Contents

Documentation

1. Introduction

OntoCloud aims to visualize mappings between BioPortal ontologies. It uses data collected through BioPortal's web services and SPARQL endpoints. OntoCloud consists of the web application with a GUI and the web service which can be called programmatically.

2. Dependecies

OntoCloud uses:

- D3.js[1] to visualize data.

- express.js[2] and jsdom.js[3] for running the web service.

- JSONExporter plugin for Gephi[4] to prepare graphs in the JSON format.

3. Data

Information about ontologies and number of mappings is gathered through BioPortal's REST service[5], while data about number of predicates in ontologies (used to calculate shape) is collected through SPARQL endpoints[6].

4. OntoCloud GUI

OntoCloud GUI offers the following functionalities

- Zooming/panning of the graph.

- Searching for ontology abbreviations in the graph.

- Main menu

    - Loading of different versions of graph (data gathered at different time points).

    - Visualizing shapes of the graph.

- Node menu (opens when clicking on a node label):

    - Show information about ontology (external link).

[1] http://d3js.org/
[2] http://expressjs.com/
[3] https://github.com/tmpvar/jsdom
[4] https://marketplace.gephi.org/plugin/json-exporter/
[5] http://www.bioontology.org/wiki/index.php/BioPortal_REST_services
[6] http://www.bioontology.org/wiki/index.php/SPARQL_BioPortal

- Show information about ontology mappings (external link).

- Show community of the current ontology.

- Show shape information (when shape mode is enabled).

5. OntoCloud web service

The OntoCloud web service can be called programmatically through the following url: [url]. Currently it accepts two parameters:

- ids: a comma separated list of ontology IDs that we wish to visualize.

- version: number that represents a version of data that we use to make the custom graph.

The web service returns a HTML file containing the graph in the SVG element.

6. Creating a new version of a graph

To create a new version of the graph the following steps have to be done:

1. Preparing node and edge data which can be imported into the Gephi[7] tool

   The /lib directory contains BioPortal wrapper files in ruby. They make it easy to prepare node and edge files that can be imported into the Gephi tool. Examples can be seen in the /lib/examples.rb file. There is also a web service available that prepare the files. It can be accessible through [url].

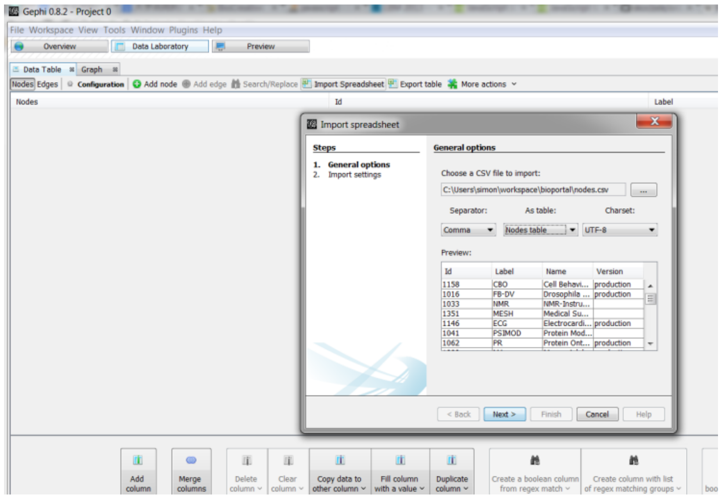
2. Importing the node and edge data into Gephi and preparing visualization data files.

   To prepare visualizations in Gephi we need to take the following steps:

   2.1 Create a new project.

   2.2 Under "Data Laboratory -> Data Table->Import Spreadsheet" we import the node file and the edge file created in Step 1 (Figure 1).

[7] https://gephi.org/, Gephi version 0.8.2 was used.



