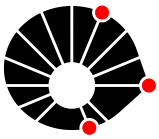
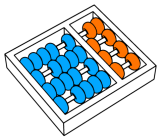




Link Visualization and Evaluation in LOD

Vitor Kenji Uema, Julio Cesar dos Reis

Relatório Técnico - IC-PFG-20-25
Projeto Final de Graduação
2020 - Dezembro

UNIVERSIDADE ESTADUAL DE CAMPINAS
INSTITUTO DE COMPUTAÇÃO

Link Visualization and Evaluation in LOD

Vitor Kenji Uema, Julio Cesar dos Reis*

December 2020

Abstract

In Linked Data, data sets should include links to others by interlinking new information with existing resources. This must improve the "discoverability" of new and existing data in LOD (Linked Open Data) cloud. However, this is somewhat overlooked because finding and maintaining links between different data sets are a costly and time consuming task. In this work, we designed and implemented an interactive system to support quality inspection of links. This must help and incentive LOD players to find and maintain links. In our approach, evaluating a link is based on two graphical representations: a radar chart and a network graph. The radar chart is used to display similarity values between the subject and the object of the link; and the network graph is used to display the link's subject and object. We present to which extent our proposed interaction mechanisms help users to evaluate the correctness of a link, which is an essential task for preserving data quality in LOD data sets.

1 Introduction

Nowadays, the web is the biggest infrastructure for sharing information. One of the aspects that made the web such a great place to share information is the Hypertext Transfer Protocol (HTTP), where hypertext documents can include hyperlinks to other resources that anyone can easily access. Moreover, HTTP together with HTML (Hyper Text Markup Language) and URL (Uniform Resource Locator) makes the internet the greatest place to publish, retrieve and discover information, at least for humans. HTML describes to computers how to display documents. In other words, how to show a web page to a human being throughout a display. That is, HTML is designed to be consumed by humans not computers, thereby different services and applications cannot freely exchange and consume all the information available in the web.

*Institute of Computing, University of Campinas, SP

Tim Berners-Lee wrote it back in 2001 wrote the following: *"Most of the Web's content today is designed for humans to read, not for computer programs to manipulate meaningfully"* [2]. A web where computers can comprehend the meaning embedded in the web is what Tim Berners called the Semantic Web [2]. In 2011, IBM Watson won the Jeopardy game-show and in the same year Apple release Siri the virtual assistant. Both IBM Watson and Apple Siri share a common trait: they rely on structured information resources such as DBpedia[1] and Geonames[2]. In this context, What was only accessed and consumed by humans started to be consumable by computers. Even though these type of systems became famous in the last ten years, in the biomedical area, the development of ontologies are dated back to 1998 when researches started to develop the gene ontology. Since then, several other ontologies have been developed such as the vaccine ontology, the disease ontology and the human phenotype ontology, to name a few.

Nonetheless, all the structured datasets openly available on the web should not remain as silos. They need to be linked as those files containing hyperlinks back in the beginning of the web. The problem of LOD linkage presents many difficulties. For example, the links between two different resources from different datasets need to be semantically correct in the sense that the semantic information should be kept intact as originally designed by its authors. Otherwise, the link may be considered as broken or incorrect.

Several algorithms, software tools and frameworks have been developed to produce links. For example, Silk [3], LogMap [9] and Limes [11]. Approaches to maintain and update links over time have also been developed [12]. Different frameworks might use different strategies and as a result the link outcome produced may also differ.

In this work, we developed an interactive software tool that aims to provide ways to users visualize and analyze produced links between LOD datasets. Our tool must help to ensure data quality which is essential because several other systems need a large amount of "good" data. One way to guarantee data quality is through manual data validation. In this perspective, our tool help users to understand and visualize links generated by link discovery frameworks or link maintenance frameworks. In our approach, necessary information to evaluate links are presented in two different formats: a radar chart and a network graph. The radar chart displays the syntactic and semantic distance value between resources involved in the link; the network graph displays the link's subject and object.

At the current status, our tool supports input links from the *Linked Open Data Maintenance Framework* (LODMF) [13]. This framework detects changes in RDF datasets and their links. RDF is a model that encodes data in the form of subject, predicate and object through URIs. The subject and the object identify the resource and the predicate informs how they are related. For each computed change, the

[1] https://wiki.dbpedia.org/

[2] https://www.geonames.org/

framework checks if the change "brokes" underlying links. In case of a broken link, the LODMF suggests some amend suggestion for the link maintenance. For each broken link, our software tool presents to the user the amend suggestions and ways of comparing each modified link, so the user can choose the most suitable amend.

The remaining of this document is organized as follows: Section 2 presents Fundamental Concepts and Related Work; Section 3 details our proposed Link Evaluator software tool; Section 4 presents implementation details of the tool; Section 5 discusses the obtained findings; finally, Section 6 presents the conclusion and summary of the work's contributions. Appendix A presents details of the Components Life Cycle.

2 Background

This Section presents the fundamental concepts for understanding this work (*cf.* Section 2.1), in addition to a synthesis of recent works developed in this line of investigation (*cf.* Section 2.2).

2.1 Fundamental Concepts

Linked Data is not just about making data available on the web, but making links between them so they can be explored. They refer to structured information meant for computers, in the sense that data semantics are computer interpretable. One way to structure linked data is via RDF[3] graph (Figure 1). RDF graph is a directed named graph, which every triple (node, edge, node) translates to a well-structured sentence (subject, predicate, object). For example, in Figure 1, the triples:

(`https://dbpedia.org/resource/Wolfgang_Amadeus_Mozart`,
`http://dbpedia.org/property/composer`,
`http://dbpedia.org/resource/Requiem_(Mozart)`)

translates to the phrase *"Wolfgang Amadeus Mozart composed Requiem"*.

RDF aims to make statements about resources using triples. The subject denotes a resource and the predicate expresses a relationship between the subject and the object. The object can be neither a resource or a literal value. A resource is a reference to real things such as: persons, books and places. Each resource is identified by a URI (Universal Resource Identifier). As a rule of thumb, each URI is made of two parts. First comes the namespace then comes the resource identifier. For example, in the URI <`https://dbpedia.org/resource/Wolfgang_Amadeus_Mozart`>, <`https://dbpedia.org/resource/`> denotes the Dbpedia resource namespace and *"Wolfgang_Amadeus_Mozart"* is an identifier. Usually the namespace is abbreviate,

[3] i.e Resource Description Framework

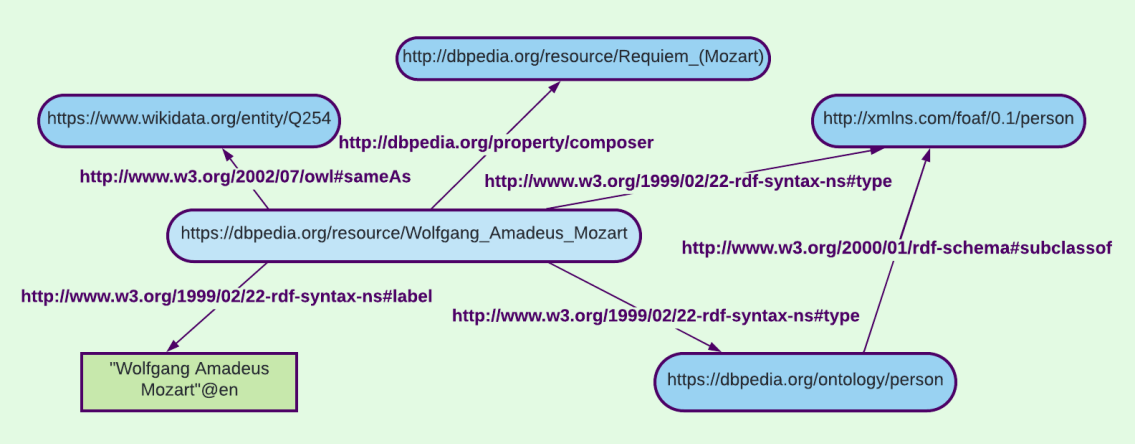


Figure 1: RDF Graph example (inspired by Dbpedia [1]).

so the <`https://dbpedia.org/resource/Wolfgang_Amadeus_Mozart`> is normaly seen as <dbr:Wolfgang_Amadeus_Mozart>.

