

# DAOLOT: A Semantic Browser

João Bruno Silva 

Faculty of Sciences, University of Porto, Portugal  
up201406063@fc.up.pt

André Santos 

CRACS & INESC Tec LA, Porto, Portugal  
Faculty of Sciences, University of Porto, Portugal  
afs@inesctec.pt

José Paulo Leal 

CRACS & INESC Tec LA, Porto, Portugal  
Faculty of Sciences, University of Porto, Portugal  
zp@dcc.fc.up.pt

---

## Abstract

The goal of the Semantic Web is to allow the software agents around us and AIs to extract information from the Internet as easily as humans do. This semantic web is a network of connected graphs, where relations between concepts and entities make up a layout that is very easy for machines to navigate.

At the moment, there are only a few tools that enable humans to navigate this new layer of the Internet, and those that exist are for the most part very specialized tools that require from the user a lot of pre-existing knowledge about the technologies behind this structure. In this article we report on the development of DAOLOT, a search engine that allows users with no previous knowledge of the semantic web to take full advantage of its information network. This paper presents its design, the algorithm behind it and the results of the validation testing conducted with users. The results of our validation testing show that DAOLOT is useful and intuitive to users, even those without any previous knowledge of the field, and provides curated information from multiple sources instantly about any topic.

**2012 ACM Subject Classification** Information systems → Web searching and information discovery; Information systems → Resource Description Framework (RDF)

**Keywords and phrases** Semantic, Web, Named Entities, Search, SPARQL, RDF, COMUNICA

**Digital Object Identifier** 10.4230/OASICS.SLATE.2020.5

**Funding** *André Santos*: Ph. D. Grant SFRH/BD/129225/2017 from Fundação para a Ciência e Tecnologia (FCT), Portugal.

*José Paulo Leal*: Fundação para a Ciência e a Tecnologia (FCT), within project UIDB/50014/2020.

## 1 Introduction

The end of the last decade, in technological terms, was marked by an expansion of the Internet beyond human users. Looking at this trend, we can say that we are entering the era of an Internet shared with machines, where programs are able to browse the Internet independently of humans. However, the Internet was not designed to be used by autonomous machines. When entering a web page, the relationship between the data found there and other links may be obvious to us, but a machine cannot understand the relationship between several different pages: it only sees a list of links.

The semantic web[1] was designed to solve this problem: an extension to the normal Internet that allows web applications to navigate a network of interconnected data, just as one would navigate links on the normal Internet. In the semantic web, instead of a



© João Bruno Silva, André Santos, and José Paulo Leal;  
licensed under Creative Commons License CC-BY

9th Symposium on Languages, Applications and Technologies (SLATE 2020).

Editors: Alberto Simões, Pedro Rangel Henriques, and Ricardo Queirós; Article No. 5; pp. 5:1–5:11



OpenAccess Series in Informatics

OASICS Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

network of links, the Internet is presented as a huge interconnected graph of information, where the nodes represent resources or properties, and the arrows represent the relationships between them.

This information is stored in RDF, a format in which the atomic information unit is called a triple[8]. A triple has the format of **Subject** → **Predicate** → **Object**, and these triples are what allow the semantic web to specify connections between entities and concepts, forming a knowledge graph.

The applications designed to explore these graphs are called semantic browsers. They are meant to help the user taking advantage of the semantic web to search for information.

The semantic browsers that currently exist and are mostly academic tools designed specifically for research, built for very specific purposes, and are mostly impossible to use without pre-existing technical knowledge.

The aim of the project presented in this article is to create a semantic search engine that is available and accessible to the general public, available to all desktop and mobile devices, which allows a human user to easily search and compare results in a simple and intuitive way. The browser developed for this goal is named DAOLOT. It consists of a web page which retrieves information from multiple endpoints, and processes this information as it arrives to be easily understood by the user. This article describes the current state of semantic browsers, followed by an overview of DAOLOT's architecture and the validation testing used to improve it.

