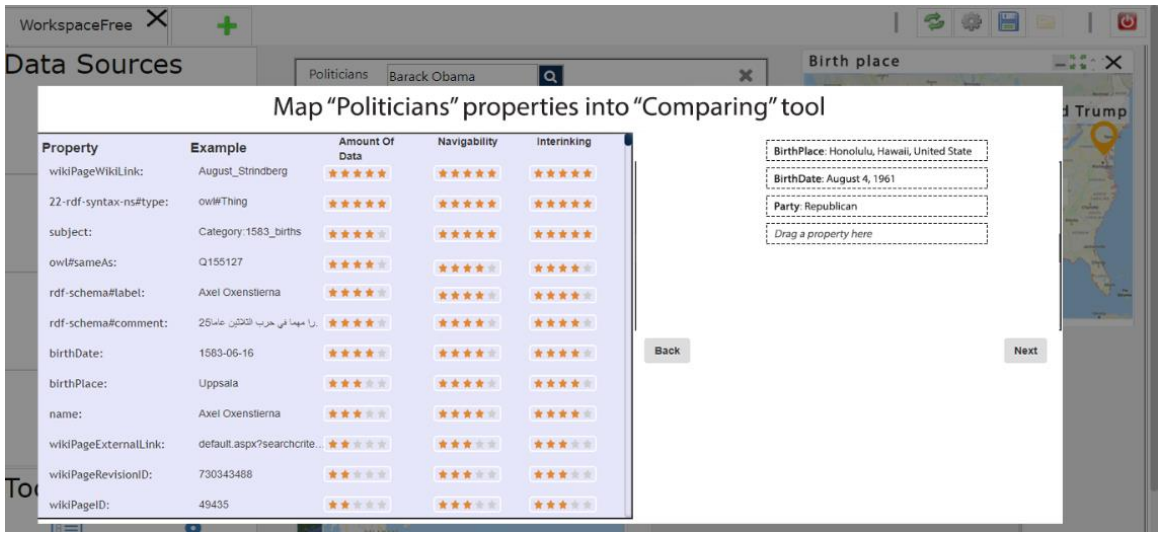


Figure 5. Training by the user of an Actionable UI Component (*Comparing* tool in the example): drag&drop actions map three MDS class properties (*Politicians properties*) onto the capability of the Actionable UI Component (textual fields displaying the attributes to compare).

3.2.6. Exploiting an LD quality model to assist class and property selection

As discussed in Section 2.2, LD are affected by quality issues that can limit their usage (Hogan et al., 2012; Zaveri et al., 2016)(Hogan et al., 2012; Zaveri et al., 2016)(Hogan et al., 2012; Zaveri et al., 2016)(Hogan et al., 2012; Zaveri et al., 2016)(Hogan et al., 2012; Zaveri et al., 2016)(Hogan et al., 2012; Zaveri et al., 2016). In (Cappiello et al., 2016) the authors propose a set of LD quality dimensions to assist the selection of LD resources. In our framework, we assume that the model is used when the system is configured in order to select quality data sources. We then use some adapted dimensions to let users quantify the quality of *i)* single classes of a dataset during the topic selection and *ii)* class attributes during the visual mapping onto the visual template.

As illustrated in Section 2.2, the original model proposes five dimensions. We selected those dimensions that could be dynamically measured on the subset of entities belonging to the classes retrieved by the system during the MDS creation. The three selected dimensions are *Amount of data*, *Navigability* and *Interlinking*. For *classes*, such dimensions are measured as in the following:

- *Class amount of data*: we compute the average number of properties that characterize entities belonging to a class.
- *Class navigability*: we consider all the triples having one of the class entities as subject, and we then compute the percentage of objects that are URIs, with respect to the total number of objects.
- *Class interlinking*: it is the total number of *owl:sameAs* links of the selected class entities.

For *class properties*, the three dimensions are measured as in the following:

- *Attribute amount of data*: it is the ratio between the number of class entities that have a value for the attribute, and the total number of entities belonging to the class.
- *Attribute navigability*: it is the ratio of the number of entities where the attribute appears as URI, and the total number of entities having a value for that attribute.
- *Attribute interlinking*: it is the total number of *owl:sameAs* links in the selected attribute.

We did not consider *completeness* as its assessment would require knowing an "ideal list" of properties that can be considered as complete with respect to the user task. Computing this measure would be feasible if the user tasks are known, i.e., the platform is configured to support only specific tasks. *Conciseness* assessment then requires a syntactic and semantic analysis of the returned entities and properties, also at the instance level. This would be too demanding to be performed on the fly; therefore, we decided to not consider this dimension in our current

prototype. Showing conciseness values would be however feasible by pre-computing the metrics (based for example on caching strategies).

4. Platform Architecture

The architecture of the platform supporting the MDS paradigm is organized along three layers (Figure 6). On top, the *UI layer* provides and manages visual composition operations expressed by users through direct manipulation actions on UI elements (Desolda et al., 2016). Two main components characterize this layer, i.e., *1) UI Components* that use *UI Templates* and *2) Actionable UI Components* that allow users to visualize and manipulate data extracted from UI Components. UI Components are widgets that expose data. We can distinguish two types of UI Components, i.e., the ones created on top of a Web service (e.g., YouTube API can be used to retrieve and visualize videos), and MDSs built on top of LD datasets. In both cases, their creation is facilitated by visual mashup mechanisms, like the one we described in Section 3.1 for MDSs, or the one to deal with Web APIs reported in (Desolda et al., 2016). The UI layer runs in the user's Web browser and communicates with the Logic and Data layer that run on the Web server.

The *Logic Layer* implements the *Mashup Engine* module that translates the actions performed by end users at the Interaction Layer into the mashup execution logic. It consists of three different components, i.e., *Source Manager*, *MDS Manager* and *Event Manager*.

The *Source Manager* is invoked each time users create UI Components on top of API data sources. It receives from the UI Layer a JSON file that codifies the user's composition actions and translates it into an object able to query the remote API. For example, if users create a UI Component by using the YouTube API, an object is instantiated that receives the user query, requests data from the API, receives the API results, filters the result attributes chosen by the user and gives back the results to the UI Component, which is finally in charge to visualize the results. This component is able to manage different types of APIs (e.g., RESTful and SOAP Web services), as well as different authentication methods like OAuth 2.0, OpenID and Custom Authentications.

The *MDS Manager* behaves in an analogous way. It receives a JSON file that codifies the actions the users perform when creating the MDS, and translates it into an object able to query the LD dataset. For instance, when Leonard creates his MDS, this module instantiates an object that receives user queries, performs the SPARQL query by using an LD endpoint[***], receives the resulting entities, applies the algorithm for entity selection (see in Section 3.2.3), filters the result attributes chosen by the users, and gives back the results to the MDS for visualization.

The *Event Manager*, instead, manages the UI Components coupling. When users define a synchronization between two UI Components *A* and *B*, it instantiates a listener that waits for an event on *A* that, when triggered, causes the execution of an action on *B*, according to the coupling defined by the user.

The *Data Layer* consists of four JSON-based repositories. The *Service Descriptor* repository stores JSON files providing abstract specifications on how to query each data source registered in the platform and how to read its results. The *UI Component* repository includes files that specify the services included in the components, the user-defined queries to integrate the services data sets, and the specification of the component UI template. The *UI Template Descriptors* repository stores JSON file reporting an abstract representation of the UI templates. The *Workspace Descriptors* repository contains files representing the workspaces created by each user and, for each workspace, it specifies the included UI components and possible UI synchronizations defined among them, as well as Actionable UI Components included in the workspace. The Workspace and UI Component descriptors are associated with the user who creates them, and thus can be accessed depending on the user's access rights. Some "default" workspace descriptors are also available to any user; they provide the specification for pre-packaged workspaces related to specific topics or domains.

EFESTO currently implements this architecture by means of the Java Spring framework[†††]. The UI was programmed by using Thymeleaf[‡‡‡], a Java HTML5 template engine, and the Bootstrap[§§§] front-end framework.

[***] In the current implementation the system is set to query DBpedia through the endpoint at the URI https://dbpedia.org/sparql

[†††] https://spring.io/.

[‡‡‡] 5http://www.thymeleaf.org/

The use of Bootstrap allowed us to build responsive UIs, which adapt their layout to the device on which they are run (e.g., PCs, smartphone, tablet). EFESTO has been deployed on a virtual machine created in the Windows Azure cloud platform (4 core, 8GB RAM, Windows Server 2012).

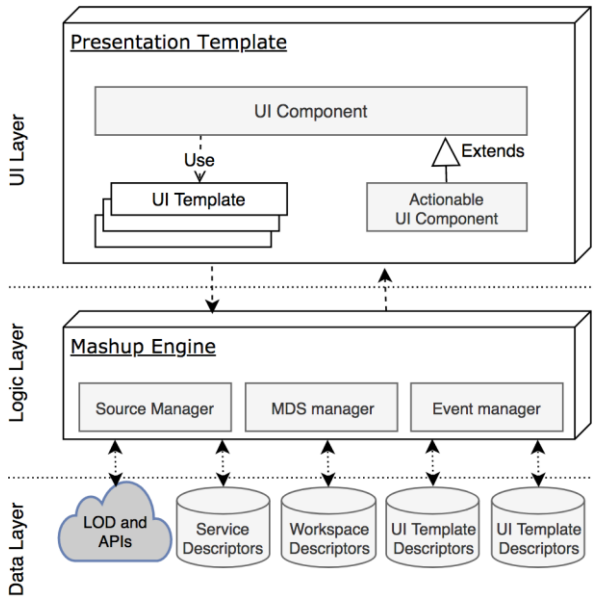


Figure 6. A high-level overview of the EFESTO three-layer architecture

5. Comparative study

We carried out an experimental study to understand if and how MDSs support users in accessing and exploring the LD space. Specifically, we compared two MDS versions, i.e., the one including quality indexes (*MDS_QM* in the following), and one without such indexes (*MDS* in the following). We conducted this comparison as we also wanted to investigate pro and cons of considering the quality model. Besides these two systems, we introduced the Google search engine (Google SE) as a baseline. Google SE is largely considered the "search engine giant", for its powerful algorithms, easy-to-use interface and personalized user experience. According to several studies (see for example the latest *netmarketshare* report, released on November 2018 (NetApplications.com, 2019)), Google SE holds the first place in search with a stunning difference of around 65% from the second place (held by Bing). A number of studies investigating strategies for information search have shown that Google SE is the search engine that better supports the behavior of Web users performing search tasks on the web (Aula et al., 2005; Aula et al., 2010; White, 2016). We also considered Wikipedia as an alternative baseline for our study, given that it offers access to the same DBpedia content. However, we chose Google SE to accommodate the search strategies that regular Web users would adopt most. It is also worth considering that Google SE grants the access to a very large content that also includes Wikipedia.

