universitätbonn

Web-based ontology analysis and partitioning tool

Ahmad Alzeitoun
Imuwahen Osazuwa
Iulia Buga

Supervised by:
Dr. Gökhan Coskun and Irlan Grangel

Agenda

- Objective
- Organization
- Solution
- Challenges
- Demo
- Q&A

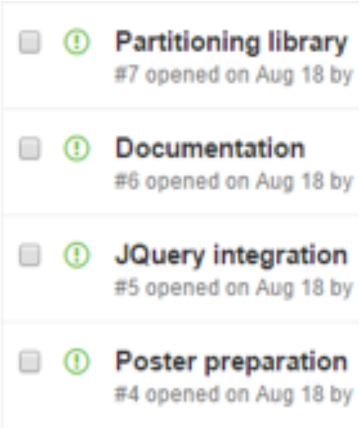
Objective

- Decompose large ontologies into smaller modules.
- Support the comprehension of an existing ontology and improve the process of ontology interlink.

Organization

- **Weekly*** team meetings
  - with mentors on Wednesdays
  - with team on Sundays

- **Github** for managing code/tasks
- **Google Drive** for managing files

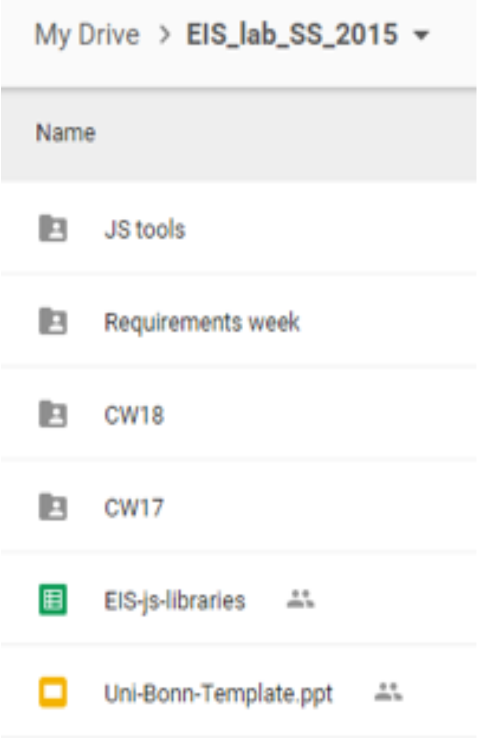


□ Partitioning library
#7 opened on Aug 18 by

□ Documentation
#6 opened on Aug 18 by

□ JQuery integration
#5 opened on Aug 18 by

□ Poster preparation
#4 opened on Aug 18 by



My Drive > EIS_lab_SS_2015

Name
JS tools
Requirements week
CW18
CW17
EIS-js-libraries
Uni-Bonn-Template.ppt

