**Scholarly Communication & Research Analytics**

## Documentation (draft – to be enriched in future)

## Table of Contents

Code description 1

API documentation 2

Database description 6

## Code Description

The code of the described project is uploaded in:

<https://github.com/nmpegetis/visualizationAPI>

The project uses the following node\_modules installed with npm:

* body-parser
* cookie-parser
* debug
* ejs
* express
* morgan
* serve-favicon
* sqlite3

Their versions are on package.json

The main **code for all web services** is all written in app.js (ignore the files: app\_acm\_openaire, app\_old, index2. These are only in experimental version)

In the directory public/ there is everything needed for **the client side**:

* **index.html:** *not created yet.* Should contain a welcoming index page, as this of <https://github.com/nmpegetis/visualizationAPI> repository (but not set in the public directory) (e.g. <http://astero.di.uoa.gr/graphs/index.html> )
* **experiments.html:** contains the client side page where is the switching among visualizations to be created using the d3js library. Also contains all the ajax calls for bringing the listings of journals/conferences/topics/corpus/authors to the equivalent web services.
* **js/ directory**: you can find all the client side javascript libraries used for the index and the experiments page, as well as all the visualizations
* **css/ directory**: you can find all the client side css libraries used for the index and the experiments page, as well as all the visualizations
* **cloud.html:** contains the client side cloud visualization using the d3js library. Also contains the ajax call to the equivalent web service.
* **spider.html:** contains the client side spider visualization using the d3js and AngularJS library. Also contains the ajax call to the equivalent web service. *(Little more work needed, estimated Oct-Dec ‘16)*
* **trends.html:** contains the client side trends visualization using the d3js library. Also contains the ajax call to the equivalent web service. *(Little more work needed, estimated Oct-Dec ‘16)*
* **newtrends.html:** contains a new bunch of visualization trends and stacks demonstration altering by clicking using the d3js library, with aiming in future work visualizations. Also contains the ajax call to the equivalent web service.
* **cloud.html:** contains the client side cloud visualization using the d3js library. Also contains the ajax call to the equivalent web service.
* **graph.html:** contains the client side graph visualization using the d3js library. Also contains the ajax call to the equivalent web service. *Still work needed here. Not working yet.*

Having already downloaded the repository (contributing to the repository would be even more desirable), you should download and install NPM and node.js.

<https://www.npmjs.com/> (for installing other node packages needed and updating the existing)

<https://nodejs.org/en/> (the above code works properly with the v4.4.7 LTS of Node)

Execution:

To run the API the server must be online and listening to a port. In the above code case the server is set to listen on port **3001** and to start you should run:

visualizationuser@prozac-MS-7924:/var/www/api/code$ node app.js

To run the server in Prozac serving whatever described in the public/ directory of the above repository, it is set to listen in the **8080** port and to start you should run:

visualizationuser@prozac-MS-7924:/var/www/api/code$ http-server

If you want credentials for the visualizationuser in Prozac, feel free to ask for them.

In future work, there should be set a task manager (e.g. grunt, <http://gruntjs.com/>) and a process manager (e.g. pm2, <http://pm2.keymetrics.io/>) to make all this easier handle. Plans for this begin from October 2016.

## API using GET Method

**Topic Modeling: webServices API for ACM**

**All webServices names:**

1. getExperiments
2. getConnections
3. getNodes
4. getTopics
5. getCloud
6. getTrends
7. getSpider
8. getEntitiesList
9. getJournalsList
10. getConferencesList
11. getTopicsList

WebServiceName (in bold, naming by method used and main case name. For example, a webservice for experiments using the get method: getExperiments)

Description: Short description

Indented by one indent and underlined is the parameter. For example: ex, s, id, set, etc

All parameters get a *yes/no* value to show if it is obligatory to exist in header of the request or not.

Indented by one more indent and placed below options list are: the possible values for each parameter, or the pattern that the parameter values should follow

Link has an example working url

Example of describing webServices:

**getExperiments**

Description: web services fetching information about experiments

ex: yes

* options:

ACM\_400T\_1000IT\_6000CHRs\_150\_40PRN100B\_4M\_4TH\_cosNoPPR, ACM\_400T\_950IT\_6000CHRs\_150\_40PRN100B\_4M\_4TH\_cosWVNoPPR

s: yes

* options:

0.5, 0.6, ..., 1

*link*: <http://prozac.madgik.di.uoa.gr:3001/getConnections?ex=ACM_400T_1000IT_6000CHRs_150_40PRN100B_4M_4TH_cosNoPPR&s=0.8>

**1. getExperiments**

Description: Brings information about all experiments run in the ACM database that can be visualized and used for the description of each experiment and mainly from this webservice we use the names of experiments to be added in all the other webservices in parameter ‘ex’.