Not all nodes are resources; some nodes are literal which represents concrete data such as numbers, strings, Boolean, etc. Usually, literals are represented in rectangular boxes, as exemplified in Figure 1 by the string *"Wolfgang Amadeus Mozart"@en*. All data resources are serialized using many formats such as: N-Triples, RDF/XML, Turtle and RDF/JSON. The Listing 1 shows the Turtle serialization of the graph in Figure 1.

```
@prefix dbr: <https://dbpedia.org/resource/> .
@prefix dbo: <https://dbpedia.org/ontology/> .
@prefix wd: <https://www.wikidata.org/entity/> .
@prefix foaf: <http://xmlns.com/foaf/0.1/> .
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .

dbr:Wolfgang_Amadeus_Mozart rdf:type dbo:person ;
    rdf:type foaf:person ;
    rdf:label "Wolfgang Amadeus Mozart"@en ;
    dbo:composer dbr:Requiem(Mozart) ;
    owl:sameAs wd:Q254 .

dbo:person rdfs:subClassOf foaf:person .
```

Listing 1: Turtle example.

Focusing on triples, we categorize them in two groups: literal triples and RDF Links [8]:

1. **Literal Triples** have an RDF literal such as: a string or a number as the object. Literal triples are used to describe the properties of resources.

2. **RDF Links** describe the relationship between two resources (subject and object). In addition, the predicate defines the type of the relationship between the resources.

RDF links can be further distinguish based on the predicate and resources location. For example, if links' predicate is *rdf:type*, the link is a type property link. If both resources reside in the same dataset, the link is an internal link. Otherwise, it is a external link.

Of all possible classifications, we emphasize the identity link, also know as the instance link. The identity link are represented by the predicates: *owl:sameAs*, *skos:exactMatch* and variations. It defines that the link's subject and object, representing the resource, stand for the same concept. An example of an identity link is *<dbr:Wolfgang_Amadeus_Mozart owl:sameAs wd:Q254>*. This link shows that both resources, although in different namespaces, represent the same thing. Identity plays an important role in LOD because it bundle related resources from multiple datasets, improving the knowledge of a specific resource.

2.2 Related Work

Haller *et al.* [7] proposed an empirical linkage analysis of 430 datasets from the LOD Cloud. They mainly analyzed the "dereferenceability" of links found in those datasets. The authors showed that a great proportion of links are broken. Neither the subject or object of links are not reachable through HTTP lookup.

Spanhiu *et al.* [14] presented a framework to profile the quality of *owl:sameAs* links in the LOD Cloud. Their work used two similarity metrics: a string similarity and a numeric similarity (*e.g.* how close two number are). To evaluate a instance link, they calculate the similarly score between each property of the instance link's subject and object, then all similarity scores are aggregated weighting all values using the geometric progression of 75% increase.

Ben Ellefi *et al.* [6] proposed a dataset recommendation approach for data link. They profile multiple datasets in order to find two datasets that models similar concepts. For example, two datasets that model geographical concepts. Then these two datasets can be linked using an automatic link discovery framework, such as Silk [3]. To profile a dataset, they retrieve a list of literals that represent the dataset's "theme", then they mesure how similar the list of literals are using a semantic metric based on the frequency of term co-occurence in a large corpus (bag of words) combined with a semantic distance based on WordNet.

Regarding general dataset qualities, such as re-use of existing terms, correct URI usage and low blank node usage, Debattista *et al.* [5] proposed a conceptual methodology to profile and assess the quality of Linked Datasets (a framework for Linked Data Quality Assessment). It provides a graphical user interface that shows metadata

about the quality of a dataset. Although it is not focused on instance link quality, this contribution was used as a motivation for our work.

Regarding the maintenance of links, Regino *et al.* [13] developed a Linked Open Data Maintenance Framework (LODMF). This framework provides mechanisms to detect evolution in RDF datasets and how to maintain link up-to-date. To this end, the framework provides maintenance actions for links, in case it found a broken link during the evolution detection process. The maintenance actions can be suggestions for modify the subject (ModifySub), predicate (ModifyPred) or object (ModifyObject) of the link, remove a link (RemoveLink) or keep the link unchanged (UnchangedLink).

Most of the literature focus on profile links with other objectives than the maintenance of an instance link. However, the techniques used to profile and assert the correctness of a link can be used to solve the maintenance of an instance link. Moreover, data quality assess frameworks can be used as motivation and inspiration to design a instance link maintenance graphical interface.

3 The Link Evaluation and Visualization Software Tool

Our tool aims to offer a way to users visualize and compare links generated based on link discovery or link maintenance services. In particular, we used the output from the link maintenance service [13], which generates a list of link suggestion to fix broken links.

The main functionality designed is the display of similarity metrics and graph visualization, so a human can assert the correctness of a link. In order to evaluate the correctness of a link, we need to compare the subject and the object in the context of the predicate. For example, to evaluate the following link: *<geonames:CityOfLondon owl:sameAs dbpedia:London>*, first, it is necessary to obtain prior knowledge on the meaning of the predicate. In this case, *owl:SameAs* means that both subject and object should represent the same resource. On this basis, it is necessary to assert the properties of both subject and object. If they have similar properties, they possibly represent the same resource.

In our tool, the evaluation can be conducted in two different ways: through a radar chart or through a network graph. The radar chart is used to display normalized syntactic or semantic similarity values. Furthermore, any other metric can be displayed in the radar chart such as an aggregated metric composed by a syntactic similarity metric and a semantic similarity metric. Additionally, the network graph works as a double checker for evaluation purpose.

In the following subsection, we explain how the user interact with the software tool occurs, starting from the initialization of the application to the evaluation of a

link. Furthermore, we explain the design decision taken during the development of the tool.

3.1 User Workflow

Figure 2 presents the main interaction flow in the software tool. When entering the system, the user has two options: continue a task or start a new task (*cf.* step A in Figure 2). If the user chooses to start a new task, (s)he must enter all required data. It includes the source data set $V0$; source data set $V1$; the target data set; the predicates list; and the background knowledge file (*e.g.* WordNet[4] or BabelNet[5]), a semantic network used to calculate semantic metrics (*cf.* step B in the Figure 2). These are the required files for the LODMF. After the input phase, it is sent to the sever, that runs all the computation. When the server finishes its computation (*e.g.* finds all broken links, generates amend suggestions, calculates all similarity metrics), it sends the results on demand to the application.

Completed the task creation phase, the user experiences the overview dashboard. This dashboard enables the user visualize the itemized percentage (%) of verified links and other statistics about the server output (*cf.* step C in the Figure 2).

The loop from step D to step G represents the process of evaluating each link. In step D, the user selects one link to analyze - the list of links are itemized according to changes in the link's subject, predicate or object detected by the LODMF (*cf.* step D in Figure 2). After the link selection, the user visualizes the link evolution dashboard (*cf.* step E in Figure 2) - this dashboard presents detailed information about the link evolution along with maintenance action suggestions. Next, the user can compare all the maintenance suggestions through the analysis of the radar chart and the network graph (*cf.* step F in Figure 2). Finally, the user decides what is the final action on the under analysis link, eg., choose any of the fix suggestions or remove the link (*cf.* step G in Figure 2). From step G, the user can select another link to analyze (go back to step D) or save the selected action.

