

---

# Scanner Documentation

*Release 1.7*

**A. Pascual Saavedra**

February 07, 2017



<b>1</b>	<b>What is Scanner?</b>	<b>3</b>
1.1	Architecture . . . . .	3
1.2	More . . . . .	4
<b>2</b>	<b>Installation</b>	<b>5</b>
2.1	Building Scanner . . . . .	5
2.2	Running Scanner . . . . .	5
<b>3</b>	<b>API Schema</b>	<b>7</b>
3.1	Schemes . . . . .	7
<b>4</b>	<b>API Usage</b>	<b>9</b>
4.1	POST /tweets . . . . .	9
4.2	GET /tweets . . . . .	9
4.3	GET /tweets/{tweetId}/metrics . . . . .	11
4.4	GET /tweets/{tweetId}/history . . . . .	11
4.5	GET /users . . . . .	12
4.6	GET /users/{userId}/metrics . . . . .	14
4.7	GET /users/{userId}/network . . . . .	14
4.8	GET /communities . . . . .	15
4.9	Adding more tweets for the same topic . . . . .	16
<b>5</b>	<b>Metrics</b>	<b>17</b>
5.1	Direct metrics . . . . .	17
5.2	Indirect metrics . . . . .	17
<b>6</b>	<b>Metrics Usage</b>	<b>19</b>
6.1	Searching for Relevant Tweets and Users . . . . .	19
6.2	Tweet Ratio . . . . .	20
6.3	Influence . . . . .	20
6.4	Follow Relation . . . . .	21
6.5	Relevance . . . . .	21
<b>7</b>	<b>Demo</b>	<b>23</b>



Contents:



---

## What is Scanner?

---

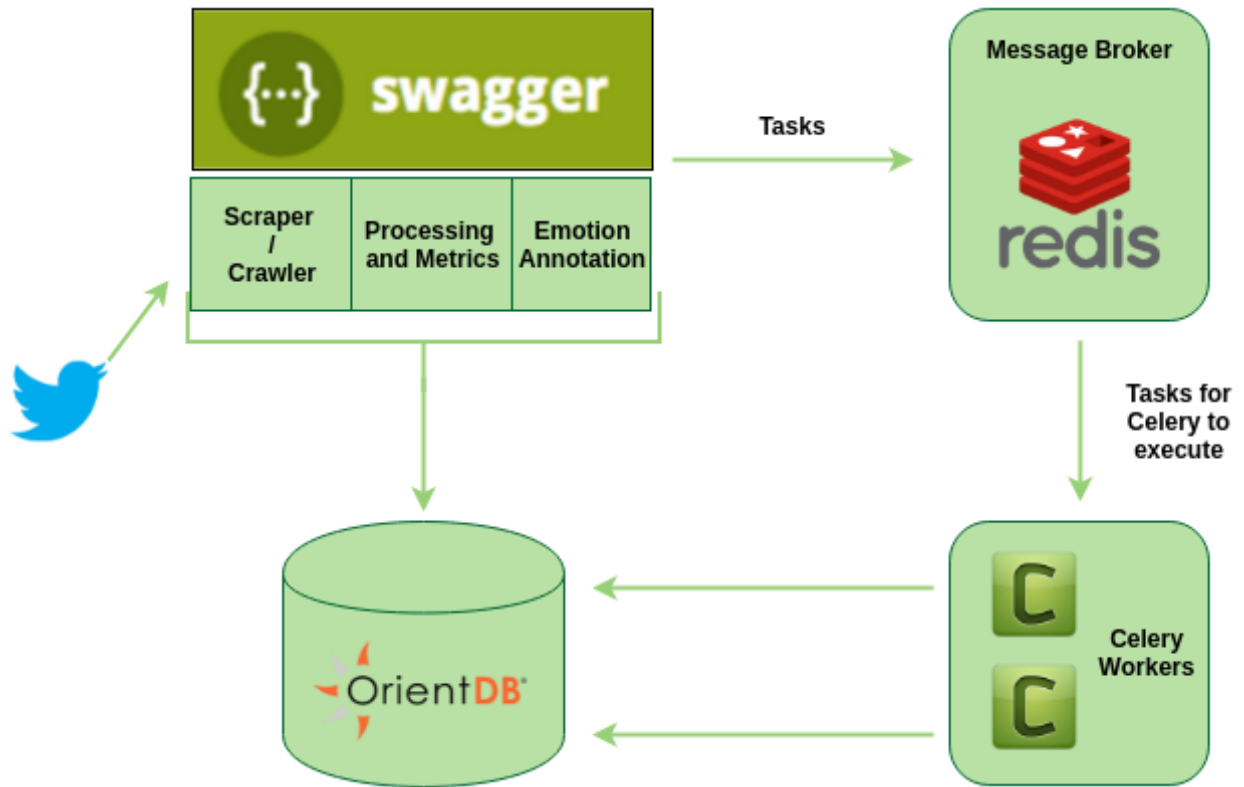
SCANNER: Social Context Analysis aNd Emotion Recognition is a platform to collect and analyse social context, i.e context of users and content in social media.

