Web-based ontology analysis and partitioning tool

Prepared by:
Ahmad Alzeitoun
Iulia Buga
Imuwahen Osazuwa

Supervised by:
Dr. Gökhan Coskun
Irlan Grangel

Technical Documentation

Date: (11/09/2015)

Table of Contents

1. Introduction

1.1 Overview

Ontologies, i.e. semantic structures encoding concepts, relations and axioms, providing a model of a given domain, are the backbone of the Semantic Web, a semantic-aware version of the World Wide Web. Ontologies allow web resources to be semantically enriched. Ontology building is a task that pertains to ontology engineers, an emerging expert profile that requires the expertise of knowledge engineers (KE) and domain experts (DE). Even though automatic ontology learning methods significantly support ontology engineers by speeding up their task, there is still the need of significant manual effort, in the completion, consolidation and validation of the automatically generated ontology [1]

With the increasing use of ontologies in many branches of science and industry not only the number of available ontologies has increased considerably but also many widely used ontologies have reached a size that overburdens development and quality control procedures. It has been argued that the maintenance of large ontologies would be greatly facilitated by decomposing large ontologies into smaller modules [2] that cover certain subtopics of the ontology. Using such models it would be much easier to avoid inconsistencies and semantic defects in the creation and maintenance when the part of the ontology to consider for modification is clearly defined [3].While there is a clear need for modularization, there are no well-defined and broadly accepted ideas about the criteria that define a good module. As a result, several approaches have been recently used to partition ontology into modules, each of them implementing its own intuition about what a module should contain and what should be its qualities [5, 4, 2].

To accomplish this, the **"Web-based ontology analysis and partitioning tool"** is developed as a scalable, open platform web application. Its design allows for quickly uploading large ontologies and partitioning the ontologies into new ones.

1.2 Purpose

The main goal of this technical documentation is to give an overview and an insight on how the system operates.This document defines the technical system documentation involved in the project

1.3 Scope

The scope of the **WAPT** project is defined in what follows :

- analyze aspects like structure and semantics of vocabularies or ontologies
- understand the content and evolution of vocabularies or ontologies
- support the partitioning through existing modularization techniques
- employ the MEAN stack with a node.js backend as its software development base(without MongoDB)
- export ontologies
- work cross platform

1.4 Definitions, Acronyms, and Abbreviations.

WAPT	- "Web-based analysis and partitioning tool"
KE	– Knowledge engineers
DE	– domain experts
MEAN	– MongoDB, Express, AngularJS and node.js
FR	– Functional requirement
NFR	– Non-functional requirement

1.5 References

[1] A software engineering approach to ontology building - AntonioDeNicola , MicheleMissikoff, RobertoNavigli.

[2] H.Stuckenschmidt and M. Klein.Structure-based partitioning of large concept hierarchies. In S. A. McIlraith, D. Plexousakis, and F. van Harmelen, editors,*Proceedings of the Third International Semantic Web Conference* (ISWC 2004),pages 289–303, Hiroshima, Japan, Nov 2004.

[3] A. Rector. Modularisation of domain ontologies implemented in description logics and related formalisms including OWL. In *Proceedings of the 16th International FLAIRS Conference.* AAAI, 2003.

[4] B. Cuenca Grau, B. Parsia, E. Sirin, and A. Kalyanpur. Modularity and web ontologies. In *Proceedings of the 10th International Conference on Principles of Knowledge Representation and Reasoning* (KR2006),2006.

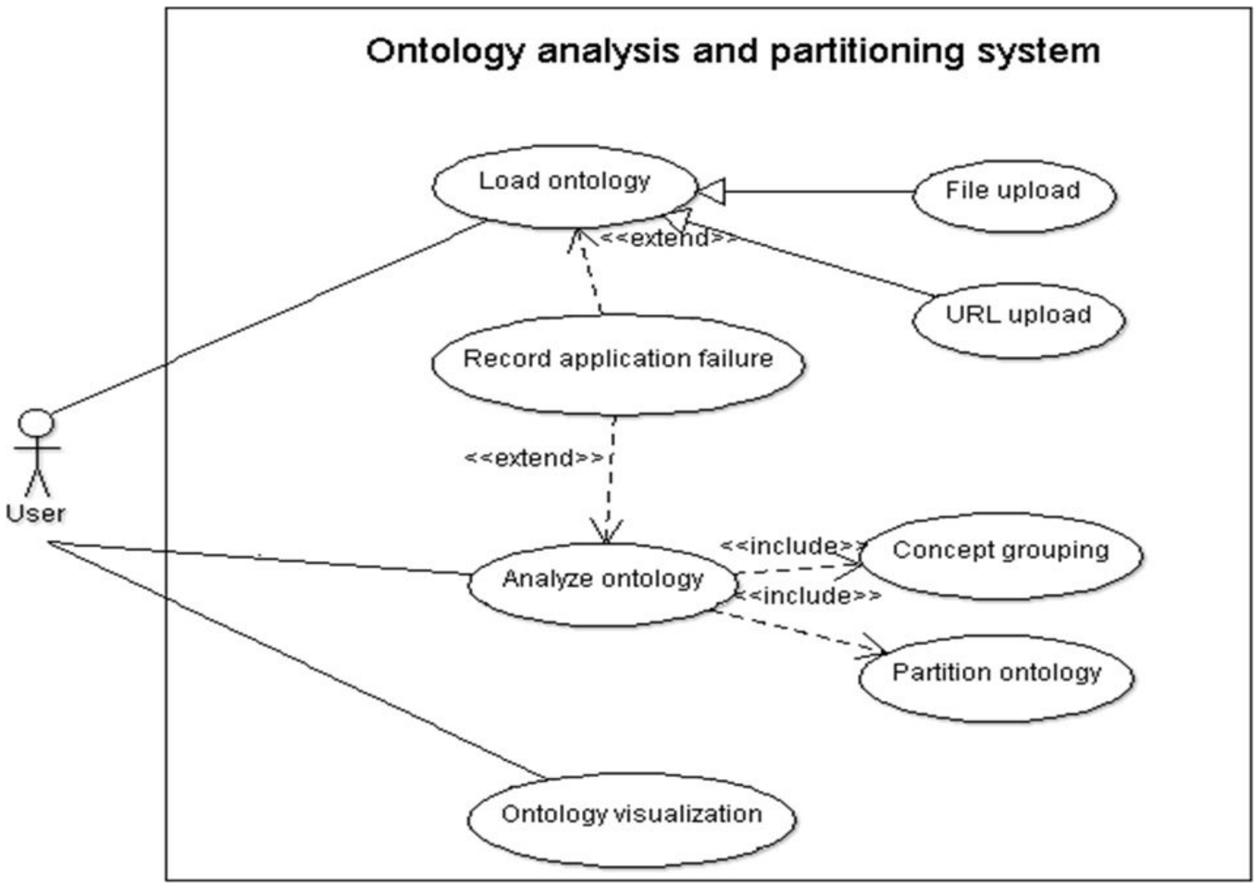
[5] B. MacCartney, S. McIlraith, E. Amir, and T. Uribe. Practical partition-based theorem proving for large knowledgebases. In *Proceedings of the Eighteenth International Joint Conference on Artificial Intelligence* (IJCAI-03), 2003.