We therefore built a UI component on top of the Google Search API, with the same look&feel and features of the original search engine (*Google* in the following). It is worth remarking that the goal of our research is not the design of a search engine and the assessment of its precision and recall. Rather, we aim at the development of a novel paradigm to simplify LD access and querying. Choosing a baseline between Google and Wikipedia was due to the unavailability of tools offering exploration paradigms comparable with MDS. Indeed, some of the LD visual browsers that we reviewed are not anymore available or are not available as open-source projects (Becker and Bizer, 2018; Berners-Lee et al., 2008; Cheng and Qu, 2009; Dadzie et al., 2011; Ding et al., 2004; Franz et al., 2010; Harth, 2010; Huynh et al., 2007; Napoleoni et al., 2014; Tummarello et al., 2007). Others are too technical

[§§§] http://getbootstrap.com/.

and thus complex for lay users (Bergman and Giasson, 2008; Bizer and Gauß, 2018; d'Aquin and Motta, 2011; Hastrup et al., 2009; Lerner and Self, 2018). FERASAT provides a similar paradigm (Khalili et al., 2018), which still is not comparable as it provides only aggregated visualizations on entity properties which are not synchronized and do not offer functions to manipulate data. We thus reached the conclusion that Google, even if it has a coverage larger than MDSs in terms of data, and even if the users are already familiar with Google searches, under certain assumptions (i.e., specific search tasks) could be the only significant baseline to gather useful indications about the MDS data exploration process.

The leading question of this research relates to *the support provided to users by the visual interaction paradigm in creating and querying MDSs*. Specifically, the research questions driving this study were:

- *What is the difference between the considered systems in terms of user performance in searching for information?*
- *What is the difference between the considered systems in terms of user satisfaction in searching for information?*

5.1. Participants and design

We recruited 12 participants (3 females, 9 males) among the students of the third year of the Bachelor Degree in Computer Science. Their mean age was 23.25 years (SD = 3.99, min = 20, max = 30). As resulting from the demographic questionnaire that they filled in at the beginning of the study, participants had a high experience in IT ($\overline{x}$ = 8.2, SD = 1.48, min = 5, max = 10) but a low experience in using linked open data ($\overline{x}$ = 3.83, SD = 2.72, min = 1, max = 7). Likert scales ranging from 1 to 10 (1 very low - 10 very high) were used to assess such skills. Since the overall goal of this study was to investigate the support offered to the users by the proposed interaction paradigm in creating and using MDSs, we purposely recruited people with different information needs, who would allow us to collect data from different domains and different perspectives and make the study results generalizable. To this goal, we defined some tasks (see Type 2 tasks in Section 5.2) so that all the participants were enabled to search content they preferred most, without any specific constraint.

The performed controlled experiment adopted a within-subject design, with the system as an independent variable and three within-subject levels, i.e., MDS, MDS_MQ, Google. Each participant used all the three systems in sequence. In order to minimize learning effects (Graziano and Raulin, 2012), based on the Latin-Square design they used the system in a different order by considering permutations of the three systems and of the experimental tasks.

5.2. Tasks

With each system, the participants performed 4 tasks, each requiring finding specific information. Such tasks can be classified in two types: 1) *Type 1* consisted in finding information that was available on the DBpedia dataset, so that the participants could have a high probability of finding the requested information; 2) *Type 2* asked the participants to freely propose the information they wanted to find and then search it. Each type was used to generate two tasks. Type 2 tasks were purposely defined to verify in which measure MDS creation and its following querying actually satisfy the users' needs, even when the information required is not available in the DBpedia dataset. To achieve higher internal validity, each task required performing three different queries. To enhance external validity, we chose information that users often search on search engines[****]. The final task list was:

- Task 1 (Type 1) – Find the date of birth, place of birth, political party of the following American presidents: Barack Obama, Donald Trump, Bill Clinton.
- Task 2 (Type 1) – Find the description, music genre, date of birth of the following music artists: U2, James Blunt, Andrea Bocelli.

[****] https://trends.google.com/trends/yis/2018/US/

- Task 3/Task 4 (Type 2) – Think to a topic to search (e.g., buildings, movies, politicians different from the one of the previous task), identify three entities (e.g., U2, James Blunt, Andrea Bocelli, not these of course) and try to find information related to them, selecting three properties as you like.

For the two MDS systems, before performing the task queries, the participants had also to create an MDS to search the specific task information. In accordance with the within-subjects design, each participant performed 12 tasks that required executing three queries each, for a total of 432 queries (3 systems x 4 tasks x 3 queries x 12 participants). Each participant also created 8 MDSs, for a total of 96 MDSs (2 systems x 4 tasks x 12 participants).

5.3. Procedure

The study took place in a quiet university room where the study apparatus was installed. Two HCI researchers were involved: one acted as an observer, the other as a facilitator. A laptop with a 15-inch retina display provided with an external mouse was available. The observation of the user interaction with the systems was facilitated by an external monitor that duplicated the laptop screen.

The comparative study lasted 4 days, 3 participants were individually observed per day. Each study session followed the same procedure. First, a 10-minute presentation was given by the facilitator to introduce the participants to the goal of the study and what they had to do. Then, they were asked to sign a consent form for video-audio recordings and photo shoots, and to fill in the questionnaire for collecting demographic data and their IT and LD competences. The facilitator introduced the first system to be evaluated demonstrating how to perform a search and, in case of the MDS systems, how to build and query an MDS. Then each single participant was invited to perform the four experimental tasks. The participant read aloud the task text and then started it. At the end of all the experimental tasks, the participant filled in an online questionnaire about the system used. Before repeating the same procedure with the next system, the participant was invited to relax for 5 minutes.

An online questionnaire was administered at the end of the participant session. It asked to rank the three systems on the basis of their usefulness, completeness, and ease of use, and to choose which system the participant would like to use in her/his activities. This procedure was preliminarily assessed by a pilot study involving two further participants.

5.4. Data Collection

Both qualitative and quantitative data were collected to analyze user performance and satisfaction on the three systems. We considered the set of notes taken by the observer during the task execution, the video recorded during the different study sessions, the questionnaire answers, the free comments participants provided during the study. Regarding qualitative analysis, two researchers transcribed the audio/video recordings, the observer's notes, and the questionnaire open questions. A thematic analysis was carried out on these data. Then, the two researchers independently double-checked the results. The initial reliability value was 91%, thus the researchers discussed the differences and reached a full agreement (Braun and Clarke, 2006). Regarding user performances, the researchers built an excel file reporting for each query task performed by each user the following data: user ID (from 1 to 12), type of system (MDS, MDS_QM, Google), task ID (T1–T4), task activity (MDS creation, query 1, query 2, query 3), time (in seconds), task success (success – partial success - failure).

Regarding user satisfaction, two online questionnaires were administered during the study. The first questionnaire, organized in 4 sections, was used to evaluate each system. The first section included the *System Usability Score (SUS)* (Brooke, 1996), which gives an overview of the user's subjective usability evaluation of a given system. It is a closed-ended questionnaire encompassing 10 statements on an ordinal 5-point Likert scale from "strongly disagree" to "strongly agree". This questionnaire was chosen for its reliability, brevity and wide adoption (Borsci et al., 2009).

The second section included the *NASA-TLX* questionnaire, used as "Raw TLX" (Hart, 2006). It is a 6-item survey that rates perceived workload in using a system through 6 subjective dimensions, i.e., mental demand, physical demand, temporal demand, performance, effort, and frustration, which are rated within a 100-points range with 5-point steps (lower is better). These ratings were combined to calculate the overall NASA-TLX workload index (Hart and Staveland, 1988).

The third section had three questions about the 1) easiness of data retrieval process related to Type 1 tasks, 2) easiness of data retrieval process related to Type 2 tasks, and 3) quality of the retrieved data. Likert scales ranging from 1 to 10 (1 very low - 10 very high) were used for these questions. This section ends with two open questions on the advantages and disadvantages of the system.

Finally, the fourth section was composed of an incremental number of questions. In particular, at the end of the use of the second system, a single open-question asked the participants to illustrate the differences they had found between the first two systems. Analogously at the end of the study session, i.e., when the participants had used the third system, a single open-question asked them to illustrate the differences between the three systems.

The second questionnaire was administered before dismissing the study participant. It asked to rank the three systems based on their utility, completeness and ease of use (from 1 to 3, 1 is the best), and to explain the reasons for the expressed order.

One-way repeated measures ANOVAs (all Greenhouse–Geisser corrected) with posthoc pairwise comparisons (Bonferroni corrected) were adopted to analyses SUS, NASA-TLX results and some efficiency measures, such as task time, task success rate. Friedman test was adopted to analyze differences in systems ranking and the learning curve for the MDS building activity, with Wilcoxon signed-rank test used as posthoc pairwise comparisons.

5.5. Results for User Performance

In order to answer the first research question, namely, if there is any difference in terms of user performance in searching for information, we structured our analyses along different measures. In the following, we report on the results for all the considered performance dimensions.

5.5.1. Time to build an MDS

We first analyzed the time participants spent in building MDSs with the two systems (MDS $\bar{x}$ = 106.33, SD =55.27; MDS_QM $\bar{x}$ = 187.41, SD =148.62). The paired-samples t-test test highlighted that there is a significant difference among these systems ($t(47) = 3.467$, $p = .001$) in favor of the MDS system. The participants spent a significantly higher amount of time when using MDS_QM due to the extra time required to examine the quality indexes for both classes and their properties. However, this result does not mean that the longer the time spent, the higher the quality of the MDS and the results found. To understand this aspect, we will examine in the next sections the MDSs queries success rates and the user satisfaction.

Another considered aspect was how fast participants learned to build the MDSs. Figure 7 shows three learning curves, the blue and green ones are related to the time to build MDSs with the MDS and the MDS_QM systems respectively, while the red curve represents their average. These curves refer to data gathered in four different moments of task execution, i.e., after the creation by participants of 2, 4, 6, and 8 MDSs. For each moment, we averaged the time the participants spent in building the related MDSs. Friedman test was applied to understand where each learning curve significantly improves. For the MDS system, this test highlighted a significant difference among the four moments ($\chi^2(3) = 14.550$, $p = .002$). The posthoc comparison also revealed significant differences between round 1-2 and round 3-4 ($Z = -1.807$, $p = 0.050$), and between round 3-4 and round 5-6 ($Z = -2.240$, $p = 0.025$). For the MDS_QM system, the test highlighted a significant difference among the four moments ($\chi^2(3) = 9.038$, $p = .029$), but the posthoc tests were not able to reveal significant differences between the different rounds.

This analysis highlighted that with the MDS system users learn very fast how to build MDSs, already after the first two sessions and that after creating 6 MDSs their learning curve stabilizes, i.e., they are able to master MDS creation. Slightly different results emerged for the MDS_QM system. Indeed, the participants performed in an analogous way during the first four sessions and only after there was a significant improvement in their learning curve. This means that the indexes introduce a mental overload that slows down the learning curve. More details about the cognitive aspects will be discussed in 5.6.