The platform is able to extract and process social media information data from Twitter, allowing us to analyse and process different metrics from tweets and users. To perform the test of the platform, it includes models of users and tweets. It creates the appropriate relationships between users and their related tweets, it calculates their influences and makes this information available through a REST API.

### 1.1 Architecture

The modular architecture of Scanner allows retrieving, storing and processing large amounts of information. The independent task system and API further contribute to decouple the modules in the platform.

- OrientDB stores all the amount of data that we needed and easily edges the information to create graphs.
- The crawler has been able to extract the necessary information from Twitter, being only limited by the Twitter API rate limit. Scanner uses [bitter](#) to implement this task.
- The task manager process all this information.



## 1.2 More

For more information visit <http://scanner.readthedocs.io/en/latest/>

### 1.2.1 Acknowledgement

This development has been partially funded by the European Union through the MixedEmotions Project (project number H2020 655632), as part of the *RIA ICT 15 Big data and Open Data Innovation and take-up* programme.



**MixedEmotions**





---

## Installation

---

Scanner requires docker and docker-compose to work. You can download Docker [here](#)

Docker-compose can be easily installed through pip.

```
$ pip install docker-compose
```

### 2.1 Building Scanner

First of all, you need to clone the Github repository:

```
$ git clone git@github.com:gsi-upm/scaner
$ cd scanner
```

Once cloned, we need to build the docker image:

```
$ docker-compose build
```

Then, it is necessary to populate **OrientDB** schema.

```
$ ./populate_schema.sh
```

### 2.2 Running Scanner

Now the image is ready to run:

```
$ docker-compose up
```



---

## API Schema

---

Base URL: /api/v1, Version: 1.0.0

Default request content-types: application/json

Default response content-types: application/json

### 3.1 Schemes

#### 3.1.1 Tag: users

API call	Description
GET /users/{userId}	Obtain information of a particular user
GET /users/{userId}/emotion	Obtain the emotion of a user
GET /users/{userId}/sentiment	Obtain the sentiment of a user
GET /users/{userId}/metrics	Obtain the metrics of a user
GET /users	Obtain list of available users
GET /users/{userId}/network	Obtain social network of a user

#### 3.1.2 Tag: tweets

API call	Description
GET /tweets/{tweetId}	Obtain information of a particular tweet
DELETE /tweets/{tweetId}	Delete a tweet from the database
GET /tweets/{tweetId}/history	Obtain the history of a particular tweet
GET /tweets/{tweetId}/sentiment	Obtain the sentiment of a tweet
GET /tweets/{tweetId}/emotion	Obtain the emotion of a tweet
GET /tweets/{tweetId}/metrics	Obtain the metrics of a tweet
GET /tweets	Obtain list of available tweets
POST /tweets	Add a tweet to the database

#### 3.1.3 Tag: topics

API call	Description
GET /topics	Obtain list of available topics
GET /topics/{topicId}	Obtain information of a particular topic
GET /topics/{topicId}/network	Obtain social network of a topic

### 3.1.4 Tag: tasks

API call	Description
GET /tasks	Obtain the list of tasks
GET /tasks/{taskId}	Obtain the status of a particular task

