RDF Graph Visualization Tools: a Survey

Francesco Antoniazzi
INFN CNAF and University of Bologna
Bologna, Italy
francesco.antoniazzi@unibo.it

Fabio Viola
University of Bologna
Bologna, Italy
fabio.viola@unibo.it

Abstract—**Semantic Web technologies are increasingly being used for the development of Future Internet applications, mainly due to the impressive growth of the Internet of Things research area. This spread pushes for effective and efficient ways to visualize the content of RDF ontologies and knowledge bases. Several strategies can be adopted to visualize semantic data and one of this consists in exploiting the graph representation intrinsic in the RDF model. In this paper, we propose a survey of the main tools for the graphical visualization of triples (being them terminological or assertional statements) exploiting a graph representation.**

I. INTRODUCTION

The Semantic Web [1] movement was born to transform the Web from a repository of human-readable information, to a world wide network of machine-understandable data. To achieve the scope, multiple protocols were introduced: RDF (Resource Description Framework) [2] allows to represent all the information as a set of triples (i.e., *subject*, *predicate*, *object*) where resources are univocally identified through URIs (Uniform Resource Identifiers). Ontologies represented according to RDFS (RDF Schema) [3] and OWL (Web Ontology Language) [4] bind meanings to RDF terms (with a set of rules expressed through RDF). Finally, SPARQL Query [5] and Update [6] languages allow to respectively retrieve data from the Knowledge Base (KB) and update it.

Semantic Web technologies are gaining momentum, due to the wide spread of two strongly linked research areas: context-aware computing [7] and the Internet of Things (IoT) [8]. Context-aware computing is aimed at developing applications able to adapt to changes in the environment and often exploits semantics to model the context with a high expressive power. The IoT is instead a world-wide network of interconnected and uniquely addressable objects, based on given communication protocols [9]. It is characterized by heterogeneity of the involved devices and by a multitude of protocols born with different aims [10]. For this reason, IoT applications are almost always compared to vertical silos [11] where the interoperability is a challenging task. In this scenario, Semantic Web technologies are often considered as interoperability enablers that allow to bridge different applications by means of a semantic representation of the involved entities.

Then, also thanks to context-aware computing and IoT, the number of applications exploiting Semantic Web technologies is costantly increasing, as demonstrated by LOV (Linked Open Vocabulary) [12], an innovative observatory of the semantic vocabularies ecosystem. Vandenbussche et al. in [13] describe the impressive growth of the repository: less than 100 ontologies in March 2011, more than 500 as of June 2015 (as of September 2018, the number is growth up to 650). Given the high spread of semantic applications, it is essential for both developers and users to have efficient tools to visualize ontologies as well as to explore semantic KBs.

Before going further, it is important to clarify two of the key concepts that we will rely on in the rest of the paper and that we already mentioned: ontology and knowledge base. Among the multiple definitions of ontology available in literature, we rely on the one provided by Noy et al. in [14]: *an ontology is a formal explicit description of concepts in a domain of discourse (classes), properties of each concept describing various features and attributes of the concept and restrictions*. Since an OWL/RDFS ontology is represented as a set of RDF triples, we refer to these triples as terminological statements (or T-Boxes). All the triples defining and specifying class instances are instead known in literature as assertional statements (or A-Boxes). A-Boxes and T-Boxes form the knowledge base.

In the rest of the paper we will consider the visualization of data from two (possibly overlapping) points of view: visualization of ontologies and visualization of RDF triples represented according to a given ontology. While the first is aimed at grasping the relevant concepts of an application domain, the second is mostly aimed at a practical inspection of data (e.g., for debug purposes). In both cases it is important to be able to dominate the complexity of a very high amount of data by means of proper visualization strategies and effective filtering mechanisms.

In this paper, we survey the available software aimed at providing a graphical visualization of semantic knowledge bases containing either terminological or assertional data. We focus only on the tools adopting the graph metaphor to represent data (other possible graphical representation methods are briefly presented in Section III).

The rest of the paper is organized as follows: in Section II, an overview of the existing surveys on the visualization of semantic knowledge bases is proposed. Section III introduces the background for this work and motivates the need for a new survey in this research area. Section IV presents the main tools for the visualization of Semantic Web datasets. For tools still actively developed and/or widely used, this paper also proposes examples based on information retrieved from DBpedia. In Section V, all the features of the analyzed tools

are summarized. Eventually, in Section VI, conclusion are drawn.

II. RELATED WORK

The graphical representation of information is a topic that has been addressed in various ways by research, since it is a matter of algorithms and user interface theory simultaneously, and, as pointed out in [15], nowadays also of Big Data.

Literature is rich in surveys and explorations of visualization methods for information represented according to Semantic Web technologies. The Semantic Web, in particular, relies on the graph theories for what concerns the viewing of knowledge, which is a topic well covered by surveys like [16]. However, the actual investigation of the Semantic content, most of the time, is centered on the usage and integration of ontologies in applications, like in [17] and [18]. Semantic content visualization, on top of them, is a particular research topic that tries to overcome the difficulties that arise at development time when direct usage and integration is required.

In their survey [19], Katifori et al. point out the need for studies in the visualization field at the various levels of usage of semantics: design, organization and navigation into resources. Similarly is done by Mutton et al. in [20], where the complexity of graph drawing is examined keeping in mind that there is implicit information hidden in a graph topology that cannot be easily observed from a plain textual representation. Another work, in this sense, is [21] by Wiens et al., in which the ontology view is split into three levels of understanding: global, filtered and more specific but fully detailed.

The main discussion of those works, however, is the ontology: a complete outline of methods is given, but the full content of the RDF knowledge base is out of the scope.

In [22], Bikakis et al. propose a description of the major requirements and challenges that should be addressed by modern exploration and visualization systems for Linked Data and propose a list of the state of the art approaches. Differently from the present work, they propose a comparison of tools offering different visualization types (i.e., bubble charts, charts, circles, graphs, maps, pies, parallel coordinates, scatter plots, streamgraphs, treemaps, timelines, trees) and the focus is mainly on Big Data applications. Akrivi et al. in [23] propose a comparison of four visualization methods (i.e., Ontoviz, Jambalaya, TGVizTab and Class Browser) characterized by different approaches. In [24], an interesting study is performed with the aim to explore pros and cons of the available tools on a real use case.

Along with tools, plain text formats for the Semantic Web are also subject to evolution over time. Recently, for instance, JSON-LD (standing for *JSON for Linked Data*) was introduced in the Semantic environment [25] as a new format graph-equivalent to represent part of a resource graph. The official JSON-LD W3C website [26], among all the material, offers also the opportunity to compare a few different visualizations obtained from any JSON-LD formatted file.

The Linked Data (LD) concept, described in [27], is a more recent research direction on Semantic Web applied to web resources. LD are constantly growing in size and new data repositories, that come also from governmental sources, which demonstrates their increasing importance. The final goal is to have LD both machine and human understandable, and therefore, they are an important test bench for any RDF graph visual exploration technique. Peña et al. in their work [28] intent the description of the available methods to deal with LD access and, in particular, the visualization.

III. BACKGROUND AND MOTIVATION

Accessing and understanding the content of a database is hardly ever a negligible task for programmers. When data is stored in a relational database, the views are obtained by transforming into a table the output of a query written in one of the various flavors of the SQL language. Smart-written queries on equally smart-built databases can efficiently perform a lot of calculations over data, as well as outline special and complex relationships even between apparently distant entries. Therefore the *know-your-data* principle, typical of Data Mining and Big Data theory, is in fact a more general and solid base from which to start any data-related implementation, though implying sometimes great study effort from the developer in the initial phase of software creation [29]. It is, as a matter of fact, common knowledge that frequently programmers have to spend more time in organizing and reformatting their data more than in the actual programming logic.

