RDF Resource Search and Exploration with LinkZoo

Marios Meimaris[1,2], Giorgos Alexiou[1,3], Katerina Gkirtzou[1], George Papastefanatos[1] and Theodore Dalamagas[1]

[1]*Institute for the Management of Information Systems, Reseach Center ATHENA, Athens, Greece*

[2]*Department of Computer Science and Biomedical Informatics, University of Thessaly, Volos, Greece*

[3]*Department of Electrical and Computer Engineering, National Technical University of Athens, Athens, Greece*

Keywords: Linked Data, RDF, Collaboration, Graph Search.

Abstract: The Linked Data paradigm is the most common practice for publishing, sharing and managing information in the Data Web. Linkzoo is an IT infrastructure for collaborative publishing, annotating and sharing of Data Web resources, and their publication as Linked Data. In this paper, we overview LinkZoo and its main components, and we focus on the search facilities provided to retrieve and explore RDF resources. Two search services are presented: (1) an interactive, two-step keyword search service, where live natural language query suggestions are given to the user based on the input keywords and the resource types they match within LinkZoo, and (2) a keyword search service for exploring remote SPARQL endpoints that automatically generates a set of candidate SPARQL queries, i.e., SPARQL queries that try to capture user's information needs as expressed by the keywords used. Finally, we demonstrate the search functionalities through a use case drawn from the life sciences domain.

1 INTRODUCTION

The Data Web has completely changed the way we create, interlink and consume large volumes of information. More and more corporate, governmental and user-generated datasets break the walls of traditional "private" management within their production site, are published, and become available for potential data consumers. The Data Web extents current Web infrastructure to a global data space containing and connecting data from very diverse domains.

The Linked Data paradigm is the most common practice for publishing, sharing and managing information in the Data Web, and offers a new way of data integration and interoperability. The main concept in Linked Data is that all resources published on the Web are uniquely identified by a URI, and typed links (instead of traditional Web hyperlinks) between URIs are used to semantically connect resources. Reusing existing URIs rather than creating new ones, and pointing from one dataset to another by referencing these URIs, forms the Linked Open Data cloud (Bizer et al., 2009).

Linked Data is mainly implemented with the Resource Description Framework (RDF). An RDF representation is a set of statements about resources, known as *triples*, i.e. expressions of the form *subject predicate object*. The *subject* refers to a resource to be described. Actually, the subject is a URI reference to that resource, which identifies it unambiguously. Predicates are usually terms from existing vocabularies and ontologies and are also identified by URIs. Finally, the *object* can be either a literal or a URI that refers to another RDF resource. We will refer to triples whose objects are literals as *entity-to-attribute* properties, and to triples whose objects are entities as *inter-entities* properties. A set of RDF triples can be represented by a directed labelled graph, known as the RDF data graph. However, in practice, RDF triples are stored in relational database systems, native triple/quad stores or graph DBMS (Faye et al., 2012; Bizer and Schultz, 2008). To query Linked Data, the SPARQL query language is used (Prud'Hommeaux and Seaborne, 2008).

In this paper, we briefly describe LinkZoo (Meimaris et al., 2014), a web-based platform for collaborative management, editing and sharing of Data Web resources, and we mainly focus on the search facilities. Two LinkZoo search services are presented: (1) an interactive, two-step keyword

Meimaris M., Alexiou G., Gkirtzou K., Papastefanatos G. and Dalamagas T..
RDF Resource Search and Exploration with LinkZoo.
DOI: 10.5220/0005499602320239
In *Proceedings of 4th International Conference on Data Management Technologies and Applications* (DATA-2015), pages 232-239
ISBN: 978-989-758-103-8

search service, where live natural language query suggestions are given to the user based on the input keywords and the resource types they match within LinkZoo, and (2) a keyword search service for exploring remote datasets, which automatically generates a set of candidate SPARQL queries that try to capture user's information need as expressed by the keywords used.

To demonstrate the services' effectiveness, we describe a real use case where the search facilities of LinkZoo are combined to effectively address user needs when working with a scientific Linked Data set. Furthermore, we perform a preliminary effectiveness study to evaluate the keyword search service with query candidates. LinkZoo is available at: `http://www.linkzoo.gr:9000` with credentials (user: *data_2015*, password: *data_2015*) for the demo account.