This article consists of a brief introduction to the concept of the Semantic Web and the goals of DAOLOT, followed by a assessment of the existing semantic browsers, their similarities and limitations.

The following section details the architecture and algorithm behind DAOLOT, followed by the validation section, which details the tests conducted with users and the feedback which was used to improve DAOLOT.

Finally, the conclusion sums up some of the difficulties that we had during development and the implications of DAOLOT in the future of the Semantic Web.

## **2** State-of-the-Art

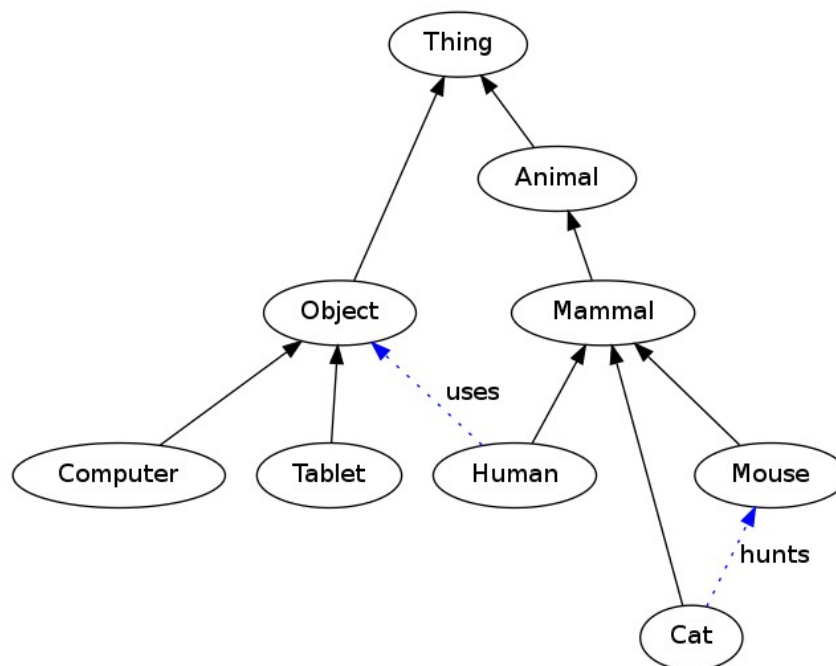
Semantic Browsers are browsers built to navigate semantic knowledge graphs, which are databases of information stored in the RDF format, and extract information from them. The RDF format is the basis of the semantic web, and it stores information in atomic structures called triples, which specify relations between entities in the format **Subject** → **Predicate** → **Object**. A group of triples referencing the same entities form a knowledge graph, as illustrated in the following image.

The existing semantic browsers are specialized tools, built mostly as either academic research or commercial use tools. The majority of semantic browsers are designed for navigating local RDF databases, not for retrieving information from existing remote endpoints.

Those that are designed to retrieve information from other endpoints require the user to type his queries in the SPARQL format[9, 3], a querying language made to retrieve information from semantic databases, requiring technical knowledge from the user. This section surveys some of the semantic browsers described in the literature, considering both their similarities to DAOLOT and their limitations:

### **/Facet**

*/Facet* is a semantic browser designed to browse *RDF* repositories[7] located on web servers, entirely based on SWI Prolog, a development environment in the Prolog programming language designed to handle the *RDF* format.



■ **Figure 1** A visual representation of a knowledge graph[5].

Developed in 2006, /facet is one of the first semantic browsers to have been published, having since then become part of a larger browser called *ClioPatria*, which is also no longer functional.

/Facet allows to graphically navigate a graph as if it were a tree of files, and even allows searching for a *keyword*, with the addition of a graphic illustrator for temporal data.

The biggest limitation of this browser is the fact that it only allows the simple search for one argument, and, because it only works with local *datasets*, it greatly limits its usefulness for the general public.