As of September 2018, this Survey aims to show to the readers the tools currently available to exploit in an effective and possibly easy way the Semantic Web. Therefore, the focus of the paper is the observation of their overall usability and capabilities in the complex task of analyzing and understanding the content of a knowledge base. For this reason, both the ontological statements (T-Boxes) and the instances (A-Boxes) have to be included in the discussion, which is a missing point in the previous similar works, as we outlined in the Related Works Section.

A. Representing data in the Semantic Web

The appearance of the Semantic Web in the panorama of information technology gave ways more than a simple new tool to explore the Web, but a new interpretation of the resources available on the Internet. Through the SPARQL language and the Resource Description Framework, the Internet network is considered as a whole a special database whose resources are interconnected in a labeled directed graph. The main idea of the Semantic Web, therefore, is to exploit the concepts of *Uniform Resource Identifier*, explained in [1], to bind resources through triple-based statements (i.e., the already mentioned subject-predicate-object triple). Any connection is in that fashion not a simple reference as hypertext linking is, but contains in addition the information given by the inner content of the resources. Then, according to W3C recommendations [30], the Semantic Web in the end takes the form of a graph, where both the contents and the statements contribute to the overall meaning. Table I shows what kind of

TABLE I. Semantic triple composition with an example

Subject	Predicate	Object	Example
URI	URI	URI	ns:Francesco foaf:knows ns:Fabio
		literal	ns:Francesco foaf:firstName "Francesco"
		blank node	ns:Francesco foaf:knows _:X
blank node	URI	URI	_:X foaf:knows ns:Francesco
		literal	_:X foaf:firstName "Francesco"
		blank node	_:X foaf:knows _:Y

TABLE II. Listing 2 SPARQL output table

?p	?count
<http://purl.org/dc/terms/subject>	13
rdfs:seeAlso	101
<http://purl.org/linguistics/gold/hypernym>	2
DBpedia2:precededBy	42
DBpedia2:author	93
DBpedia2:publisher	21
DBpedia2:books	101
foaf:isPrimaryTopicOf	1

```
PREFIX rdf: <http://www.w3.org/1999/02/..>
PREFIX dbp: <http://dbpedia.org/resource/>
SELECT (count(?o) as ?count)
WHERE {
    dbp:The_Lord_of_the_Rings rdf:type ?o
}
```

Listing 1. SPARQL query to all classes parent of the resource dpb:The_Lord_of_the_Rings

item can be subject, object or predicate in a semantic graph, with a reader-friendly example based on `foaf` ontology [31].

A few considerations are needed, however, when we start discussing about the possibility to store information in a semantic graph. In fact, a few critical points are present, and have to be highlighted. First of all, as it is depicted in [32], without regulations, the semantic graph is doomed to chaos, i.e. to an unpredictable information taxonomy. The solution to this issue is the ontological description of knowledge, that consists in the formal definition of all the classes, relationships and statements that can be present in the graph. Once the programmers agree on the ontology, there is no uncertainty on how the data is organized. All the information needed to query the graph is stored in an OWL file, standing for Web Ontology Language. OWL is, according to W3C, a "semantic markup language for publishing and sharing ontologies", and has been widely used to define all sort of ontologies and vocabularies (which are smaller ontologies): an interesting repository, in this field of study, is the Linked Open Vocabularies website which has been already presented in the Introduction.

A second critical point is connected to the dimension of the graph, which can be considerable not only when we are discussing about web-located knowledge bases like DBpedia, or the Internet itself, but also in smaller applications exploiting RDF and SPARQL. To make an example, let us perform the query of Listing 1 to DBpedia.

As of September 2018 the output of the query, which is a simple request to count all the classes that "The Lord of the Rings" resource belongs to, is equal to 25. Clearly, far from a naive and optimistic expectation of a few outputs similar to `:Book`, `:Novel` and so on. Listing the actual values consequently results in a 25-rows table that outlines the evidence of the hidden complexity in the results analysis,

even in simple situations. The complexity grows considerably if we proceed querying on the following level (Listing 2).

```
PREFIX rdf: <http://www.w3.org/1999/02/..>
PREFIX dbp: <http://dbpedia.org/resource/>
SELECT ?p (count(?t) as ?count)
WHERE {
    dbp:The_Lord_of_the_Rings ?p ?o.
    ?o rdf:type ?t
    FILTER (?p != rdf:type)
}
```

Listing 2. SPARQL query to every link (except `rdf:type`) with dpb:The_Lord_of_the_Rings as origin, and the destination's class

The query available in Listing 2 with the variable `?count` outputs the number of links of type `?p` outgoing from the resource `dbp:The_Lord_of_the_Rings`, except from `rdf:type` links, that can be viewed by performing the query in Listing 1.

The direct outcome after running the queries in the Listings 1 and 2 is given by the possibility to observe the available variability of results and to look for their meanings. The output of those simple SPARQL queries highlights, for instance, that the "Lord of the Rings" resource is individual of at least 25 classes which we expect to have a specific meaning and a description of their own. Going further with the latter query, moreover, the number of elements connected to the resource is even more increasing, both as connectivity spread, and in diversity of ontological classes involved. In general the description of all those resources can be as pragmatic as an algorithm, or philosophical, or mathematical. However, it is clear that without the Semantic Web it would be hardly achievable to obtain such a multi-layered description of a resource, apart from using natural language. In fact, when it comes to exploit the tools of Semantic Web, a frequent

feeling is that it is not possible to reuse previously available data, because it would imply to understand completely all the resources, all the classes, and all the ontologies that are standing behind. According to [33], expressiveness, in this situation, is a bottleneck for Semantic Web.

This is where visualization tools for the semantic graph come to help: they provide a step by step approach to the knowledge base that, together with filtering techniques, and the possibility to see the contents, are useful to go through the relevant concepts.

B. Possible visualizations

As we said in the previous Subsection, the most frequent way to produce a view of a database is the tabular representation. This is a possible solution also for queries made in SPARQL language to RDF triple stores like Blazegraph, Fuseki and Virtuoso. The view of a SPARQL `SELECT` is a direct consequence of the number of variables concerned by the inquire: i.e., the number of variables in the `SELECT` clause is the same as the number of columns contained in the results. To be more precise, to obtain the column number either (i) it is necessary to count the variables queued after the `SELECT` keyword, like in listings 1 and 2; or (ii), variables have to be obtained from the `WHERE` clause, as in the `SELECT *` `WHERE {...}` case.

The drawbacks with table views are, unfortunately, already quite visible when the number of rows reaches as little as few dozen entries. Aside from the fact that there is not a group view of the overall query result, the table often is required to contain more than one row for the same conceptual entity. This happens for instance when a resource is connected to another via more than one predicate, or when it is connected to different objects, through the same predicate.

In such situations, the result table can contain not only plenty of lines with the same meaning disturbing the overall understanding of the query output, but also, as already said, a high number of columns. Moreover, some entries in the table can also be empty, as an effect of `OPTIONAL` statements in the query. Detecting particular cases, in sparse and large tables, becomes a time-consuming and error-prone task in those situations. On the other hand, a few workarounds are available in SPARQL language to crunch into a single line the occurrence of multiple table lines for a single concept, but they usually imply slowing down the performances, and have the effect to concatenate the values into strings. That is, we lose the possibility to check if they are represented as IRIs, literals, or blank nodes.