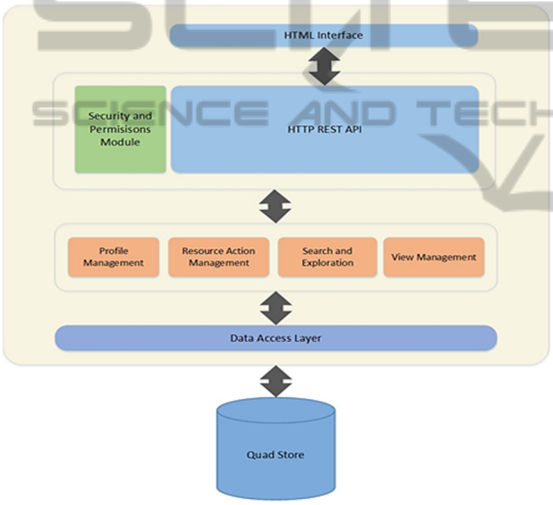

Figure 1: Linkzoo Architecture.

2 LinkZoo OVERVIEW

2.1 Architecture

The architecture of LinkZoo is shown in Figure 1. The Storage Layer is built on top of a persistent quad store, while the system's functionality is based on four basic modules: the *Profile Management module*, the *Resource Action Management module*, the *View Management module,* and the *Search module*. The *Profile Management* module is responsible for the administration of user accounts, handling actions such as user administration, account management, ascribing namespaces and named graphs to users. The *Resource Action Management* module provides processing and editing of resources such as importing, annotating and dereferencing. It is responsible for handling all actions associated with each type of resource. Also, it provides resource sharing functionality among users. The *View Management* module controls the lifecycle of views and folders; it also manages containment relationships between resources, folders and views. Views can be considered as different workspaces where the same resources can be organized in various ways. Finally, the *Search & Exploration Module* is responsible for the searching facilities implemented in LinkZoo. Specifically, it implements different search mechanisms as well as faceted browsing capabilities for private and public user graphs. A more in-depth discussion of the Search module is presented in Section 3.

2.2 Resource Model

The LinkZoo Resource Representation Model captures the following aspects: (i) resource descriptive metadata, (ii) resource interlinking, and (iii) view definitions and containment relationships of resources in views and folders. Common vocabularies such as RDFS, Dublin Core Terms and FOAF are used to model non-functional metadata (e.g. labels, creators, etc.). Moreover, users can import existing ontologies or define new ones under their own schema namespace. Given that a resource can co-exist in many user accounts (e.g., in case two users happened to import the same resource), resource definitions and views in LinkZoo depend on user context. Multiple parallel versions of resource definitions are stored in their owners' named graphs. LinkZoo handles a variety of resource types, e.g., files, URLs, contacts, RDF datasets and remote SPARQL endpoints. Furthermore, folders, i.e., resource collections, are also modelled as a special resource type, and, thus, can be annotated, shared and linked accordingly. We have defined an extensible taxonomy of resource types that includes various levels of specialization for each type. This way, we allow for different handling of each resource type or sub-type.

2.3 Resource Annotations

In LinkZoo, users can annotate resources and enrich their definition with new triples. Many established ontologies and vocabularies have been imported in the tool for quick access, while new properties can also be created on demand, under each user's custom schema namespace. Annotation can be performed manually and collaboratively, as well as

automatically. In the case of URL resources, external APIs are used to automatically enrich the imported URLs by parsing their content and extracting related Linked Data entities (specifically DBPedia and Freebase resources). In particular, LinkZoo utilizes the Alchemy API (Alchemy API, 2015) but other similar services can be used as well.

2.4 Sharing and Collaboration

LinkZoo resources can be collaboratively annotated and enriched with new knowledge. This is achieved by sharing resources with other users, with appropriate roles and privileges. Three levels of privileges, represented by three user roles, ensure proper sharing and usage among users. These are the *owner*, *editor* and *viewer*. Owners and editors of resources are able to share/unshare, annotate and delete them. Viewers cannot perform any kind of write-related operation that alters the state of the resource in the storage, and are thus limited to read-only actions of their shared resources. Furthermore, resources can be *private* or *public*. Shared directories pass on their sharing status to their contained items, and whenever a new resource is inserted into a folder, it automatically becomes available to the folder's shared users.