Another difference of this system from DAOLOT is the presentation of the search results, which are presented at the end of the semantic tree of which they are part, requiring the user to scroll the graph up to the sheet for each individual result of the *query*.

### Mspace

Mspace is a semantic browser developed in the University of Southampton, US, and consists of an interface and framework designed to explore and manipulate information in semantic datasets[6, 10]. Its interface system is based on columns, which present each property and possible values.

Although Mspace has a more complete search functionality than existing semantic browsers, it remains simply based on a rudimentary keyword search, and the interface is unnecessarily complex, while also having no visualization tools for aggregating data.

### OpenLink Data Explorer

OpenLink Data Explorer is a browser that works as *add-on* for conventional web browsers like Chrome or Firefox.

Its main feature is to be able to extract metadata from any resource on the Internet at the user's command.

Although functional, this engine only gives us the *RDF* meta-data that describes the resource, always depending on manual search, and also does not offer any functionality that facilitates the navigation of this data.

## 5:4 DAOLOT: A Semantic Browser

Browser	/Facet	Mspace	Openlink data explorer
GUI	Yes	Yes	Yes
Search by keywords	Limited	Yes	No
Data browsing	Limited	Yes	No
Graphical representation of data	Yes	No	No
Access to the semantic web	No	Yes	Yes
Data analysis	No	No	No
Custom endpoints	Yes	No	No

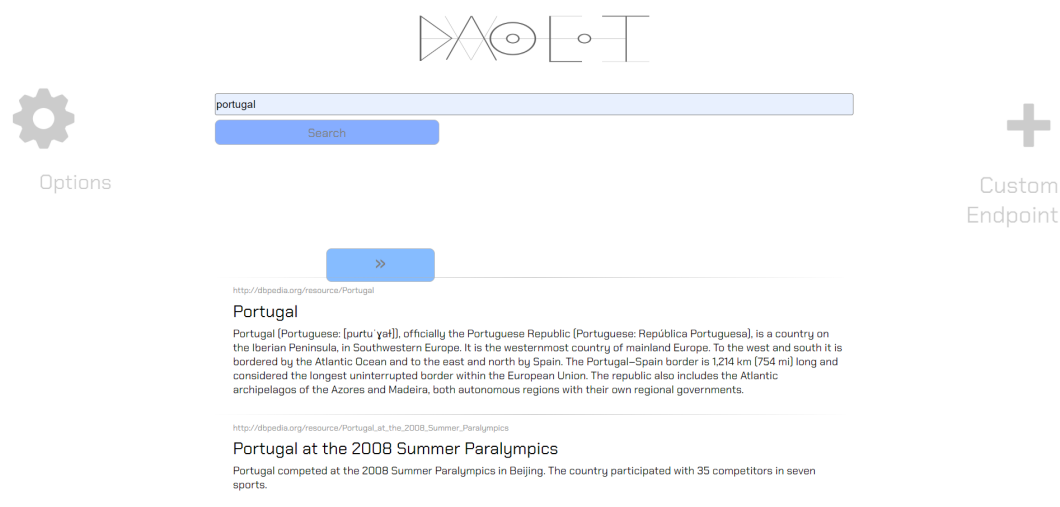
The table above compares the features and limitations of each of the semantic Browsers mentioned:

There are a few more semantic browsers out there, but these are representative of the semantic browsers that exist at the moment and the current limitations that DAOLOT aims to correct.

### 3 System Overview

The DAOLOT search engine<sup>1</sup> is a responsive web page, available for desktop and mobile devices.

DAOLOT transforms the user-typed search term into a SPARQL query, and uses asynchronous requests to send it to several semantic endpoints. As the information arrives, it compiles and curates the results to minimize redundancy and maximize usefulness.



■ **Figure 2** The DAOLOT Interface.