The multi-table approach is a graph visualization technique that tries to address the problem of having limited control over the complete data table. Let's consider a query selecting all triples in the RDF store: `SELECT * WHERE {?a ?b ?c}`, and let's suppose that in the store only 5 distinct resources might correspond to the `?b` variable. With this background a full-table approach would return an $n = 3$ column table, where n is the number of variables. Instead a multi-table approach would outcome with 5 smaller tables,

one for each one of the `?b` resources, each of them built up of $n - 1 = 2$ columns. An interesting work about the complexity of translation from SPARQL to other languages, included table view, was provided by Chebotko et al. in [34]: among all the contributions, this paper perfectly shows the complexity of a multi-table approach.

Finally, last but not least, the RDF knowledge base representation can be performed through a labeled graph visualization. Although the RDF concept is defined for directed graphs, in most of the cases the label is sufficient to get at view time the direction of the connection. This allows the usage of algorithms for undirected graphs. Nevertheless, the drawbacks of this approach are also related to the knowledge base dimension, as the understanding of contents is tightly bound to the possibility of identify paths and node types easily and effectively. In the next Subsection a few techniques for graph drawing available in literature are presented.

C. Graph drawing algorithms

There is a complex relationship between the domain of the semantic application, the tool that is being used, and the algorithm that is implemented to visualize the graph. To make an example, let's consider a knowledge base in which information about some people is stored. If the application working on the knowledge base is not interested in literal terms, the sight of the graph would be effectively simplified and clearified by just removing all the links towards literal terms, e.g. names, surnames and birth dates.

In other RDF triple stores more than one unique ontology may have been used to define resources, exploiting for instance simultaneously the `foaf` ontology and the Dublin-Core ontology (DC) [35]. If an application is interested only in the `foaf`-related connections, and in a small part of the DC's, there would be no use in trying to represent everything.

A full description of all the algorithms available for graph drawing is out of the scope of this paper. In this Section, nevertheless, a brief overview of a few works available in literature is given, before proceeding in Section IV to the analysis of the tools.

A complete theoretical overview of the main algorithm logic available to draw graphs is given by Kobourov in [36]. Spring algorithms and their variations for instance are explained: they usually aim to reproduce an aesthetically pleasant view, even if their best performance is obtained in most of the cases when the graph has less than 40 vertices. However, as it has been said, the semantic graph is definitely a large graph, or very large, and for this reason it demands particular approaches that imply multiple scale algorithms. Nodes organization is not necessarily done on a plane: possible alternatives are to dispose them on a sphere or other geometrical objects. In [37] more than one plane is used, which can be a technique also to represent the evolution of data over time. How to show in an effective way dynamic evolution of contents in a graph is also the topic of survey [38] by Beck et al.

IV. GRAPH VISUALIZATION TOOLS

This Section proposes a detailed analysis of the main tools for the visualization of RDF knowledge bases and ontologies. We focus on the tools providing a graph visualization of RDF statements. The tools presented in this Section are reported in alphabetical order.

A. CytoScape

Cytoscape [39] is a tool for network data integration, analysis and visualization. Support to Semantic Web technologies is provided by a set of extensions hosted on CytoScape's App Store, such as General SPARQL, SemScape and Vital AI Graph Visualization. General SPARQL allows to navigate semantic web KBs through an extensible set of pre-defined queries. The plugin is pre-configured to retrieve and visualize data from public endpoints (e.g., Reactome, Uniprot, HGNC, NCBI Taxonomy, Chembl). SemScape supports the interaction with remote SPARQL endpoints by means of SPARQL queries. In this way, CytoScape can be employed to visualize the results of a query. Vital AI Graph Visualization is not limited to semantic databases, but provides access also to SQL and NoSQL databases as well as Apache Hadoop instances. To the best of authors' knowledge, this tool only allows the visualization of data compatible with the BioPAX format.

B. Fenfire

Fenfire [40] was a tool for the visualization and editing of RDF graphs aimed at an interactive exploration of the graph. Authors face the problem of scalability by limiting the exploration of the graph to one thing at a time. The visualization in facts, diplays only one central node and its surroundings. The central node, at the beginning of the exploration is selected exploiting the `foaf:primaryTopic` property (if present), otherwise is selected by the user. The nodes surrounding the central one (named *focus*) are placed on the plane according to a simple strategy: on the left, all the nodes being subjects of the statements linking to the focus. On the right, those being objects of the statements. Development of Fenfire stopped in 2008.

C. Gephi

Gephi [41] is a very powerful tool designed to represent not only semantic graphs, but every kind of graph or network. Support to RDF graphs is provided by two external plugins, VirtuosoImporter and SemanticWebImport (this one developed by INRIA). Gephi is able to retrieve data from SPARQL endpoints (through REST calls) as well as to load RDF files. Gephi supports filtering the KB through SPARQL queries. The look of the graph visualized by Gephi is fully customizable, in terms of colors and layouts; moreover the tool supports grouping similar nodes and this helps achieving better results when dealing with very complex graphs. As regard exporting the graph, Gephi is the tool that supports the highest number of file formats for exporting the graph. Among these, it is worth mentioning `csv`, `pdf` and `svg`.

In Figure 1 we can see a view of the graph that Gephi is able to retrieve from DBpedia by using the SPARQL `CONSTRUCT` available in Listing 3. The tool performs the representation very quickly, and implements various possible algorithms to build the graph. Unfortunately, as it can be seen, it is quite difficult t o g et t he o verall i dea o f t he c omposition. Although there is the possibility to add the labels of nodes and edges, the output is not reader-friendly, and the research in it is a rather impossible task. A practical example can be observed also in Fig. 2, where we highlighted the nodes related to the novel "The Black Cauldron" by L. Alexander. Eventually, a number of statistical functions can be applied to the network, like the *Network Diameter*, the *Density* and the *Average Path Lenght*: the only problem is that they have, as for the Authors' knowledge, very limited use when applied to a Semantic Graph.
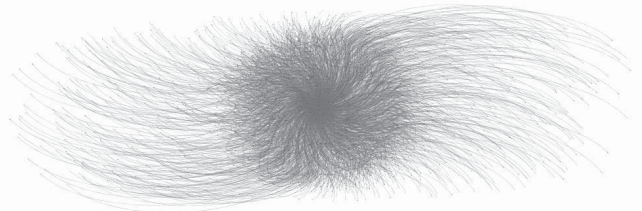

Fig. 1. Gephi [41] is capable to query DBpedia and show the result graph. The Figure is the output of `CONSTRUCT` in Listing 3 (see Appendix). According to Gephi's logger, the triples represented in this graph are 6529
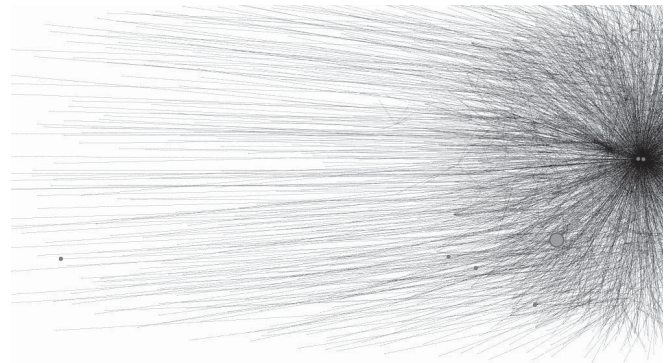

Fig. 2. With Gephi [41] some nodes can be highlighted, to help the user to go through the knowledge base. When the number of edges and nodes is high, however, it's not easy to outline the information. The nodes in red are related to L. Alexander's novel "The Black Cauldron"

D. GLOW

Glow [42] is a visualization plugin for the ontology editor Protégé. Force-directed, Node-link tree and Inverted radial tree are the three layout algorithms provided by GLOW. The items are arranged automatically with every layout, and cannot be moved. The tool is able to represent a set of ontologies and optionally their individuals. To the best of authors' knowledge,

this tool is not developed anymore. No information about the license could be found.