2.5 Linked Data Publication

Creating and publishing resources as Linked Data is a key feature of LinkZoo. This means that created resources are automatically assigned dereferenceable URIs, which can be used for external linking and referencing. These URIs follow a simple minting schema that takes into account the type of resource as well as a unique identifier created dynamically.

Dereferencing is performed when there are appropriate permissions, thus restricting external users with no authorization from getting access to descriptions of private resources. Unauthorized dereferencing returns a limited description. However, for a private resource, a public dereferenceable URI can be generated on demand, allowing the user to offer access to others without changing its status. If the user owns a LinkZoo account, he can choose to import the item to his account. Also, the platform offers serialized RDF exporting facilities for selected resources.

2.6 Static and Dynamic Views

The default exploring and browsing mode of LinkZoo follows the traditional folder-based approach of file systems with visual interfaces. However, LinkZoo exploits the semantics of the resources to provide multiple ways of organization. Users are able to organize their resources based on their properties and store the results as linked views. Views leverage the semantic web by offering intuitive means for organizing, searching and discovering new resources either within the platform or the entire LOD cloud. In essence, views act as workspaces and can be specialized in two sub-types, namely *static* and *dynamic*. These can be parallelized with materialized and non-materialized views in relational models respectively. Dynamic views are result sets of particular queries that are associated with the views. This way, the various annotations of resources are used as organizational factors, depending on the user's needs. For instance, the user can create a dynamic view with the query "*Find all hairpins that produce mature with name hsa*-mir-147a". This will organize into a dedicated workspace all resources that are matched by the evaluation of this query. Updating the dynamic view will result in repopulating the view based on the updated query result set.

3 RESOURCE SEARCH AND EXPLORATION

Search and exploration in LinkZoo combines keyword-based search with property-based faceted browsing. Specifically, we have implemented an "*on-the-go*" search mechanism that serves suggestions based on the taxonomy of resource types as well as the properties of resources. This type of search is applied on LinkZoo resources that have been imported to or shared with a user's account. Furthermore, we have implemented a "*search–with-query-candidates*" mechanism that can be used to query remote endpoints imported in LinkZoo. This way, LinkZoo allows exploring, importing, and annotating remote datasets. Therefore, by combining the search functionalities, users can first find relevant endpoints and then query them explicitly.

LinkZoo also provides exploration by faceted browsing. In every folder shown to the user, the system also shows all properties from triples with the contained resources as subjects. Then, upon selection of a property, the objects in the triples of that property will be listed in the form of virtual folders for further exploration. For instance, in a folder that contains MP3 audio files, the property `mo:genre` will be selectable for faceted browsing. Then, the MP3 resources will be organized to

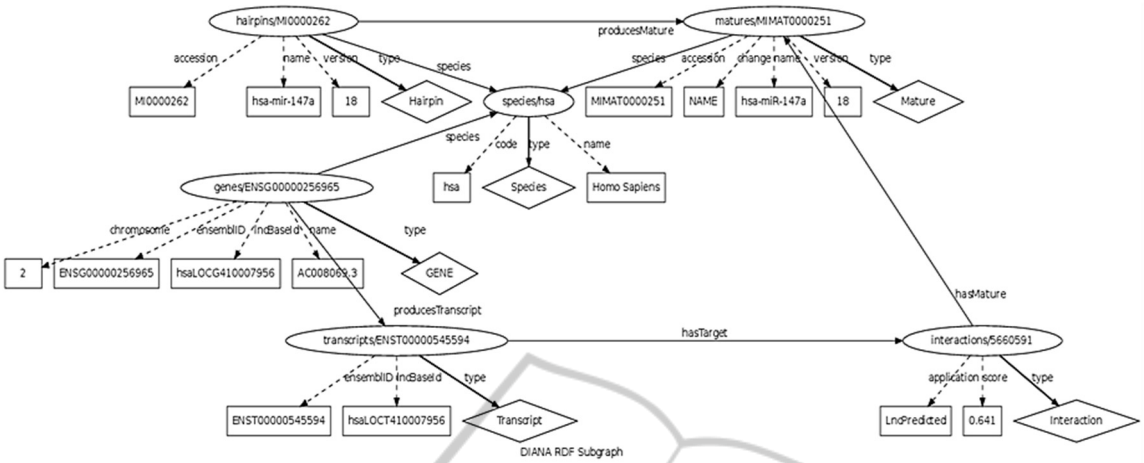

Figure 2: Example of an RDF data graph.