The inputs available to the user, presented in Figure 2, are:

#### The Search Bar

Allows the user to input his query;

#### The Options Menu

A pop-up menu on the side allows the user further control over the search parameters,

<sup>1</sup> DAOLOT: <https://dao.lot.dcc.fc.up.pt/>

such as the number of results to display, language, sources to be included in the search and even a manual mode for advanced users, which allows to directly write the SPARQL query to be sent;

### Custom Endpoint Menu

This menu allows the user to add endpoints not included in the default selection of DAOLOT. Before adding the endpoint, DAOLOT runs a “compatibility check” on the endpoint, checking whether it responds to request and if it recognizes standard RDF ontologies which are essential for DAOLOT. Examples of such ontologies are the RDFS vocabulary[2], which specifies basic types of predicates such as `rdfs:comment` (a small summary about the subject), or the Web Ontology Language (OWL)[4], which helps to find aliases for a subject with the predicate `owl:SameAs`, both which are essential to the functioning of DAOLOT. If successful, the endpoint is added to the list.

While still providing in-depth options for more advanced users, all these options are handled automatically if the user does not modify them.

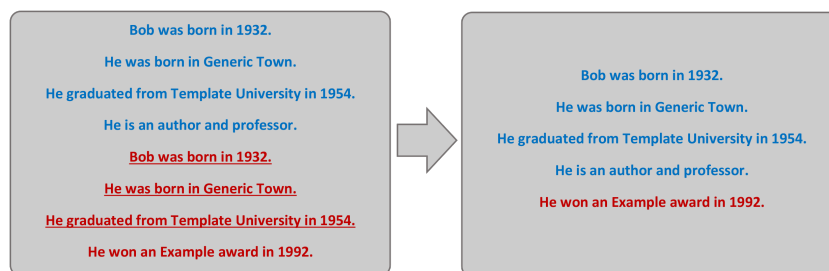
## 3.1 Result Gathering

DAOLOT’s search works on a 2-phase system: In the first phase, the user inserts a search term about the topic or subject of interest. This query can be just a single word or a several keywords. When it is submitted, the algorithm prepares a query that searches the endpoints for subjects that contain any of the keywords either on its name or in any description associated with its name.

Then this query is sent to every endpoint selected in the configurations. As responses arrive, DAOLOT generates a list, each entry containing the name of the subject, its aliases on different endpoints, and a description compiled from all the descriptions linked to that particular subject. This first step presents to the subject a list of subjects that might be related to his search, and to help him choose which one to inquire further about.

## 3.2 Reduce redundancy

Because these descriptions usually have a high level of overlap, DAOLOT eliminates redundant sentences in these sections. Figure 3 provides an example on how redundant data from two sources is merged.



■ **Figure 3** A visual representation of the redundancy reduction function.

## 3.3 Property Gathering

When the user selects a subject, this engages the search engine’s second phase. In this phase, the engine will try to find all the available information and properties (e.g.: age, place of birth, related work) on this subject. As subject can be referred to by different labels across

the endpoints, the algorithm will build a query about all the triples that involve any of this subject's aliases, and send it to all the selected endpoints. In this phase DAOLOT relies on the use of the Comunica framework[11], which supports both SPARQL endpoints and Triple Pattern Fragments[13] endpoints (a simpler and lighter server interface for semantic data), allowing for a larger pool of available endpoints beyond the default ones, such as DBPedia, DBWik or YAGO.

After the query is sent to all the endpoints, DAOLOT adds to a list of information about this subject as the answers arrive. This profile of the subject will aggregate all the information, each property grouped with all the values associated with that property (if multiple different values appear). In addition, if any of the values of properties of this subject are images, DAOLOT will display these images as part of the results, as seen in Figure 4.

### 3.4 Information Checking

At this point, DAOLOT is able to check the information quality. For example, suppose that endpoint X returns the information that this subject's birth-date is <S> <birthDate> 1932. Suppose also that endpoint Y returns the same information. When the same value for the same property is returned by multiple sources, DAOLOT assumes that this piece of information is more trustworthy, and that property will be marked visually as containing trusted information, highlighting the trusted values.