### 3.1.5 Tag: communities

API call	Description
GET /communities/{communityId}	Obtain information of a particular community
GET /communities/{communityId}/emotion	Obtain the emotion of a community
GET /communities/{communityId}/sentiment	Obtain the sentiment of a community
GET /communities	Obtain list of available communities
GET /communities/{communityId}/users	Obtain users that conforms a community

## 4.1 POST /tweets

### Example request:

```
POST /api/v1/tweets/ HTTP/1.1
Host: localhost
'Content-Type': 'application/json',
'Accept': 'application/json'
```

### Example response:

```
HTTP/1.1 200 OK
'Content-Type': 'application/json',
'Vary': 'Accept'

{
  "metadata": { "parameters": {},
                "url": "http://localhost:5000/api/v1/tweets"},
  "result": "Tweet added to DB."
}
```

## 4.2 GET /tweets

### Example request:

```
GET /api/v1/tweets?limit=1 HTTP/1.1
Host: localhost
'Accept': 'application/json'
```

### Example response:

```
HTTP/1.1 200 OK
'Content-Type': 'application/json'
'Vary': 'Accept'

{
  "metadata": {
    "count": 20,
    "parameters": {
```

```

    },
    "url": "http://localhost:5000/api/v1/tweets"
  },
  "statuses": [
    {
      "created_at": "Mon Dec 15 23:29:35 +0000 2014",
      "entities": {
        "hashtags": [

        ],
        "symbols": [

        ],
        "urls": [
          {
            "display_url": "bit.ly/1AeESXb",
            "expanded_url": "http://bit.ly/1AeESXb",
            "indices": [
              19,
              41
            ],
            "url": "http://t.co/1JWX4VDvmz"
          }
        ],
        "user_mentions": [

        ]
      },
      "id": 000000000000,
      "id_str": "000000000000",
      "lang": "ja",
      "metadata": {
        "iso_language_code": "ja",
        "result_type": "recent"
      },
      "text": " http://t.co/1JWX4VDvmz",
      "timestamp_ms": 1418686175.0,
      "topics": [
        "bigdata"
      ],
      "user": {
        "created_at": "Tue Sep 02 01:27:55+0000 2014",
        "followers_count": 7254,
        "following": null,
        "friends_count": 7277,
        "id": 0000000000,
        "id_str": "0000000000",
        "lang": "ja",
        "protected": 0,
        "screen_name": "*****",
        "statuses_count": 9328
      }
    }
  ]
}

```

**Query parameters:**

- query (f) fields: Comma-separated list of fields to retrieve e.g 'screen\_name' 'following'

- query (l) limit: Get only this many users per request by default limit is 20 tweets
- query (t) topic: Only retrieve users related to a certain topic e.g 'LaboralKutxa'
- query (s) sort\_by: Sort users using this criterion. Prepending a minus sign reverses the order. e.g. '-tweet\_count'.

### 4.3 GET /tweets/{tweetId}/metrics

#### Example request:

```
GET /api/v1/tweets/{tweetId}/metrics HTTP/1.1
Host: localhost
'Content-Type': 'application/json'
'Accept': 'application/json'``
```

#### Example respond:

```
HTTP/1.1 200 OK
'Content-Type': 'application/json'
'Vary': 'Accept'

{
  "metadata": {
    "parameters": {

    },
    "url": "http://localhost:5000/api/v1/tweets/00000000000000/metrics"
  },
  "result": {
    "complete": true,
    "date": "2016-12-07",
    "id": 00000000000000,
    "influence": 0.496603800169,
    "lastMetrics": true,
    "relevance": 0.10244804983,
    "timestamp": 1481107250.4926267,
    "topic": "bigdata"
  }
}
```

#### Path parameters:

- path (t) tweetId (required): Tweet id to filter by

### 4.4 GET /tweets/{tweetId}/history

#### Example request:

```
GET /api/v1/tweets/{tweetId}/history HTTP/1.1
Host: localhost
'Content-Type': 'application/json',
'Accept': 'application/json'``
```