2. The Overall Description

2.1 Product Perspective

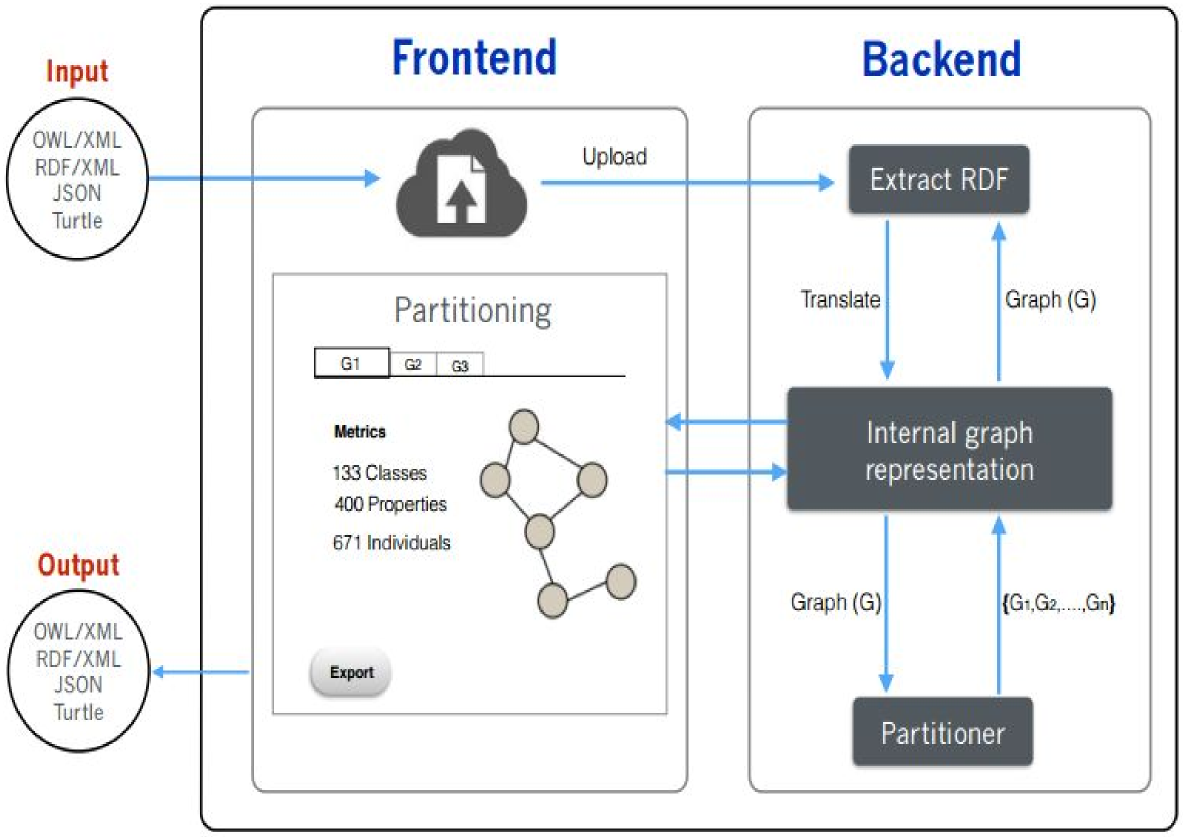
Ontologies are the backbone of the Semantic Web, a semantic-aware version of the World Wide Web. The availability of large-scale high quality domain ontologies depends on effective and usable methodologies aimed at supporting the crucial process of ontology building. Ontology building exhibits a structural and logical complexity that is comparable to the production of software artefacts. [1] In the context of existing Ontology Modularization Frameworks, there exists a Java based application that can perform this task. However, due to emerging new technologies, a version of a product that delivers a quick, scalabale and fast interface is highly needed. For this purpose, the **WAPT** is being developed.

The WAPT use case diagram is presented in what follows.



2.2 Architecture Specification

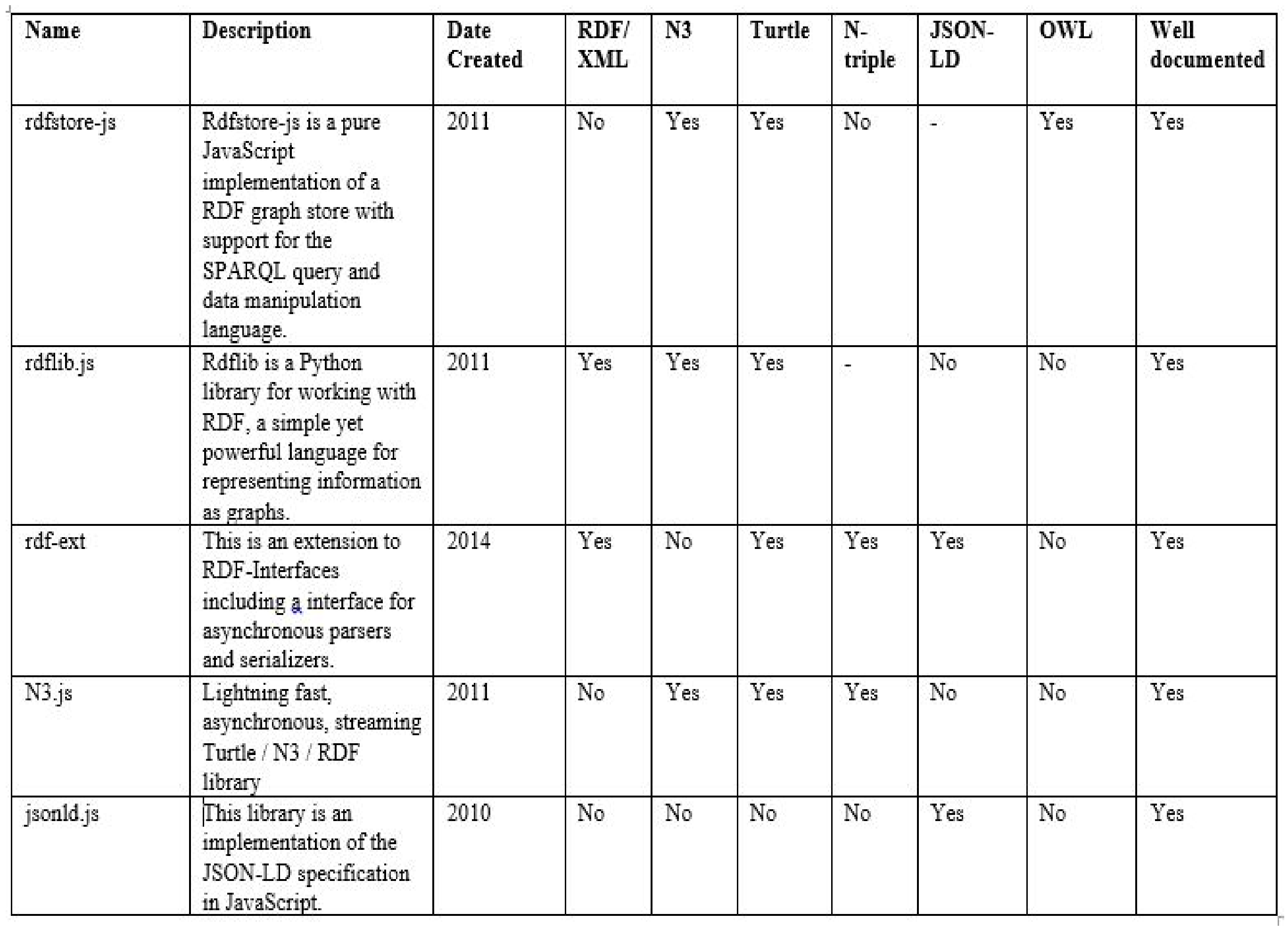
The following diagram gives a general overview of the architecture of WAPT;