On the other hand, DAOLOT is also able to detect contradictions. When a new property is introduced, the algorithm makes a "functionality check" to it, sending a request for all triples containing that property, and calculating the average amount of different values per subject on that property. If it determines that the property usually only has one value per subject, it will add it to a watch-list of properties that are supposed to only have one value.

Consider the previous example. Statistically, most people only have one birth-date. Hence, if a third endpoint Z says that the subject's birth-date is 1978, the property <birthDate> would appear in the list with a visual indicator that it is contradictory, and will have the source of each value displayed directly over it.

### 3.5 Navigating the results

If the value of a triple is also another subject the result will be presented as a link, and the user can now click this value in order to see all the information about that related subject. This process can be repeated indefinitely. As the user browses from one related subject to another, DAOLOT forms a search chain containing the sequence of subjects. This allows the user to use the arrow keys either on the screen or on a keyboard to navigate quickly through all the gathered information, or to go back to a previous point in the search and follow a different path. Each time the user performs a new step in its search, the URL of the page is updated, allowing the user to save the location, to come back to this specific subject later or to share the results found.


Figure 5 below illustrates the different layers of the algorithm and its stages.

<https://dbpedia-live.openlinksw.com/sparql/>

## Andrew Bobola

---

<https://dbpedia-live.openlinksw.com/sparql/>  
<https://fragments.dbpedia.org/2015/en>



Information backed by multiple sources 


---

<https://dbpedia-live.openlinksw.com/sparql/>  
<https://fragments.dbpedia.org/2015/en>

**Abstract**

[Andrew Bobola, S.J. \(Polish: Andrzej Bobola, 1591 – 16 May 1657\)](#) was a Polish missionary and martyr of the Society of Jesus, known as the Apostle of Lithuania and the hunter of souls. He was tortured to death during the Khmelnytsky Uprising. He was canonized in 1938 by Pope Pius XI.

[Andrew Bobola, S.J. \(Polish: Andrzej Bobola, 1591 – 16 May 1657\)](#) was a Polish missionary and martyr of the Society of Jesus, known as the Apostle of Lithuania and the hunter of souls.

Information backed by multiple sources 

---

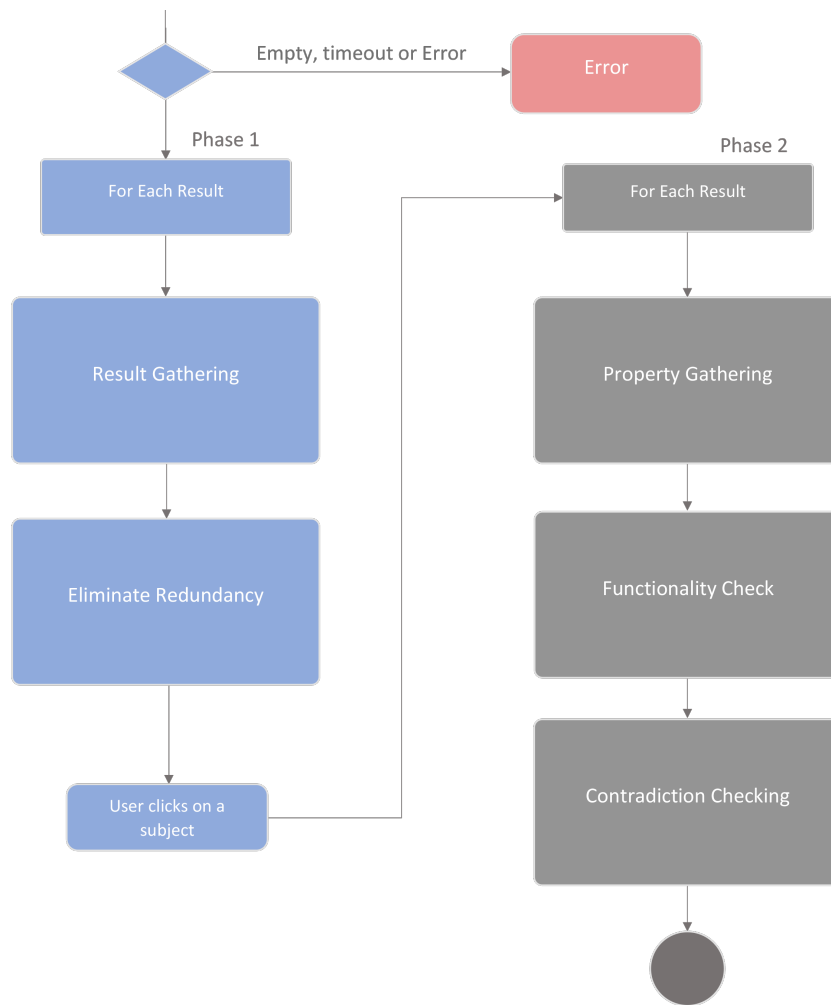
<https://dbpedia-live.openlinksw.com/sparql/>  
<https://dbpedia.org/sparql>  
<https://fragments.dbpedia.org/2015/en>