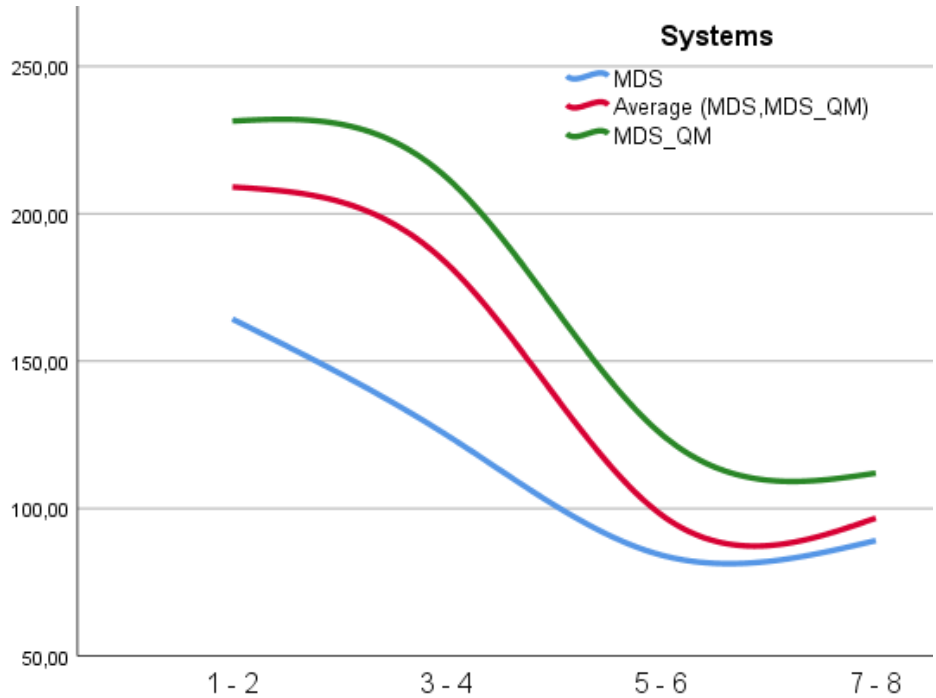


Figure 7. The three learning curves represent the time the participants spent to build MDSs with the two systems and the average between them. The x-axis represents the number of MDSs created.

5.5.2. Time to retrieve the information

The first indication comes from the time the participants spent to retrieve the information, i.e., the time to specify the task queries and detect all information required by the task (Google: $\bar{x}$ = 52.47, SD =35.89; MDS: $\bar{x}$ = 16.07, SD =11.02; MDS_QM: $\bar{x}$ = 19.63, SD =11.86). The ANOVA test highlighted a significant difference among the systems (F(1.237, 144.718) = 88.671, p < .000, partial η^2 = .431), and posthoc test revealed that the participants were faster with MDS than with both the MDS_QM (-3.568, SE = 1.398, p = 0.36) and Google (-36.407, SE = 3.526, p < 0.000), as well as they were faster with MDS_QM than Google (-32.839, SE = 3.587, p < 0.000).

A more detailed analysis focused on the average time spent on performing the tasks of Type 1 and Type 2 with the three systems (see Figure 8). For the time taken for Type 1 queries (Google $\bar{x}$ = 46.30, SD =29.05; MDS: $\bar{x}$ = 13.46, SD =8.03; MDS_QM: $\bar{x}$ = 16.11, SD =8.15), the ANOVA test highlighted a significant difference among the systems (F(1.230, 86.094) = 69.823, p < .000, partial η^2 = .431), and posthoc test revealed that the participants were faster with MDS than with Google (-32.831, SE = 3.637, p < 0.000), and they were faster with MDS_QM than Google (-30.183, SE = 3.656, p < 0.000).

For the time taken for Type 2 queries (Google $\bar{x}$ = 56.91, SD =44.58; MDS: $\bar{x}$ = 18.87, SD =13.86; MDS_QM: $\bar{x}$ = 25.37, SD =14.37), the ANOVA test found a significant difference among the systems (F(1.187, 53.398) = 22.000, p < .000, partial η^2 = .328), and posthoc test showed that participants were faster with MDS than with Google (-38.043, SE = 7.341, p < 0.000) or with MDS_QM (-6.5, SE = 2.547, p = 0.043), and faster with MDS_QM than with Google (-31.543, SE = 7.249, p < 0.000).

Both for general analysis and for the analysis of the two types of tasks, the lower Google performance can be explained by considering the significant amount of time that the participants spent first to identify the right page among the returned query results, and then the required information inside single pages. However, a not so trivial difference also emerged between the MDS and MDS_QM systems, with a lower MDS_QM performance mainly caused by queries for Type 2 tasks. For this aspect, even considering the data gathered through the satisfaction questionnaire, we were not able to identify a reasonable motivation. This aspect will be the object of further studies.

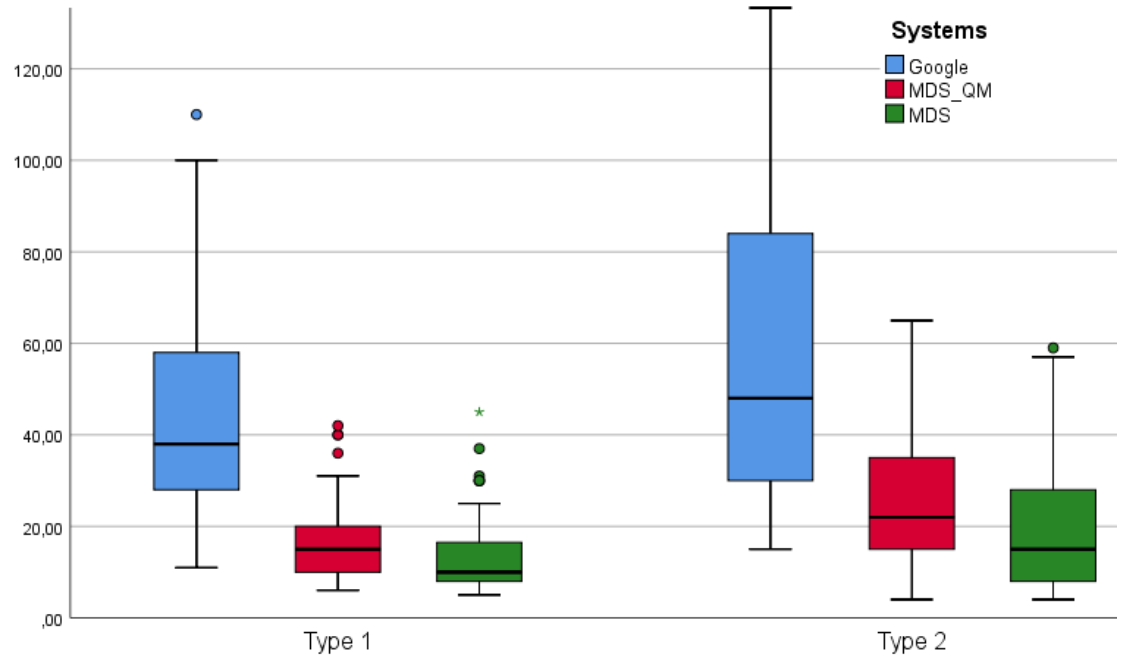


Figure 8. Average time to find the information through queries for tasks of Type 1 and Type 2.

From the previous analyses, it clearly emerged that queries performed with MDSs are significantly faster than the ones performed with traditional search engines. However, our aim is not to claim that the MDSs are in general more efficient than a search engine like Google. In contrast, we want to discuss how the MDS approach can be more effective in some specific situations it was conceived for, i.e., queries that domain experts have to perform repeatedly on the same topic.

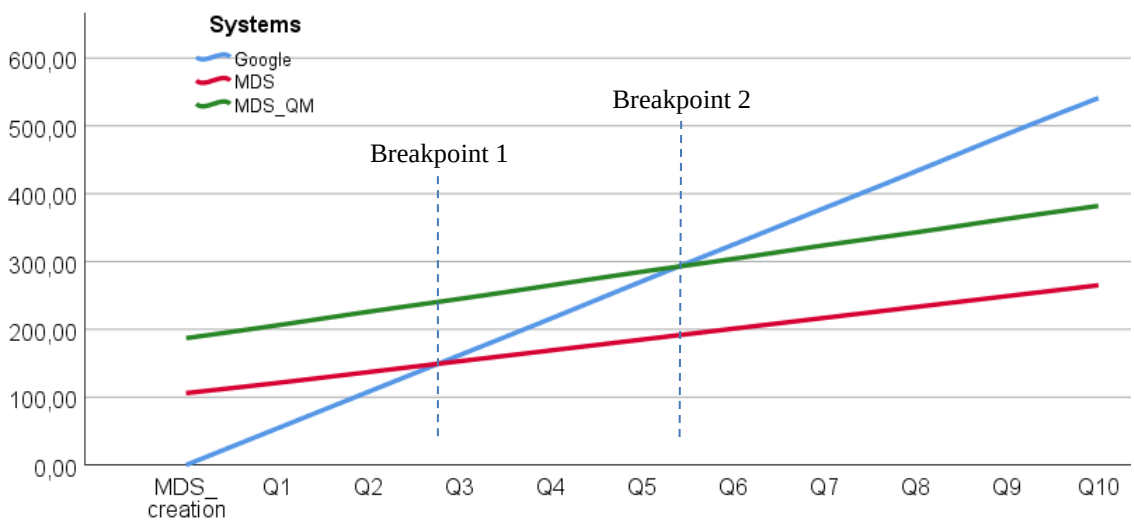


Figure 9. Time progression of the execution times of 10 queries for the three systems studied

To this aim, we determined a *breakpoint*, i.e., the minimum number of queries beyond which it makes sense to consider MDSs in place of traditional search engines. We considered the average time taken by the participants to carry out 10 different queries with MDS_QM, MDS and Google. As shown in Figure 9, of course, both MDS and MDS_QM require a certain time for MDS creation. After that, each query time Q_n is calculated summing up the

times taken from the previous query, plus the MDS creation process in case of the MDS systems. The first breakpoint occurs slightly before the third query (Q3 in Figure 9), where the time the participants needed to perform three queries is the same between Google and MDS. From that moment on, the use of the MDS for searching for information on the same topic becomes significantly faster that Google of around 36 seconds for each query. In case of MDS_QM, a higher time is required to justify its use instead of Google; indeed its breakpoint is between the fifth and sixth queries (Q5 and Q6 in Figure 9). After that, the participants become significantly faster than Google of around 32 seconds for each query.