#### Example respond:

```
HTTP/1.1 200 OK
'Content-Type': 'application/json'
'Vary': 'Accept'

{
  "metadata": {
    "parameters": {

    },
    "url": "http://localhost:5000/api/v1/tweets/0000000000000000/history"
  },
  "result": [
    {
      "complete": true,
      "date": "2016-12-07",
      "id": 0000000000000000,
      "influence": 0.496603800169,
      "lastMetrics": true,
      "relevance": 0.10244804983,
      "timestamp": 1481107250.4926267,
      "topic": "bigdata"
    },
    {
      "complete": true,
      "date": "2016-12-05",
      "id": 0000000000000000,
      "influence": 0.0,
      "lastMetrics": false,
      "relevance": 0.0,
      "timestamp": 1480940543.4212337,
      "topic": "bigdata"
    }
  ]
}
```

**Path parameters:**

- path (t) tweetId (required): Tweet id to filter by

## 4.5 GET /users

**Example request:**

```
GET /api/v1/users?limit=3 HTTP/1.1
Host: localhost
'Content-Type': 'application/json'
'Accept': 'application/json'``
```

**Example respond:**

```
HTTP/1.1 200 OK
'Content-Type': 'application/json'
'Vary': 'Accept'

{
  "metadata": {
    "count": 3,
    "parameters": {
```



```

    },
    "url": "http://localhost:5000/api/v1/users"
  },
  "users": [
    {
      "community": 1156,
      "created_at": "Mon Nov 10 16:10:39 +0000 2014",
      "followers_count": 7,
      "friends_count": 23,
      "id": 0,
      "id_str": "0",
      "lang": "ru",
      "protected": "0",
      "screen_name": "*****",
      "statuses_count": 38,
      "topics": [
        "bigdata"
      ]
    },
    {
      "community": 560,
      "created_at": "Mon Nov 10 19:57:30 +0000 2014",
      "followers_count": 3,
      "friends_count": 12,
      "id": 1,
      "id_str": "1",
      "lang": "ru",
      "protected": "0",
      "screen_name": "*****",
      "statuses_count": 56,
      "topics": [
        "bigdata"
      ]
    },
    {
      "community": 4,
      "created_at": "Sun Jan 17 16:12:59 +0000 2010",
      "followers_count": 936,
      "friends_count": 1154,
      "id": 2,
      "id_str": "2",
      "lang": "ja",
      "protected": "0",
      "screen_name": "****",
      "statuses_count": 20637,
      "topics": [
        "bigdata"
      ]
    }
  ]
}

```

**Query parameters:**

- query (f) fields: Comma-separated list of fields to retrieve e.g 'screen\_name' 'following'
- query (l) limit: Get only this many users per request by default limit is 20 tweets
- query (t) topic: Only retrieve users related to a certain topic e.g 'LaboralKutxa'

- query (s) sort\_by: Sort users using this criterion. Prepending a minus sign reverses the order. e.g. '-tweet\_count'.

## 4.6 GET /users/{userId}/metrics

### Example request:

```
GET /api/v1/users/{userId}/metrics HTTP/1.1
Host: localhost
'Content-Type': 'application/json'
'Accept': 'application/json'`
```

### Example response:

```
HTTP/1.1 200 OK
'Content-Type': 'application/json'
'Vary': 'Accept'

{
  "metadata": {
    "parameters": {},
    "url": "http://localhost:5000/api/v1/users/59390872/metrics"
  },
  "result": {
    "complete": true,
    "date": "2016-12-07",
    "followRelationScore": 0.140941982233,
    "followers": 43017,
    "following": 43445,
    "id": 59390872,
    "impact": 0.000002767175,
    "influence": 1,
    "influenceUnnormalized": 0.00499198376,
    "lastMetrics": true,
    "relevance": 2.105912,
    "statuses_count": 39233,
    "timestamp": 1481114116.3879638,
    "topic": "bigdata",
    "tweetRatio": 0.00015293248,
    "voice": 0.030976330457,
    "voice_r": 0.00000003215
  }
}
```