**Death place**

<a href="#">Ivanava</a>	<a href="http://dbpedia.org/resource/Ivanava">http://dbpedia.org/resource/Ivanava</a>
<a href="#">Grand Duchy of Lithuania</a>	<a href="http://dbpedia.org/resource/Grand_Duchy_of_Lithuania">http://dbpedia.org/resource/Grand_Duchy_of_Lithuania</a>
<a href="#">Polish–Lithuanian Commonwealth</a>	<a href="http://dbpedia.org/resource/Polish%E2%80%93Lithuanian_">http://dbpedia.org/resource/Polish%E2%80%93Lithuanian_</a>
<a href="#">Polish–Lithuanian Commonwealth</a>	<a href="http://dbpedia.org/resource/Polish–Lithuanian_Commonwealth">http://dbpedia.org/resource/Polish–Lithuanian_Commonwealth</a>

Might contain contradictory information 

■ **Figure 4** An example phase 2 search displaying images, contradicted and confirmed information.



■ **Figure 5** A diagram of the main data-processing function.

#### 4 Validation

The validation of DAOLOT's usability followed the Nielsen Usability Model[12]. It was conducted with 7 participants, with ages between 20 and 25, without any experience or knowledge about web semantic.

Each was given a list of simple tasks to guide them through different features, each making them explore a different functionalities of DAOLOT:

##### Task 1

Find more about a certain person;

##### Task 2

Find his hometown and it's population;

##### Task 3

Find his main interests;

##### Task 4

Increase the search limit and remove an Endpoint from the settings;



**Task 5**

Find artists with the same name as the first search subject;

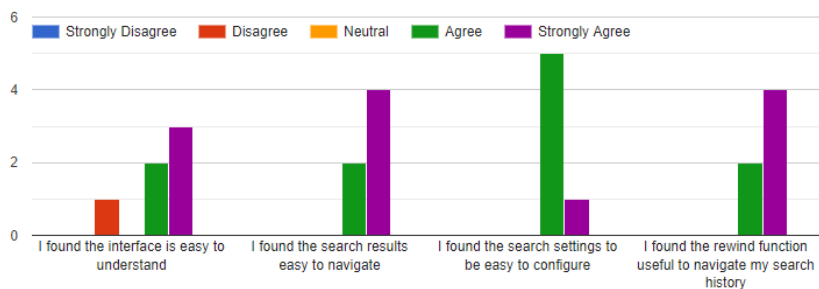
**Task 6**

Remove a specific endpoint and see if the results change;