3. Research for Libraries

3.1 Ontology Library

The table below shows the existing Javascript libraries that were researched.They were selected because of their features and their supported file types. From these libraries, we selected *rdfstore-js*;



Name	Description	Date Created	RDF/ XML	N3	Turtle	N-triple	JSON-LD	OWL	Well documented
rdfstore-js	Rdfstore-js is a pure JavaScript implementation of a RDF graph store with support for the SPARQL query and data manipulation language.	2011	No	Yes	Yes	No	-	Yes	Yes
rdflib.js	Rdflib is a Python library for working with RDF, a simple yet powerful language for representing information as graphs.	2011	Yes	Yes	Yes	-	No	No	Yes
rdf-ext	This is an extension to RDF-Interfaces including a interface for asynchronous parsers and serializers.	2014	Yes	No	Yes	Yes	Yes	No	Yes
N3.js	Lightning fast, asynchronous, streaming Turtle / N3 / RDF library	2011	No	Yes	Yes	Yes	No	No	Yes
jsonld.js	This library is an implementation of the JSON-LD specification in JavaScript.	2010	No	No	No	No	Yes	No	Yes

3.2 Graph Libraries

From below libraries, *vis.js* library has been selected. The library is designed to handle large amounts of dynamic data, and enable manipulation of the data and it supports custom shapes, styles, colors, sizes, images, and more. The major components that we worked with in vis.js is DataSet and DataView to Add, update, remove items and to filter and/or format view.

Name	**Date created**	**Description**	**Metrics**	**Import/export formats**	**Visualization support**
Vis.js	2013	The library is designed to be easy to use, to handle large amounts of dynamic data, and to enable manipulation of and interaction with the data.	Yes	JSON	Yes
D3	2010	JavaScript library for producing dynamic, interactive data visualizations.	No	JSON	Yes
jOWL	2009	load OWL from url - still under development - suported only OWL-RDFS	No	OWL format	Yes

3.3 Partitioning Algorithms

There is no library used for partitioning, we applied a partitioning algorithm based on a research paper by S.Ghafourian, A.Rezaeian, and M.Naghibzadeh "*Modularization of Graph-Structured Ontology with Semantic Similarity*".

In the research paper a weighted matrix has been constructed by giving weighs to different relationships of the ontology, the weights values have been specified by another research, then the weights of the edges in the graph are normalized to be between zero and one. Thus, the weight of the edge outgoing from a node v is divided by the sum of the weights of outgoing edges from node v. This is needed for input matrix of random walk in which every element must be between zero and one.

$$W_{i,v}^{normal} = \frac{w_{i,v}}{\sum_{j \in out\_edges(V)} W_{j,v}}$$

Where W(i,v) the weight of the edge outgoing from a node v that is normalizing and the denominator is the sum of the weights of outgoing edges from node v.

Then the paper used the neighborhood random walk method to measure vertex closeness.

$$d(v_i, v_j) = \sum_{T:v_i \to v_j} P(T)c(1-c)^{Length(T)}$$

Where□ P is transition probability matrix, □□□□□length(□T) is length of random walk where □□□□□ □□□□□length(□T) ≤ l □, l is the length that a random walk can go, □C is restart probability where C□ ∈ (0,1), □□□□d(Vi,Vj) is the neighborhood random walk distance from Vi to Vj □□. It measures vertex closeness and T is a path from Vi to Vj whose length is □□□□□length(□T) with transition probability □P(□T).

The based algorithms depend on the chosen similarity criterion ;

```
Algorithm. Modularization (adjacencyMatrix)
//C represents whole modularization
C = put every node in a separate module
for K=N down to 2 // K shows number of modules
    // for all the pairs of nodes that could be merged maxS= -2
    for i=1 to K-1
        for j=i+1 to K
            c=Union(i,j);//merge modules i and j into module C C=C∪{c}\{i,j};
            S=Score(C); //according to equation above of S(i)
            //if this new score is better, save it
            if S>maxS
                maxS = S;
                modularization=C;
            end if
        end for
    end for
bestModularization=modularization;
end for
```

4. Technological Specification

4.1 Interfaces

WAPT product is based on the MEAN stack(without MongoDB) and employing technologies which are managed as standalone projects with dedicated organization. The backend is Node.js and the frontend is AngularJS.

Node.js is an open-source, cross-platform runtime environment for developing server-side web applications. Node.js applications are written in JavaScript and can be run within the Node.js runtime. Node.js allows the creation of web servers and networking tools using JavaScript and a collection of "modules" handling the various core functionality. Modules are similar to libraries in the C programming language. Each module contains functions specific to the "subject" of the module.

Additionally, the Bootstrap package is used in the scope of constructing a fluid and intuitive user interface. The Express.js framework is used to simplify the development of the web application. With just a few lines of code, Express.js can listen for an parse an HTTP request, route its according to the URL, create a well formatted response and send it back to the client.

The coding language for the new development is mostly influenced by the coding language of the used modules. Licence compliance review is done at the selected stages of the project life cycle. The goal is to verify the open source packages.

4.1.1 Software Interfaces

The minimal requirements for running the application are:

- Node.js v0.12.2 (release date may 2015)
- Java

These programs should be installed on the PC on which the application is running.

The frontend is handled by:

- AngularJS v1.3.15
- Bootstrap v3.3.4
- Underscore v1.8.3

A framework is used to accelerate the development of the application:

- Express.js v4.13.3

Express.js is one of the most widely used Node.js modules and it forms the backbone of many web applications which you visit day-to-day such as Paypal. Furthermore, other Node.js modules are used to ease the development and to accomplish different functions, such as file handling;

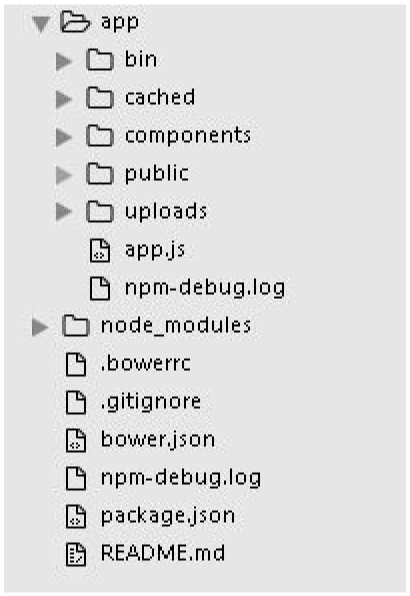
- BodyParser v.1.12.4
- BufferConcat v.0.0.1
- ConnectBussboy v0.0.2
- Debug v2.2.0
- Glob v.4.13.3
- Rdfstore v.0.9.7
- Request v2.60.0
- Morgan v1.6.1

All these modules are already included in the app, at the aforementioned versions. Their definition is in the file `package.json` which is found in the root of the application.