### Path parameters:

- path (t) userId (required): User id to filter by

## 4.7 GET /users/{userId}/network

### Example request:

```
GET /api/v1/users/{userId}/network
HTTP/1.1 Host: localhost
'Content-Type': 'application/json'
'Accept': 'application/json'
```

Example respond:

## 4.8 GET /communities

Example request:

```
GET /api/v1/communities/ HTTP/1.1
Host: localhost
'Content-Type': 'application/json'
'Accept': 'application/json'
```

Example respond:

```
HTTP/1.1 200 OK
'Content-Type': 'application/json'
'Vary': 'Accept'

{
  "communities": [
    {
      "emotion": "joy",
      "id": 0,
      "sentiment": "positive",
      "user_count": 12
    },
    {
      "emotion": "joy",
      "id": 1,
      "sentiment": "positive",
      "user_count": 32
    },
    {
      "emotion": "joy",
      "id": 2,
      "sentiment": "positive",
      "user_count": 8
    }
  ],
  "metadata": {
    "count": 3,
    "parameters": {

    },
    "url": "http://localhost:5000/api/v1/communities"
  }
}
```

Query parameters:

- query (f) fields: Comma-separated list of fields to retrieve e.g 'screen\_name' 'following'
- query (l) limit: Get only this many users per request by default limit is 20 tweets
- query (t) topic: Only retrieve users related to a certain topic e.g 'LaboralKutxa'
- query (s) sort\_by: Sort users using this criterion. Prepending a minus sign reverses the order. e.g. '-tweet\_count'.

## 4.9 Adding more tweets for the same topic

Once Scanner has calculated the metrics for a certain topic, the tool allows to retrieve instant information of relevance of a tweet in that topic.

### Example request:

```
POST /api/v1/tweets/ HTTP/1.1
Host: localhost
'Content-Type': 'application/json'
'Accept': 'application/json'`
```

### Example respond:

```
HTTP/1.1 200 OK
'Content-Type': 'application/json'
'Vary': 'Accept'

{
  "metadata": {
    "parameters": {

    },
    "url": "http://localhost:5000/api/v1/tweets"
  },
  "result": {
    "result": "Tweet added to DB",
    "topic": "bigdata",
    "tweet_relevance": "0.010244804983"
  }
}
```

Internally, metrics are classified in two different types: direct and indirect metrics.

## 5.1 Direct metrics

Direct metrics are directly obtainable from the extracted data, such as the number of followers a user has. The Social Context Analysis module obtains direct metrics as soon as new social media content is stored in the database, and these metrics are updated when new information arrive. For instance, the Social Context module is configured to refetch general information about users periodically, so these metrics would be updated as well.

## 5.2 Indirect metrics

Indirect metrics are obtained through data processing, for example User Influence. These metrics are calculated periodically, as they have a high processing cost and require accessing all the information in the database.

### 5.2.1 User relevance

We define the user relevance score based in the tweet rate, the user influence and follow relation score of each user.

#### **Tweet Rate (TR) score**

This metric measures the proportion of tweets related to the topic that a user posts or retweets. Some of the topic-related users usually retweet tweets relevant to the topic originally posted by others, which means they play a role of “filter” searching for valuable relevant tweets and sharing them with their followers.

#### **User Influence (UI) score and Tweet Influence (TI) score**

How much each tweet is paid attention to by others is measured according to the retweet and reply activities and the follow relation. Based on this idea, we define not only the UI score of each user but also “tweet influence (TI) score” of each tweet. The UI score of each user is calculated using the TI score of the user’s tweets and retweets, and the TI score of each tweet is calculated using the UI score of users who pay attention to the tweet.

### Follow Relation (FR) score

A reference graph consisting of user nodes and directed edges each of which connects two of the user nodes, called “follow relation graph”, is created from the follow relation.

### 5.2.2 Tweet relevance

They describe a method for finding relevant tweets to the target topic.

#### Voice and Impact score calculation

In order to judge the relevance of each tweet to the target topic, we have the following assumptions about tweets relevant to the topic (relevant tweets).

1. The relevant tweets are posted or retweeted by the topic-related users.
2. The relevant tweets are paid attention to (retweeted or replied to) by many topic-related users.