Import spreadsheet

Steps
1. General options
2. Import settings

General options

Choose a CSV file to import: C:\Users\simon\workspace\bioportal\nodes.csv

Separator: Comma
As table: Nodes table
Charset: UTF-8

Preview:

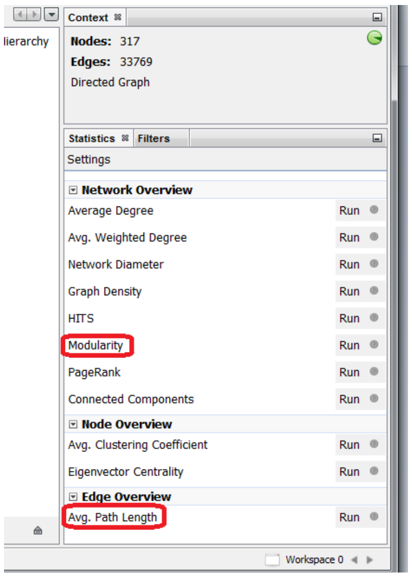
Id	Label	Name	Version
1158	CBO	Cell Behavi...	production
1016	FB-DV	Drosophila ...	production
1033	NMR	NMR-Instru...	
1351	MESH	Medical Su...	
1146	ECG	Electrocardi...	production
1041	PSIMOD	Protein Mod...	
1062	PR	Protein Ont...	production

Figure 1: Importing data to Gephi

2.3 Under "Overview" we can notice unstructured graph created from our data. To create our visualization we need to run the following metrics in the statistics tab (Figure 2):

- Modularity: is a measure of structure in graphs. Graphs with high modularity have separate communities of densely connected nodes inside the communities and sparse connection across communities. With this feature we classified BioPortal ontologies into groups of highly related ontologies.

- Avg. Path Length help: helps to calculate betweenness centrality. Betweenness centrality is a measure of the frequency of occurrence of a particular node in all shortest paths between any two nodes. The feature was used for identifying "bridging" ontologies.

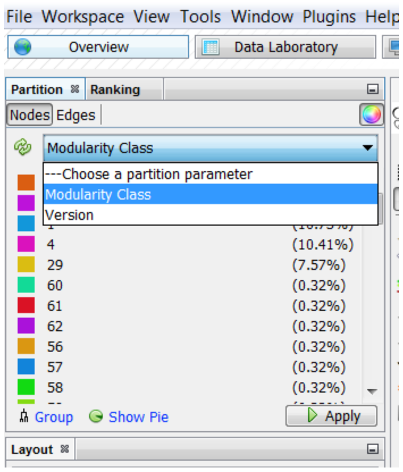


Nodes: 317
Edges: 33769
Directed Graph

Network Overview	
Average Degree	Run
Avg. Weighted Degree	Run
Network Diameter	Run
Graph Density	Run
HITS	Run
Modularity	Run
PageRank	Run
Connected Components	Run
Node Overview	
Avg. Clustering Coefficient	Run
Eigenvector Centrality	Run
Edge Overview	
Avg. Path Length	Run

Figure 2: Statistics metrics

2.4 Under "Overview->Partition->Nodes" we choose "Modularity Class" as the partition parameter (Figure 3). This will color the nodes of the same community. If Modularity Class is not available, try to press the refresh button.



🟥	4	(10.41%)
🟨	29	(7.57%)
🟩	60	(0.32%)
🟥	61	(0.32%)
🟪	62	(0.32%)
🟧	56	(0.32%)
🟦	57	(0.32%)
🟩	58	(0.32%)

Figure 3: Selecting Modularity Class as node partition parameter.

2.5 Under "Overview->Ranking->Nodes" we choose "Betweenness Centrality" as the partition parameter (Figure 4). This will adjust node sizes according to betweenness centrality. Also set the desired minimum and maximum sizes. Existing visualizations use 20 and 60 for minimum and maximum respectively.

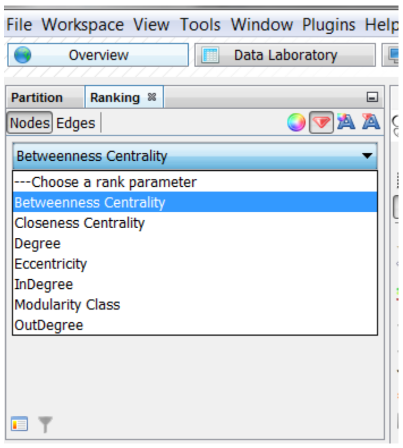
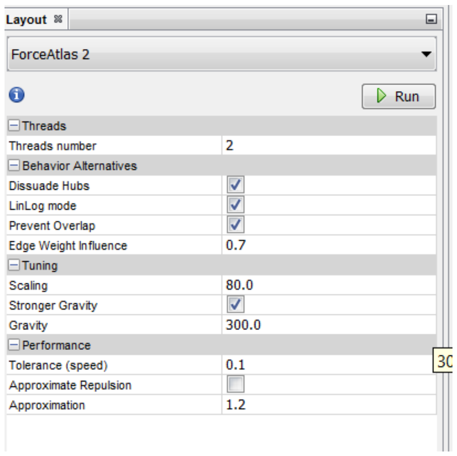


Figure 4: Selecting Betweenness Centrality as Node ranking parameter.