The semantic web specific libraries used are:
- RDFSTORE
- VisJS
- OWL2VOWL

4.1.2 Project structure

The project is structured into an intuitive and easy to understand way. The folders are separated by scope, functionality and role in the application. Below, a screenshot of the folder structure can be seen.



```
▼ app
  ▶ bin
  ▶ cached
  ▶ components
  ▶ public
  ▶ uploads
    app.js
    npm-debug.log
▶ node_modules
  .bowerrc
  .gitignore
  bower.json
  npm-debug.log
  package.json
  README.md
```

The significance of each folder is detailed below:

- **bin-** Node.js relevant folder, where the server is created.
- **cached-** Node.js relevant folder, where the cached ontologies are saved.
- **components-** Node.js relevant folder, where the Java VOWL2OWL components are.
- **public-** The front end for the Node.js application is organized here (AngularJS and all its folders are included here).
- **uploads-** Node.js relevant folder where the ontology is uploaded.
- **node_modules-** Node.js relevant folder, where all the dependencies for the Node.js are created.

The main entry point for the project is the file `app.js.`

Error handling is accomplished using two methods:
1) Node.Js console logs
2) Visual logs

In the *upload* URI page, the user is prompted to enter a valid URI. Verifications are made against empty URI and valid URI. Several REST endpoints are used. These endpoints expose critical information for the application to run. An exhaustive list of all the REST endpoints can be seen below:

Endpoint	Scope
/ontology	Raw data as generated by the RDFSTORE library
/ontology/nodes	JSON object with information about the nodes ● `id` ● `label`
/ontology/edges	JSON object with information about the edges ● `from` ● `to` ● `label` ● `arrows` ● `filter` where filter is an array with the property about the node, with the following possible values: ● `object-property`

	• `data-property` • `class` • `individual` • `literal` • `blanknode` • `resource`
/ontology/graph	JSON object with information about the currently parsed ontology, custom written to encompass information about the ontology including filter, metrics etc. • `ontologyName` • `triples` • `nodes` • `edges` • `filter` • `metrics` • `id` where triples, nodes, edges, filter and metrics are arrays with the respective information.
/metrics	JSON object with information about the metrics resulted from using the OWL2VOWL library • `name` • `value`
/files	JSON object listing all the files in the `/uploads` folder
/options	JSON object listing vital information about the application: • `hostname` • `path` • `dirPath` • `fileName` • `url` • `agent` where: `hostname` — the current hostname, such as `localhost`

`path`	the path to the currently selected ontology, such as `/uploads/filename.owl`
`dirPath`	the directory path to the uploads folder (full path) as represented on a computer such as `e:\\labEIS\\app\\uploads`
`fileName`	the name of the currently selected ontology `(simple_family.owl)`
`url`	the URL of the ontology (if the ontology was uploaded using the Upload URL option)
`agent`	`false`

/parser	Returns the parsed ontology using VOWL2OWL

4.1.3 Assumptions and Dependencies

It is assumed that the user has Javascript activated in the browser. This is a precondition for the application to run. Otherwise, the application will not run. Only browsers that support Javascript are targeted.

Also, it is assumed that the operating system has Java installed. This is a critical dependency for calculating the metrics of the ontology.

4.1.4 External Interfaces

WAPT allows the input of an OWL file. The output will be in an text file of triples..

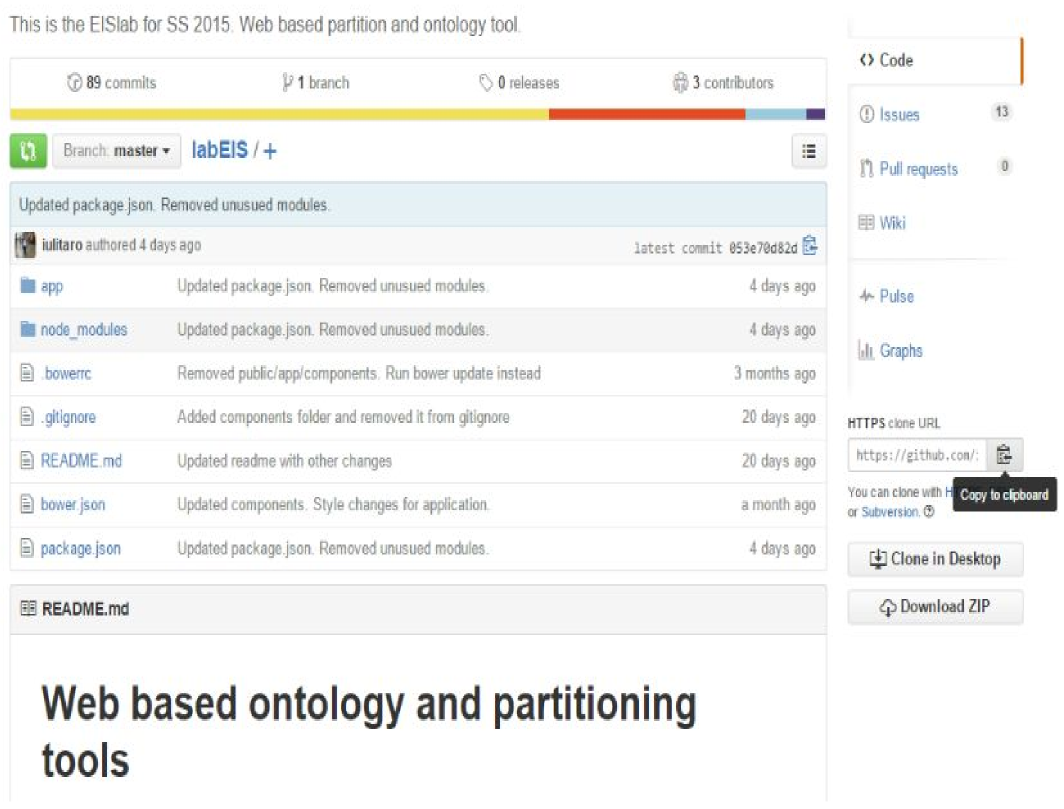
5. Operational Specifications

5.1 Installing WAPT

- The tool can be downloaded from : **https://github.com/EIS-Bonn/MA-INF3232-Lab-SS2015.git** and installed using the following steps;

Step 1:

First *'Copy the link to clipboard'*


This is the EISlab for SS 2015. Web based partition and ontology tool.

89 commits | 1 branch | 0 releases | 3 contributors

Branch: master ▾ labEIS / +

Updated package.json. Removed unused modules.

iulitaro authored 4 days ago — latest commit 053e70d82d

app	Updated package.json. Removed unused modules.	4 days ago
node_modules	Updated package.json. Removed unused modules.	4 days ago
.bowerrc	Removed public/app/components. Run bower update instead	3 months ago
.gitignore	Added components folder and removed it from gitignore	20 days ago
README.md	Updated readme with other changes	20 days ago
bower.json	Updated components. Style changes for application.	a month ago
package.json	Updated package.json. Removed unused modules.	4 days ago