*link*: <http://prozac.madgik.di.uoa.gr:3001/getExperiments>

**2. getConnections**

Description: Brings all the connections-linking in groups of two nodes (authors for the case study of ACM). Also, shows the category field that each node (author) belongs to and the similarity between the two nodes in values from 0 to 1. When setting the ‘s’ parameter, this is the similarity of groups-of-2 nodes threshold above which the results are fetched.

ex: yes

* options:

ACM\_400T\_1000IT\_6000CHRs\_150\_40PRN100B\_4M\_4TH\_cosNoPPR, ACM\_400T\_950IT\_6000CHRs\_150\_40PRN100B\_4M\_4TH\_cosWVNoPPR

s: yes

* options:

0.5, 0.6, ..., 1

*link*: <http://prozac.madgik.di.uoa.gr:3001/getConnections?ex=ACM_400T_1000IT_6000CHRs_150_40PRN100B_4M_4TH_cosNoPPR&s=0.8>

**3. getNodes**

Description: Brings information about the nodes. In case of ACM brings information about the authors.

ex: yes

* options:

ACM\_400T\_1000IT\_6000CHRs\_150\_40PRN100B\_4M\_4TH\_cosNoPPR, ACM\_400T\_950IT\_6000CHRs\_150\_40PRN100B\_4M\_4TH\_cosWVNoPPR

*link*: <http://prozac.madgik.di.uoa.gr:30011/getNodes?ex=ACM_400T_1000IT_6000CHRs_150_40PRN100B_4M_4TH_cosNoPPR>

**4. getTopics**

Description: Brings information about the topics, theirs consisting words and the weight of each one.

ex: yes

* options: ACM\_400T\_1000IT\_6000CHRs\_150\_40PRN100B\_4M\_4TH\_cosNoPPR, ACM\_400T\_950IT\_6000CHRs\_150\_40PRN100B\_4M\_4TH\_cosWVNoPPR

id (multiple): yes

* options:

all, 0, 1, ..., 32, 34, ...

*link*: <http://prozac.madgik.di.uoa.gr:3001/getTopics?ex=ACM_400T_1000IT_6000CHRs_150_40PRN100B_4M_4TH_cosNoPPR&id=all>

**5. getCloud**

Description: Brings all the information needed to create a cloud visualization. The output is set to fit exactly the input for the visualization.

ex: yes

* options: ACM\_400T\_1000IT\_6000CHRs\_150\_40PRN100B\_4M\_4TH\_cosNoPPR, ACM\_400T\_950IT\_6000CHRs\_150\_40PRN100B\_4M\_4TH\_cosWVNoPPR

id (multiple): yes

* options:

all, 0, 1, ..., 32, 34, ...

*link*: <http://prozac.madgik.di.uoa.gr:3001/getCloud?ex=ACM_400T_1000IT_6000CHRs_150_40PRN100B_4M_4TH_cosNoPPR&id=32,34>

**6. getTrends**

Description: Brings all the information needed to create a trend visualization. The output is set to fit exactly the input for the visualization.

ex: yes

* options: ACM\_400T\_1000IT\_6000CHRs\_150\_40PRN100B\_4M\_4TH\_cosNoPPR, ACM\_400T\_950IT\_6000CHRs\_150\_40PRN100B\_4M\_4TH\_cosWVNoPPR

set: yes

* options:

corpus, conference, journal

id (multiple): no

* options:

all, {for journals: 0163-5948, 0362-1340, 0163-5999, ... }, {for conferences: SERIES402, SERIES473, SERIES324 }, {for corpus: no id}

*link*: <http://prozac.madgik.di.uoa.gr:3001/getTrends?ex=ACM_400T_1000IT_6000CHRs_150_40PRN100B_4M_4TH_cosNoPPR&set=journal&id=0163-5948>

**7. getSpider**

Description: Brings all the information needed to create a spider visualization. The output is set to fit exactly the input for the visualization.

ex: yes

* options: ACM\_400T\_1000IT\_6000CHRs\_150\_40PRN100B\_4M\_4TH\_cosNoPPR, ACM\_400T\_950IT\_6000CHRs\_150\_40PRN100B\_4M\_4TH\_cosWVNoPPR

set: yes

* options:

topics, authors

id (multiple): yes

* options:

all, 81100002314, 81100235850, 81100145665

s: no

* options:

0.5, 0.6, ..., 1. If no 's' given then s=0.9

*link*: <http://prozac.madgik.di.uoa.gr:3001/getSpider?ex=ACM_400T_1000IT_6000CHRs_150_40PRN100B_4M_4TH_cosNoPPR&set=authors&id=81100002314,81100235850&s=0.8>

**8. getEntitiesList**

Description: Brings all information needed for selecting authors for the visualization. A list of authors.

ex: yes

* options:

ACM\_400T\_1000IT\_6000CHRs\_150\_40PRN100B\_4M\_4TH\_cosNoPPR, ACM\_400T\_950IT\_6000CHRs\_150\_40PRN100B\_4M\_4TH\_cosWVNoPPR

*link*: <http://prozac.madgik.di.uoa.gr:3001/getEntitiesList?ex=ACM_400T_1000IT_6000CHRs_150_40PRN100B_4M_4TH_cosNoPPR>

**9. getJournalsList**

Description: Brings all information needed for selecting journals for the visualization. A list of journals.

ex: yes

* options:

ACM\_400T\_1000IT\_6000CHRs\_150\_40PRN100B\_4M\_4TH\_cosNoPPR, ACM\_400T\_950IT\_6000CHRs\_150\_40PRN100B\_4M\_4TH\_cosWVNoPPR

*link*: <http://prozac.madgik.di.uoa.gr:3001/getJournalsList?ex=ACM_400T_1000IT_6000CHRs_150_40PRN100B_4M_4TH_cosNoPPR>

**10. getConferencesList**

Description: Brings all information needed for selecting conferences for the visualization. A list of conferences.

ex: yes

* options:

ACM\_400T\_1000IT\_6000CHRs\_150\_40PRN100B\_4M\_4TH\_cosNoPPR, ACM\_400T\_950IT\_6000CHRs\_150\_40PRN100B\_4M\_4TH\_cosWVNoPPR

*link*: <http://prozac.madgik.di.uoa.gr:3001/getConferencesList?ex=ACM_400T_1000IT_6000CHRs_150_40PRN100B_4M_4TH_cosNoPPR>

**11. getTopicsList**

Description: Brings all information needed for selecting topics for the visualization. A list of topics.

ex: yes

* options:

ACM\_400T\_1000IT\_6000CHRs\_150\_40PRN100B\_4M\_4TH\_cosNoPPR, ACM\_400T\_950IT\_6000CHRs\_150\_40PRN100B\_4M\_4TH\_cosWVNoPPR

*link*: <http://prozac.madgik.di.uoa.gr:3001/getTopicsList?ex=ACM_400T_1000IT_6000CHRs_150_40PRN100B_4M_4TH_cosNoPPR>

## Database Description

The queries executed from the API in the database of the described project are uploaded in config.js:

<https://github.com/nmpegetis/visualizationAPI/blob/master/config.js>

They are separated in groups of queries according to the webservice that executes them. There is set an alias for each column of all the query results. The reason why we do this is that there shouldn’t be changes in code in future but only in the queries. This is handy, as in this way if there are changes with new databases then the only thing needed is to find the correlations between the new and the old database and just make the query work by assigning the same name aliases to the new columns (as the webservices see the aliases in order to work) and in the exact same order as they are now.

From the database of June 2016 the tables that are finally used in order to work all the visualizations are:

Tables:

Experiment

entityTopicDistribution

TopicDescription

topicdetails

topicanalysis

Conference

Journal

Views:

EntitySimilarityView\_authors

Running commands:

<http://localhost:9000/trends.html?ex=ACM_400T_1000IT_6000CHRs_150_40PRN100B_4M_4TH_cosNoPPR&set=corpus&id=all&type=area>

<http://localhost:9000/trends.html?ex=ACM_400T_1000IT_6000CHRs_150_40PRN100B_4M_4TH_cosNoPPR&set=corpus&id=all&type=stream>

<http://localhost:9000/trends.html?ex=ACM_400T_1000IT_6000CHRs_150_40PRN100B_4M_4TH_cosNoPPR&set=corpus&id=all&type=line>

* http-server –p9000
* node app.js

<http://localhost:8080/trends.html?ex=ACM_400T_800IT_4000CHRs_300_20PRN100B_4M_4TH_cosPPR_&set=conference&id=SIGGRAPH&type=area>

<http://88.197.53.136:8080/trends.html?ex=ACM_400T_800IT_4000CHRs_300_20PRN100B_4M_4TH_cosPPR_&set=corpus&id=all&type=area>