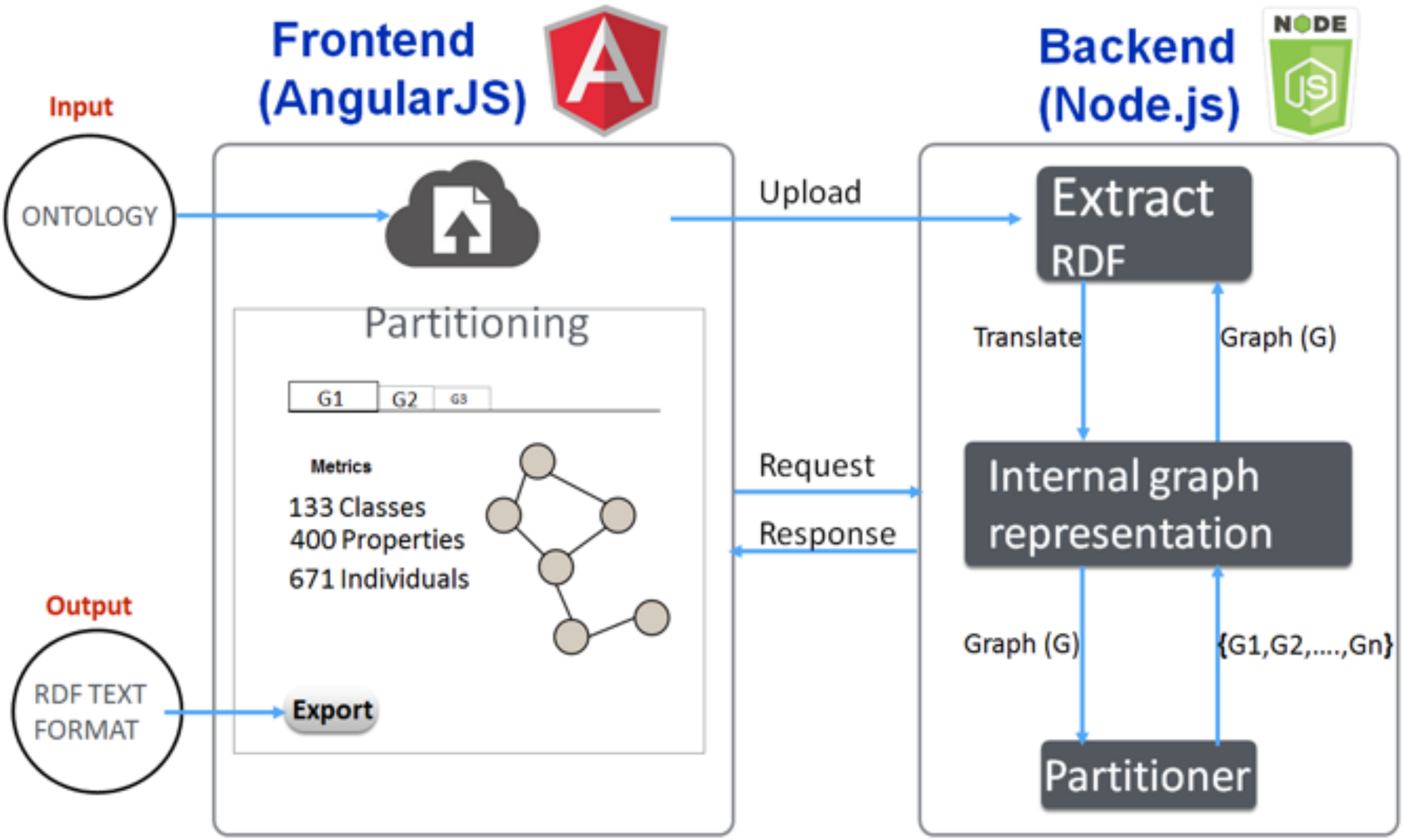
Solution

- Programming language: Javascript
- Environment: Windows, Mac Os
- Tools: SublimeText, Atom, GitSCM

Tasks

1. Technology stack
2. Upload
3. Parse
4. Metrics
5. Filtering
6. Visualize and highlights
7. Partition and export
8. Document
9. Test
10. Research

Application flow



Technology stack

- MEAN
  - MongoDB
  - ExpressJS
  - AngularJS
  - Node.js

Image credit: http://adrianmejia.com/blog/2014/10/03/mean-stack-tutorial-mongodb-expressjs-angularjs-nodejs/

- Package managers:
  - Bower (frontend)
  - npm (backend)

- Libraries
  - RdfStore (backend)
  - VisJS (frontend)

Upload

Local upload

- OWL file as input
- processed by NodeJS and RDFStore
- dedicated folder: *uploads*

URL upload

- processed by NodeJS and RDFStore
- *no storage*

Output: triples (subject, predicate, object)

Parse

OWL2VOWL

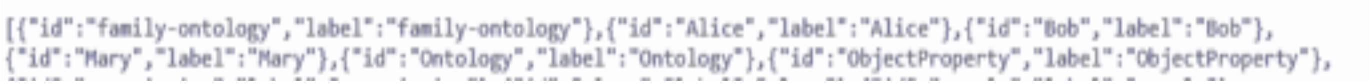
- local/URI upload
- generate JSON in VOWL format