README.md

Web based ontology and partitioning tools

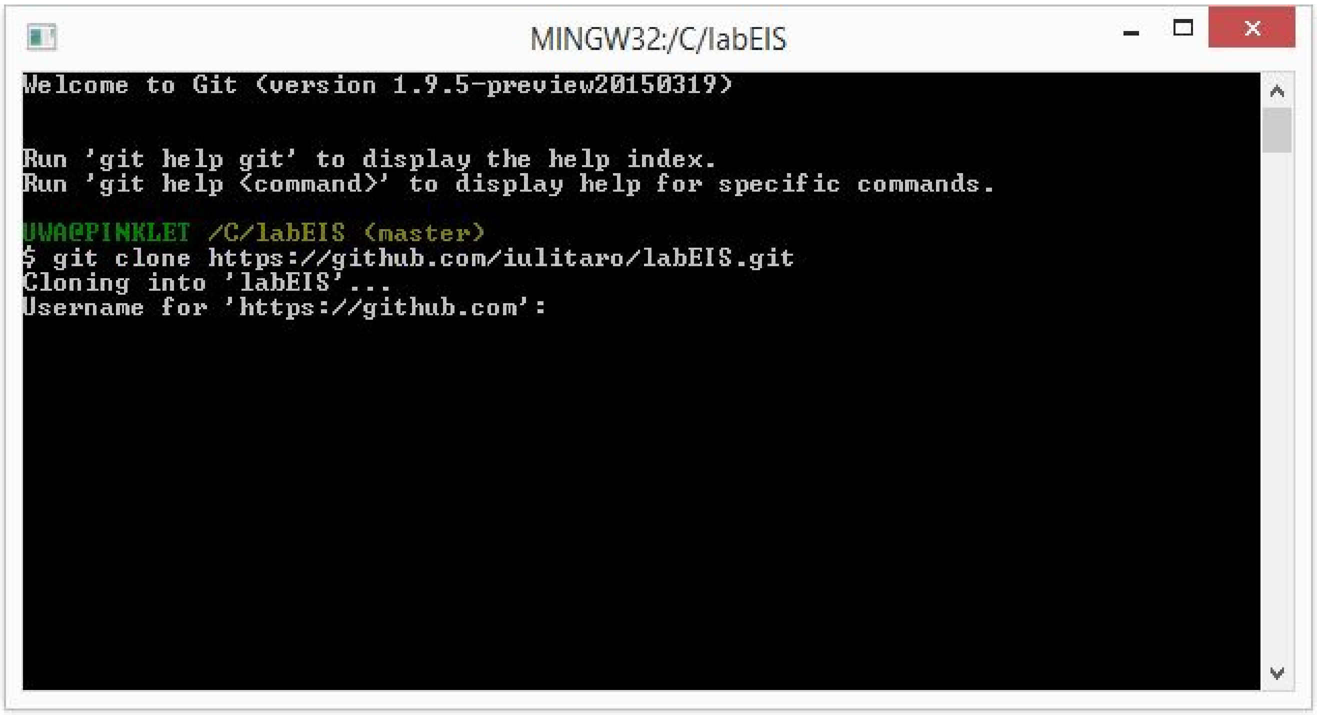
Step 2:

Open *Git Bash* and clone the repository. This can be done by entering '*git clone'* and inserting the link which was copied. Some file names will be reported as "*filename too long*" by Windows because of the package managers (bower and npm). Please perform the following setting:

```
git config core.longpaths true
```

It enables long filenames in the path.

The image shows how to clone the repository:



```
Welcome to Git (version 1.9.5-preview20150319)


Run 'git help git' to display the help index.
Run 'git help <command>' to display help for specific commands.

UWA@PINKLET /C/labEIS (master)
$ git clone https://github.com/iulitaro/labEIS.git
Cloning into 'labEIS'...
Username for 'https://github.com':
```

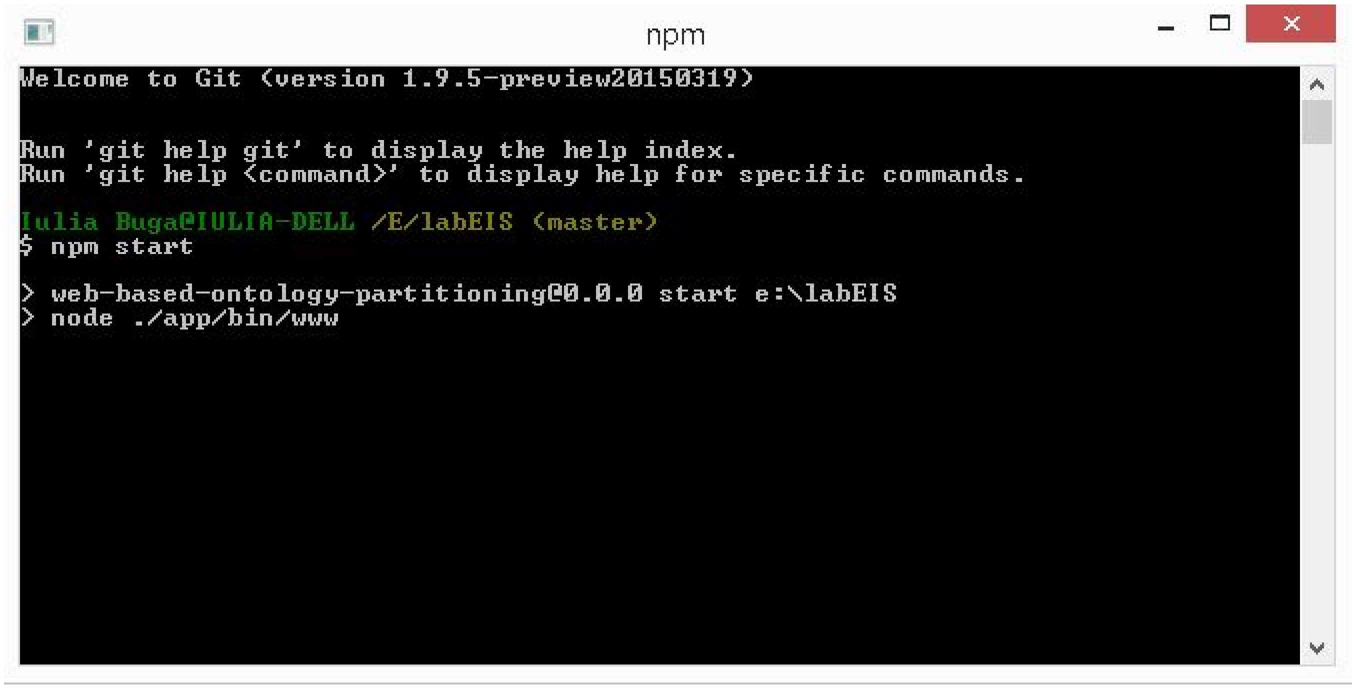
Step 3:

Then following commands can be used after the repository has been cloned in *Git Bash*;

Step 4:

To run the application,the following command can be used in *Git Bash*:

npm start

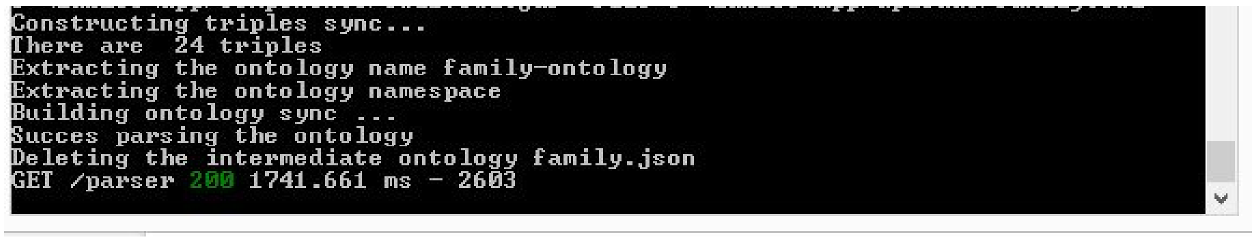


npm

```
Welcome to Git (version 1.9.5-preview20150319)


Run 'git help git' to display the help index.
Run 'git help <command>' to display help for specific commands.

Iulia Buga@IULIA-DELL /E/labEIS (master)
$ npm start

> web-based-ontology-partitioning@0.0.0 start e:\labEIS
> node ./app/bin/www
```