The Impact score is used for the estimation based on the first idea, and the Voice score is used for the estimation based on the second idea.

#### Tweet Ranking

The tweet relevance score of each tweet is calculated from the Voice score and the Impact score, then select the top-M tweets ranked by the tweet relevance score as the tweets relevant to the target topic.

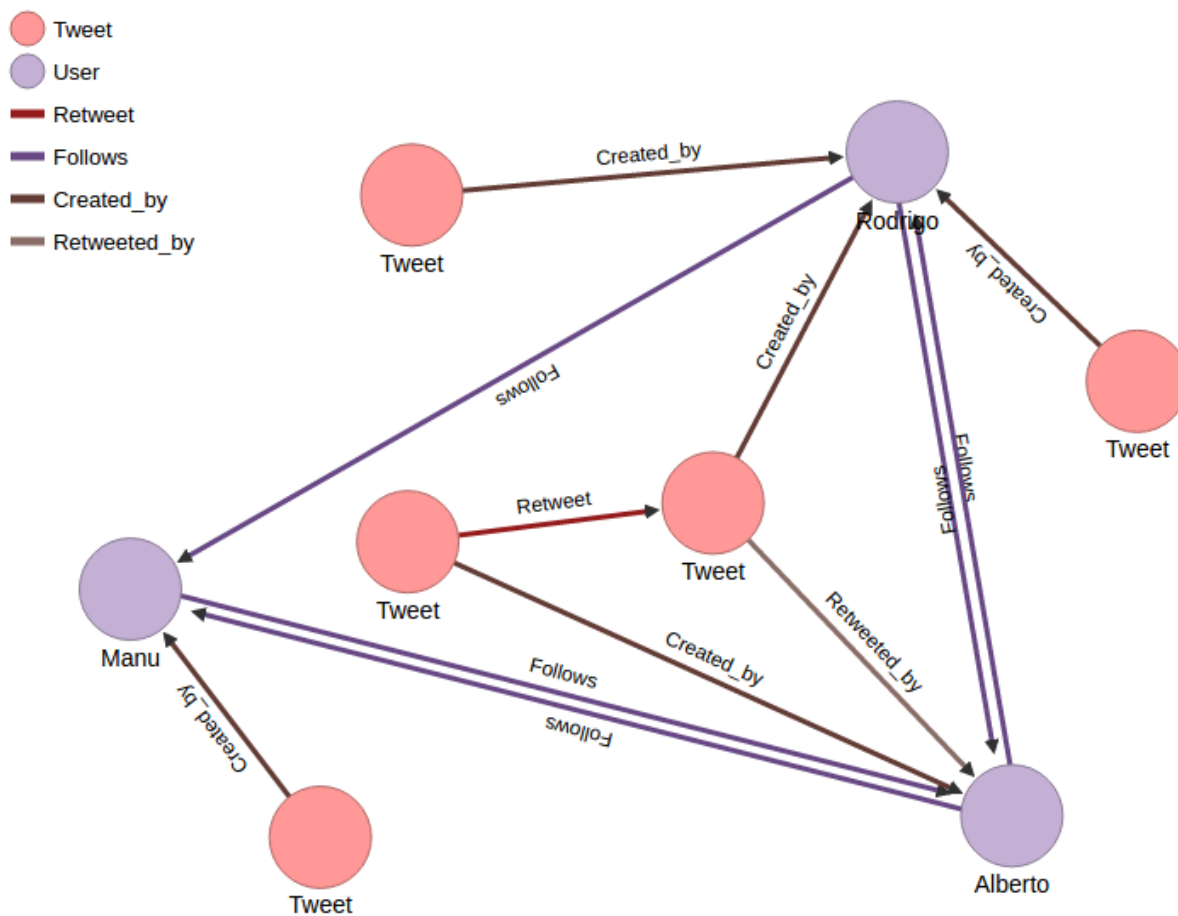
---

**Metrics Usage**


---

### 6.1 Searching for Relevant Tweets and Users

Suppose we have 3 twitter users: **Manu**, **Alberto** and **Rodrigo**. This three users are expressing their opinions about a concrete topic in Twitter, i.e about nuclear energy. They have published tweets concerning this topics and they follow each other as it is shown in the following graph.