In the following subsections, we present the tool interfaces related to each of the steps illustrated in the interaction flow (Figure 2). For each subsection, we indicate in the storyboarding the interface of the system which it represents.

3.1.1 Step A and B: Create New Task and Submit Input Form

First of all, the system presents to the user an welcome screen (Figure 3a) in which (s)he can take two actions: continue a link analysis task or create a new one. The user can continue a previous link analysis task by clicking in one of the recently opened task or by clicking on the open task. (S)he can create a new link analysis

[4]https://wordnet.princeton.edu

[5]https://babelnet.org

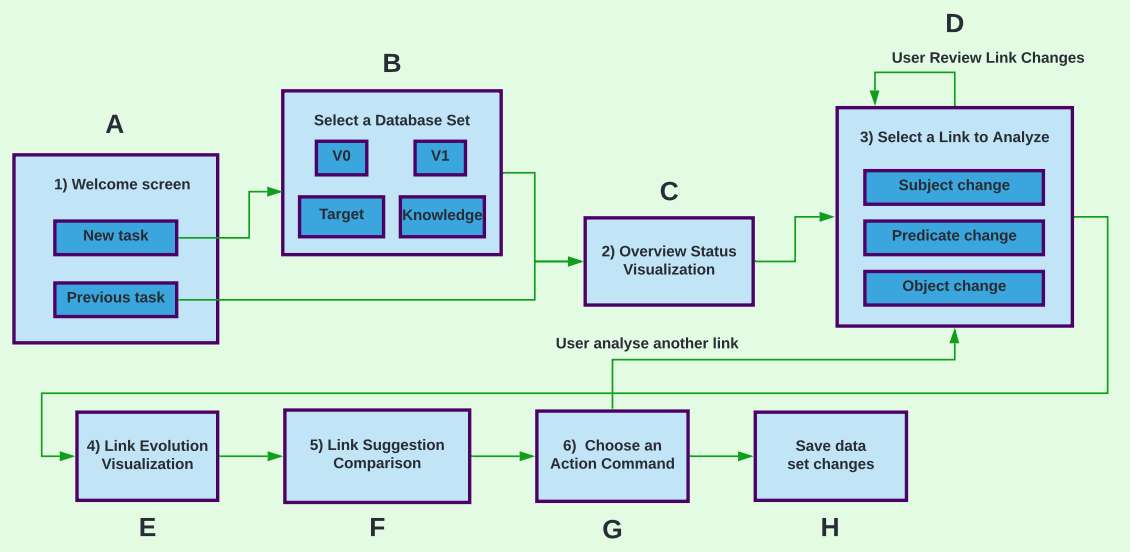


Figure 2: User Interaction Flow.

task by clicking to create a new task. By doing so, an upload screen appears (Figure 3b), then the user can click cancel and go back to the welcome screen; or (s)he can fill all input formats and click on upload. After click on upload, all data sets and the predicates are sent to the server.

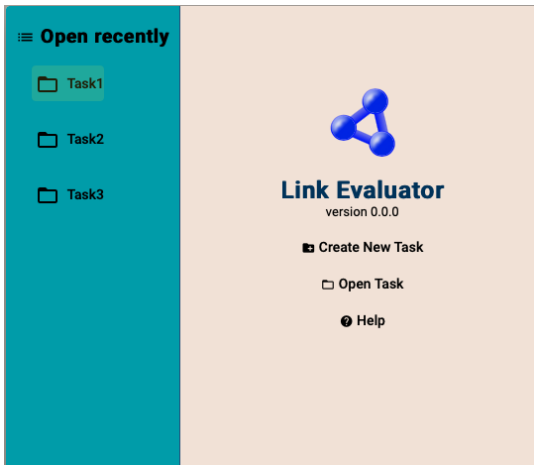
3.1.2 Step C: Overview Status Visualization

After the serve response, the system presents to the user an overview dashboard (Figure 4). In the left side, it stays the links itemized by the changed happened. In the right side, it appears an overview dashboard. Two pieces of information compose the overview dashboard. First, general information about the task - which data sets were used and overview statistics about the output (*e.g.*, total number of unchanged links and total number of removed links). Second, how many links the user have verified, discarded and unchecked, presented in a donut chart type.

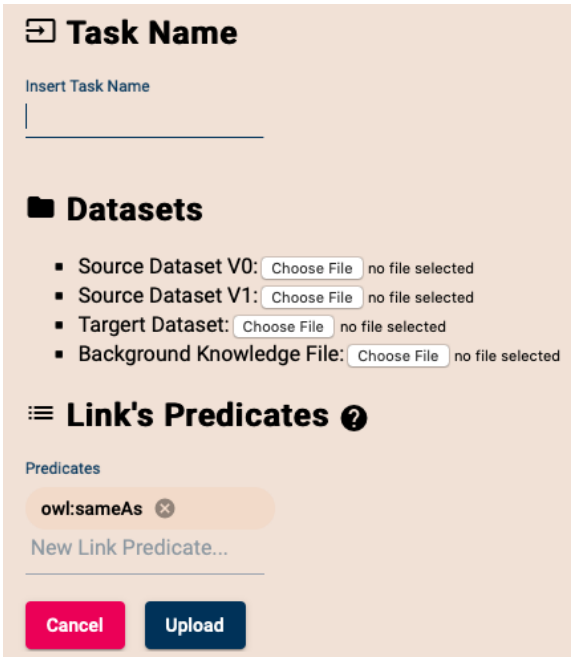
The main objective of the overview dashboard is to remind the user of the current status of the task. That is why it is shown right after the initialization of the application. In summary, the overview dashboard shows how much work has to be done and what percentage of the work has been done. Furthermore, the progress bar in the bottom left tracks the work completed.

3.1.3 Step D: Click on a Link in the Index

The left side of Figure 4 can be expanded to show the links (Figure 5); then the user can select a link to analyze. The links are divided into three groups: (1) modified



(a) Welcome Dialog.



(b) Upload Dialog.

Figure 3: Welcome and upload dialogues.

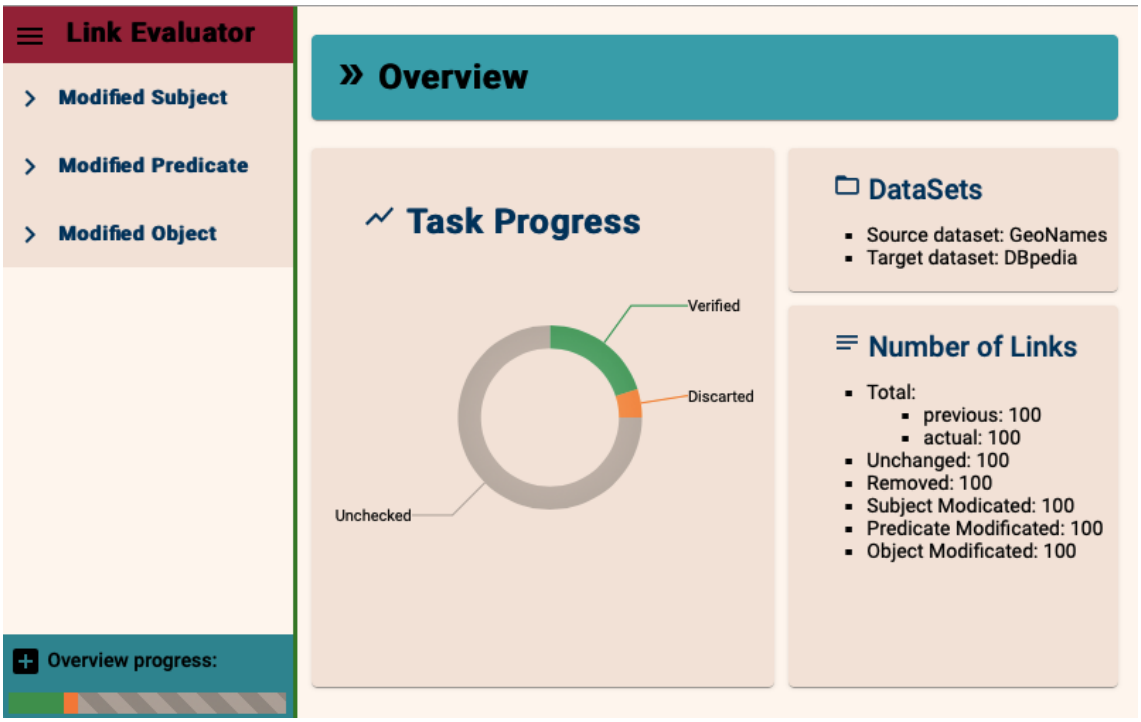


Figure 4: Overview status screen.

subject, (2) modified predicate and (3) modified object. These are links whose the *LODMF* consider to be broken. As a form of abbreviation, the predicate of each link is replaced by a mathematical symbol representing the semantic meaning of the predicate.