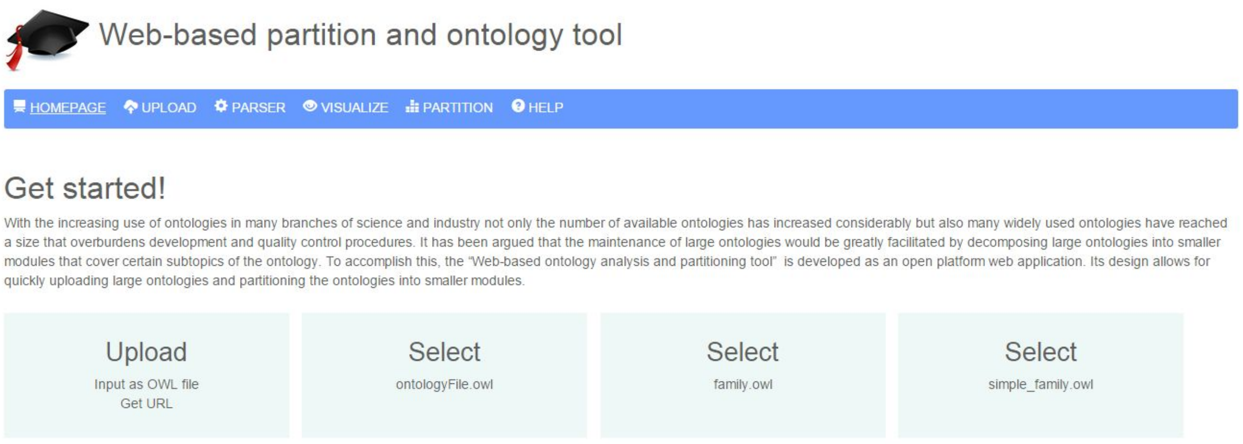
5.2 Getting started

It is recommended that when the application is running, the console is also analyzed. The console displays all relevant information for the application. Here error messages are logged, as well as success messages. The Morgan module is used for error reporting and, in general, for reporting information about the application.



```
Constructing triples sync...
There are  24 triples
Extracting the ontology name family-ontology
Extracting the ontology namespace
Building ontology sync ...
Succes parsing the ontology
Deleting the intermediate ontology family.json
GET /parser 200 1741.661 ms - 2603
```

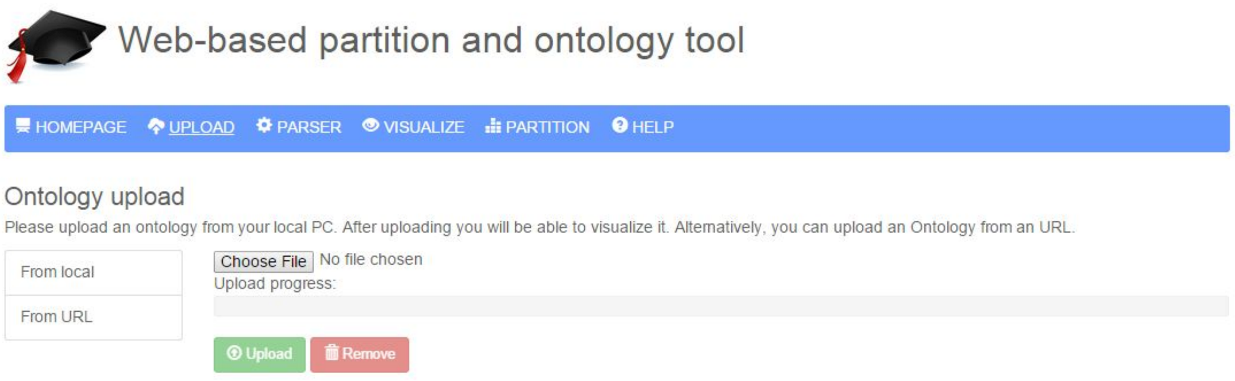
1. ***HOMEPAGE* -** An ontology can be chosen from the list of ontology displayed or the user can also load any ontology by clicking *Upload* . In the homepage the most 3 recent ontologies are displayed in the order in which they are accessed. A message is also available and the user is encouraged to either upload an ontology or choose an existing one. The ontology that were chosen from a URI will not be displayed in this list.



Get started!

With the increasing use of ontologies in many branches of science and industry not only the number of available ontologies has increased considerably but also many widely used ontologies have reached a size that overburdens development and quality control procedures. It has been argued that the maintenance of large ontologies would be greatly facilitated by decomposing large ontologies into smaller modules that cover certain subtopics of the ontology. To accomplish this, the "Web-based ontology analysis and partitioning tool" is developed as an open platform web application. Its design allows for quickly uploading large ontologies and partitioning the ontologies into smaller modules.

2. ***UPLOAD*** - An ontology can be uploaded from the user's local PC into the tool by clicking *Choose File* and can also be removed.The user can also upload ontology using a URL.



Ontology upload

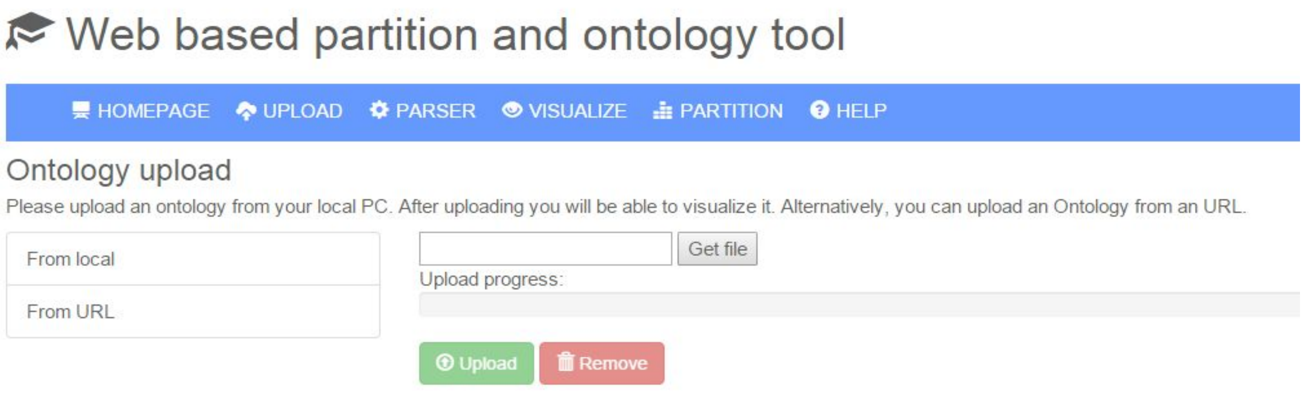
Please upload an ontology from your local PC. After uploading you will be able to visualize it. Alternatively, you can upload an Ontology from an URL.