(This data can easily be loaded in Scanner tool using “populate\_db.py” script).

Now let's find more information about this data. The metrics used are based on Noro et al. article [Noro et al, 2016].

- *Tweet Ratio*
- *Influence*
- *Follow Relation*
- *Relevance*
- *Voice*
- *Impact*

## 6.2 Tweet Ratio

To calculate Tweet ratio of each user Scanner takes the lapse of time of the tweets data introduced and relations the number of tweets published in that lapse with the number of tweets related to the topic that are in the Scanner database.

Let's calculate Rodrigo's tweet ratio. To do so we choose two intervals of time to measure (**t0** y **t1**). In **t0** Rodrigo had 300 tweets in his timeline. In **t1** Rodrigo had 333 tweets. Of this 33 new tweets, 3 of them are related to nuclear energy topic and that is the data in Scanner. So the tweet Ratio was:

```
import numpy as np
```

```
statuses_count_t0 = 300
statuses_count_t1 = 333
tweets_in_scanner = 3

#TWEET RATIO
TR = tweets_in_scanner/(statuses_count_t1-statuses_count_t0)
TR
```

```
0.09090909090909091
```

## 6.3 Influence

To find out the influence of users and tweet in this data we need to create three matrix: on one hand with the number of tweets created by the users, on the other, with the retweets and replies and the last matrix with the replies and the tweets created, retweeted or replied by the followers of the users. Then Scanner calculate the influence. This influence is normalized assigning 1 to the maximum influence value.

For our example above, Rodrigo would be the most influence user, so Rodrigo's influence is 1. The influence vector calculated. We could observe that the second most influent was Alberto, that is because Alberto has retweeted a tweet from Rodrigo.

```
# [UI_Rodrigo UI_Manu UI_Alberto]
UI_vector = np.array([1, 0.6, 0.8])
UI_vector
```

```
array([ 1. ,  0.6,  0.8])
```

To find out the influence of each tweet the process is the same, the matrix are the same and the results are correlated to the user influence.



## 6.4 Follow Relation

Follow Relation shows information about the amount of follows an user has. We can see in the graph that Alberto has the most amount of Follows arrows, so the follow relation is 1 for Alberto. The follow Relation of Rodrigo is the lowest. That's because Rodrigo has only one user in the data that follow him. Let's probe this suposition calculating Scanner Follow Relation vector:

```
# [FR_Rodrigo FR_Manu FR_Alberto]
FR_vector = np.array([0.54054,0.7702, 1])
FR_vector
```

```
array([ 0.54054,  0.7702 ,  1.        ])
```

## 6.5 Relevance

### 6.5.1 User Relevance

User relevance is calculated with a ponderated sum of the previous ones. Each metric has a weight depending its importance. For example, user relevance of Rodrigo will be:

```
w_tr = 0.4
w_i = 0.4
w_fr = 0.2
user_relevance = TR**w_tr + UI_vector[0]**w_i + FR_vector[0]**w_fr
user_relevance
```

```
1.9674710190829381
```

### 6.5.2 Voice

The voice of an user is calculated according to the quantity of tweets and retweets the user has. There are two types of this score, voice of tweets, and voice of retweets. The calculation of both are equivalent. Here we are going to calculate the voice of Manuel in nuclear energy topic:

```
Tweet = 1
sigma = 1
TI = 0.37500000161 # Calculated together with Manuel user influence
Sumatorio_tweets = 1
Voice_tweet = (1/(1+1))*1*TI
Voice_tweet
```

```
0.187500000805
```

### 6.5.3 Impact

The impact score is calculated related to the user influence and the interactions of the user in the tweets collected (Related interactions: retweets and replies). For example the impact score of Alberto would be:

```
UI_Alberto = 0.8 #Calculated above
d = 0.15 # Dumping factor
Relate_Alberto = 1 # Alberto has retweeted one tweet
Sigma = 1 # Smoothing parameter
```

```
Number_tweets = 4
IMPACT = (UI_Alberto/(Relate_Alberto+Sigma))*(1-d) + (UI_Alberto/4)*d
IMPACT
```

```
0.37
```

### 6.5.4 Tweet relevance

Tweet relevance is the main phase of Scanner. The purpose of this score is to find the relevance of a new introduced tweet in Scanner instantaneously based on the scores calculated above. This score allow us to rank the new tweets in real time. Let's make an easy example.