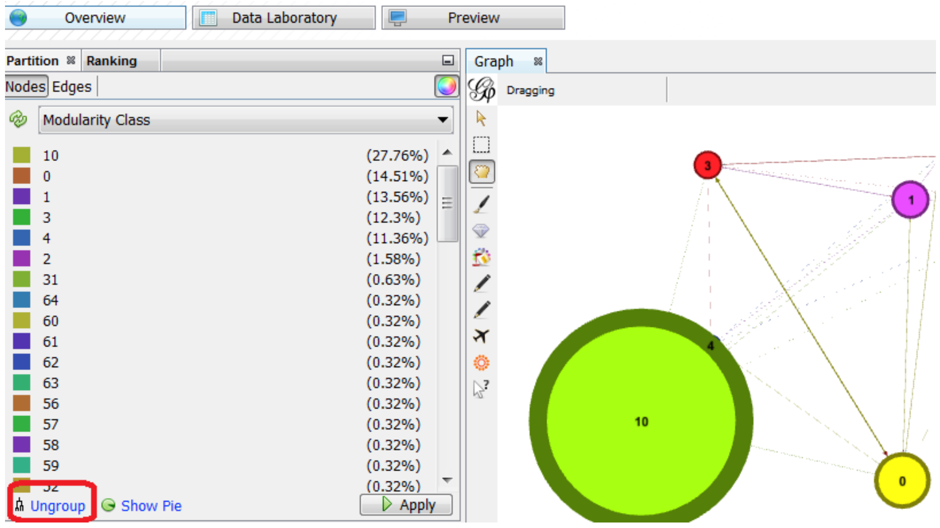
2.6 Under "Overview->Layout" select Force Atlas 2 as the layout algorithm and set the parameters according to visualization results that you wish to acquire. Existing visualizations use parameters as shown on Figure 5. Run the layout until you are satisfied with the results.



Threads	
Threads number	2
Behavior Alternatives	
Dissuade Hubs	✓
LinLog mode	✓
Prevent Overlap	✓
Edge Weight Influence	0.7
Tuning	
Scaling	80.0
Stronger Gravity	✓
Gravity	300.0
Performance	
Tolerance (speed)	0.1
Approximate Repulsion	☐
Approximation	1.2

Figure 5: Force Atlas 2 parameters.

2.7 After running the animation some manual repositioning of the nodes might be needed. Grouping nodes according to their community (color) can be helpful in this process (Figure 6).



Modularity Class

🟨	10	(27.76%)
🟫	0	(14.51%)
🟪	1	(13.56%)
🟩	3	(12.3%)
🟦	4	(11.36%)
🟪	2	(1.58%)
🟩	31	(0.63%)
🟦	64	(0.32%)
🟨	60	(0.32%)
🟪	61	(0.32%)
🟦	62	(0.32%)
🟩	63	(0.32%)
🟫	56	(0.32%)
🟩	57	(0.32%)
🟪	58	(0.32%)
🟩	59	(0.32%)
🟨	52	(0.32%)

Figure 6: Grouping nodes that belong to the same community.

2.8 When satisfied with the visualization, save it as a JSON file through “File->Export->Graph file”. The file should be then added to OntoCloud /json directory.

3. When the file is added to the /json directory, the following files have to be configured so that the new version of graph is included in the OntoCloud’s GUI:

- /onto_cloud/index.html: search for the following line: “<!-- Fixed sub menu content div -->”

- /js/versions.js

4. Final notes on adding a new version of graph into OntoCloud

   - the current version of OntoCloud uses the viewBox parameter[8] with the following parameters "-500 -550 1680 1050". Please keep this in mind when exporting the graph from Gephi.

7. Contact

OntoCloud is a work of Database Center for Life Sciences. For any questions or comments please contact "skocbek at gmail dot com" or "jdkim at dbcls dot rois dot ac dot jp".

[8] http://www.w3.org/TR/SVG/coords.html#ViewBoxAttribute