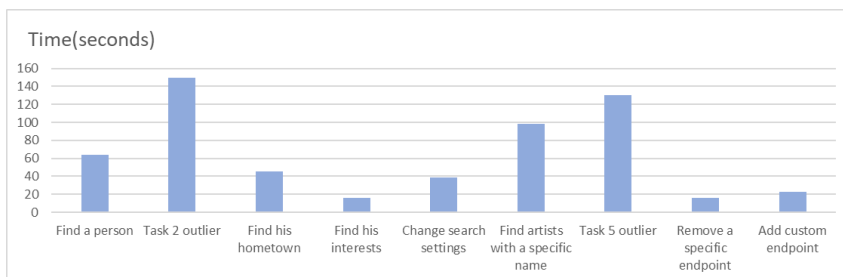
**Task 7**

Add this custom endpoint to the settings;

After attempting those tasks, participants were asked to fill out a survey asking how much they agreed or disagreed with some statements about the usefulness and ease of use of DAOLOT. The screen of the participants was recorded for analysis, and the goal was to understand if they managed to use all the features of DAOLOT correctly without help.



**Figure 6** The responses given in the survey by the users.



**Figure 7** The average time of completion of each task in the survey.

In general, all participants managed to do most of the tasks with no guidance, except for the custom endpoint feature, which was due to their lack of knowledge on semantic web concepts.

As seen in Figure 7, most tasks were completed in under a minute, with the final two being completed on average under 30 seconds.

We can conclude from this data that after figuring out the core functionalities, the users were able to accomplish tasks much faster, as evidenced by the last tasks being completed much faster than the first two.

As Figure 6 illustrates, with one exception, all users found the interface easy to use and the results easy to navigate, despite their lack of technical knowledge.

After grasping the fundamentals about DAOLOT user interface with the first task, participants were able to complete most of the basic tasks in less than 25 seconds. The task which took longer, which consisted of finding artists with a particular name, was mainly due to a problem with DAOLOT’s earlier version, which had trouble finding exact matches

for the participants's search, which led to unpredictable results. In response to this issue, the search algorithm was updated to have a secondary function designed to fetch only exact matches and put them on top of the list, before trying to search approximated results. The outlier that occurred in the task "find his hometown" with one participant was that he tried to navigate between multiple subject as results were still loading, and thus ended up missing additional information. The solution was to implement a graphical queue on top of the subject's label, to indicate that more results are still incoming, which improved participant's reception.

The participants' feedback provided in the survey was also used to improve DAOLOT in several points. Some of them reported the UI as being "too colorless", and as such the UI was redesigned to have a more colorful aspect.

In the initial testing, some participants took longer to find the options, as the original UI just had the icons of the options visible, with no label as to what these icons did.

In response to this feedback, the UI was redesigned to have permanent labels on all the options, in addition to a loading icon to indicate to the user that more results are still arriving for the search.

Outside the programmed tasks, all the participants found DAOLOT to be easy to use and to produce relevant results.

## **5** Conclusions

The goal of DAOLOT is of being an intuitive, easy-to-use and reliable semantic browser for the average user. Based on the validation results, it can be concluded that DAOLOT has achieved this goal.

There were several major challenges in the implementation of this browser, such as dealing with contradictory information provided by several different sources, finding which entities (from distinct sources) were actually referring to the same subject, and how to maximize compatibility with endpoints. DAOLOT is unable to determine which information is correct, but it is able to point out contradictions and possible confirmations, by doing a statistical analysis of the variety of values of a property. It is also able to identify similar subjects by relying on a property of the OWL vocabulary, `owl:sameAs`. However, this represents one of the challenges of DAOLOT, which is how to deal with the lack of use of standard vocabularies. This browser requires that endpoints employ standard vocabularies such as RDFS and OWL in order to function correctly, and any endpoint who doesn't comply would require a custom implementation. This is why WikiData cannot be used as an endpoint on DAOLOT, despite being by far the largest and most complete general scope semantic source. These difficulties illustrate the challenge of building a completely autonomous agent to exploit resources from different sources – the goal of the semantic web – which would be significantly more complex than a semantic browser, such as DAOLOT.