5.5.3. Query success rate

The success rate is a measure typically adopted to evaluate the effectiveness of a system in supporting the completion of a task. In our study, each query executed by the participants was coded as "Success" if the information was completely found, "Partial success" in case of some missing data in the retrieved information, or "Failure" if the participant did not found anything or if the retrieved information was wrong. Figure 10 reports the query success rate for each system, also detailed for queries of Type 1 and queries of Type 2. The participants successfully completed all the queries by using Google. On the contrary, the MDSs systems were obviously more prone to partial success or failure due to the minor amount of data available and to their lower quality with respect to Google. In general, the participants adopting the MDS_QM system successfully completed 79 queries, partially completed 39 queries and failed in 26 queries. When using the MDS system, they successfully completed 74 queries, partially completed 47 queries and failed in 23 queries. A more detailed analysis was carried out by focusing on the different types of tasks. In case of Type 1 tasks with MDS_QM, the participants successfully completed 55 queries and partially completed 17 queries, without failing in any query, while when using the MDS system they successfully completed 53 queries, partially completed 18 queries and failed just 1 query. More critical results were found in case of Type 2 tasks: with MDS_QM, the participants successfully completed 24 queries, partially completed 22 queries and failed 26 queries, while when using the MDS system they successfully completed 21 queries, partially completed 29 queries and failed just 22 queries.

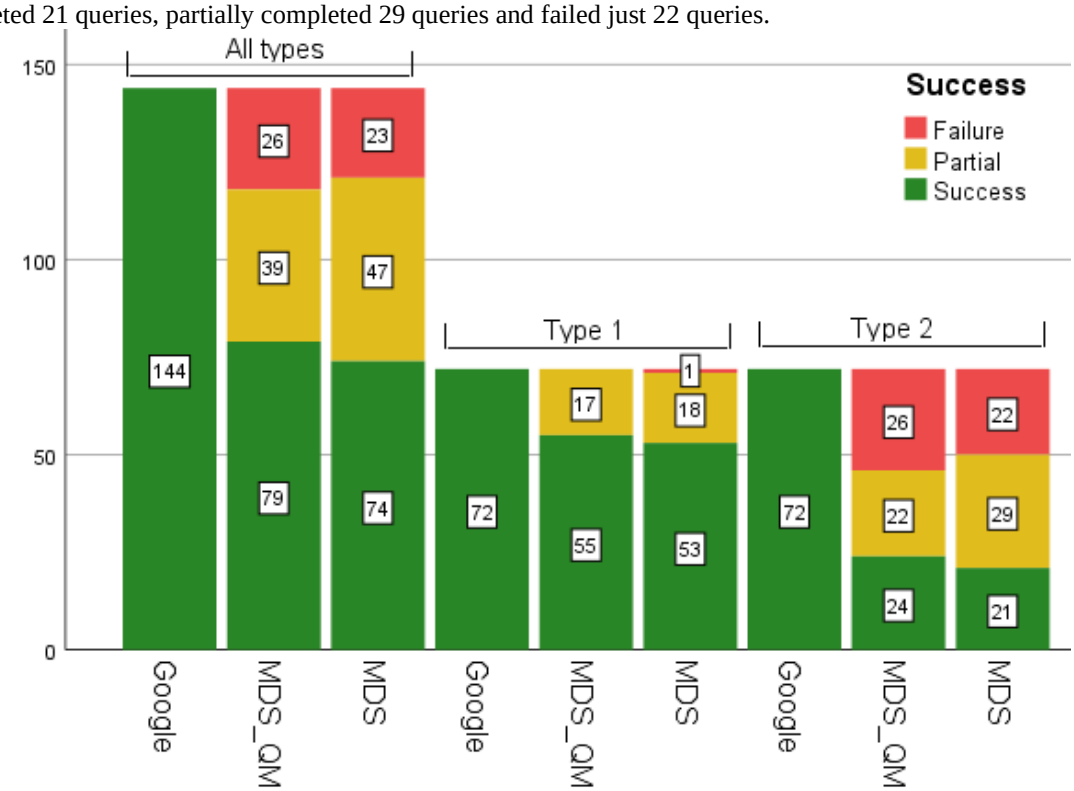


Figure 10. Bar chart reporting the query success rate of each system (first three bars) also detailed for the query of Type 1 (the fourth, fifth and sixth bars) and of Type 2 (the last three bars).

While the time analysis highlighted a lower performance of Google, an opposite result emerged here. Indeed, as expected, Google never failed in assisting users to find the requested information, while the MDS systems often caused a partial or a total failure. No great differences appeared between the two MDS systems. On the contrary, a clear difference emerged between the two types of tasks, since it is evident that Task 2 queries have brought out the limits of the MDS. It is worth remarking that the results of this analysis are dependent on the specific dataset used to build the MDS, i.e., DBpedia in the current implementation of our platform. Inevitably, the MDS data quality, and especially data completeness, depends on the quality of the underlying datasets. Thus, we are confident that this problem can be alleviated or almost totally solved by combining multiple datasets, thus maximizing the possibility to retrieve results for the user queries. However, as already pointed out, the integration of multiple LD datasets poses further challenges that are out of the scope of this paper.

5.6. *Results with user satisfaction*

In order to answer the second research question, namely, if there is any difference in terms of user satisfaction in searching for information, we structured our analyses along different measures. In the following, we report on the results of all the satisfaction dimensions.

5.6.1. Usability

SUS scores revealed the perceived usability of the three systems (Google: $\bar{x}$ = 84.08, SD =5.79; MDS: $\bar{x}$ = 68.00, SD =16.77; MDS_QM: $\bar{x}$ =68.08, SD =18.22), and the ANOVA test highlighted a significant difference among these scores (F(1.681, 18.492) = 5.095, p = .021, partial η^2 = .317). Posthoc test only shown that Google was perceived better than both the MDS (+16.000, SE = 5.629, p = 0.48) and the MDS_QM (+16.093, SE = 4.699, p = 0.17) systems.

According to (Lewis and Sauro, 2009), we also split the overall SUS score into two factors, i.e., *SUS-Learnability* considering SUS statements #4 and #10 (Google: $\bar{x}$ = 86.25, SD =6.08; MDS: $\bar{x}$ = 65.00, SD =21.74; MDS_QM: $\bar{x}$ = 66.67, SD = 21.67) and *SUS-Usability* considering all the other statements (Google: $\bar{x}$ = 83.54, SD = 6.55; MDS: $\bar{x}$ = 68.85, SD =18.12; MDS_QM: $\bar{x}$ =68.33, SD =16.83), as shown in Figure 11. Regarding the SUS-Learnability, the ANOVA test showed that there is a significant difference among the systems (F(1.677, 18.448) = 5.559, p = .016, partial η^2 = .336) and posthoc test discovered that Google was perceived better than both the MDS_QM (+19.583, SE = 6.468, p = 0.34) and MDS (+21.250, SE = 6.065, p = 0.15) systems. Regarding the SUS-Usability, the ANOVA test highlighted that there is a significant difference among the systems (F(1.703, 18.730) = 4.278, p = .035, partial η^2 = .280) and posthoc test revealed that Google was perceived better than only the MDS_QM system (+15.208, SE = 4.794, p = 0.27).

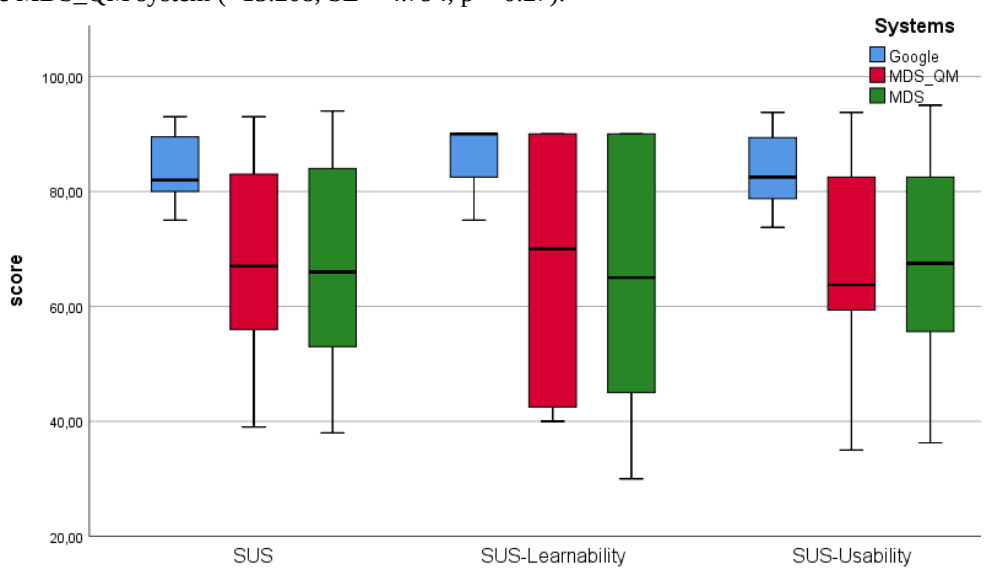


Figure 11. SUS score of the three systems and its decomposition in SUS-Usability and SUS-Learnability. A higher score is better.

It is evident, and not surprising, that Google SUS scores were better than the ones of the MDS systems. However, the aims of the SUS analysis were to 1) quantify the differences between the different systems and 2) compare the MDS scores. For the first point, the Google SUS score were significantly better thanks to factors like the habit of use, the reduced complexity given the absence of the creation phase, the professional look&feel. On the contrary, MDSs scores were negatively affected by the prototypal state of the systems, the required MDS creation phase, the use of a novel process to retrieve information. Anyway, the MDS scores are in line with SUS scores (69.5) of one thousand studies reported in (Bangor et al., 2009), thus it can be considered a promising result. Regarding the second point, the absence of differences between the two systems revealed that the use of quality indexes does not downgrade the perceived usability.

5.6.2. Workload

Differently from the SUS score, the workload caused by each system and estimated through the NASA-TLX was quite similar (Google: $\bar{x}$ = 23.06, SD =13.22; MDS_QM: $\bar{x}$ = 29.44, SD =17.51; MDS: $\bar{x}$ =28.06, SD =15.29), as also demonstrated by the ANOVA test that did not reveal any significant difference between the three systems ($F(1.865, 20.515) = .541$, $p = .578$, partial $\eta 2 = .047$). We also analyzed possible differences between the systems by considering each single NASA-TLX dimension, whose mean and standard deviation are reported in Table 2 and depicted in Figure 12.

Table 2. Mean and the standard deviation of the NASA-TLX dimensions for the three systems.