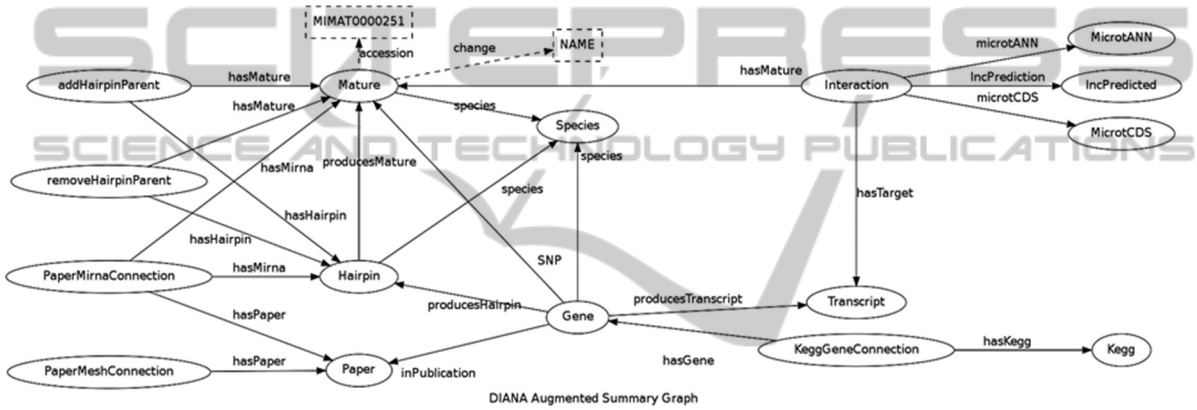

Figure 3: One of the 5 possible Augmented Summary Graphs for the keywords MIMAT0000251, name and hasTarget for the data graph shown in Figure 2.

virtual folders based on the distinct values of the `mo:genre` property. The keyword search and property filtering methods can be combined and applied in an exploratory manner.

3.1 Search with On-the-Go Suggestions

An interactive, two-step search process is implemented for the exploration of the user's imported and shared data within LinkZoo. Natural language query suggestions are given to the user based on the input keywords and the resource types they match. For instance, by typing "*Research*", the user will be prompted to select a suggestion from a list that contains queries of the form "*find URLs with rdfs:seeAlso dbpedia:Research*", if a similar pattern exists in the user's data. Upon selection of a suggested query, the relevant results will be shown in a virtual folder. These can then be further processed in bulk for annotation, moving, deleting and other resource-specific operations. This kind of search works incrementally, on the results previously fetched. For example, after the user selects "*find URLs with rdfs:seeAlso dbpedia:Research*" and the relevant results are fetched, further keyword exploration will suggest queries based only on the current result set and not on the whole set of user data. The above points can be summarized in the following steps: *(1)* Each keyword entered by the user is matched to objects of the triples of the user's resources. The distinct predicate-object pairs that are matched are ordered by their resource types. *(2)* The system feeds back suggestions of the form "find {resource_type} with {predicate} {object} ". For example, "*find URLs with rdfs:seeAlso dbpedia:Research"*. *(3)* The user selects a suggestion and the system builds a query based on the resource type and the predicate object pairs found in (1). *(4)* The system feeds back the results of the query in (3) to the user. *(5)* The user goes back to (1) and enters a new keyword in order to refine the results.