In fact, *owl:SameAs*, *skos:exactMatch* are represented by the equal sign (=); *skos:closeMatch* by the approximate sign (≈) and *owl:differentFrom* by the different sign (≠). The subject and object are abbreviated to the data set name followed by a label. All these abbreviations aim to shorten the link for display reasons and to bring semantic tips because the complete URL of a link can be quite long and hard to understand. For example, the link:

<`https://sws.geonames.org/2643743`
owl:sameAs
`http://dbpedia.org/resource/London`>

is too long and the meaning is not obvious, therefore it is abbreviate to:

<gn:London = dbr:London>.

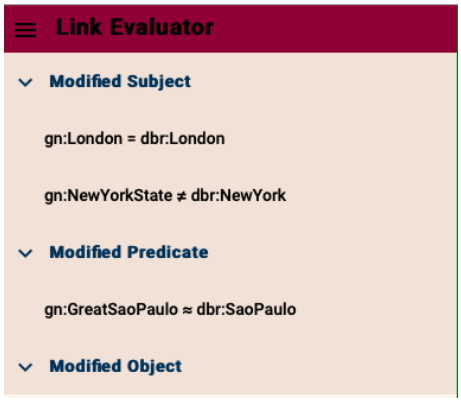


Figure 5: Index Example.

3.1.4 Step E: Link Evolution Visualization

When the user clicks at any link, specific information about the link appears in the right side (Figure 6), along with a new tab - blue indicates which tab the user is currently seeing (item 1 in Figure 6). This dashboard shows the link evolution. In other words, how the link was in the source data set $V0$ and how it is at the current version in the source data set $V1$ (item 3 in Figure 6). The link evolution dashboard shows the maintenance suggestions (item 2 in Figure 6); more suggestions appear when the user clicks on the show more suggestions. Additionally, in this dashboard, the user can select multiple links suggestions to compare (item 4 in Figure 6).

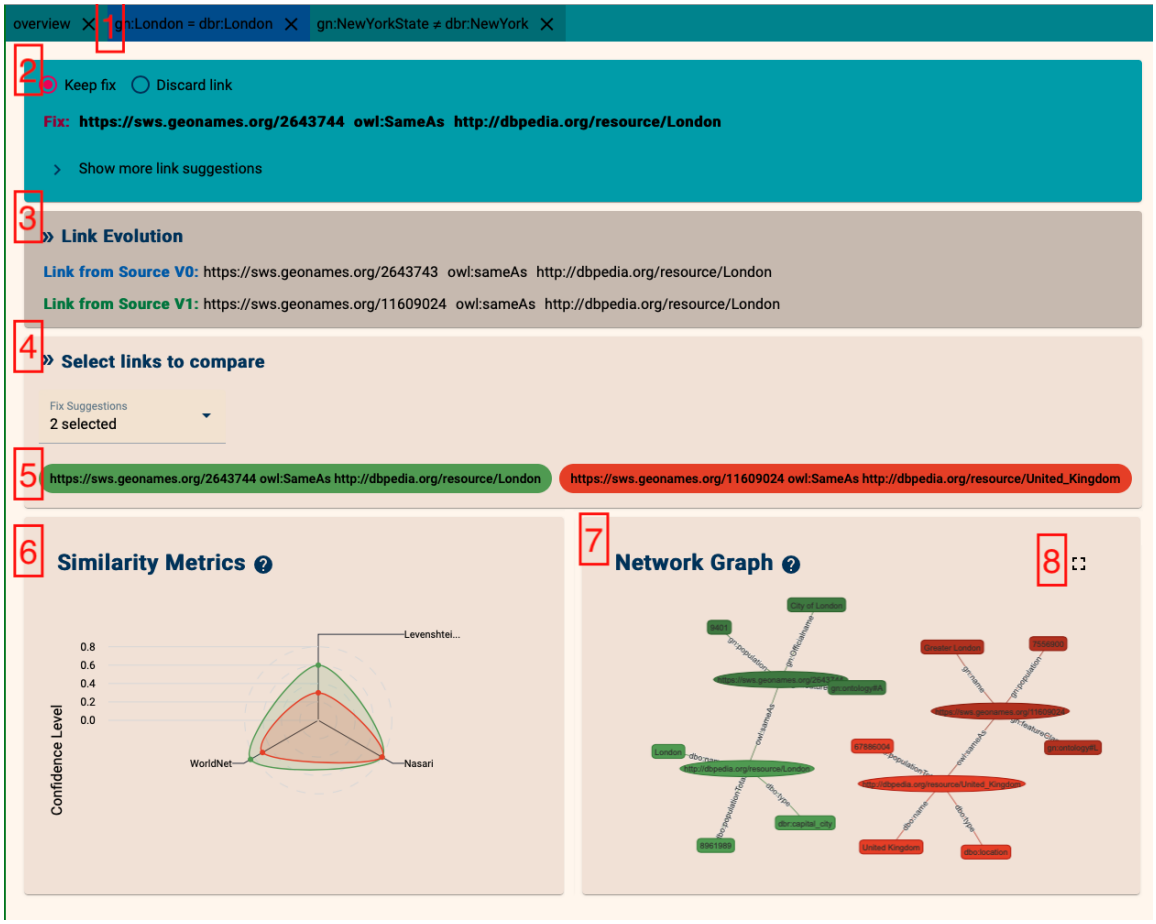


Figure 6: Link Evaluation.

3.1.5 Step F: Link Suggestion Comparison

The user can select multiple link amend suggestion from the selector (Figure 7). All selected links appear in the chip list bellow the selector, each one with a different color (item 5 in Figure 6), matching the color used in the radar chart and network graph. When choosing multiple amend suggestion, the user can use the radar chart and the network graph to compare multiple amend suggestion, then s(he) can choose the best suggestion.

In the radar chart, the user can hover through to analyze the precise metric value. In the network graph, the user can use the mouse to navigate and zoom the graph. Clicking on the expand icon (item 8 in Figure 6), the network graph enters in full-screen mode.

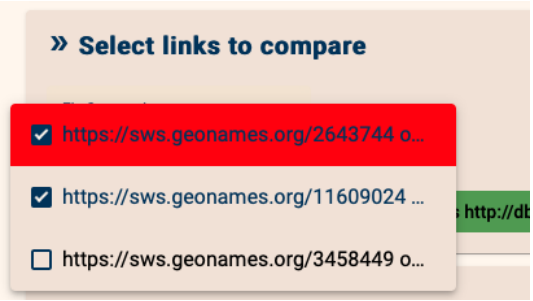


Figure 7: Select links to compare.

The radar chart displays three different metrics: one string similarity metric (Levenshtein distance [10]) and two semantic similarity metrics. One using the NASARI semantic vector [4] representations and other using the WordNet knowledge base, the was calculated based on how many nodes are between two terms.