E. IsaViz

IsaViz [43] is a 2.5D tool for the visualization of RDF graphs originally developed by E. Pietriga (INRIA) in collaboration with Xerox Research Centre Europe. IsaViz, as the name suggests, is based on GraphViz [44] and allows importing and exporting from/to RDF/XML, Notation 3 and N-Triple files. The result of the visualization can be also exported as a `png` or `svg` file. In the *Graph view* it is possible to select resources and access a textual list of properties (this view is named *Property Browser*). A third view is named *Radar* and presents an overview of the graph, since the graph view may contain only a portion of it. Finally, it is worth mentioning the search tool provided by IsaViz, whose results are highlighted one by one in the graph view. Unfortunately, the last development version of this tool dates back to 2007.

F. Jambalaya

Jambalaya [45] is a Protégé plugin for the visualization of ontologies. Jambalaya is characterized by the integration of the SHriMP (Simple Hierarchical Multi-Perspective) [46] visualization technique, designed to improve the user experience in browsing, exploring, modelling and interacting with complex information spaces. This technique, originally born to help programmers understanding software, was applied to Protégé to build a powerful visualization of classes and relationships. The tool proposes a nested graph view and the nested interchangeable views. Nesting is used to represent the sub-class relationships among classes as well as the link between classes and their instances (different colors allow to distinguish between classes and instances). Jambalaya also provide an easy way to search for items in the ontology.

Despite being an interesting tool developed with support from the National Center for Biomedical Ontology (NCBO), Jambalaya is not developed anymore.

G. LOD Live

LOD Live [47] is a web-based tool for the incremental navigation of Linked Data available on a selected SPARQL Endpoint (e.g., DBpedia). Endpoints can be configured through a JSON map of their parameters, similarly to what happens in Tarsier [37]. The purpose of this tool is to demonstrate that the powerful Semantic Web standards are also easy to understand; the aim is to foster the spread of Big Data. Every resource drawn by LOD Live is surrounded by a set of symbols representing different kinds of relationship (e.g., direct relations, group of direct relations, inverse relations and group of inverse relations). The incremental navigation, joined to the ability of the tool to group properties allows to draw a very clean graph. No support for statistics or advanced filtering (e.g., based on SPARQL) is provided. To the best of our knowledge, directly exporting the graph is not possible. In Figure 3 it is shown how LOD Live performs a similar task as the one in Figure 2: exploring data is easier, but there is no way to perform requests like the one in Listing 3.
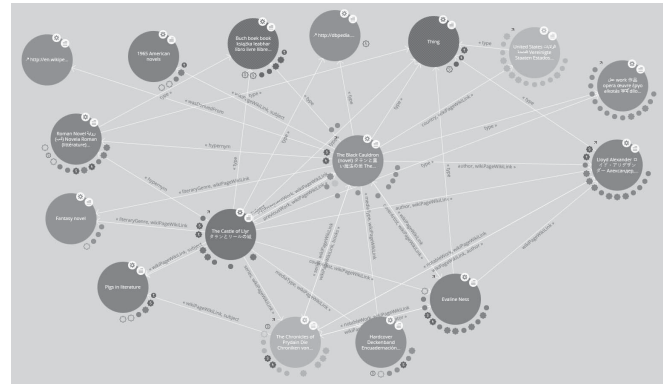

Fig. 3. To use LOD Live [47] a resource must be fixed. Then, the knowledge related to the resource can be expanded as shown. Like in Figure 2, the example here is based also on L. Alexander's novel "The Black Cauldron"

H. Ontograf

Ontograf [48] is one of the visualization tools provided by the famous ontology editor Protégé. The tool allows to build a custom visualization of the ontologies loaded in Protégé by iteratively enabling or disabling the desired classes. Ontograf proposes a grid layout (with classes sorted in alphabetical order), a spring layout and a (vertical or horizontal) tree layout. Individuals of a class can be visualized in its tooltip, but this is uncomfortable when dealing with a high number of assertional statements. Ontograf allows to export the visualized graph as a png, jpeg, gif or dot file. This tool exploits the layout library provided by Jambalaya.

Fig. 4 shows a graph created with OntoGraf using the DBpedia ontology. Classes `work` and `written work` were initially selected. Then, a double click on the latter allowed to expand it and visualize all the subclasses (solid blue line), and all the classes linked to it by means of an object property (dashed lines).

The last version of Ontograf dates back to April 2010, but is still included in the last stable version of Protégé (the 5.2.0, as of September 2018). The tool is useful to select and visualize (a small number of) classes from the ontologies loaded in Protégé and the existing relationships.

I. OntoSphere

OntoSphere [49] is one of the two tools (the other is Tarsier [37]) that proposes a three-dimensional visualization of the graph. The rational behind OntoSphere is that exploiting a 3D space it is possible to better arrange items. Moreover, the 3D visualization is quite natural for humans and the exploration can then be more intuitive. Colors allow to easily convey information about the different nature of represented items. OntoSphere is aimed at representing both terminological and assertional statements. Four scene types are proposed to fulfill different requirements. The `RootFocus` scene shows all the concepts and their relationships on a sphere. The `TreeFocus` scene draws the tree originating from a concept, while the `ConceptFocus` scene proposes a view containing all the
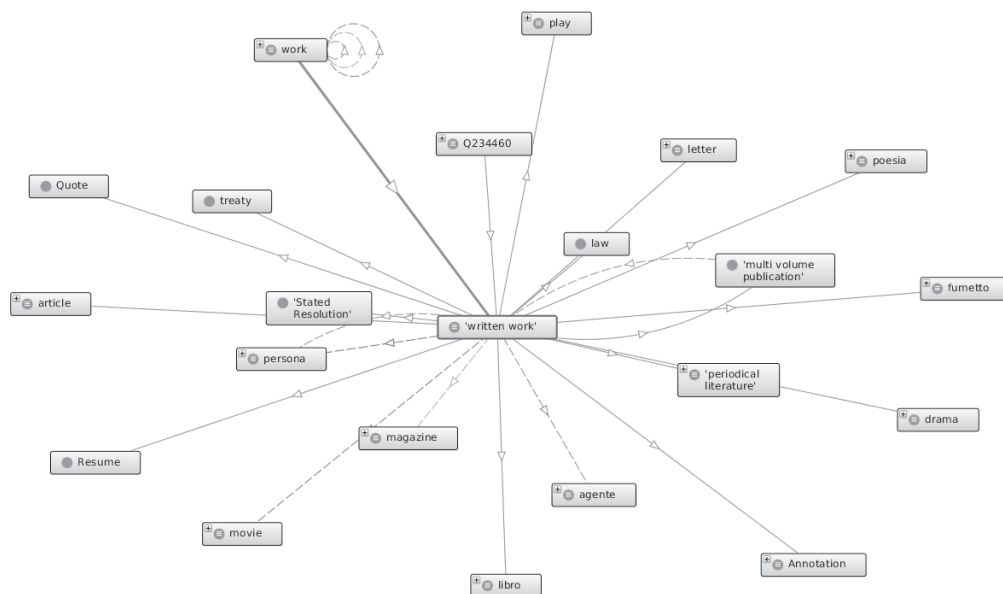
Fig. 4. A portion of the DBpedia ontology visualized in Ontograf [48]

items linked to a concept. The tool is aimed at domain experts dealing with the development and review of ontologies, as well as novice users that wants to understand the represented data and the links among concepts. OntoSphere is a standalone applications, but can also be run inside Protégé and Eclipse. The last version on the source code repository is dated 2008, so the development stopped ten years ago.