3.2 Search with Query Candidates

A key feature of LinkZoo is a search service that assists the user to explore remote RDF data sources and to retrieve RDF entities, which, in turn, can be imported in LinkZoo. Given a set of keywords, LinkZoo returns a set of candidate SPARQL queries that try to capture user's information need as expressed by the keywords. Briefly, given a set of *n* keywords, we perform the following steps: *(1)* For each keyword k_i, we retrieve all its matches M_i on the RDF data graph. *(2)* We calculate all possible combinations $C = M_1 \times M_2 \times \cdots \times M_n = \{ c = (m_1, \cdots, m_n) | m_i \in M_i \, \forall i = 1, \cdots, n \}$ of all the matched elements M_i where $i = 1, \cdots, n$. *(3)* For each combination $c \in C$ that contains one matched element m_i per keyword k_i, we create an *augmented summary graph* G_c. *(4)* From each augmented summary graph G_c, we generate the *query pattern graph* G_c^{QP} and finally *(5)* we translate each query pattern graph G_c^{QP}. into a SPARQL query. Next, we elaborate on the details.

Let's consider that we have an RDF dataset as the one shown in Figure 2. The dataset is depicted as an RDF data graph, where oval shape vertices represent RDF entities, diamond shape vertices represent RDF classes and rectangle shape vertices represent literals. Similarly, dashed edges represent entity-to-attribute properties, while solid ones represent inter-entities properties. Let's us assume that the user has provided the keywords *MIMAT0000251, name* and *hasTarget*. The first step is to match the keywords to elements in the RDF data graph: (1) *MIMAT0000251* matches to the literal "MIMAT0000251" that is connected via the property "accession" with an RDF entity of "Mature" type, (2) *name* matches to the literal "NAME" that is connected via the property "change" met with RDF entity of type "Mature" and to the entity-to-attribute property "name" met with entities of type "Hairpin", "Mature", "Species" and "Gene", resulting in 5 possible matches, and (3) *hasTarget* matches to the inter-entities property "hasTarget" met with subject of type "Interaction" and object of type "Transcript". The second step is to calculate all possible combinations of the matched elements and for each combination *c* create the augmented summary graph G_c. In this example, there are 5 possible combinations. Let's examine one combination, where the *name* keyword matches to the literal "NAME".

Augmented Summary Graph. The augmented summary graph G_c is a combination of an aggregated representation of the RDF data graph G, enriched with graph elements for each matched element $m_i, i = 1, \cdots, n$. More specifically, all entities from the RDF data graph G that have the same type of RDF class are represented by a vertex labelled with the name of the RDF class. Similarly, all inter-entities properties of the same type are represented by a directed edge between the aggregated vertex representation of the subjects and the aggregated vertex representation of the objects. The edge is also labelled with the property's name. Note that entity-to-attribute properties as well as literal values are omitted from the summary representation. Overall, the augmented graph is actually an abstraction of the RDF data graph G.

The augmented summary graph G_c contains also graph elements for each element m_i from the set of matched elements c. More specifically, if the matched element m_i is a literal value, then the graph is extended by a directed edge and a vertex. The edge represents the entity-to-attribute property that the matched element is met with in the RDF data graph, while the vertex is the matched element m_i itself. Note that the edge is attached from the aggregated vertex representation of the subject to the newly inserted vertex. Similarly, if the matched element m_i is an entity-to-attribute property, then the graph is extended by a directed edge and a vertex. The edge represents the entity-to-attribute property, i.e. the matched element m_i, and it is attached from the aggregated vertex representation of the subject to the newly inserted vertex. The difference from the previous case is that the latter vertex represents the unknown literal of the property. Note that if the same entity-to-attribute property is met with multiple RDF entities of different RDF types in the RDF data graph that would lead to different sets c. An example of the Augmented Summary Graph for the combination under investigation is shown in Figure 3.

Query Pattern Graph. In order to extract the query pattern graph G_c^{QP} from the augmented summary graph G_c, we calculate the shortest paths between every pair of matched elements and we combine all of them into one connected subgraph. Note that during the shortest path calculations we ignore the directionality of the edges. Moreover, since a matched element m_i , i.e. a source or sink of the shortest path algorithm, can also be an edge, then the distance between two matched elements counts the number of both vertices and edges that needs to traverse across the augmented summary graph G_c.

For the Augmented Summary Graph of Figure 3, since we have three keywords, we need to calculate three shortest paths. We then combine the shortest

paths into a single connected component, resulting to the query pattern graph shown in Figure 4. Note that the extra node "Transcript" is attached to the property "hasTarget" in order to form a complete triple pattern, although it is not part of neither of the shortest paths.