In the radar chart (item 6 Figure 6), the user can visualize all computed values of metrics at once. Therefore, (s)he can decide his own aggregation function. The user can decide which metrics are more important and mentally assign weights to each metric and then make a decision about the correctness of an link amend suggestion based on his own judgment. Moreover, the user Conclusively, the user Judges which link amend suggestion is the correct one.

Depending on the information available or the lack of information, only the metrics might not be enough to evaluate the link. Concerning the string similarity metric, two different concepts might use the same label. For example, the Label “London” could represent the capital of the United Kingdom or a city in Ohio, USA. Both labels would yield the same Levenshtein distance value, even though they represent two different things. That is, the metrics might mislead the user and additional information are needed. Hence, the network graph aims to fill any possible gaps and doubts left by the similarity metrics.

3.1.6 Step G: Choose an Action Command

After comparing the amend links suggestion and electing one, the user can choose what s(he) wants to do with the broken link (Figure 8). (S)he can take three actions: keep the first suggestion, change to any other suggestion or discard the link. To change the amend link, the user can click on show more suggestions and choose any link from the list; the selected link appears above the list of suggestions. Furthermore, the user can choose if (s)he wants to use one of the amend options or (s)he wants to remove the link from the data set.

Figure 8: Select Maintenance Action.

4 Implementation

Figure 9 presents the overall system architecture. The back-end side needs to implement a REST API and provides three services: (1) metrics and statistics calculator, which calculates all the similarity metrics used in the radar chart; (2) a database, which generates the network graph; (3) A link generator service, which generates the link list.

4.1 Technologies

We present the main technologies explored in our development.

Angular[6] is currently in the 11.0.0 version. It is a popular framework also knows as Angular 2+, which is the successor of AngularJS (Angular 1+). From now and so on, we refer to Angular 2+ as Angular.

[6] `https://angular.io`

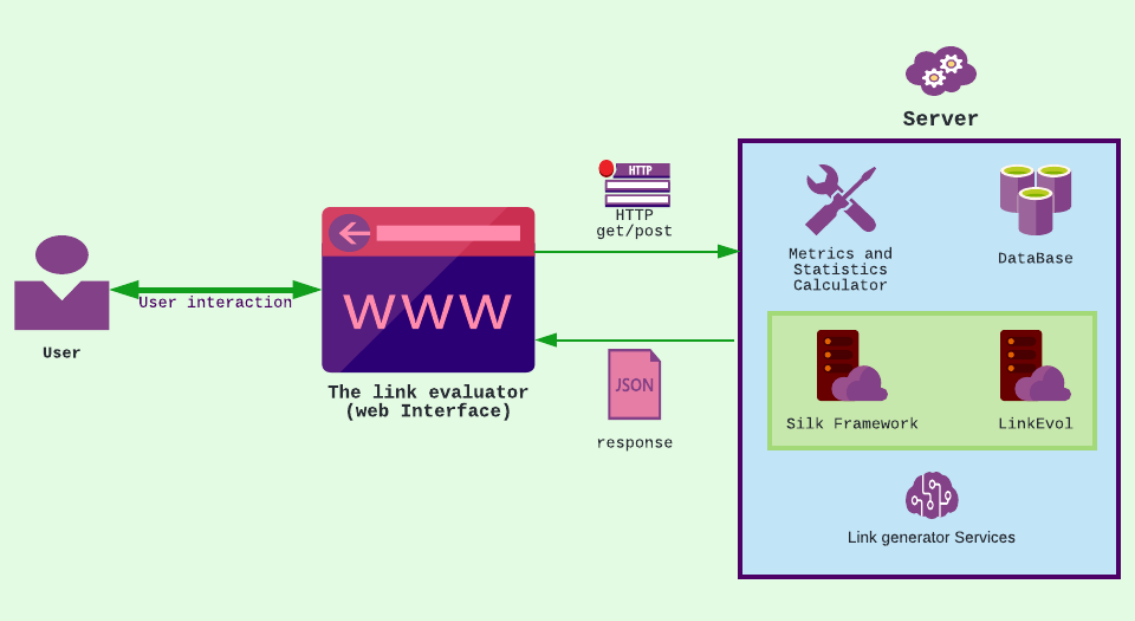


Figure 9: System Architecture.

An Angular application is a composition of modules. A module is a mechanism to group components, directives, pipes and related services. A component is the main building block of an Angular application and consists of a typescript class; HTML files and a CSS file. Likewise, the services, pipes and directives are the connectors that stay between components. They provide ways to components interact with each other.

Angular Material[7] provides UI components based on Google's Material Design and it was used to bootstrap and accelerate our development process.

Ngx-charts[8] based on d3[9], ngx-charts was used to render all charts. Ngx-charts provides a common chart which can be used out of the box, but it provides ways to generate the custom chart. Ngx-charts was used instead of pure d3.

Vis.js[10]: is a dynamic, browser based visualization library. The library is designed to be easy to use and to enable manipulation of and interaction with the data. The library consists of the components: DataSet, Timeline, Network, Graph2d and Graph3d. However, only the Network component was used in our implementation.

REST API[11] is based on representational state transfer (REST), which is an architectural style and approach to communications often used in web services de-

[7] `https://material.angular.io`
[8] `https://swimlane.gitbook.io/ngx-charts/`
[9] `https://d3js.org`
[10] `https://visjs.org`
[11] `https://www.redhat.com/en/topics/api/what-is-a-rest-api`

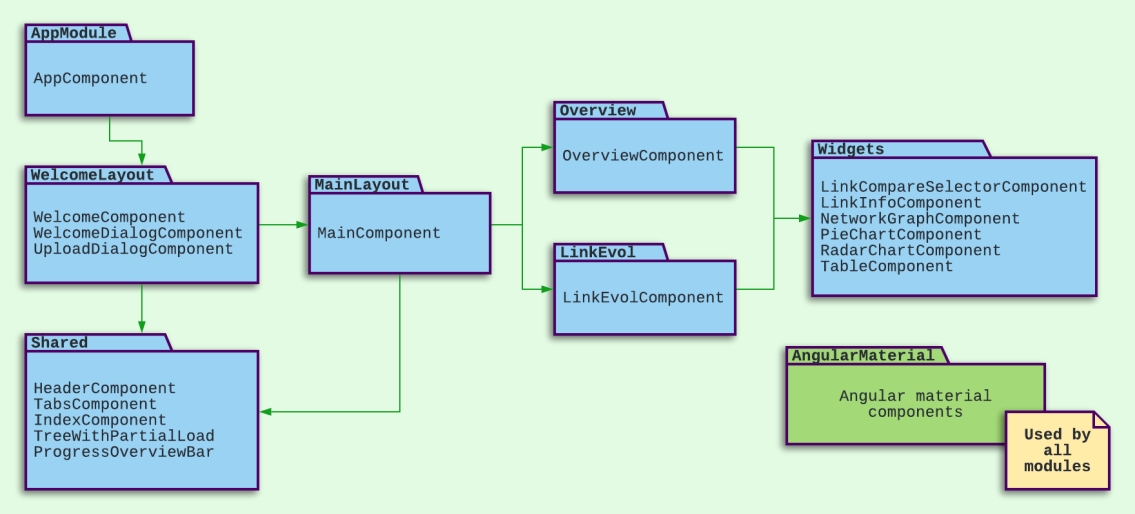


Figure 10: Modules Diagram.

velopment. All the requests managed through HTTP. That is, the HTTP requests: GET, PUT, POST and DELETE refer respectively to: reading, updating, creating and deleting data operations.

4.2 Code Design

The front-end application contains eight modules, as shown in Figure 10. Four modules: Welcome Layout Module; Main Layout Module; Overview Dashboard Component; and LinkEvol Dashboard Module deal with the organization, layout and interconnection of components. Two modules: the shared module; and the widgets module have all the components used in our development.