Imagine that we introduce another tweet retweeted by Alberto. We don't need to recalculate the tweet influence score of this new tweet. We calculate the tweet relevance as follows:

```
Voice_retweet_Alberto = 0.5
VR_t = 0.5
IR_t = 0.37
alpha = 0.4
Tweet_relevance = alpha*VR_t + (1-alpha)*IR_t
Tweet_relevance
```

```
0.42200000000000004
```

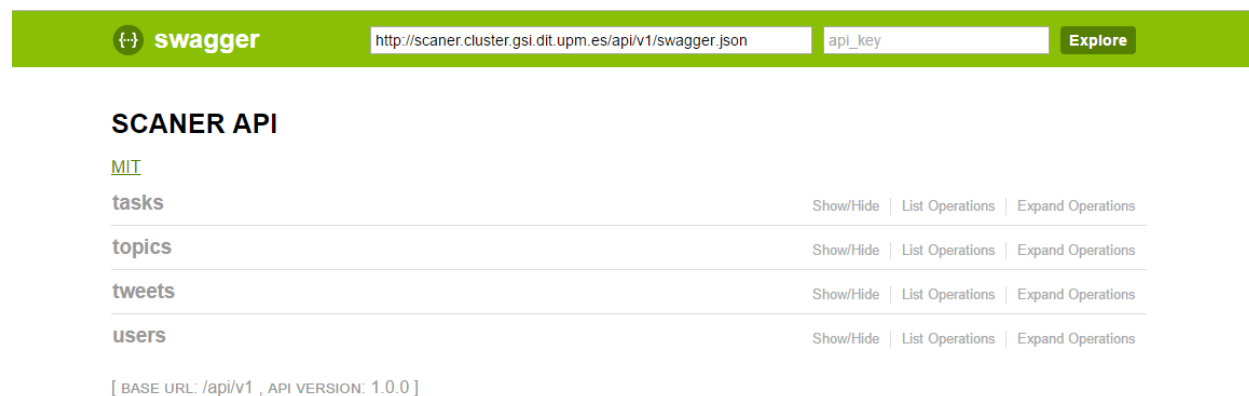
Noro, T., Ru, F., Xiao, F., & Tokuda, T. 2016. Searching for relevant based on topic- related user activities. *Journal of Web Engineering*, 15 (3&4), 249-276,.

---

## Demo

---

There is a demo available on <http://scanner-demo.cluster.gsi.dit.upm.es/>, this demo has a sintetic dataset. You can explore all the possibilities of the Scanner API.



The image shows a screenshot of the Swagger UI interface. At the top, there is a green header with the Swagger logo and the text "swagger". Below the header, there are two input fields: the first contains the URL "http://scanner.cluster.gsi.dit.upm.es/api/v1/swagger.json" and the second is labeled "api\_key". To the right of these fields is a button labeled "Explore". Below the header, the main content area displays the API title "SCANNER API" and a link to the MIT license. A table lists the API endpoints: "tasks", "topics", "tweets", and "users". Each endpoint has three links: "Show/Hide", "List Operations", and "Expand Operations". At the bottom of the interface, there is a footer that reads "[ BASE URL: /api/v1 , API VERSION: 1.0.0 ]".

**SCANNER API**

[MIT](#)

<b>tasks</b>	Show/Hide	List Operations	Expand Operations
<b>topics</b>	Show/Hide	List Operations	Expand Operations
<b>tweets</b>	Show/Hide	List Operations	Expand Operations
<b>users</b>	Show/Hide	List Operations	Expand Operations

[ BASE URL: /api/v1 , API VERSION: 1.0.0 ]