No file chosen

Upload progress:

For testing, the following URLs was used:
http://dbpedialite.org/titles/Lisp_%28programming_language%29
http://www.dbpedialite.org/things/18016
http://dbpedia.org/resource/Tim_Berners-Lee

Only valid URIs will be accepted. An HTML5 validation is made to serve this purpose.



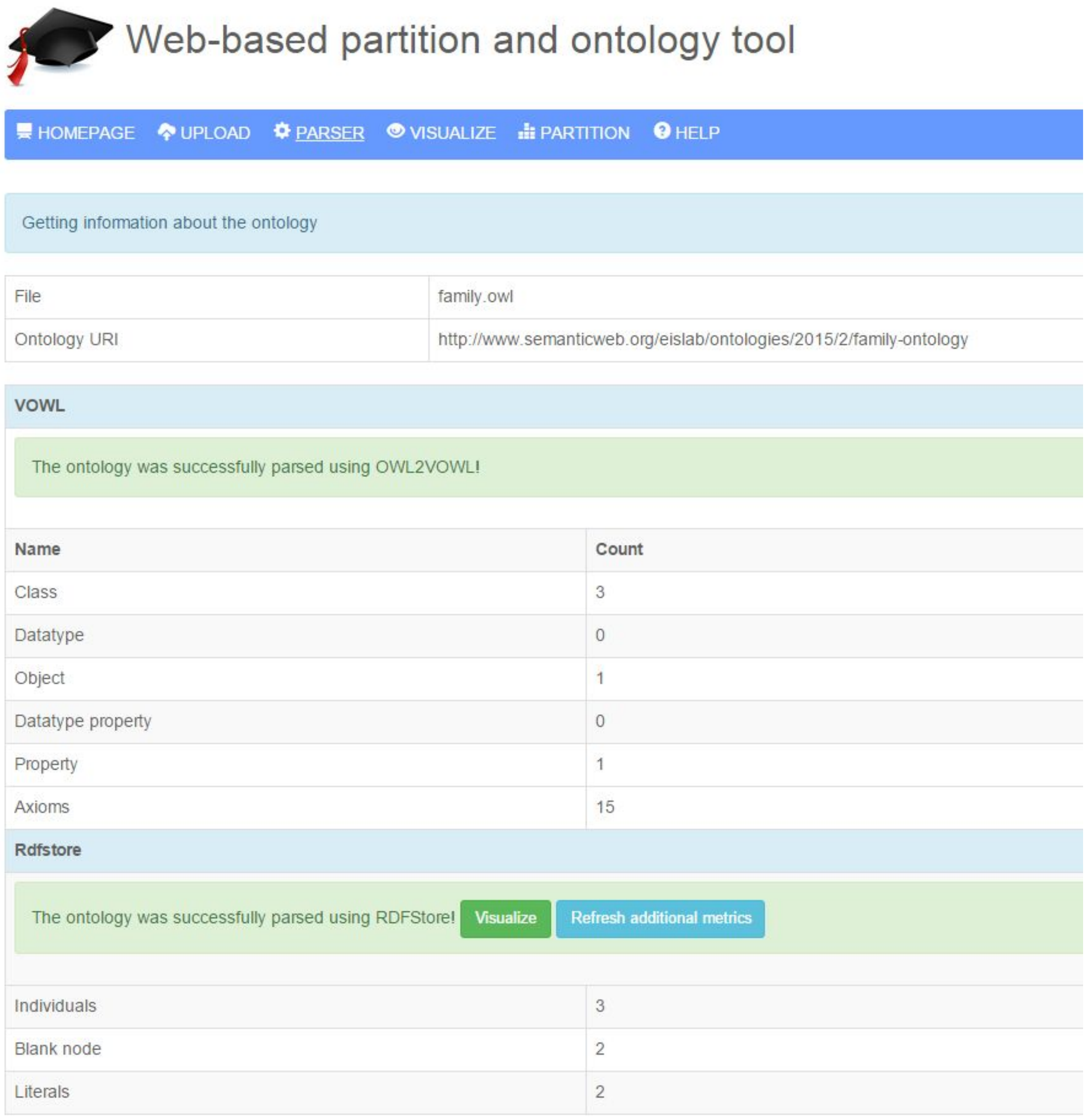
Ontology upload

Please upload an ontology from your local PC. After uploading you will be able to visualize it. Alternatively, you can upload an Ontology from an URL.

Upload progress:

3. ***PARSE*R** - It shows metrics obtained from the uploaded ontology; If the ontology was already parsed, the VOWL2OWL will not run again. Instead, a cached version of the parser ontology will be loaded.

   If you want to visualize the additional metrics, click on the "Refresh additional metrics" button.


Web-based partition and ontology tool
HOMEPAGE
UPLOAD
PARSER
VISUALIZE
PARTITION
HELP
Visualize
Refresh additional metrics

Getting information about the ontology

File	family.owl
Ontology URI	http://www.semanticweb.org/eislab/ontologies/2015/2/family-ontology

VOWL

The ontology was successfully parsed using OWL2VOWL!

Name	Count
Class	3
Datatype	0
Object	1
Datatype property	0
Property	1
Axioms	15

Rdfstore

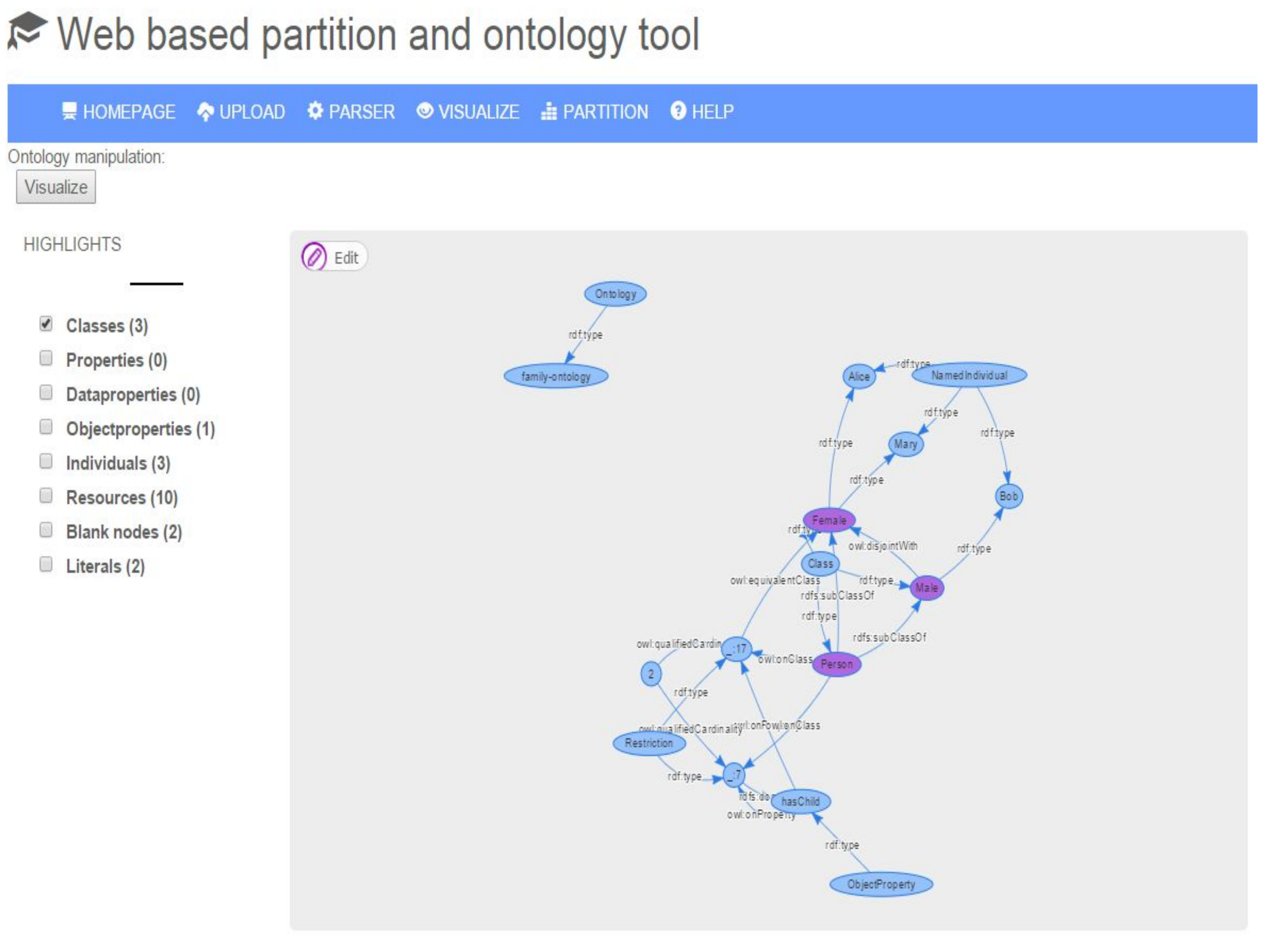
The ontology was successfully parsed using RDFStore!