The AppModule contains only the AppComponent, the start point of the tool. The SharedModule presents all the components used by the Welcome Layout Module and Main Layout Module. All components in this model deal mostly with app navigation. For example, the TabsComponent displays the opened tabs and it does the navigation through the opened tabs, similar to the tabs in a web browser.

The Widgets Module has all the components used by the Overview and LinkEvol Modules. All the components in this module neither deals with data display or action selection. The *NetworkGraphComponent*, the *PieChartComponent*, the *RadarChartComponent* and the *TableComponent* receive input data and displays it. Furthermore, the *LinkInfoComponent* and the *LinkCompareSelectorComponent* receive a list of link amend suggestions as input and output the list of selected links IDs. The output can be passed to *RadarChartComponent* and *NetworkGraphComponent* as an input.

The Main Layout Module is responsible to anchor the *OverviewComponent* and

the *LinkEvolComponent*. By doing so, all components in the shared and widgets module can be reused and a new layout (screen) can be easily created. The only things that need to change is the routers[12] to fit the new layout into the navigation flow. The Welcome Layout Module defines the welcome and upload dialogues, along with the interaction between the two dialogues.

The Overview and LinkEvol Modules behave similarly. They define the position of components from the Widgets Module. Depending on which component they use, these modules need to handle the components' input/output flow. For example, the *LinkCompareSelector* generate a list of link IDs, which *RadarChartComponent* and the *NetworkGraphComponent* use as input. The *RadarChartComponent* and the *NetworkGraphComponent* use the list of link IDs to fetch data from the server.

The AngularMaterialModule keeps organized all the angular material components used in the application. The WelcomeLayoutModule is responsible to display the first two dialogues. More details of all components are present in Appendix A.

5 Discussion

Data has more value if it can be connected to more data. However, many data sets in LOD cloud show a low number of instance links due to the difficulties of the linkage task. Despite the complexity of this task, most link discovery frameworks do not present a way to users evaluate generated links. In addition, link maintenance frameworks may also need to provide a way for a humans evaluate the quality of links.

This work developed a link evaluator software tool to allow users to profile the output produced by those frameworks. In particular, a LOD data set manager or a record linkage researcher can use our link evaluator tool to verify the correctness of theirs links created. Our link evaluator tool can be useful in this context because wrong links in LOD data sets can be harmful to other services, such as: a recommendation service or knowledge discovery services.

The radar chart and the network graph complement each other and present information to users rank link suggestions. Comparing two links with different predicates might be problematic and mislead the user evaluation. For example, considering the links: (1) <*example:London(UK) owl:sameAs example:London(Ohio, USA)*>; and (2) <*example:London(UK) owl:differentFrom example:Londrina(Brazil)*>; the user might choose link (1) as being correct, since the radar chart could present a greater Levenshtein distance value for link (1), however link (2) is the correct one. The radar chart can mislead the user, because the radar chart guides the user to choose links with greater similarity values. But in a case were a link represents how *unsimilar* the subject and object are, the similarity values are low, even though the link is correct. To maneuver around this problem, new metrics that consider the meaning of multiple

[12] `https://angular.io/guide/router`

predicates should be developed or links with different predicate meaning should not be compare.

Regarding the code quality, new components and page layout can be added to the tool without major problems, due to the modularity that Angular provides. However, how the routers are set to navigate through the page structure may need changes, which can cause some difficulties. The server communication may be improved.

Only the front-end side was developed in this work. The server-side representation are left open. How the back-end side must be arranged was not addressed in this work. Upon the back end development, the services and the data exchange interface can be further developed.

For improvements, a usability inspection could be performed to identify problems in the design and learn how the users experience the implemented tool. This can indicate improvements in how to refine the features in the tool.

6 Conclusion

Links play an essential role in LOD, especially instance links. They interconnect different datasets thus enable the creation of larger and richer datasets. Although several frameworks have appeared for link discovery and maintenance, we observed a lack of tools to help users understanding, visualizing and analyzing generated links. We designed and implemented a software tool to provide metrics, statistics and metadata to users evaluate and analyze the correctness of instance links. Similarity metrics are presented in a single radar chart. This enables users to compare multiple links and rank them for selection and curation. The tool implements a network graph containing information on the properties of the subject and object of the link. Our software tool was implemented in a modular way and all components are reusable, which allows further improvements in the tool with minor changes.

Acknowledgements

We thank the São Paulo Research Foundation (FAPESP) (grant #2017/02325-5)[13].

[13]The opinions expressed in here are not necessarily shared by the financial support agency.

A APPENDIX: Components Life Cycle

This section explains how each component works. In summary, for each component, we present its sequence of events, services used and interactions with other components. Of the total 16 components presented and implemented, we discuss 6 of them.

A.1 Welcome Component Life Cycle

The welcome component takes care of steps A and B in Figure 2). Since these steps are composed of two dialogues, one for the initial screen and one for the upload screen, we have two more components to handle each dialog. The welcome dialog components represents the Figure 3a; the upload dialog component represents the Figure 3b (presented in the Section 3.1).

Figure 11 shows the sequence of events from the start of the application to the upload of the user input. When the application starts, the welcome dialog starts, because only the creation of a task event is implemented. Figure 11 does not show any sequence of interactions of opening a task. The upload dialog starts right after the user clicks on to create a task. When all inputs are fulfilled, the user can click on upload, which triggers three events. First, the pload dialog component calls the Links service's upload input method. This method makes a HTTP POST to the server with all input files. Second, it calls the Tabs Service's *addTab* method to add the overview tab. Finally, the Upload Dialog Component closes both dialogues and calls the Router service to navigate to the overview dashboard in the main layout (Figure 12).

A.2 Overview Component Life Cycle

The overview component takes care of the step C in Figure 2). Figure 12 represents the sequence of events that happens from the user upload to the presentation of the overview dashboard. The overview component uses two services and two components. The pie chart component is used to display the number of keep, discarded and not verified links. The overview stats component is used to display general information about the generated links. Since both components receive data from the overview component, first the overview component needs to fetch all data. To do so, it uses two services: overview stats service and saves link service. The overview stats service is responsible to make the HTTP GET request; it returns an observable to the overview component. The save link service returns the status of all links. In summary, how many links should be kept, discarded or unchecked. After the return of all data, the pie chart and the overview stats show the results to the user.

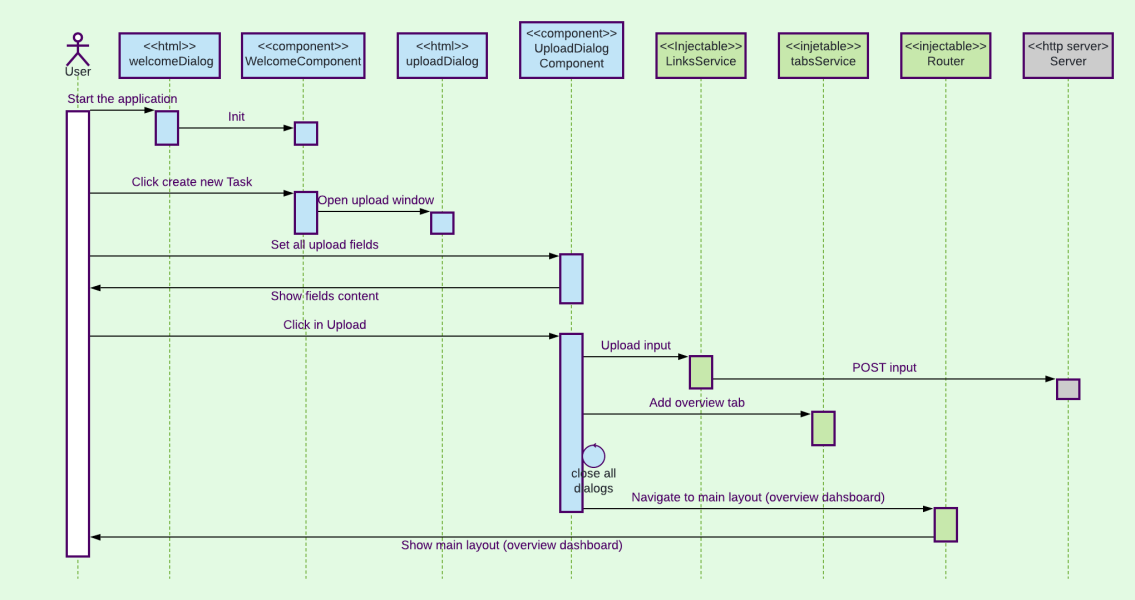


Figure 11: Welcome Component Sequence Diagram.