RDFStore

- local/URI upload
- generate custom JSON of edges, nodes, graph

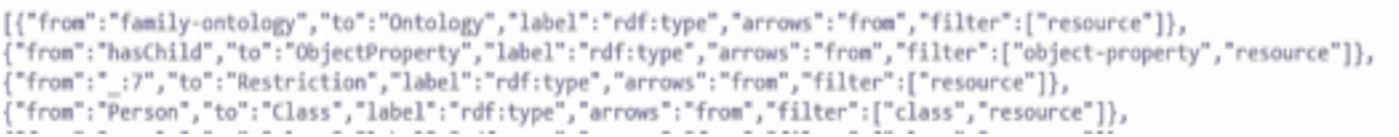
Output: REST endpoints

- nodes



```
[{"id":"family-ontology","label":"family-ontology"},{"id":"Alice","label":"Alice"},{"id":"Bob","label":"Bob"},
{"id":"Mary","label":"Mary"},{"id":"Ontology","label":"Ontology"},{"id":"ObjectProperty","label":"ObjectProperty"},
```

- edges

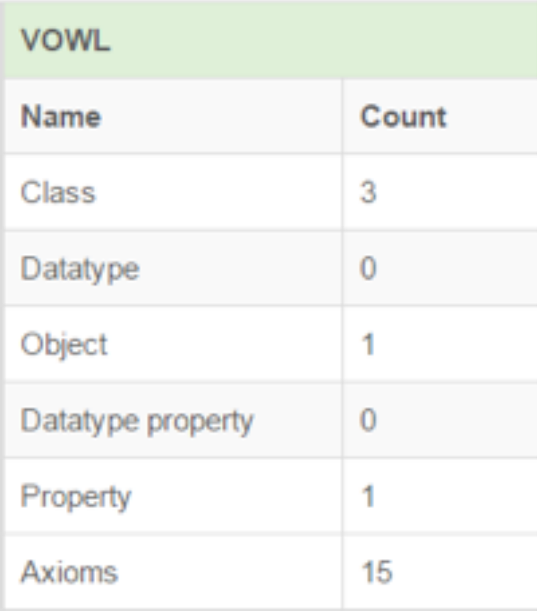


```
[{"from":"family-ontology","to":"Ontology","label":"rdf:type","arrows":"from","filter":["resource"]},
{"from":"hasChild","to":"ObjectProperty","label":"rdf:type","arrows":"from","filter":["object-property","resource"]},
{"from":"_:7","to":"Restriction","label":"rdf:type","arrows":"from","filter":["resource"]},
{"from":"Person","to":"Class","label":"rdf:type","arrows":"from","filter":["class","resource"]},
```

Metrics

OWL2VOWL

- already available
- generate VOWL JSON

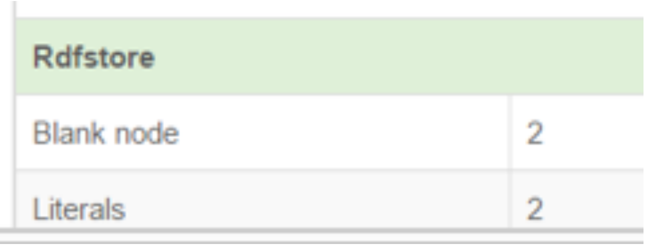


VOWL

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

RDFStore

- not available
- custom code to add bonus metrics



Rdfstore

Blank node	2
Literals	2

Filtering

- Generate input for visualization
- Process the nodes and edges
- Add a filter to each edge
- Implementation:
  - Node.js using RDFStore

Visualize and highlights

VisJS visualization

- graph visualization
  - Nodes:

    ```
    [ {id: 1, label: 'Class'},{id: 2, label: 'Male'}]
    ```

  - Edges:

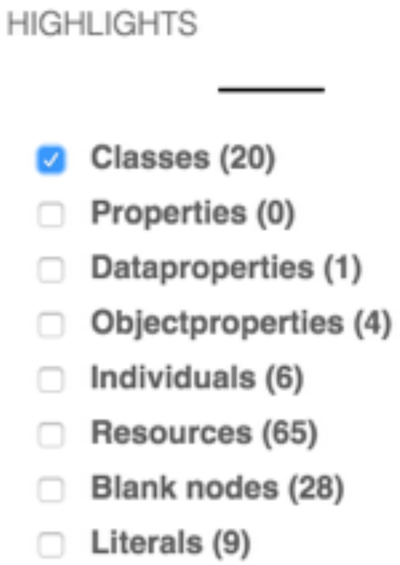
    ```
    [ {from: 2, to: 1, label:'rdf:type', arrows:'from', filter:['class','resource']}]
    ```