	Google		**MDS_QM**		**MDS**	
	Mean	*SD*	*Mean*	*SD*	*Mean*	*SD*
Effort	28.33	24.06	34.17	21.52	33.33	22.93
Frustration	23.33	24.98	22.50	14.85	24.17	16.76
Mental Demand	26.67	16.14	32.50	26.67	32.50	20.06
Performance	15.83	7.93	39.17	17.81	32.50	13.57
Physical Demand	17.50	8.67	24.17	21.52	20.83	19.75
Temporal Demand	26.67	25.70	24.17	18.89	25.00	24.68

The ANOVA test found a significant difference only for the Performance dimension ($F(1.779, 19.567) = 8.290$, $p = .003$). Posthoc analysis discovered that Google was scored better than the MDS_QM (-23.333, *SE=5.817*, *p = 0.006*) and MDS (-16.667, *SE=4.975*, *p = 0.019*) systems. No significant differences emerged for Effort ($F(1.515, 25.759) = 1.377$, $p = .265$, partial $\eta 2 = .075$), Frustration ($F(1.944, 33.048) = 1.634$, $p = .211$, partial $\eta 2 = .088$), Performance ($F(1.885, 32.052) = .328$, $p = .710$, partial $\eta 2 = .019$), Physical Demand ($F(1.274, 21.665) = .126$, $p = .787$, partial $\eta 2 = .007$) and Temporal Demand ($F(1.509, 25.654) = 2.457$, $p = .117$, partial $\eta 2 = .126$).

The NASA-TLX analysis complements the SUS one, focusing on different workload aspects. It is interesting to notice that, in contrast to the SUS, no differences exist between the three systems for the overall NASA-TLX score. Thanks to the triangulation with the other results, we can safely assume that two main factors contributed to equalize the workload of the three systems. The first one was the creation phase required in the MDS systems but not needed in Google. The second one was the exploration phase that is required in Google but not needed in both the MDS systems. Thus, in a different way, these factors increased the perceived user's workload balancing the final score.

Regarding the detailed analysis, the only notable difference emerged in the Performance dimension, which revealed that Google was the best system. Considering that in the NASA-TLX questionnaire the Performance item asks: "How hard did you have to work to accomplish your level of performance", a plausible explanation for the identified difference can be found in the task success rate. Indeed, participants felt more satisfied with Google because this was the unique system that did not create problems in terms of success of the queries, thus significantly improving the perceived performance.

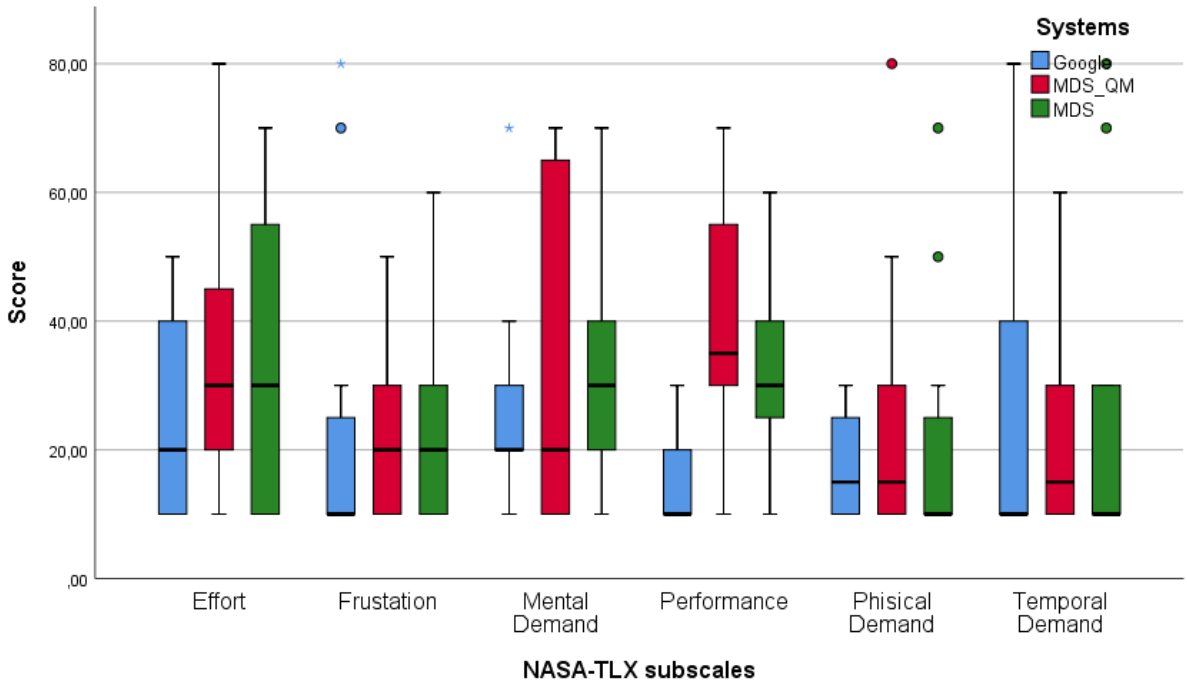


Figure 12. NASA-TLX dimensions workload of the three systems. Lower score is better.

5.6.3. Perceived quality of the data retrieval process

Specific qualities of the data retrieval process were also evaluated (see Figure 13). The ANOVA tests did not found significant differences among the systems both in case of easiness of the data retrieval process related to tasks of Type 1 ($F(1.604, 17.646) = 3.795$, $p = .051$, partial $\eta^2 = .257$) and Type 2 ($F(1.314, 14.453) = 3.554$, $p = .071$, partial $\eta^2 = .244$). Regarding the quality of the retrieved data, the ANOVA test detected a significant difference among the systems ($F(1.157, 12.731) = 10.192$, $p = .006$, partial $\eta^2 = .481$) and posthoc test revealed that Google was perceived better than both MDS_QM (+3.333, SE =.396, $p < 0.000$) and MDS (+3.083, SE =.848, $p = 0.12$). Similar to the NASA-TLX Performance, the explanation lies in the task success rate, which led to a higher perception of the quality of the results.

The last indications about the quality of the data retrieval process come from the third questionnaire, which revealed differences in how the participants considered the systems in relation to *Completeness*, *Utility* and *Ease of Use* and their overall preference on one of the three systems.

Regarding Completeness (Google: $\bar{x} = 1.54$, MDS_QM: $\bar{x} = 1.92$, MD: $\bar{x} = 2.54$), the Friedman test revealed a significant difference between the three systems ($\chi^2(2) = 6.682$, $p = 0.035$) and posthoc test found only that MDS_QM was significantly better than *MDS* ($Z = -2.333$, $p = 0.020$). Regarding the Utility (Google: $\bar{x} = 1.79$, *MDS_QM* $\bar{x} = 2.04$, *MD* $\bar{x} = 2.17$), there are no significant differences between the three systems ($\chi^2(2) = .977$, $p = .614$). Concerning the Ease of Use (Google: $\bar{x} = 1.50$, *MDS_QM* $\bar{x} = 2.21$, *MD* $\bar{x} = 2.29$), the Friedman test revealed a significant difference between the three systems ($\chi^2(2) = 6.229$, $p = .044$) but posthoc tests were not able to detect significant differences in the pairwise comparisons. There is only a notable result close to the p-value threshold ($p < .05$), since Google was better than *MDS_QM* ($Z = -1.941$, $p = .052$). Even if the p-value is not strictly below the threshold, we highlight this trend as we will discuss it in the following sections.

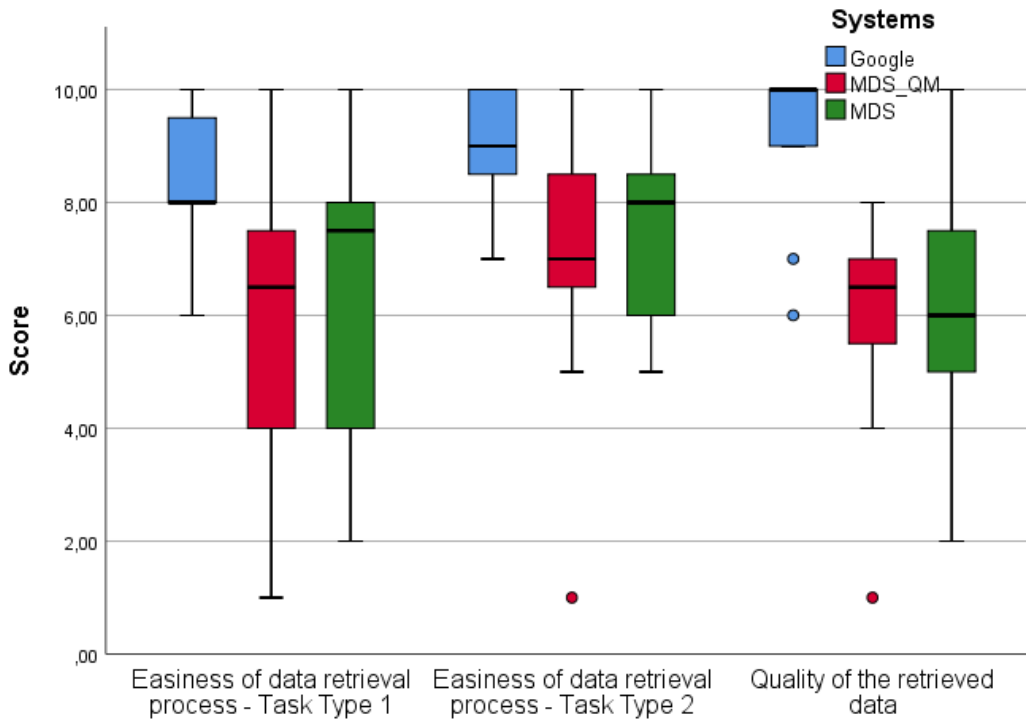


Figure 13. Qualities about the search process. A higher score is better.

5.7. Qualitative data analysis

The participants appreciated very much MDSs because they permit to perform specific queries on a specific topic, without losing time and devoting effort in exploring different Web pages, as it happens when using Google. In other words, MDS and MDS_QM permit to lighten the effort for searches in large amounts of data as they allow users to select pertinent properties and exclude what might not be of interest already when the user configures the search query. This aspect drastically reduces two problems known in the literature, i.e., getting lost in hyperspace (Otter and Johnson, 2000) and cognitive overload (Kirsh, 2000). The participants found MDSs quite simple to use, and only two participants said that, in order to properly use it, a minimum of computer skills is required.

The most severe problem of MDSs relates to the obsolescence of data. This problem, however, cannot be ascribed to MDSs per se. It could be alleviated by means of systematic quality assessments applied on the entire data set the system rely on and on the choice a-priori of data sources that feature a high quality. A further disadvantage concerns the language of the results, which sometimes is different from the search language - in some cases results were presented in Arabic. The participants also pointed out that the results of the queries were not up to date (e.g., a participant created an MDS to find F1 drivers and when he found Lewis Hamilton the MDS visualized only 3 Word Championships titles instead of 5) or even missing, and the waiting time was often long. Finally, the participants suggested creating a more attractive UI.