Individuals	3
Blank node	2
Literals	2

4. ***VISUALIZE***- The uploaded ontology can be visualized as a graph. The metrics of classes, properties, individuals ,resources , blank nodes and literals can be seen on the left side of the page. The metrics here is generated by calculations on the graph which means there is no use for VOWL2OWL like in the parser page.

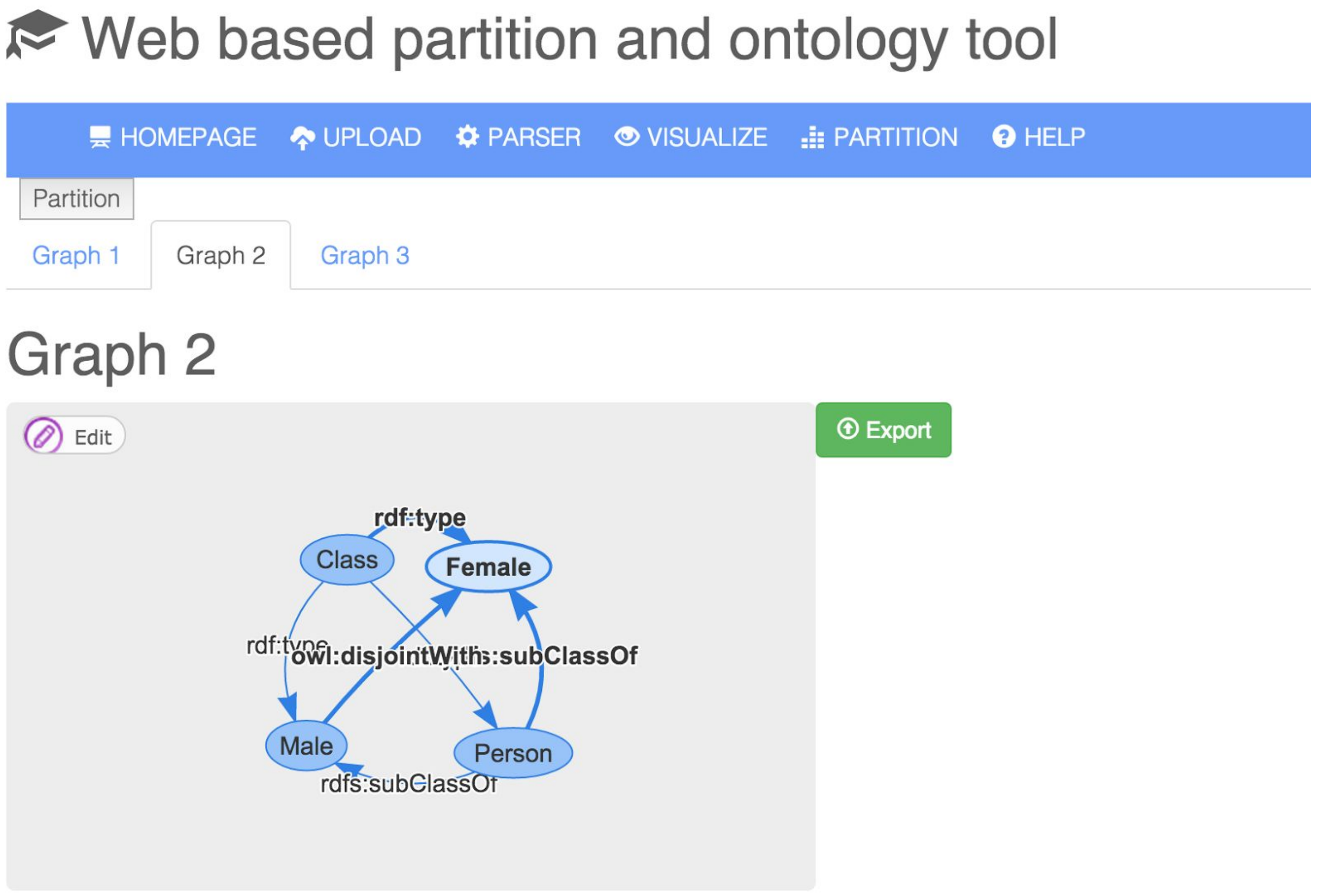
- To visualize the current ontology, the button *Visualize* must be clicked. Instantly, a graph representation of the ontology is created. It is possible to zoom in or zoom out in the dedicated area for the visualization.For an improved understanding of the ontology, certain key items of the ontology can be highlighted. Clicking on an item triggers a highlight into the right hand size.

- To manipulate the graph, click the edit button in the upper left corner of the graph, where you can add, edit and update nodes and edges in the visualized graph.



- [✓] Classes (3)
- [ ] Properties (0)
- [ ] Dataproperties (0)
- [ ] Objectproperties (1)
- [ ] Individuals (3)
- [ ] Resources (10)
- [ ] Blank nodes (2)
- [ ] Literals (2)

5. ***PARTITION*** - It shows the partition of the main graph separated into different tabs based on the applied partitioning algorithm. Each graph in these tabs can be exported as triples in a plain text file. To show the partitions you need to click Partition button.


Web based partition and ontology tool
HOMEPAGE
UPLOAD
PARSER
VISUALIZE
PARTITION
HELP
Partition
Graph 1
Graph 2
Graph 3
Graph 2
Edit
Export
rdf:type
Class
Female
rdf:type
owl:disjointWith
rdfs:subClassOf
Male
Person
rdfs:subClassOf

6. Future work

- Enable editing of ontology (i.e adding and removing nodes).
- Improve Partitioning by adding more algorithms.
- Generate OWL file from a JSON file.
- Add database support (MongoDb).