- manipulation
  - add, edit and delete nodes and edges
  - export as image

Visualize and highlights

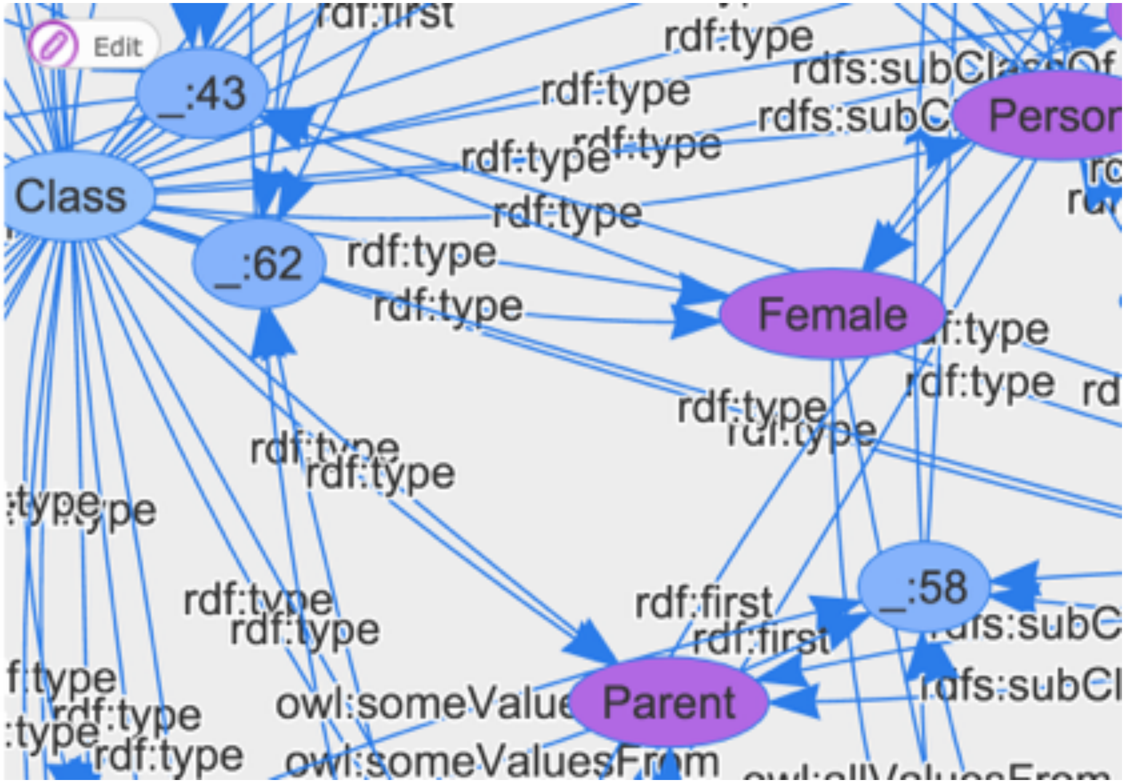
Highlights

Node: shape, color



HIGHLIGHTS

☑ Classes (20)
□ Properties (0)
□ Dataproperties (1)
□ Objectproperties (4)
□ Individuals (6)
□ Resources (65)
□ Blank nodes (28)
□ Literals (9)