Candidate SPARQL Generation. The final step of the mapping process is to translate the query pattern graph G_c^{QP} into a SPARQL query. Note that the vertices of the query pattern graph G_c^{QP} are either known or unknown literal values and aggregated representations. We need to connect the latter type of vertices and the vertices of unknown literal values with variables in order to form the SPARQL triple patterns. Note that labels of the vertices can be used as constants in the triple patterns, while the labels of the edges as predicates. To produce conjunctive SPARQL queries, given the above observations for every vertex $v \in G_c^{QP}$, we perform the following:

- if v is a literal, do nothing
- if v is an unknown literal, then connect the vertex into a new variable $var(v)$.
- If v is a aggregated representation for entities of RDF type *class* with label $label(v) = class$, then the vertex is connected into a new variable $var(v)$ and produce the following SPARQL triple $var(v)$ `rdf:type` $label(v)$.

Similarly, for every edge $e \in G_c^{QP}$:

- If e represents an inter-entities property between a vertex $subject$ and a vertex $object$, then we produce the triple pattern $var(subject)$ $label(e)$ $var(object)$.
- If e represents an entity-to-attribute property between a vertex $subject$ and a vertex $object$ that is a literal, then we produce the triple pattern $var(subject)$ $label(e)$ $label(object)$.
- If e represents an entity-to-attribute property between a vertex $subject$ and a vertex $object$ that is an unknown literal, then we produce the triple pattern $var(subject)$ $label(e)$ $var(object)$.

In our example, the pattern graph of Figure 4 is mapped to the following SPARQL query.

```
SELECT ?I ?M ?T WHERE
{?I a diana:Interaction.
?M a diana:Mature.
?T a diana:Transcript.
?I diana:hasMature ?M.
?I diana:hasTarget ?T.
?M diana:accession "MIMAT0000251".
?M diana:change "NAME".}
```

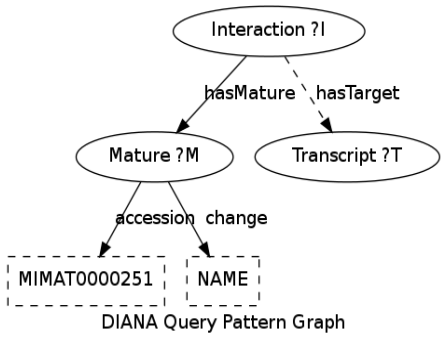

Figure 4: The Query Pattern Graph extracted from the Augmented Summary Graph of Figure 3. In dashed style, we depict the matched elements.

4 DEMONSTRATION

In this section we demonstrate the search capabilities of our tool. We employ a use case taken from the DIANA linked dataset. The dataset contains aggregated information from well-known biology databases, including ENSEMBL, miRBase and KEGG pathway, of the microRNA world published in RDF, available at the endpoint `http://leonardo.imis.athena-innovation.gr:8891/diana/sparql`.

Let us consider the following scenario: a user is engaged in a bioinformatics research project which concerns control mechanisms for cancer studies, and more specifically it focuses on the regulatory microRNA molecules. To this end, the user has gathered resources and data from a variety of sources, such as publications from PubMed, and data from the Gene Expression Atlas, the Experimental Factor Ontology and DIANA. Some of these resources have been imported by the user himself, while others have been shared to him by his collaborators. Publications are modelled either with the *file* or *URL* type and are annotated with metadata provided by the user and collaborators as well as external enrichment services. On the other hand, the imported datasets are modelled as resources of the type *DataCollection*, which allows the exploration of a remote RDF dataset via our *search-with-query-candidates* mechanism. Similarly to other resource types, *DataCollection* resources are annotated with descriptive metadata.

We consider that the user has either limited knowledge of the RDF vocabulary used to describe the datasets, or limited experience with SPARQL. To overcome this problem, LinkZoo offers the capability of keyword search for identifying and exploring RDF datasets. The user can identify

potential datasets that fit his criteria, based on metadata annotations. In this case, he is looking for datasets that contain microRNA data, and to that end he uses the *search-on-the-go* utility to eventually identify the DIANA dataset.