J. OWLViz

OWLViz [50] is a plugin for Protégé that enables the incremental visualization of the classes in the class hierarchy. As the name suggests, this tool, like IsaViz, is based on the famous library GraphViz developed by the AT&T and allows exporting the visualized graph as `png`, `jpeg` and `svg`. Through OWLViz is easy to visualize classes and `is-a` relationships. OWLViz is not developed anymore, but is still included in the last version of Protégé (September 2018).

K. Paged Graph Visualization

Paged Graph Visualization (PGV) [51] is a Java software for the visualization of RDF graphs. It is based on [52], a high performance RDF storage. With PGV, the exploration starts from a point of interest and then incrementally includes more data. Such point of interest can be selected interactively from a list or using a complex SPARQL query. Then, it is drawn in the center of the graph using the color green, and its direct neighbors are shown as blue rectangles placed around it. Literals on the other hand, are represented with the white color. The user is able to explore nodes by double-clicking on them: explored nodes are then displayed in green, while edges connecting explored nodes are depicted in red.

Deligiannidis et al. [51] declare that the tool's strength relies in helping the user willing to explore data without knowing the exact information and graph patterns he is looking for,

while in other situation a standard visualizer could be more appropriate.

To the best of authors' knowledge, this tool is not developed anymore.

L. RelFinder

RelFinder [53] is a web tool developed using Adobe Flex and can be tried using the web instance linked in the homepage of the project (configured to access DBpedia). RelFinder differs from the other tools proposed in this survey, since it is aimed at visualizing all the paths connecting two resources. So, its purpose is to answer a very specific question, rather than providing a tool for the free exploration of the knowledge base. The tool supports filtering to increase or reduce the number of relationships shown simultaneously. It also implements a smart drawing algorithm to reduce overlapping and the user is allowed to move and pin items. To the best of authors' knowledge, this tool is not actively developed but the online instance is still available for tests on the DBpedia endpoint.

Fig. 5 reports an example of this application where all the paths between two DBpedia resources, i.e, "JRR Tolkien" and "The Lord of the Rings", are shown. Fig. 6 shows the filtering panel proposed by RelFinder to show/hide elements in the visualization. Paths can be filtered by length, class of the RDF terms, property and connectivity level.

M. Tarsier

Tarsier [37], is a tool developed by the authors of the present paper. It is a software for the interactive exploration of an RDF graph in a three-dimensional space, aimed at the visualization of small and medium-sized knowledge bases. The main contribution of the tool is the introduction of the metaphor of semantic planes that group RDF terms sharing a common concept. The purpose of the tool is threefold: 1)
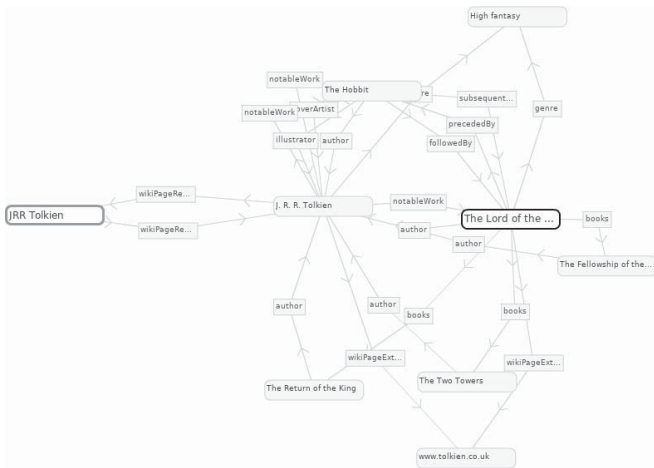
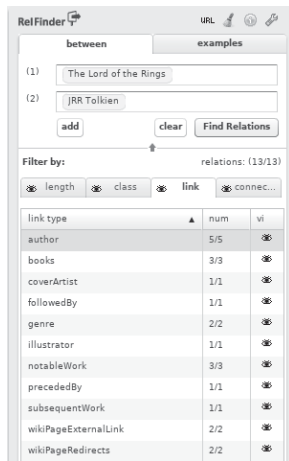Fig. 5. RelFinder [53] showing all the paths from "JRR Tolkien" to "The Lord of the Rings"

Fig. 6. RelFinder [53] filtering panel

Tarsier can be used as a support for didactic (e.g., to help newcomers to deal with Semantic Web technologies); 2) It is useful to figure out the nature of a new KB for developers (i.e., activity known as "sensemaking" [54]) ; 3) It allows debugging of semantic knowledge bases.

Tarsier retrieves data from SPARQL endpoints. The initial knowledge base can be determined through a SPARQL Contruct query: this pre-filtering stage allows to efficiently interact also with very large knowledge bases (e.g., DBpedia [55], that contains more than 6.6M entities). Tarsier proposes a classification of all the RDF terms among classes, resources, blank nodes, literals, object and datatype properties. This grouping is exploited by Tarsier's web interface to provide a set of controls for advanced filtering: through them, the user is allowed to toggle visibility of items or to move them across semantic planes.

An example of Tarsier is shown in Figg. 7 and 8. Tarsier was set up to retrieve data from DBpedia, and in particular to extract all the fantasy books published between 1900 and 2018

and their authors. While Fig. 7 shows the unfiltered knowledge base, in Fig. 8 is shown one of the peculiarities of Tarsier: the semantic planes. Two semantic planes were created over the main knowledge base to extract respectively books and one of the authors, i.e., Marion Zimmer Bradley. In this way, it is easy to notice how this instance of the class `foaf:Person` is linked with the rest of the graph.
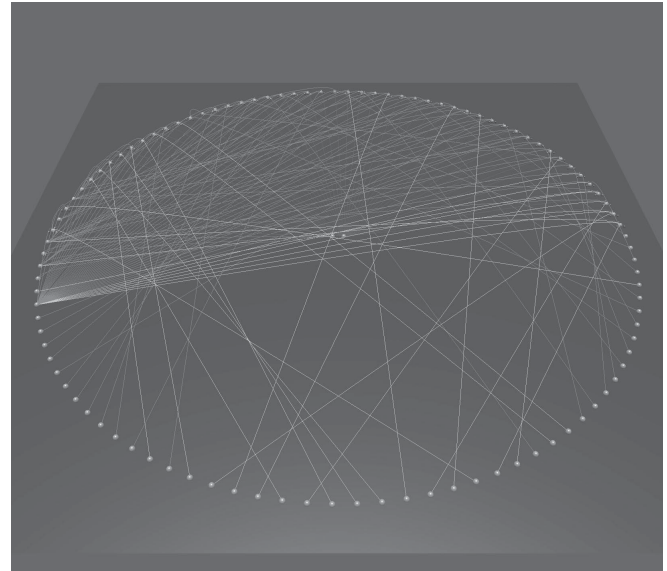

Fig. 7. Tarsier [37] showing the graph of all the fantasy books published from 1900 to 2018 and their authors. This subgraph is retrieved from DBpedia

Summarizing, Tarsier is a tool for the three-dimensional multi-planar visualization of small/medium-sized knowledge bases with advanced filtering mechanism based on the multiple selection of RDF terms or SPARQL queries.