The scientific contribution of this article is that DAOLOT provides a solution towards most of the problems that surround web semantics, by providing a way to easily convert the content of the semantic web into information that a human user can easily understand.

In the future, DAOLOT can be further improved by expanding its methods of data visualization, such as adding graphical representation of certain types of data and display statistics about certain properties when appropriate, such as a timeline of presidents in the history of a country.

DAOLOT is easily expandable, requiring little to no maintenance when dealing with new endpoints, and with the fast growth of the semantic web, it will only get more useful over time, as the amount of information available to it increases.

---

**References**


---

- 1 Tim Berners-Lee, James Hendler, and Ora Lassila. The semantic web. *Scientific american*, 284(5):34–43, 2001.
- 2 Dan Brickley, Ramanathan V Guha, and Andrew Layman. Resource description framework (RDF) schema specification. Technical report, World Wide Web Consortium (W3C), 1999.
- 3 Kendall Grant Clark, Lee Feigenbaum, and Elias Torres. SPARQL Protocol for RDF. W3c recommendation, W3C, January 2008. URL: <http://www.w3.org/TR/rdf-sparql-protocol/>.
- 4 W3C Owl Working Group et al. Owl 2 web ontology language document overview, 2009. URL: <http://www.w3.org/TR/owl2-overview/>.
- 5 Sébastien Harispe, Sylvie Ranwez, Stefan Janaqi, and Jacky Montmain. Semantic measures for the comparison of units of language, concepts or entities from text and knowledge base analysis, October 2013.
- 6 C. Harris, A. Owens, A. Russell, and D. A. Smith. mSpace: Exploring the semantic web. a technical report in support of the mspace software framework. Master’s thesis, Faculty of Engineering, Science and Mathematics, University of Southampton, 2004.
- 7 Michiel Hildebrand, Jacco Van Ossenbruggen, and Lynda Hardman. /facet: A browser for heterogeneous semantic web repositories. In *International Semantic Web Conference*, pages 272–285. Springer, 2006.
- 8 Ora Lassila, Ralph R Swick, et al. Resource description framework (RDF) model and syntax specification. Technical report, World Wide Web Consortium (W3C), 1998.
- 9 Eric Prud’hommeaux and Andy Seaborne. SPARQL Query Language for RDF. W3C Recommendation, January 2008. URL: <http://www.w3.org/TR/rdf-sparql-query/>.
- 10 MC Schraefel, Daniel A Smith, Alisdair Owens, Alistair Russell, Craig Harris, and Max Wilson. The evolving mspace platform: leveraging the semantic web on the trail of the memex. In *Proceedings of the sixteenth ACM conference on Hypertext and hypermedia*, pages 174–183, 2005.
- 11 Ruben Taelman, Joachim Van Herwegen, Miel Vander Sande, and Ruben Verborgh. Comunica: a modular sparql query engine for the web. In *International Semantic Web Conference*, pages 239–255. Springer, 2018.
- 12 Carl W Turner, James R Lewis, and Jakob Nielsen. Determining usability test sample size. *International encyclopedia of ergonomics and human factors*, 3(2):3084–3088, 2006.
- 13 Ruben Verborgh, Miel Vander Sande, Olaf Hartig, Joachim Van Herwegen, Laurens De Vocht, Ben De Meester, Gerald Haesendonck, and Pieter Colpaert. Triple pattern fragments: a low-cost knowledge graph interface for the web. *Journal of Web Semantics*, 37:184–206, 2016.

## **A** Press Summary

DAOLOT is a search engine directly provides the user with the information it finds about a subject, providing sources and even qualifying pieces of information as trustworthy or not, and neatly presenting it in a convenient format, instead of just providing the user with a list of links to websites.

DAOLOT can provide the user with what would be usually a prolonged research in moments, instead of having to spend time collecting information about a subject across multiple websites.