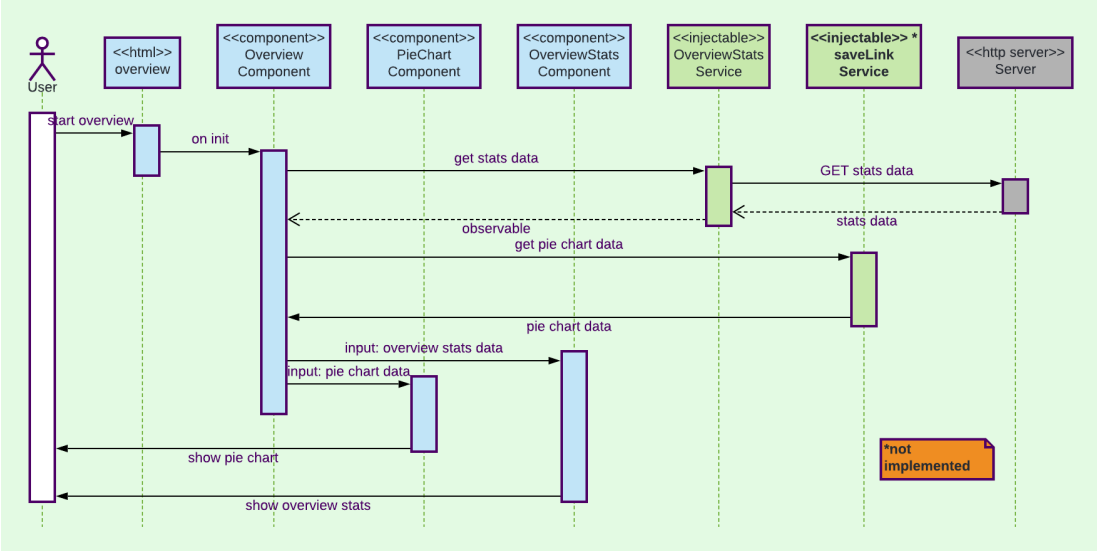


Figure 12: Overview Component Sequence Diagram.

A.3 Index Component Life Cycle

The index component takes care of step D in Figure 2). Figure 13 represents the sequence of events that happens when a user clicks on a link to evaluate it. The index component uses the links Service to make a HTTP GET request, then after the server response it displays the link list. When the user clicks on the link, it triggers two events: first a new tab is added using the tabs service, then the index component uses the router service to navigate to the link evol tab (Figure 6).

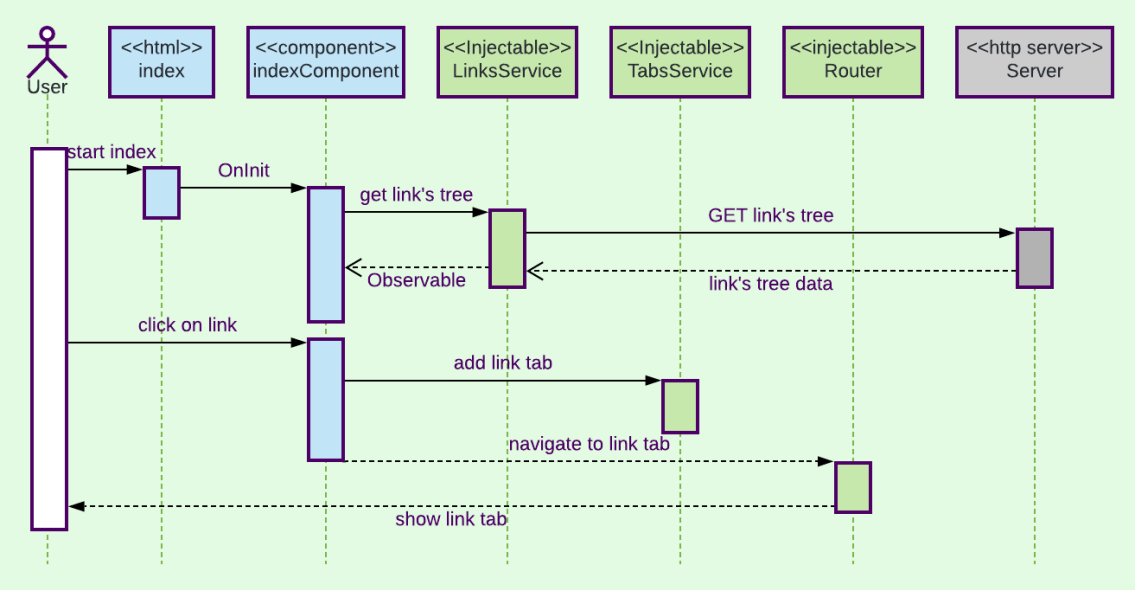


Figure 13: Index Component Sequence Diagram.

A.4 LinkEvol Component Life Cycle

The link evolution component takes care of E and F in Figure 2). Figure 14 represents the sequence of events that happens from the user clicks on a link in the index to the display of all information. The link evolution component arranges the position and the input/output of all widgets used to evaluate the correctness of the suggested links. On start, the link evolution component uses the link service the make a HTTP GET request to get the link information and maintenance suggestions (Figure 6 first to cards). When the server sends the response, the link evol component passes the link information and suggestions to the link info and link compare components, respectively. By default, the selected link to compare is the first link in the suggestions list. This link is used to start the radar chart component and the network graph component.

In the link compare component, multiple links can be selected. When the list of selected list change, all selected link *ids* are sent back to the link evolution component, which uses the output list to feed to the radar chart and network graph. When the radar chart component and network graph receive the input, they use the radar chart service and the network graph, respectively, to fetch data from the server. These two services send the list of links *id* to the server to get the radar chart data and the network graph data. When the server sends the response, the radar chart and the network graph are displayed.

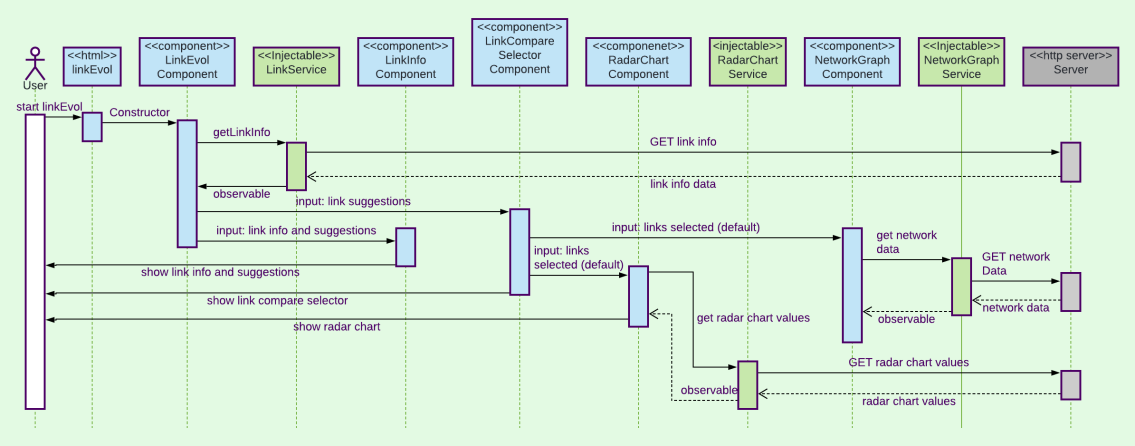


Figure 14: LinkEvol Component Sequence Diagram.