N. TGVizTab

This is yet another visualization plugin for the ontology editor Protégé. TGVizTab [56] designed to be lightweight and support both T-Boxes and A-Boxes visualization. TGVizTab relies on TouchGraph, an open source Java environment aimed at creating and navigating network graphs in an interactive way. The tool supports exporting the graph in an XML file, to be loaded in other TouchGraph applications. The graph is drawn using the spring layout: similar nodes are drawn close to each other. TGVizTab, like other tools (e.g., Fenfire), asks the user to select a focal node among classes and instances to generate the graph. Then, the user is able to further modify the graph by right-clicking on the represented nodes: in this way the so-called *Node Menu* is shown, containing four options (i.e., expand, collapse, hide, view). Then TGVizTab allows to incrementally build the desired visualization

O. VOWL

VOWL (Visual OWL) [57] is available as a web-based tool (WebVOWL [58]), a plugin for Protégé (ProtégéVOWL [59]), a tool able to directly interact with Linked Data endpoints

being developed and at the moment do not allow to perform a customized request to DBpedia.
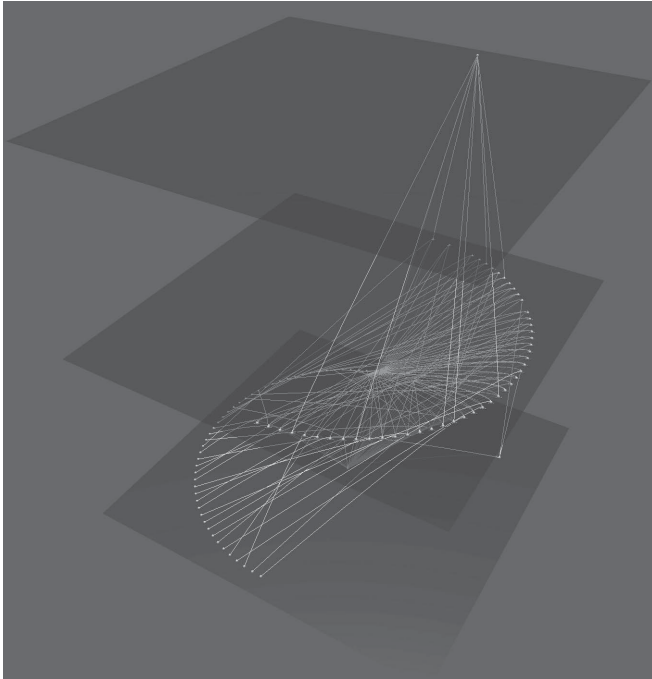

Fig. 8. Tarsier [37] showing two semantic planes over the main knowledge base: one showing books, the other (the topmost) showing the author Marion Zimmer Bradley

Fig. 9. Overview of the DBpedia ontology in WebVOWL2 [58]

(LD-VOWL [60]), and as a visual query language tool (QueryVOWL [61]). In this paper, we will refer to the web based version, WebVOWL. As the name suggests, software in the VOWL toolkit are designed to graphically represent ontologies. They propose a force-directed graph layout. The basic representation rules adpoted by VOWL consists in:

- Classes are depicted using circles where the color depends on the type: light blue for OWL classes, purple for RDFS classes, dark blue for those imported by other ontologies, gray for deprecated classes.
- OWL object and datatype properties are represented with black solid lines with, respectively, light blue and green labels, while RDFS properties have purple labels.
- Relationships subClassOf are depicted with a dashed line.

The graph drawn by VOWL can be exported as an svg image or as a json file. A click on a node or edge allows visualizing the associated metadata and statistics. Statistics also report the number of individuals of the selected class, but unfortunately this is the only information about individual that is possible to obtain using VOWL. As regards filtering, VOWL provides a basic support to filters that allows to show/hide object/datatype properties, solitary classes, class disjointness and set operators.

VOWL is actively developed and an online instance is available. As the tool is designed for ontologies, importing the output of the CONSTRUCT in Listing 3 results in representing only the two rdf:type relationships. The other tools are still
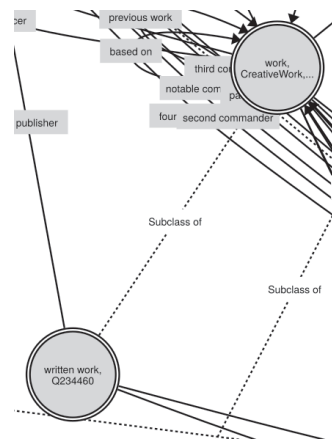

Fig. 10. WebVOWL2 [58]: Close-up on two of the classes defined in the DBpedia ontology

V. SUMMARY

Table III summarizes the main features of the analyzed software. Columns of the table are:

- **Software** – reports the name of the software;
- **T-Boxes** – this column tells if the tool supports the visualization of terminological statements;
- **A-Boxes** – this column shows if the tool supports the visualization of assertional statements (and can then be used to explore a knowledge base, rather than just ontologies);

- **Statistics** – a boolean field showing if the tool provides or not statistics on the visualized data;
- **Filtering** – filtering allows to show/hide elements in the visualization according to a set of user-defined criteria. Filtering can be implemented in very different ways (e.g., SPARQL queries, or UI controls to select classes, just to name a few). This column indicates whether the related tool provides at least one filtering mechanism.
- **Editing** – This paper surveys visualization tools for semantic data, but some of them also offer editing functionalities. This column states whether or not the related tool supports the manipulation of the ontology/knowledge base;
- **Standalone** – Many of the surveyed tools were born as plugins for the ontology editor Protégé. Other can be run as standalone software. This column tells if the related software is embedded in other tools or is a standalone application.
- **Plugin** – Not all the presented tools were born to visualize semantic knowledge bases. Then, some of them need additional plugins to achieve this task.
- **Domain** – This column contains the specific domain (if any) where the related application can be applied.
- **Reference** – This column reports the reference number of the paper(s) describing the tool.

Table IV summarizes information about the entities that started the development of the tool, the license and the current status of the project.

VI. CONCLUSION

This paper presented an analysis on the current status of the tools for the visualization of ontologies and knowledge bases exploiting a graph representation. Results of this study contrast with the wide spread of Semantic Web technologies: in fact, among the 15 software surveyed by the authors, only 5 are still active. It is also worth noting that 6 of them are available as visualization plugins for the popular ontology editor Protégé. Advanced filtering through SPARQL queries is provided, to the best of our knowledge, only by Gephi and Tarsier. Future work will consist in the analysis of other tools exploiting different representations with a detailed set of use cases for their evaluation.

APPENDIX

In Listing 3 we report the SPARQL CONSTRUCT query adopted in the practical examples shown in this Paper.

ACKNOWLEDGEMENTS

REFERENCES

[1] Tim Berners-Lee, Roy Fielding, and Larry Masinter. Uniform resource identifier (uri): Generic syntax. Technical report, 2004.
[2] Ora Lassila, Ralph R Swick, et al. Resource description framework (rdf) model and syntax specification. 1998.

```
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
PREFIX xsd: <http://www.w3.org/2001/...>
PREFIX rdfs: <http://www.w3.org/2000/...>
PREFIX rdf: <http://www.w3.org/1999/...>
PREFIX : <http://dbpedia.org/resource/>
PREFIX dbpo:<http://dbpedia.org/ontology/>
CONSTRUCT {
  ?book rdf:type dbpo:Book;
    dbpo:literaryGenre :Fantasy_novel;
    dbpo:author ?author;
    dbpo:releaseDate ?date;
    rdfs:comment ?comment;
    rdfs:label ?label;
    dbpo:numberOfPages ?num;
    foaf:isPrimaryTopicOf ?topic.
  ?author rdf:type foaf:Person }
WHERE {
  ?book rdf:type dbpo:Book;
    dbpo:literaryGenre :Fantasy_novel;
    dbpo:author ?author.
  ?author rdf:type foaf:Person.
  ?book rdfs:comment ?comment;
    rdfs:label ?label;
    dbpo:numberOfPages ?num;
    foaf:isPrimaryTopicOf ?topic.
  FILTER langMatches(lang(?label), "EN")
  FILTER langMatches(lang(?comment), "EN")
  FILTER (isIRI(?author))
  FILTER (xsd:integer(?num) > 200)
  OPTIONAL {
    ?book dbpo:releaseDate ?date .
    FILTER (?date > 1900)
    FILTER (?date < 2018)
    FILTER (isLiteral(?date))
    FILTER (datatype(?date) = xsd:integer)
  }
}
```