After identifying DIANA as the dataset to work with, the user is interested in collecting information about zebrafish miRNAs, in order to evaluate a potential correlation with human cancer cell metastasis. To achieve this, he types the words "*zebrafish hairpin*" into the keyword search box and as a result he gets two possible SPARQL queries. Both generated queries will search for publications that are annotated with the mesh term "*zebrafish*" and are related with miRNAs of hairpin type. In the first query, the "*hairpin*" keyword matches to the RDF class `diana:Hairpin` and imposes a direct constraint that the property `diana:hasMirna` of the RDF class `diana:PaperMirnaConnection` will retrieve only Hairpin entities, while in the second one the "*hairpin*" keyword matches to the literal value of the property `diana:mirnaType` of the RDF class `diana:PaperMirnaConnection`, imposing an indirect constraint to the property `diana:hasMirna`. Moreover, the first query will also retrieve the RDF entities of the connected Hairpins, while the second will not. The data he retrieved from the keyword search request, could provide useful information that would allow the user to further explore the dataset. Also, the user can retrieve results by selecting one of the generated queries, and incorporate them as new resources in his LinkZoo account, in order to annotate them and share them with his collaborators.

4.1 Preliminary Evaluation

In order to evaluate the search with query candidates, we perform an effectiveness study. We have asked our biologists collaborators to provide keyword queries along with a description in natural language of the required information. We have aggregated 5 queries for the DIANA dataset. An example query is "*"Alzheimer's disease" mature"* and the corresponding description is *"Retrieve all mature miRNAs that are related with Alzheimer's disease"*. To evaluate the effectiveness of our generated queries we order them in reverse order given the number of triple patterns they contain and we calculate the Reciprocal Rank metric defined as RR = 1/r where r is the rank of the correct query. Given our problem definition, a query is correct if it matches the information needs as explained in the provided natural language description. Figure 5 shows the Reciprocal Rank we have calculated for the 5 queries for the DIANA dataset. In the 4 out of 5 queries, we got an RR of 1 meaning that we were able to get the information required by the users.

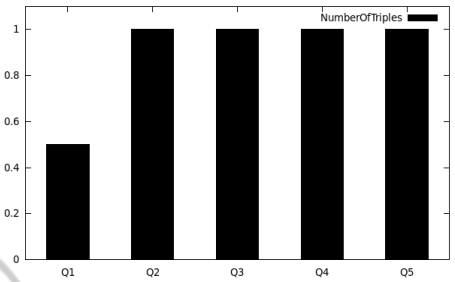

Figure 5: Reciprocal Rank for the DIANA dataset.

5 RELATED WORK

Collaborative editing and annotating has been explored and addressed thoroughly on the schema level. Tools available for collaborative ontology editing are presented in (Auer et al., 2006; Farquhar et al., 1997; Tudorache et al., 2013). However, these require expertise on the schema level. Regarding the management of heterogeneous resources, Personal Information Management (PIM) systems and tools have been implemented, employing common representation semantics as an abstraction layer (Bernardi et al., 2011; Franz et al., 2007; Sauermann et al., 2006). However, these address the management of resources in non-collaborative communities and are thus limited to individual usage.

On the other hand, the keyword search problem over structured data, tree structured (Cohen et al., 2003; Kimelfeld and Sagiv, 2006) or graph structured (He et al., 2007; Bhalotia et al., 2002), is a problem that has widely been explored. Basic steps in those works involve 1) mapping the keyword elements to data elements 2) searching for substructures on the data that connect the keyword elements and 3) return as output the substructures given a scoring function. (Tran et al., 2009) proposed a different solution to the keyword search problem, where instead of computing for the answers directly, it computes structured queries allowing the user to choose the appropriate one. LinkZoo's approach on keyword search with query candidates follows (Tran et al., 2009) approach to generate SPARQL queries, but uses a different exploratory method. In particular, we create multiple augmented graphs one per keywords combination

and use the notion of shortest paths to create a query pattern graph.