In the comparison between MDS and MDS_QM, in few cases the participants did not notice any difference: they perceived that a quality-based ranking was provided for MDS too, and declared that they “automatically” associated the classes or attributes listed at the first positions with data with a better quality. In any case, since the MDS_QM explicitly allowed the participants to understand which attributes could increase the quality of data, they preferred this version of the system. For MDS_QM system the participants showed difficulties in interpreting Interlinking and Navigability quality indexes. The two indexes were often confused since their meaning is quite similar, being related to the possibility to navigate starting from the resulting entities. It is reasonable to think that users would need prior knowledge related to the LD paradigm to understand this difference, thus a different approach should be adopted to simplify this aspect. For example, an aggregate index can be used to immediately give the users an

overview of the topic and property quality, leaving them the possibility to visualize on-demand the details on the three indexes, e.g., by moving the mouse pointer on the aggregate index icon.

Google was appreciated for its ease of use and the speed in providing query results. The participants are used to query Google for their daily searches, it is a system that nowadays everyone knows. Another Google advantage, highlighted by many participants, was the richness of its results. Also, users can compare the results with each other to verify their correctness and, if they are not satisfactory, they can easily change the keywords. The same richness of results was however highlighted as a Google disadvantage: the results were considered in many cases too generic, and this forced participants to navigate through the different returned Web pages in order to identify the correct ones.

5.8. Threats to Validity

In this section, we analyze some issues that may threaten the validity of the experimental study, also to highlight under which conditions the study design offers benefits that can be exploited in other contexts, and under which circumstances it might fail.

5.8.1. Internal validity

Internal validity can be threatened by some hidden factors compromising the achieved conclusions:

Learning effect. In our experiment, this factor was minimized by counterbalancing the order of the systems and tasks according to a Latin Square design.

Subject-expectancy effects. Students are not the best participants for an experimental study due to the subject-expectancy effect they can produce, i.e., a form of reactivity occurring when a research subject expects a given result and therefore unconsciously affects the outcome. We mitigated this effect by masking details that could produce bias. In particular, we presented the experiment to the participants in a way that suggests that we had no stake in the outcome. For example, we introduced all the experimental systems as already available tools; furthermore, in order to foster the credibility of this aspect, we developed our systems with a professional look-and-feel.

Method authorship. We eliminated the biases that different facilitators running the experiment could introduce, as we had the same facilitator for every session of the study. In this way, we avoided any variability in the initial training as well as in the way participants were observed.

Information exchange. Since the study took place over 4 days, it is difficult to be certain whether the involved subjects did not exchange any information. However, the participants were recruited during the exams period thus, for many of them, it was difficult to communicate. The participants were asked to return all the material (e.g., the booklet) at the end of each session. We asked the participants that typically study and travel together to perform the test in the same session.

Understandability of the material. A pilot study involving two further participants was performed to evaluate the system reliability and the research methodology (e.g., time constraints, coding techniques, video-recording activities), as well as the understandability of experimental procedures and materials.

5.8.2. External validity

External validity refers to the possible approximation of truth of conclusions in the attempt to generalize the results of the study in different contexts. With this respect, the main threats of our study are:

Users age and domain experience. Since the study participants were young people, with different information needs, and not experienced with LD, we have to take into account three potential limitations of the study results. The first one is the participants' age that limits the prediction of the benefits of the systems to older people. Thus, we can safely accept the experiment results for *digital natives* (Prensky, 2001) but further studies have to be carried out including older people. The second potential limitation is related to the participants' background: it would be interesting to extend the sample to people with different education, background and also with skills in LD. The last limitation regards the participants' domains: to foster the generalization of the study results, we purposely recruited participants with generic interests in content spanning different domains. However, further validations with domain experts of specific communities would be helpful to assess the MDS performance in relation to the needs of specific working domains.

Task Complexity. The two types of tasks administered during the study guaranteed a wide coverage of real cases: Type 1 was designed considering the most trend topics on search engines, while Type 2 tasks covered the specific needs of each user. Despite that, due to the time constraints of the study, the participants were asked to perform only 3 queries, thus it will be useful to explore more real situations where user perform dozens of tasks every day. To this aim, longitudinal studies or field studies are appropriated.

Adoption of a wider number of LD datasets. In the current implementation, the MDS is built on top of DBpedia because federated queries useful to search on the entire LD cloud pose challenges out of the scope of this paper (Görlitz and Staab, 2011; Hartig et al., 2009). This aspect limits the great potential of MDS to exploit the plethora of information available in LD. It is therefore important to empower the MDS giving the users the possibility to exploit on a higher number of datasets.

5.8.3. Conclusion validity

Conclusion validity refers to the validity of the statistical tests applied for the analysis of the collected data. In our study, this validity was ensured by applying the most common tests that are traditionally employed in Empirical Software Engineering (Juristo and Moreno, 2013).

6. Conclusion

In this paper, we have presented the MDS paradigm to facilitate the consumption of linked data by users without technical competences on the RDF data model and the related query languages. With this paradigm, we aim to contribute towards spreading the adoption of linked data to a larger number of users. One of the most severe problems afflicting LOD today is indeed the scarce utilization of such data model outside research communities.

We illustrated how the MDS approach revolves around some key challenges for interacting with linked data (Dadzie et al., 2011), which we addressed as follows:

- *Exploration starting point*: Identifying the starting point of the exploration is facilitated by allowing users to specify keywords.
- *Combating information overload*: The users are assisted in tailoring their customized access points over LD, which can then be reused in several situations when the same or similar data are needed.
- *Returning something useful*: We adopt mechanisms that filter out irrelevant resources by taking into account *i)* the semantic similarities between ontological entities and *ii)* the context of the user's information seeking process, as derived from the query history and the data already included in the information workspace under construction. This last point greatly characterizes our approach from others, as we propose linked data consumption in a context where LD datasets can be integrated with other heterogeneous data sources and Web APIs. Considering the quality of class entities and their attributes also improves the usability and the usefulness of the retrieved data.
- *Enabling interaction*: By means of Actionable UI Components, we provide interactive mechanisms to manipulate linked data and extract useful insights from them.

The comparative study that we conducted to assess the validity of the MDS paradigm featured some limitations, first of all, the lack of tools to be used for the comparison which led us to adopt Google as baseline. This compromised some results, especially those related to the usability of the two MDS systems: Google is indeed the most popular search engine and offers an incomparable data coverage and effective algorithms for information retrieval that greatly enhance the user experience. However, the study shed light on some important aspects of the MDS paradigm and, more in general, for linked data consumption. In the following, we summarize the most relevant findings and, also taking into account some limits of our approach, we present some lessons learned that can guide the design of systems for linked data consumption. After, we conclude the paper by outlining our future work.

6.1. Lessons learned

MDS for domain-specific search tasks, but not a panacea. The comparative study demonstrated that, besides being an approach to reduce LD exploration complexity, MDSs can also support search tasks like in traditional search engines. However, MDS promotes a different approach: on one hand, once MDSs are in place, users can immediately find specific information without exploring different results and being overwhelmed by useless information; on the other hand, they have to spend time in building their MDSs. It is clear that, as already pointed out, MDSs cannot be considered as a substitute for search engines and are not effective for any type of query. MDSs are beneficial in those contexts and domains where users repeatedly perform queries on the same or similar topics. This emerged from the time analysis reported in Section 5.5.2 that highlighted that the user performance with MDSs in terms of time improves of around 30 seconds per query after the third query on the same topic (6 queries in case of MDS_QM). These analyses can help understand in which contexts and situations the use of MDS is better than the traditional search engines. More in general, some considerations can be applied to the LD model. Linked data, at least in their current format, cannot replace the huge availability of data reachable through the traditional Web and through search engines. This led us to reflect also on the need of easy paradigms for creating and maintaining LD datasets: if this aspect is improved, it can have an impact on the creation and maintenance of LD, thus on their coverage and quality.

The quality matters. Data quality is a crucial aspect of spreading the adoption of LD datasets (Cappiello et al., 2016; Zaveri et al., 2016). The limitations related to the quality of MDS data clearly emerged when the participants were completely free to build custom MDSs (Type 2 tasks), with evident failures caused by missing data in DBpedia. The quality indexes introduced in the MDS_QM were taken into account by the users, as emerged by the higher time they took to build MDSs, and influenced their opinion about the completeness, which was better perceived for MDS_QM. However, in the end, quality indexes for entity choice did not significantly help participants build high-quality MDSs, as demonstrated by the same number of partial or failure queries for both MDS and MDS_QM. The analysis of notes and video clarified that participants read and evaluated the entity indexes, but very often the system did not actually return multiple topics with similar meaning to choose from. This is because MDS systems currently operate only on the DBpedia ontology, where it is reasonable that single topics are represented by single classes. We are confident that, by introducing more datasets and ontologies, quality indexes can better support users in the selection of topics. If multiple datasets are available, it also becomes important to evaluate the quality of the whole datasets, an activity for which the quality model presented in (Cappiello et al., 2016) was originally conceived. Regarding the difficulties that participants had in interpreting Interlinking and Navigability indexes, we believe that their meaning would become more evident if multiple datasets are included in the system. Also, more significant abstractions are needed to help users understand better their meaning, without requiring technical knowledge about LD and their underlying data model.

Adequate mechanisms to tailor personalized entry points can reduce LD complexity. Visual composition paradigms inherited by mashup platforms lowered the LD exploration complexity, which is one of the factors limiting the diffusion of LD to the mass. However, not only the visual mechanisms helped users deal with LD complexity, but also the opportunity to define customized entry points that can then be reused to retrieve data. The benefits of the algorithms presented in Section 3.2.1 would be even more evident in case of adoption of multiple LD ontologies and datasets: thousands of classes would be reduced to a set of dozen of topics, also thanks to the composition context used to rank and reduce the retrieved topics (Section 0), and to the LD quality model that assists topic and property selection (Section 3.2.6). The adequateness of both the visual paradigms and of the algorithms assisting user selections of data also emerged from the analysis of the time the users spent in building the MDSs. Indeed, the learning curves presented in Section 5.5.1 show that users quickly become able to build MDSs.

Trade-off between results richness and data retrieval easiness. We intentionally reduced the complexity of LD exploration by creating an access point on top of LDs via the MDS, which is in charge of displaying a small but essential, i.e., responding to the user needs, amount of information. However, although the search for the specific data for which MDS is tailored is more precise and faster (if present), it then becomes difficult to explore further information related to the returned results. This was clear from the user's comments who pointed out that Google makes results exploration easier, as it highlights also related information. Displaying too many information contrasts our idea of making the LD access and navigation light. However, this problem put the emphasis on the

need for mechanisms for the progressive exploration of results without worsening the MDS performance. A possible solution can be the replacement of the entity summarization currently adopted in the system, the one provided by the SUMMA API (Hogan et al., 2012), with other ad-hoc solutions providing further information about a selected entity.