Listing 3. SPARQL CONSTRUCT that identifies in DBpedia the Fantasy-genre books written between 1900 and 2018 having more than 200 pages

[3] Dan Brickley, Ramanathan V Guha, and Brian McBride. Rdf schema 1.1. *W3C recommendation*, 25:2004–2014, 2014.
[4] Deborah L McGuinness, Frank Van Harmelen, et al. Owl web ontology language overview. *W3C recommendation*, 10(10):2004, 2004.
[5] Eric Prud, Andy Seaborne, et al. Sparql query language for rdf. 2006.
[6] Paul Gearon, Alexandre Passant, and Axel Polleres. Sparql 1.1 update. *W3C recommendation*, 21, 2013.
[7] Gregory D Abowd, Anind K Dey, Peter J Brown, Nigel Davies, Mark Smith, and Pete Steggles. Towards a better understanding of context and context-awareness. In *International symposium on handheld and ubiquitous computing*, pages 304–307. Springer, 1999.
[8] Kevin Ashton et al. That internet of things thing. *RFID journal*, 22(7):97–114, 2009.
[9] Alessandro Bassi and Geir Horn. Internet of things in 2020: A roadmap for the future. *European Commission: Information Society and Media*, 22:97–114, 2008.
[10] Nitin Naik. Choice of effective messaging protocols for iot systems: Mqtt, coap, amqp and http. In *Systems Engineering Symposium (ISSE), 2017 IEEE International*, pages 1–7. IEEE, 2017.
[11] P. Desai, A. Sheth, and P. Anantharam. Semantic gateway as a service

TABLE III. SUMMARY OF THE FEATURES OF THE TOOLS FOR THE VISUALIZATION OF SEMANTIC KNOWLEDGE BASES.
LEGEND: ✔ = YES, ✘ = NO, ! = PARTIAL, ☑ = MULTIPLE OPTIONS AVAILABLE, ? = UNKNOWN, ▬ = N OT APPLICABLE

Software	T-Boxes	A-Boxes	Statistics	Filtering	Editing	Standalone	Plugin	Domain	Reference
CytoScape	✔	✔	✔	✔	✔	✔	General SPARQL, SemScape, Vital AI	Biology	[39]
Fenfire	✔	✔	✘	✘	✔	✔	▬	General	[40]
Gephi	✔	✔	✔	✔	✘	✔	SemanticWebImport, VirtuosoImporter	General	[41]
Glow	✔	✔	✘	✔	✘	✘	▬	General	[42]
IsaViz	✔	✔	✘	✔	✔	✔	▬	General	[43]
Jambalaya	✔	✔	✘	✔	✘	✘	▬	General	[45]
LOD Live	✔	✔	✔	✘	✘	✔	▬	General	[47]
Ontograf	✔	!	✘	✘	✘	✘	▬	General	[48]
OntoSphere	✔	✔	✘	✘	✘	☑	▬	General	[49]
OWLViz	✔	✔	✘	✘	✘	✘	▬	General	[50]
PGV	✔	✔	✘	✘	✘	✔	▬	General	[51]
RelFinder	✘	✔	✔	✔	✘	✔	▬	General	[53]
Tarsier	✔	✔	✘	✔	✘	✔	▬	General	[37]
TGVizTab	✔	✔	✘	✘	✘	✘	▬	General	[56]
VOWL	✔	!	✔	✔	✘	☑	▬	General	[57]–[59]

TABLE IV. INFORMATION ABOUT DEVELOPERS, DEVELOPMENT
AND LICENSES. LEGEND: ✔ = YES, ✘ = NO, ? = UNKNOWN

Software	Developed by	License	Active	Reference
CytoScape	CytoScape Consortium	GPL	✔	[39]
Fenfire	University of Jywäskylä and Digital Enterprise Research Institute of the National University of Galway	GPL	✘	[40]
Gephi	Gephi Consortium	GPL	✔	[41]
Glow	Erasmus University Rotterdam	?	✘	[42]
IsaViz	INRIA in collaboration with Xerox Research Centre Europe	GPL	✘	[43]
Jambalaya	Chisel Lab (University of Victoria)	Individual	✘	[45]
LOD Live	lodlive.it	MIT	✔	[47]
Ontograf	Stanford Center for Biomedical Informatics Research	LGPL	✘	[48]
OntoSphere	Politecnico di Torino	LGPL	✘	[49]
OWLViz	University of Manchester	LGPL	✘	[50]
PGV	LSDIS Lab and Computer Science (University of Georgia), Kno.e.sis Center (Wright State University)	?	✘	[51]
RelFinder	Visualization and Interactive Systems (University of Stuttgart), Agile Knowledge Engineering and Semantic Web (University of Leipzig), Interactive Systems and Interaction Design (University of Duisburg-Essen)	GPL	✘	[53]
Tarsier	Advanced Research Center on Electronic Systems (University of Bologna)	GPL	✔	[37]
TGVizTab	IAM Group (University of Southampton)	GPL	✘	[56]
VOWL	Visualization and Interactive Systems (University of Stuttgart), Alexandru Ioan Cuza University	MIT	✔	[57]–[59]

architecture for iot interoperability. In *IEEE International Conference on Mobile Services*, pages 313–319. IEEE, 2015.

[12] Linked open vocabularies (lov).

[13] Pierre-Yves Vandenbussche, Ghislain A Atemezing, María Poveda-Villalón, and Bernard Vatant. Linked open vocabularies (lov): a gateway to reusable semantic vocabularies on the web. *Semantic Web*, 8(3):437–452, 2017.

[14] Natalya F Noy, Deborah L McGuinness, et al. Ontology development 101: A guide to creating your first ontology, 2001.

[15] Shixia Liu, Weiwei Cui, Yingcai Wu, and Mengchen Liu. A survey on information visualization: recent advances and challenges. *The Visual Computer*, 30(12):1373–1393, 2014.

[16] Ivan Herman, Guy Melançon, and M Scott Marshall. Graph visualization and navigation in information visualization: A survey. *IEEE Transactions on visualization and computer graphics*, 6(1):24–43, 2000.

[17] Jinghai Rao and Xiaomeng Su. A survey of automated web service composition methods. In *International Workshop on Semantic Web Services and Web Process Composition*, pages 43–54. Springer, 2004.

[18] Natalya F Noy. Semantic integration: a survey of ontology-based approaches. *ACM Sigmod Record*, 33(4):65–70, 2004.

[19] Akrivi Katifori, Constantin Halatsis, George Lepouras, Costas Vassilakis, and Eugenia Giannopoulou. Ontology visualization methodsa survey. *ACM Computing Surveys (CSUR)*, 39(4):10, 2007.

[20] Paul Mutton and Jennifer Golbeck. Visualization of semantic metadata and ontologies. In *Information Visualization, 2003. IV 2003. Proceedings. Seventh International Conference on*, pages 300–305. IEEE, 2003.

[21] Vitalis Wiens, Steffen Lohmann, and Sören Auer. Semantic zooming for ontology graph visualizations. In *Proceedings of the Knowledge Capture Conference*, page 4. ACM, 2017.

[22] Nikos Bikakis and Timos Sellis. Exploration and visualization in the web of big linked data: A survey of the state of the art. *arXiv preprint arXiv:1601.08059*, 2016.

[23] Katifori Akrivi, Torou Elena, Halatsis Constantin, Lepouras Georgios, and Vassilakis Costas. A comparative study of four ontology visualization techniques in protege: Experiment setup and preliminary results. In *Information Visualization, 2006. IV 2006. Tenth International Conference on*, pages 417–423. IEEE, 2006.