6 CONCLUSIONS

In this paper, we have presented LinkZoo, an IT infrastructure for collaborative management of heterogeneous resources on the Web. LinkZoo provides an environment for modelling and publishing data such as files, websites, datasets and people as RDF, and allows for their coexistence in shared contexts. Furthermore, we have presented the various types of search capabilities implemented in the platform. These span from trivial text searching within a user's data to more elaborate data-guided exploration and searching over imported RDF data collections. Finally, we have demonstrated the usability of the search functions through a use case drawn from the life sciences domain.

Currently, keyword search expects exact matches of terms. In the future, we will extend this functionality to automatically suggest terms from the RDF data graph. Another direction will be to extend the matching procedure by enabling also ontology matching (Euzenat and Shvaiko, 2013). Furthermore, to assist user understanding of the produced candidate SPARQL queries, we intend to also show natural language descriptions of the generated candidates. Finally, we also plan to perform an extensive evaluation of our search services, in term of completeness of the results, time and memory requirements for indices creation, performance.

ACKNOWLEDGEMENTS

This study has been partially supported by LODGOV project, Research Programme ARISTEIA (EXCELLENCE), General Secretariat for Research and Technology, Ministry of Education, Greece and the European Regional Development Fund, and the Operational Program "Education and Lifelong Learning" of the National Strategic Reference Framework (NSRF) - Research Funding Program: Thales. Investing in knowledge society through the European Social Fund.

REFERENCES

Alchemy API, 2015 Available from: <http://www.alchemyapi.com/>. [2015].

Auer, S., et al. 2006, 'OntoWiki–a tool for social, semantic collaboration', *Proceedings of ISWC,* ed. Springer Berlin Heidelberg, pp. 736-749.

Bernardi, A., et al. 2011, 'The NEPOMUK semantic desktop', *Proceedings* of *CSKM,* Springer Berlin Heidelberg, pp. 255-273.

Bhalotia, G., et al. 2002, 'Keyword searching and browsing in databases using BANKS', *Proceedings of ICDE,* pp. 431-440.

Bizer, C., et al. 2009, 'Linked data-the story so far', *Int. J. Semantic Web Inf. Syst.,* vol 5, no 3, pp.1-22.

Bizer, C., & Schultz, A. 2008, 'Benchmarking the performance of storage systems that expose SPARQL endpoints', *Proceedings of WWW*.

Cohen, S., et al. 2003, 'XSEarch: A semantic search engine for XML', *Proceedings of VLDB*, pp. 45-56.

Euzenat, J., & Shvaiko, P. 2013, *Ontology matching* (2nd edition), ed. Heidelberg: Springer-Verlag.

Farquhar, A., Fikes, R., & Rice, J. 1997, 'The ontolingua server: A tool for collaborative ontology construction', *International journal of human-computer studies*, vol. *46, no.* 6, pp. 707-727.

Faye, D. C., Cure, O., & Blin, G. 2012, 'A survey of RDF storage Approaches', *ARIMA Journal*, vol.*15*, pp.11-35.

Franz, T., Staab, S., & Arndt, R. 2007, 'The X-COSIM integration framework for a seamless semantic desktop', *Proceedings of K-CAP,* pp. 143-150.

He, H., et al. 2007, 'BLINKS: ranked keyword searches on graphs', *Proceedings of SIGMOD,* pp. 305-316.

Kimelfeld, B., & Sagiv, Y. 2006, 'Finding and approximating top-k answers in keyword proximity search', *Proceedings of SIGMOD-SIGACT-SIGART* pp. 173-182.

Meimaris, M., Alexiou, G., & Papastefanatos, G. 2014, 'LinkZoo: A linked data platform for collaborative management of heterogeneous resources', *Proceedings of ESWC,* pp. 407-412.

Prud'Hommeaux, E., & Seaborne, A. 2008, 'SPARQL query language for RDF', *W3C recommendation.*

Sauermann, L.,et al. 2006, 'Semantic desktop 2.0:The gnowsis experience', *Proceedings ISWC,* pp.887-900.

Tran, T., et al. 2009, 'Top-k exploration of query candidates for efficient keyword search on graph-shaped (rdf) data', *Proceedings of ICDE,* pp. 405-416.

Tudorache, T., et al. 2013, 'WebProtégé: A collaborative ontology editor and knowledge acquisition tool for the web', *Semantic web*, vol. 4, no. 1, pp. 89-99.