6.2. Future work

Our future work will especially address the MDS limits highlighted by the study. First of all, we will plug into the system more LD datasets, to provide the users with more significant results and improve the motivation for using the tool. Adding further datasets will, first of all, imply extending the platform to handle federated queries and the fusion of the returned data (Görlitz and Staab, 2011). However, this will allow us to deepen some features that could not be completely assessed with our current version of the system, such as the effectiveness of the algorithms for entity and attribute filtering and the impact of the quality model on the user selection of data. Further user studies will be conducted to evaluate these aspects. The availability of several LOD datasets would also allow us to validate the MDS approach with domain experts that in a greater number of datasets can find information that DBpedia might not provide, for example, related to laws, molecules, government stats.

The study reported in this paper considered as failed those queries that returned both wrong results and missing data. Due to the limited number of queries, it is not worth to distinguish between the two situations to estimate MDS precision and recall. Thus, a further contribution towards improving the effectiveness of the algorithms will be the evaluation of these two measures of relevance on MDSs built on different datasets and on a wider set of queries, also investigating correlations with LD quality.

Another contribution in relation to LD quality would be the definition of paradigms for LD publishing that can help create and maintain LD, thus improving their coverage and quality. This aspect is rarely covered by the works present in the literature. Among the approaches that we surveyed for our research, only a few also cover RDF writing through intuitive, visual mechanisms (see for example (Jusevičius, 2018)). To promote the adoption of MDSs to non-technical users, we will also investigate how to simplify the quality model interpretation through the aggregation of its indexes, providing the user with an overall score that can be inspected to see the original indexes, if needed.

Another aspect that we aim to improve relates to the visualization of data. The quality of the provided visualizations was not central to the research that we conducted so far. However, from the user study we realized that, now that the data access mechanisms are in place, it is fundamental to invest more efforts on mechanisms for the visualization of the whole set of properties for the user-selected class, as well as of the class instances. Adequate visualizations will allow users to get useful insights from the displayed data, and to take advantage also of the interlinked nature of linked data, giving them the opportunity to effectively understand how to navigate throughout linked entities.

Finally, we plan to conduct further user studies to fully assess the validity of the features offered by the MDS paradigm. The study reported in this paper focused only on tasks for MDS creation and querying. Also, since our goal was primarily to assess the usability of the proposed visual paradigm, we involved regular Web users with domain-independent interests in the content to be searched and not specifically motivated to consume LD data sets. Also enabled by the extension of the platform for the integration of different LOD datasets, future studies will involve domain experts, to validate the acceptability of the proposed framework by people who have specific interest in LOD content. We will also focus on the data manipulation functions offered by the Actionable UI Components, as the conducted study did not adequately address their possible benefits.

References

Alahmari, F., Thom, J.A., Magee, L., Wong, W. (2012). Evaluating semantic browsers for consuming linked data. In *Proc. of the Twenty-Third Australasian Database Conference (ADC '12)*. Australian Computer Society, Inc., Darlinghurst, Australia, Australia, 89-98.

Ardito, C., Bottoni, P., Costabile, M.F., Desolda, G., Matera, M., Picozzi, M. (2014a). Creation and Use of Service-based Distributed Interactive Workspaces. *Journal of Visual Languages & Computing* 25(6), 717-726.

Ardito, C., Costabile, M.F., Desolda, G., Lanzilotti, R., Matera, M., Piccinno, A., Picozzi, M. (2014b). User-Driven Visual Composition of Service-Based Interactive Spaces. *Journal of Visual Languages & Computing*, 25(4), 278-296.

Ardito, C., Costabile, M.F., Desolda, G., Latzina, M., Matera, M. (2015). Making Mashups Actionable Through Elastic Design Principles. In: Díaz, P., Pipek, V., Ardito, C., Jensen, C., Aedo, I., Boden, A. (Eds.) *End-User Development - Is-EUD 2015.* LNCS, Vol. 9083, Springer Verlag, Berlin Heidelberg, 236-241.

Atzori, M., Gao, S., Mazzeo, G.M., Zaniolo, C. (2016). Answering End-User Questions, Queries and Searches on Wikipedia and its History. *IEEE Data Eng. Bull.*, 39(3), 85-96.

Atzori, M., Zaniolo, C. (2012). SWiPE: searching wikipedia by example. In *Proc. of the International World Wide Web Conference - Demo session (WWW '12)*, 309-312.

Aula, A., Jhaveri, N., Käki, M. (2005). Information search and re-access strategies of experienced web users. In *Proc. of the International conference on World Wide Web (WWW '05)*. ACM, New York, NY, USA, 583-592.

Aula, A., Khan, R.M., Guan, Z. (2010). How does search behavior change as search becomes more difficult? In *Proc. of the SIGCHI Conference on Human Factors in Computing Systems (CHI '10)*. ACM, New York, NY, USA, 35-44.

Bangor, A., Kortum, P., Miller, J. (2009). Determining what individual SUS scores mean: Adding an adjective rating scale. *Journal of usability studies*, 4(3), 114-123.

Batini, C., Cappiello, C., Francalanci, C., Maurino, A. (2009). Methodologies for data quality assessment and improvement. *ACM Comput. Surv.*, 41(3), 1-52.

Becker, C., Bizer, C. *marbles*. Retrieved from http://mes.github.io/marbles/ Last Access February 12, 2018.

Bender, M.A., Farach-Colton, M., Pemmasani, G., Skiena, S., Sumazin, P. (2005). Lowest common ancestors in trees and directed acyclic graphs. *Journal of Algorithms*, 57(2), 75-94.

Bergman, M., Giasson, F. (2008). zLinks: Semantic Framework for Invoking Contextual Linked Data. In *Proc. of the Linked Data on the Web (LDOW '08)*.

Berners-Lee, T. *Linked Data - Design Issues.* Retrieved from http://www.w3.org/DesignIssues/LinkedData.html Last Access 20th of December, 2006.

Berners-Lee, T., Chen, Y., Chilton, L., Connolly, D., Dhanaraj, R., Hollenbach, J., Lerer, A., Sheets, D. (2006). Tabulator: Exploring and analyzing linked data on the semantic web. In *Proc. of the Proceedings of the 3rd international semantic web user interaction workshop (SWUI '06).* Athens, Georgia, 2006, 159.

Berners-Lee, T., Hollenbach, J., Lu, K., Presbrey, J., Prud'hommeaux, E., Schraefel, M.M.C. (2008). Tabulator Redux: Browsing and Writing Linked Data. In *Proc. of the Linked Data on the Web (LDOW '08)*. CEUR-WS.org, 369.

Bikakis, N., Liagouris, J., Krommyda, M., Papastefanatos, G., Sellis, T. (2016). graphVizdb: A scalable platform for interactive large graph visualization. In *Proc. of the International Conference on Data Engineering (ICDE '16)*, 1342-1345.

Bikakis, N., Sellis, T.K. (2016). Exploration and Visualization in the Web of Big Linked Data: A Survey of the State of the Art. In *Proc. of the International Workshop on Linked Web Data Management (LWDM '16)*.

Bizer, C., Gauß, T. *Disco - Hyperdata Browser*. Retrieved from http://wifo5-03.informatik.uni-mannheim.de/bizer/ng4j/disco/ Last Access February 12, 2018.

Bizer, C., Heath, T., Berners-Lee, T. (2009). Linked data-the story so far. *Semantic Services, Interoperability and Web Applications: Emerging Concepts*, 205-227.

Borsci, S., Federici, S., Lauriola, M. (2009). On the dimensionality of the System Usability Scale: a test of alternative measurement models. *Cognitive processing*, 10(3), 193-197.

Braun, V., Clarke, V. (2006). Using Thematic Analysis *Psychology Qualitative Research in Psychology*, 3(2), 77-101.

Brooke, J. (1996). SUS: A quick and dirty usability scale. In: Jordan, P.W., Weerdmeester, B., Thomas, A., McLelland, I.L. (Eds.) *Usability evaluation in industry.* Taylor and Francis, 189-194.

Camarda, D.V., Mazzini, S., Antonuccio, A. (2012). LodLive, exploring the web of data. In *Proc. of the International Conference on Semantic Systems (I-SEMANTICS '12)*. ACM, New York, NY, USA, 197-200.

Cappiello, C., Di Noia, T., Marcu, B.A., Matera, M. (2016). A Quality Model for Linked Data Exploration. In *Proc. of the International Conference on Web Engineering (ICWE '16)*. Springer International Publishing, 397-404.

Cappiello, C., Matera, M., Picozzi, M. (2015). A UI-Centric Approach for the End-User Development of Multidevice Mashups. *ACM Transaction Web*, 9(3), 1-40.

Casati, F. (2011). How End-User Development Will Save Composition Technologies from Their Continuing Failures. In: Costabile, M.F., Dittrich, Y., Fischer, G., Piccinno, A. (Eds.) *International Symposium on End-User Development - Is-EUD 2011.* Lecture Notes in Computer Science, Vol. 6654, Springer Berlin Heidelberg, 4-6.

Cheng, G., Qu, Y. (2009). Searching linked objects with falcons: Approach, implementation and evaluation. *International Journal on Semantic Web and Information Systems*, 5(3), 49-70.

d'Aquin, M., Motta, E. (2011). Watson, more than a Semantic Web search engine. *Semant. web*, 2(1), 55-63.

Dadzie, A.-S., Rowe, M. (2011). Approaches to visualising linked data: a survey. *Semant. web*, 2(2), 89-124.

Dadzie, A.-S., Rowe, M., Petrelli, D. (2011). Hide the Stack: Toward Usable Linked Data. In *Proc. of the The Semantic Web: Research and Applications (ESWC '11)*. Springer Berlin Heidelberg, 93-107.

Daniel, F., Matera, M. 2014. *Mashups - Concepts, Models and Architectures*. Springer.

Desolda, G., Ardito, C., Costabile, M.F., Matera, M. (2017a). End-user composition of interactive applications through actionable UI components. *Journal of Visual Languages & Computing*, 42(October 2017), 46-59.

Desolda, G., Ardito, C., Jetter, H.-C., Lanzilotti, R. (2019). Exploring spatially-aware cross-device interaction techniques for mobile collaborative sensemaking. *International Journal of Human-Computer Studies*, Volume 122(February 2019), 1-20.

Desolda, G., Ardito, C., Matera, M. (2015). EFESTO: A platform for the End-User Development of Interactive Workspaces for Data Exploration. In: Daniel, F., Pautasso, C. (Eds.) *Rapid Mashup Development Tools - Rapid Mashup Challenge in ICWE 2015.* Communications in Computer and Information Science, Vol. 591, Springer Verlag, Berlin Heidelberg, 63 - 81.

Desolda, G., Ardito, C., Matera, M. (2016). EFESTO: A Platform for the End-User Development of Interactive Workspaces for Data Exploration. In: Daniel, F., Pautasso, C. (Eds.) *Rapid Mashup Development Tools - ICWE '15*. CCIS, Vol. 591, Springer, 63-81.

Desolda, G., Ardito, C., Matera, M. (2017b). Empowering end users to customize their smart environments: model, composition paradigms and domain-specific tools. *ACM Transactions on Computer-Human Interaction*, 24(2), Article 12 (April 2017), 52 pages.