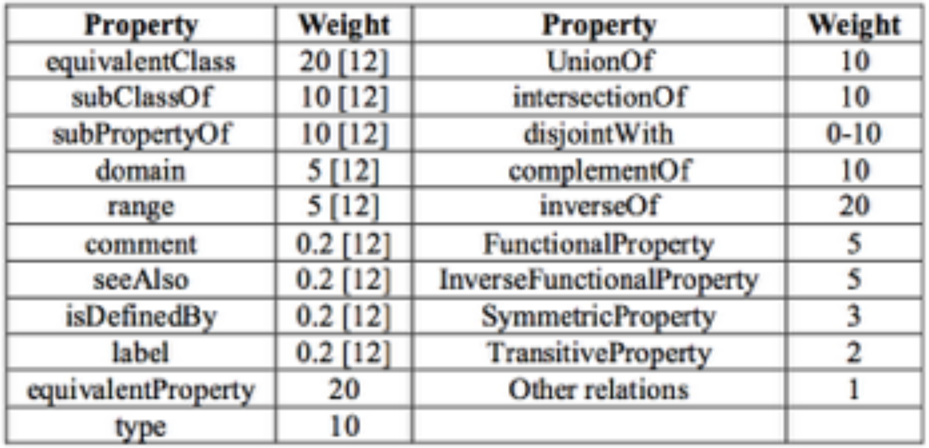
Partitioning

1. constructing weighted matrix
2. neighborhood random walk distance method
3. Silhouette: criterion function
4. Agglomerative algorithm

"*Graph-based Partitioning of Ontology with Semantic Similarity*"
A. Rezanaeian, M. Naghibzadeh, 2013

Partitioning

1. constructing weighted matrix



Property	Weight	Property	Weight
equivalentClass	20 [12]	UnionOf	10
subClassOf	10 [12]	intersectionOf	10
subPropertyOf	10 [12]	disjointWith	0-10
domain	5 [12]	complementOf	10
range	5 [12]	inverseOf	20
comment	0.2 [12]	FunctionalProperty	5
seeAlso	0.2 [12]	InverseFunctionalProperty	5
isDefinedBy	0.2 [12]	SymmetricProperty	3
label	0.2 [12]	TransitiveProperty	2
equivalentProperty	20	Other relations	1
type	10		

Normalization: NRWD input

$$W_v = \frac{WeightofOutgingEdgeFromNode(V)}{\sum_{v \in V} WeightsofOutgouingedgesFromNode(V)}$$

Partitioning

2. neighborhood random walk distance method

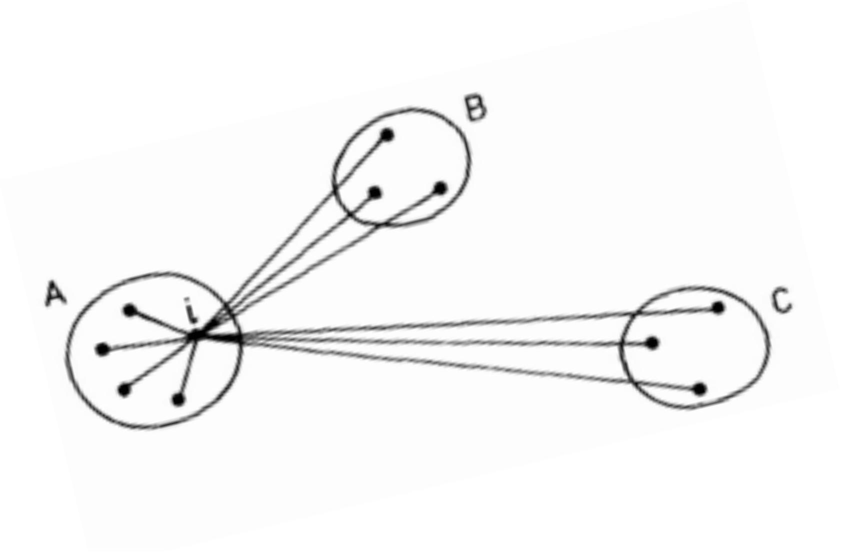
$$P_A = \begin{array}{c} \\ r_1 \\ r_2 \\ \cdots \\ \cdots \\ r_{10} \\ r_{11} \\ v_{11} \\ v_{12} \end{array} \begin{array}{c} \begin{array}{ccccccc} r_1 & r_2 & \ldots & r_{10} & r_{11} & v_{11} & v_{12} \end{array} \\ \left(\begin{array}{ccccccc} 0 & 1/3 & \ldots & 0 & 0 & 1/3 & 0 \\ 1/3 & 0 & \ldots & 0 & 0 & 1/3 & 0 \\ \ldots & \ldots & \ldots & \ldots & \ldots & \ldots & \ldots \\ \ldots & \ldots & \ldots & \ldots & \ldots & \ldots & \ldots \\ 0 & 0 & \ldots & 0 & 1/3 & 0 & 1/3 \\ 0 & 0 & \ldots & 1/3 & 0 & 0 & 1/3 \\ 1/8 & 1/8 & \ldots & 0 & 0 & 0 & 0 \\ 0 & 0 & \ldots & 1/4 & 1/4 & 0 & 0 \end{array} \right) \end{array}$$