A.5 LinkInfo Component Life Cycle

The link info component is used to show the currently selected fix link. It also shows other link suggestions along with the action the user can take (keep fix link or discard link). On start the link info component receives the input from the link evol component and displays the default select fix link. Then the user can take three actions: (1) click on show more suggestions; (2) select a new link from the suggestions; (3) decide if (s)he keeps one of the fix suggestions or discard the link. Figure 15 shows one possible permutation of the three actions mentioned.

A.6 Tabs component life cycle

The tabs component is used to change the displayed dashboard. Figure 16 shows one one possible sequence of event. On start, the tabs component gets the list of opened tabs from the tabs service. Then the user can choose between two actions: (1) click to navigate to any opened tab; or (2) close a tab. The user can re-arrange the tabs order by dragging a tab, event not shown in Figure 16.

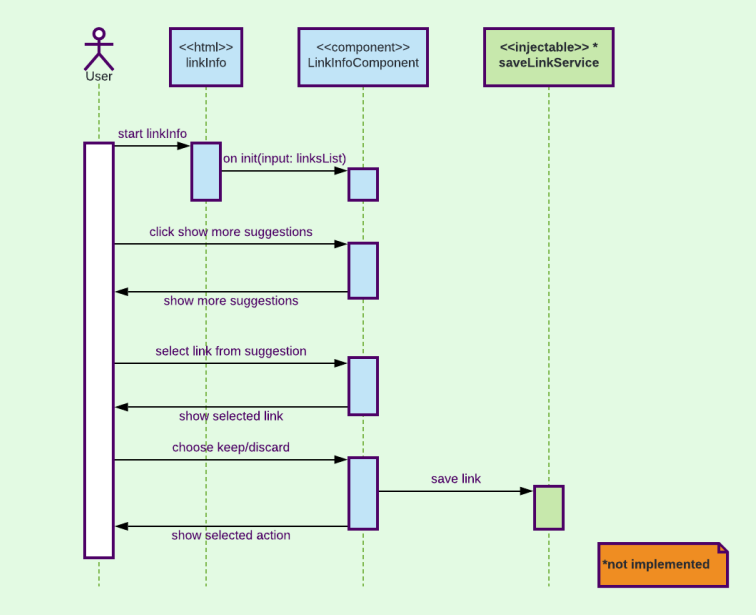


Figure 15: Keep/discard Link Sequence Diagram.

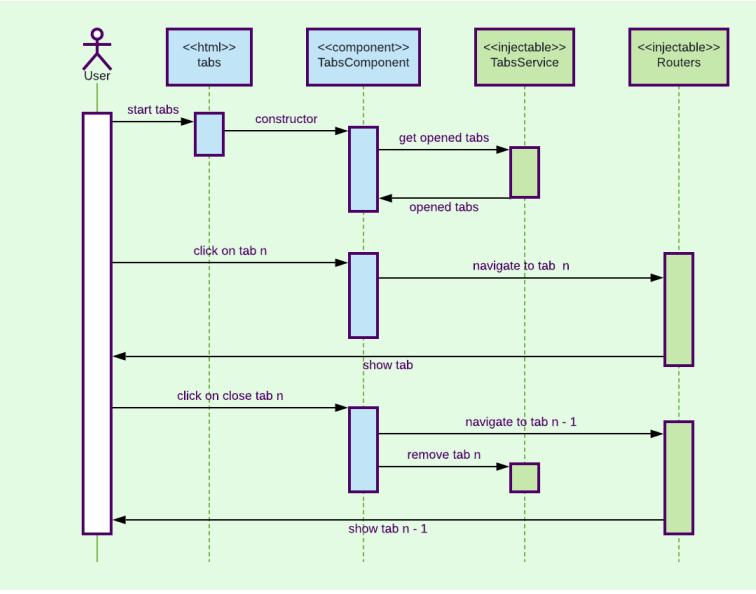


Figure 16: Tabs Component Sequence Diagram.

References

[1] Dbpedia: About: Wolfgang amadeus moza. `https://dbpedia.org/page/Wolfgang_Amadeus_Mozart`. Accessed January 2021.

[2] Tim Berners-Lee, James Hendler, and Ora Lassila. The semantic web: a new form of web content that is meaningful to computers will unleash a revolution of new possibilities. 2001. *Cited on*, page 18, 2011.

[3] Christian Bizer, Julius Volz, Georgi Kobilarov, and Martin Gaedke. Silk-a link discovery framework for the web of data. In *18th International World Wide Web Conference*, volume 122, 2009.

[4] José Camacho-Collados, Mohammad Taher Pilehvar, and Roberto Navigli. NASARI: a novel approach to a semantically-aware representation of items. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 567–577, Denver, Colorado, May–June 2015. Association for Computational Linguistics.

[5] Jeremy Debattista, Sören Auer, and Christoph Lange. Luzzu–a framework for linked data quality assessment. In *2016 IEEE Tenth International Conference on Semantic Computing (ICSC)*, pages 124–131. IEEE, 2016.

[6] Mohamed Ben Ellefi, Zohra Bellahsene, Stefan Dietze, and Konstantin Todorov. Dataset recommendation for data linking: An intensional approach. In *European Semantic Web Conference*, pages 36–51. Springer, 2016.

[7] Armin Haller, Javier D Fernández, Maulik R Kamdar, and Axel Polleres. What are links in linked open data? a characterization and evaluation of links between knowledge graphs on the web. *Working Papers on Information Systems, Information Business and Operations*, (2/2019), 2019.

[8] Tom Heath and Christian Bizer. Linked data: Evolving the web into a global data space. *Synthesis lectures on the semantic web: theory and technology*, 1(1):1–136, 2011.

[9] Ernesto Jiménez-Ruiz and Bernardo Cuenca Grau. Logmap: Logic-based and scalable ontology matching. In *International Semantic Web Conference*, pages 273–288. Springer, 2011.

[10] Vladimir I Levenshtein. Binary codes capable of correcting deletions, insertions, and reversals. In *Soviet physics doklady*, volume 10, pages 707–710. Soviet Union, 1966.

[11] Axel-Cyrille Ngonga Ngomo, Mohamed Ahmed Sherif, Kleanthi Georgala, Mofeed Mohamed Hassan, Kevin Dreßler, Klaus Lyko, Daniel Obraczka, and Tommaso Soru. Limes-a framework for link discovery on the semantic web. *Journal of Web Semantics*, 2018.

[12] André Gomes Regino, Julio Kiyoshi Rodrigues Matsoui, Júlio Cesar dos Reis, R. Bonacin, A. Morshed, and Timos Sellis. Link maintenance for integrity in linked open data evolution: Literature survey and open challenges. *Social Work*, pages 1–25, 2020.

[13] André Gomes Regino, Julio Cesar dos Reis, and Rodrigo Bonacin. Lodmf: A linked open data maintenance framework. In *Semantic Technologies for Smart Information Sharing and Web Collaboration (Web2Touch) at the 30th IEEE International Conference on Enabling Technologies: Infrastructure for Collaborative Enterprise (WETICE'20) (accepted for publication)*, 2020.

[14] Blerina Spahiu, Cheng Xie, Anisa Rula, Andrea Maurino, and Hongming Cai. Profiling similarity links in linked open data. In *2016 IEEE 32nd International Conference on Data Engineering Workshops (ICDEW)*, pages 103–108. IEEE, 2016.