[24] V Swaminathan and R Sivakumar. A comparative study of recent

ontology visualization tools with a case of diabetes data. *International Journal of Research in Computer Science*, 2(3):31, 2012.

[25] Manu Sporny, Dave Longley, Gregg Kellogg, Markus Lanthaler, and Niklas Lindström. Json-ld 1.0. *W3C Recommendation*, 16, 2014.

[26] Json-ld playground.

[27] Christian Bizer, Tom Heath, and Tim Berners-Lee. Linked data: The story so far. In *Semantic services, interoperability and web applications: emerging concepts*, pages 205–227. IGI Global, 2011.

[28] Oscar Peña, Unai Aguilera, and Diego López-de Ipiña. Linked open data visualization revisited: a survey. *Semantic Web Journal*, 2014.

[29] Jiawei Han, Jian Pei, and Micheline Kamber. *Data mining: concepts and techniques*. Elsevier, 2011.

[30] Rdf 1.1 concepts and abstract syntax.

[31] Dan Brickley and Libby Miller. Foaf vocabulary specification 0.91, 2010.

[32] Natalya Noy. Order from chaos. *Queue*, 3(8):42–49, 2005.

[33] A. Maedche and S. Staab. Ontology learning for the semantic web. *IEEE Intelligent Systems*, 16(2):72–79, March 2001.

[34] Artem Chebotko, Xubo Fei, Cui Lin, Shiyong Lu, and Farshad Fotouhi. Storing and querying scientific workflow provenance metadata using an rdbms. In *e-Science and Grid Computing, IEEE International Conference on*, pages 611–618. IEEE, 2007.

[35] Stuart Weibel. The dublin core: a simple content description model for electronic resources. *Bulletin of the American Society for Information Science and Technology*, 24(1):9–11, 1997.

[36] Stephen G Kobourov. Spring embedders and force directed graph drawing algorithms. *arXiv preprint arXiv:1201.3011*, 2012.

[37] Fabio Viola, Luca Roffia, Francesco Antoniazzi, D'Elia Alfredo, Cristiano Aguzzi, and Tullio Salmon Cinotti. Interactive 3d exploration of rdf graphs through semantic planes. *Future Internet*, 10(9):36, 2018.

[38] Fabian Beck, Michael Burch, Stephan Diehl, and Daniel Weiskopf. A taxonomy and survey of dynamic graph visualization. In *Computer Graphics Forum*, volume 36, pages 133–159. Wiley Online Library, 2017.

[39] Paul Shannon, Andrew Markiel, Owen Ozier, Nitin S Baliga, Jonathan T Wang, Daniel Ramage, Nada Amin, Benno Schwikowski, and Trey Ideker. Cytoscape: a software environment for integrated models of biomolecular interaction networks. *Genome research*, 13(11):2498–2504, 2003.

[40] Tuukka Hastrup, Richard Cyganiak, and Uldis Bojars. Browsing linked data with fenfire. 2008.

[41] Mathieu Bastian, Sebastien Heymann, Mathieu Jacomy, et al. Gephi: an open source software for exploring and manipulating networks. *Icwsm*, 8(2009):361–362, 2009.

[42] Walter Hop, Sven de Ridder, Flavius Frasincar, and Frederik Hogenboom. Using hierarchical edge bundles to visualize complex ontologies in glow. In *Proceedings of the 27th Annual ACM Symposium on Applied Computing*, pages 304–311. ACM, 2012.

[43] Emmanuel Pietriga. Isaviz: A visual authoring tool for rdf. *World Wide Web Consortium.[Online]. Available: http://www. w3. org/2001/11/IsaViz*, 2003.

[44] John Ellson, Emden Gansner, Lefteris Koutsofios, Stephen C North, and Gordon Woodhull. Graphvizopen source graph drawing tools. In *International Symposium on Graph Drawing*, pages 483–484. Springer, 2001.

[51] Leonidas Deligiannidis, Krys J Kochut, and Amit P Sheth. Rdf data exploration and visualization. In *Proceedings of the ACM first workshop*

[45] Margaret-Anne Storey, Natasha F Noy, Mark Musen, Casey Best, Ray Fergerson, and Neil Ernst. Jambalaya: an interactive environment for exploring ontologies. In *Proceedings of the 7th international conference on Intelligent user interfaces*, pages 239–239. ACM, 2002.

[46] M-A Storey, Casey Best, and Jeff Michand. Shrimp views: An interactive environment for exploring java programs. In *Program Comprehension, 2001. IWPC 2001. Proceedings. 9th International Workshop on*, pages 111–112. IEEE, 2001.

[47] Diego Valerio Camarda, Silvia Mazzini, and Alessandro Antonuccio. Lodlive, exploring the web of data. In *Proceedings of the 8th International Conference on Semantic Systems*, pages 197–200. ACM, 2012.

[48] Sean Falconer. Ontograf protege plugin.

[49] Alessio Bosca, Dario Bonino, and Paolo Pellegrino. Ontosphere: more than a 3d ontology visualization tool. In *Swap*. Citeseer, 2005.

[50] Matthew Horridge. Owlviz. *Available on: http://protegewiki.stanford.edu/wiki/OWLViz*, 2010. on CyberInfrastructure: information management in eScience, pages 39–46. ACM, 2007.

[52] Maciej Janik and Krys Kochut. Brahms: a workbench rdf store and high performance memory system for semantic association discovery. In *International Semantic Web Conference*, pages 431–445. Springer, 2005.

[53] Philipp Heim, Sebastian Hellmann, Jens Lehmann, Steffen Lohmann, and Timo Stegemann. Relfinder: Revealing relationships in rdf knowledge bases. In *International Conference on Semantic and Digital Media Technologies*, pages 182–187. Springer, 2009.

[54] Enrico Motta, Paul Mulholland, Silvio Peroni, Mathieu dAquin, Jose Manuel Gomez-Perez, Victor Mendez, and Fouad Zablith. A novel approach to visualizing and navigating ontologies. In *International Semantic Web Conference*, pages 470–486. Springer, 2011.

[55] Jens Lehmann, Robert Isele, Max Jakob, Anja Jentzsch, Dimitris Kontokostas, Pablo N Mendes, Sebastian Hellmann, Mohamed Morsey, Patrick Van Kleef, Sören Auer, et al. Dbpedia–a large-scale, multilingual knowledge base extracted from wikipedia. *Semantic Web*, 6(2):167–195, 2015.

[56] Harith Alani. Tgviztab: an ontology visualisation extension for protégé. 2003.

[57] Steffen Lohmann, Stefan Negru, Florian Haag, and Thomas Ertl. Visualizing ontologies with vowl. *Semantic Web*, 7(4):399–419, 2016.

[58] Steffen Lohmann, Vincent Link, Eduard Marbach, and Stefan Negru. WebVOWL: Web-based visualization of ontologies. In *Proceedings of EKAW 2014 Satellite Events*, volume 8982 of *LNAI*, pages 154–158. Springer, 2015.

[59] Steffen Lohmann, Stefan Negru, and David Bold. The ProtégéVOWL plugin: Ontology visualization for everyone. In *Proceedings of ESWC 2014 Satellite Events*, volume 8798 of *LNCS*, pages 395–400. Springer, 2014.

[60] Marc Weise, Steffen Lohmann, and Florian Haag. Ld-vowl: Extracting and visualizing schema information for linked data. In *2nd International Workshop on Visualization and Interaction for Ontologies and Linked Data, Kobe, Japón*, pages 120–127, 2016.

[61] Florian Haag, Steffen Lohmann, Stephan Siek, and Thomas Ertl. Queryvowl: A visual query notation for linked data. In *International Semantic Web Conference*, pages 387–402. Springer, 2015.