Ding, L., Finin, T., Joshi, A., Pan, R., Cost, R.S., Peng, Y., Reddivari, P., Doshi, V., Sachs, J. (2004). Swoogle: a search and metadata engine for the semantic web. In *Proc. of the International Conference on Information and Knowledge Management (CIKM '04)*. ACM, 652-659.

Djidjev, H.N., Pantziou, G.E., Zaroliagis, C.D. (1991). Computing shortest paths and distances in planar graphs. In *Proc. of the international colloquium on Automata, languages and programming (ICALP '91)*. Springer-Verlag New York, Inc., New York, NY, USA, 327-338.

Franz, T., Koch, J., Dividino, R.Q., Staab, S. (2010). LENA-TR: Browsing Linked Open Data Along Knowledge-Aspects. In *Proc. of the AAAI spring symposium: Linked data meets artificial intelligence (AAAI '10)*.

Görlitz, O., Staab, S. (2011). Federated Data Management and Query Optimization for Linked Open Data. In: Vakali, A., Jain, L.C. (Eds.) *New Directions in Web Data Management 1*. Springer Berlin Heidelberg, Berlin, Heidelberg, 109-137.

Graziano, A.M., Raulin, M.L. 2012. *Research Methods: A Process of Inquiry (8th Edition)*. Pearson.

Hart, S.G. (2006). Nasa-Task Load Index (NASA-TLX); 20 Years Later. *Proceedings of the Human Factors and Ergonomics Society Annual Meeting*, 50(9), 904-908.

Hart, S.G., Staveland, L.E. (1988). Development of NASA-TLX (Task Load Index): Results of empirical and theoretical research. *Advances in psychology*, 52, 139-183.

Harth, A. (2010). VisiNav: A system for visual search and navigation on web data. *Web Semantics: Science, Services and Agents on the World Wide Web*, 8(4), 348-354.

Hartig, O., Bizer, C., Freytag, J.-C. (2009). Executing SPARQL Queries over the Web of Linked Data. In *Proc. of the International Semantic Web Conference (ISWC '09)*. Springer Berlin Heidelberg, 293-309.

Hastrup, T., Cyganiak, R., Bojars, U. (2009). Browsing Linked Data with Fenfire. In *Proc. of the Linked Data on the Web workshop, in conjunction with WWW 2008 conference (LDOW2008)*.

Heim, P., Lohmann, S., Stegemann, T. (2010). Interactive Relationship Discovery via the Semantic Web. In *Proc. of the Extended Semantic Web Conference (ESWC '10)*. Springer Berlin Heidelberg, 303-317.

Herman, I., Melançon, G., Marshall, M.S. (2000). Graph visualization and navigation in information visualization: A survey. *IEEE Transactions on visualization and computer graphics*, 6(1), 24-43.

Hoang, D.D., Paik, H.-y., Benatallah, B. (2010). An analysis of spreadsheet-based services mashup. In *Proc. of the Conference on Database Technologies (ADC '10)*. Australian Computer Society, Inc., Darlinghurst, Australia, Australia, 141-150.

Hogan, A., Umbrich, J., Harth, A., Cyganiak, R., Polleres, A., Decker, S. (2012). An empirical survey of Linked Data conformance. *Web Semantics: Science, Services and Agents on the World Wide Web*, 14(Supplement C), 14-44.

Huynh, D., Mazzocchi, S., Karger, D. (2007). Piggy Bank: Experience the Semantic Web inside your web browser. *Web Semantics: Science, Services and Agents on the World Wide Web*, 5(1), 16-27.

Joachims, T. (2002). Optimizing search engines using clickthrough data. In *Proc. of the The International Conference on Knowledge Discovery and Data Mining (KDD '02)*. ACM, New York, NY, USA, 133-142.

Juristo, N., Moreno, A.M. 2013. *Basics of software engineering experimentation*. Springer Science & Business Media.

Jusevičius, M. *AtomGraph*. Retrieved from https://github.com/AtomGraph/Web-Client Last Access February 12, 2018.

Kelly, D., Teevan, J. (2003). Implicit feedback for inferring user preference: a bibliography. *SIGIR Forum*, 37(2), 18-28.

Khalili, A., van den Besselaar, P., de Graaf, K.A. (2018). FERASAT: A Serendipity-Fostering Faceted Browser for Linked Data. In *Proc. of the European Semantic Web Conference (ESWC '18)*. Springer International Publishing, 351-366.

Kirsh, D. (2000). A Few Thoughts on Cognitive Overload. *Intellectica*, 1(30), 19-51.

Lee, Y.W., Strong, D.M., Kahn, B.K., Wang, R.Y. (2002). AIMQ: a methodology for information quality assessment. *Information & Management*, 40(2), 133-146.

Lerner, J., Self, T. *Objectviewer*. Retrieved from http://projects.semwebcentral.org/projects/objectviewer/ Last Access February 12, 2018.

Lewis, J., Sauro, J. (2009). The Factor Structure of the System Usability Scale. In: Kurosu, M. Ed. *Human Centered Design - HCD 2009*. Lecture Notes in Computer Science, Vol. 5619, Springer Berlin Heidelberg, 94-103.

Marie, N., Gandon, F. (2014). Survey of linked data based exploration systems. In *Proc. of the International Conference on Intelligent Exploration of Semantic Data (IESD '14)*. CEUR-WS.org, Aachen, Germany, Germany, 66-77.

Miller, G.A. (1995). WordNet: a lexical database for English. *Communications of the ACM*, 38(11), 39-41.

Namoun, A., Nestler, T., Angeli, A.D. (2010). Service Composition for Non-programmers: Prospects, Problems, and Design Recommendations. In *Proc. of the IEEE European Conference on Web Services (ECOWS '10)*. IEEE Computer Society, Washington, DC, USA, 123-130.

Napoleoni, G.L., Pazienza, M.T., Turbati, A. (2014). HORUS: A configurable reasoner for dynamic ontology management. In *Proc. of the Sixth International Conference on Advanced Cognitive Technologies and Applications (COGNITIVE '14)*, 66-71.

NetApplications.com. *NetMarketShare*. Retrieved from https://netmarketshare.com/ Last Access 22nd of July, 2019.

Otter, M., Johnson, H. (2000). Lost in hyperspace: metrics and mental models. *Interacting with computers*, 13(1), 1-40.

Paredes‐Valverde, M.A., Alor‐Hernández, G., Rodríguez‐González, A., Valencia‐García, R., Jiménez‐Domingo, E. (2015). A systematic review of tools, languages, and methodologies for mashup development. *Software: Practice and Experience*, 45(3), 365-397.

Peña, O., Aguilera, U., López-de-Ipiña, D. (2014). Linked open data visualization revisited: a survey. *Semantic Web Journal*.

Pérez, J., Arenas, M., Gutierrez, C. (2009). Semantics and complexity of SPARQL. *ACM Transactions on Database Systems (TODS)*, 34(3), 16.

Pietriga, E. *IsaViz*. Retrieved from https://www.w3.org/2001/11/IsaViz/ Last Access February 14, 2018.

Pipino, L.L., Lee, Y.W., Wang, R.Y. (2002). Data quality assessment. *Commun. ACM*, 45(4), 211-218.

Prensky, M. (2001). Digital natives, digital immigrants part 1. *On the horizon*, 9(5), 1-6.

Rocchio, J.J. (1971). Relevance feedback in information retrieval. In: Salton, G. Ed. *The Smart retrieval system - experiments in automatic document processing.* Englewood Cliffs, NJ: Prentice-Hall, 313--323.

Shen, X., Tan, B., Zhai, C. (2005). Context-sensitive information retrieval using implicit feedback. In *Proc. of the Conference on Research and Development in Information Retrieval (SIGIR '05)*. ACM, New York, NY, USA, 43-50.

Stuhr, M., Roman, D., Norheim, D. (2010). LODWheel - JavaScript-based visualization of RDF data. In *Proc. of the International Conference on Consuming Linked Data (COLD '11)*. CEUR-WS.org, 73-84.

Sugiyama, K., Hatano, K., Yoshikawa, M. (2004). Adaptive web search based on user profile constructed without any effort from users. In *Proc. of the Conference on World Wide Web (WWW '04)*. ACM, New York, NY, USA, 675-684.

Thalhammer, A., Stadtmüller, S. (2015). SUMMA: A Common API for Linked Data Entity Summaries. In: Cimiano, P., Frasincar, F., Houben, G.-J., Schwabe, D. (Eds.) *Engineering the Web in the Big Data Era - ICWE 2015.* Springer International Publishing, Cham, 430-446.

Tummarello, G., Cyganiak, R., Catasta, M., Danielczyk, S., Delbru, R., Decker, S. (2010). Sig.ma: Live views on the Web of Data. *Web Semantics: Science, Services and Agents on the World Wide Web*, 8(4), 355-364.

Tummarello, G., Delbru, R., Oren, E. (2007). Sindice.com: Weaving the Open Linked Data. In *Proc. of the Conference on Semantic Web Conference (ISWC '07)*. Springer Berlin Heidelberg, 552-565.

Virtuoso. *LOD Cloud Cache* Retrieved from http://lod.openlinksw.com/fct/ Last Access Februay 12, 2018.

Wang, R.Y. (1998). A product perspective on total data quality management. *Commun. ACM*, 41(2), 58-65.

Wang, R.Y., Strong, D.M. (1996). Beyond accuracy: what data quality means to data consumers. *J. Manage. Inf. Syst.*, 12(4), 5-33.

White, R.W. 2016. *Interactions with search systems*. Cambridge University Press.

White, R.W., Jose, J.M., van Rijsbergen, C.J., Ruthven, I. (2004). A Simulated Study of Implicit Feedback Models. In *Proc. of the Conference on IR Research (ECIR '04)*. Springer Berlin Heidelberg, 311-326.

Yu, J., Benatallah, B., Casati, F., Daniel, F. (2008). Understanding Mashup Development. *IEEE Internet Computing*, 12(5), 44-52.

Zang, N., Rosson, M.B. (2008). What's in a mashup? And why? Studying the perceptions of web-active end users. In *Proc. of the IEEE Symposium on Visual Languages and Human-Centric Computing (VLHCC '08)*. IEEE Computer Society, Washington, DC, USA, 31-38.

Zaveri, A., Rula, A., Maurino, A., Pietrobon, R., Lehmann, J., Auer, S. (2016). Quality assessment for linked data: A survey. *Semantic Web*, 7(1), 63-93.

Zhai, C., Lafferty, J. (2001). Model-based feedback in the KL-divergence retrieval model. In *Proc. of the Tenth International Conference on Information and Knowledge Management (CIKM '01)*, 403-410.

Zobel, J., Dart, P. (1996). Phonetic string matching: lessons from information retrieval. In *Proc. of the ACM SIGIR conference on Research and development in information retrieval (SIGIR '96)*. ACM, New York, NY, USA, 166-172.