- NRWD algorithm: closeness measurement

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

Partitioning

3. Silhouette: criterion function



- $$s(i) = \frac{a(i) - b(i)}{\max\{a(i), b(i)\}}$$

$$-1 \leq s(i) \leq 1.$$

- Score of cluster c:

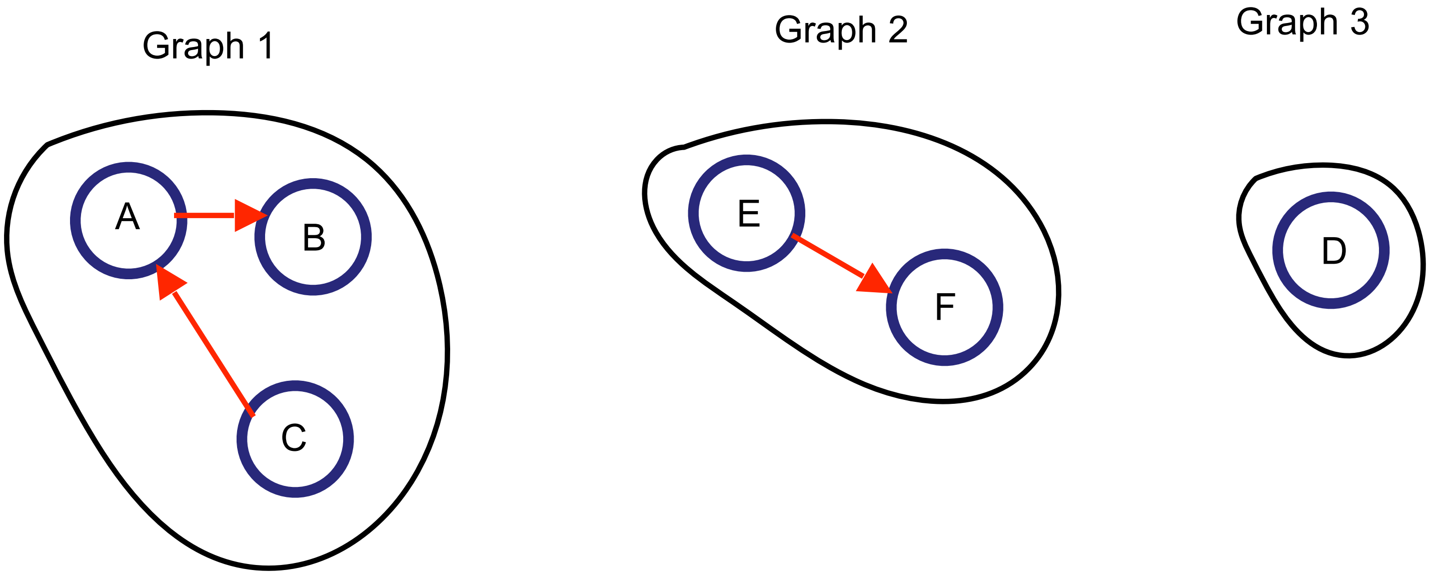
$$s_c = \frac{\sum_{i \in c} s(i)}{n_c}$$

- Partitioning score for all clusters C:

$$score(C) = average_{c \in C}(s_c)$$

Partitioning

4. Agglomerative algorithm $O(n^3.complexity_{scoring})$



Export as RDF triples:

<http://example.org/#spiderman> <http://www.perceive.net/schemas/relationship/enemyOf> <http://example.org/#green-goblin> .

Future work

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

Challenges

- Technology stack: setting up the stack.
- AngularJS and Node.js connection.
- Javascript libraries shortcomings .
- Converting JSON to OWL.
- No partitioning algorithm implemented in Javascript.

Q&A

Image credit: http://blog.mypermissions.com/new-study-shows-comprehension-rates-for-tos-of-googlefacebook-is-less-than-40/