Gestión de Información en la Web
Máster en Ingeniería Informática
Parte I: Redes Sociales On-line

Práctica 2: Redes de Twitter

Oscar Cordón García
Dpto. Ciencias de la Computación e Inteligencia Artificial
ocordon@decsai.ugr.es

¿QUÉ ES TWITTER?

Es una plataforma de microblogging basada en el envío de mensajes cortos entre usuarios

Cada usuario decide a quien seguir, permitiendo establecer listas por temáticas, construyendo su propia red social en función a sus intereses o gustos

Permite enviar mensajes de hasta 140 caracteres y así mantener conversaciones con otros usuarios o simplemente publicar contenido

FUNCIONAMIENTO BÁSICO (1)

- **Página de inicio:**

FUNCIONAMIENTO BÁSICO (2)

- **Perfil:** cada usuario tiene un perfil personal donde aparece su información básica: nombre de usuario (precedido por @), imagen (avatar), lugar, página web, etc.

FUNCIONAMIENTO BÁSICO (3)

- **Tweets**: Mensajes escritos por el usuario
- **Siguiendo (*following*)**: Usuarios a los que seguimos
- **Seguidores (*followers*)**: Usuarios que nos siguen
- **Listas**: Grupos de los que formamos parte

FUNCIONAMIENTO BÁSICO (4)

- **Cronología (*Timeline*):** Es la parte principal de la pantalla. En ella veremos los mensajes (*tweets*) que publican aquellos usuarios a los que seguimos

- Del mismo modo, nuestros **tweets** aparecerán en el **timeline** de los usuarios que nos siguen

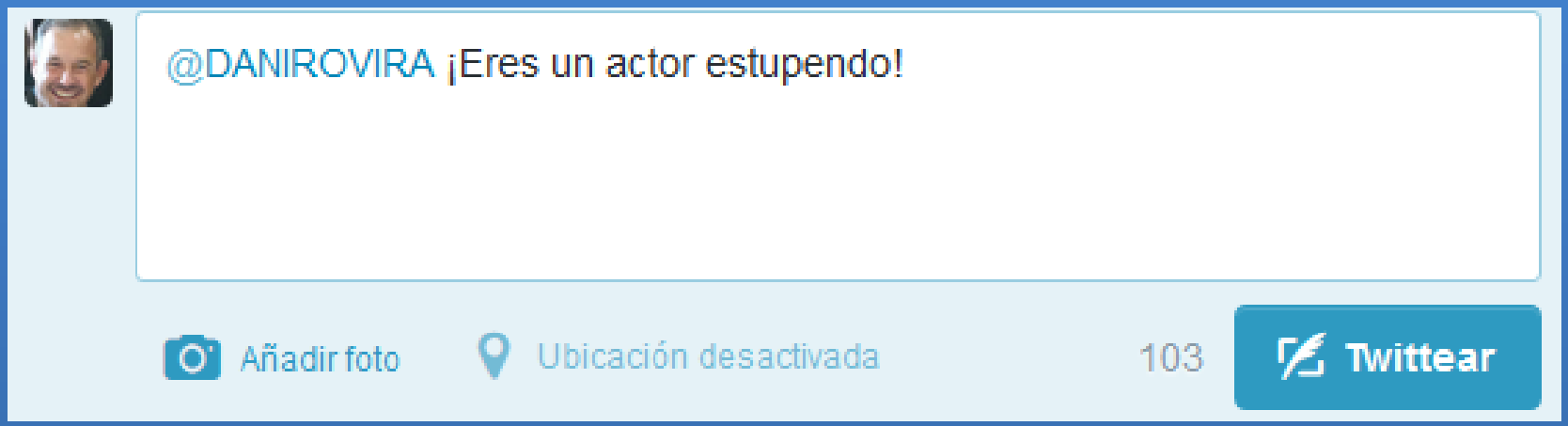
- Para añadir un **tweet**, escribiremos nuestro mensaje en la caja de texto ***¿Qué está pasando?***

FUNCIONAMIENTO BÁSICO (5)

- **Menciones (*mentions*)**: Cuando queremos interactuar con algún usuario, nos referiremos a él de la siguiente forma:

@nombreDeUsuario mensaje

Ejemplo:

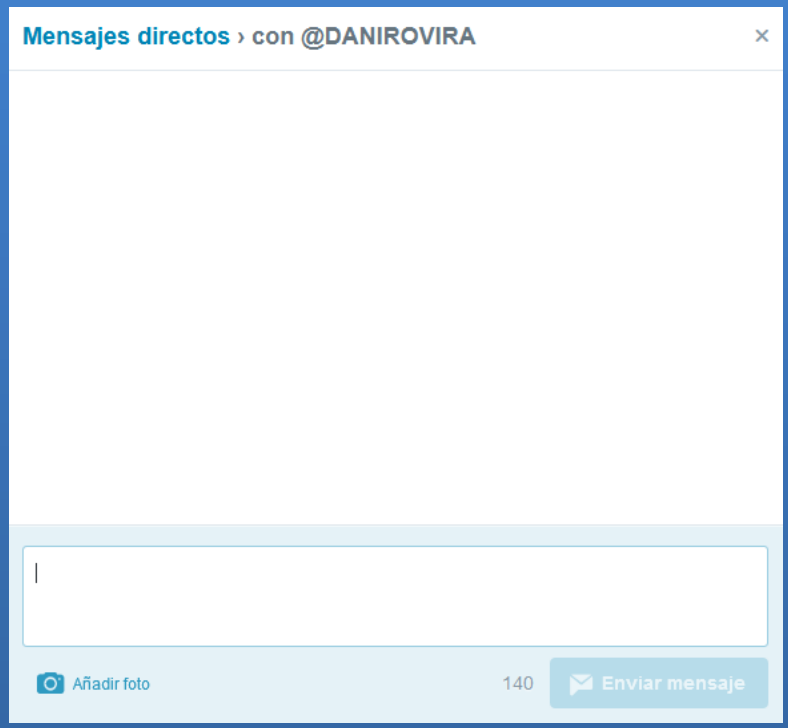


De esta forma el destinatario sabrá que nos hemos referido a él, llegándole una notificación a su sección ***Menciones***

Éste es el sistema usado para conversar públicamente entre usuarios

FUNCIONAMIENTO BÁSICO (6)

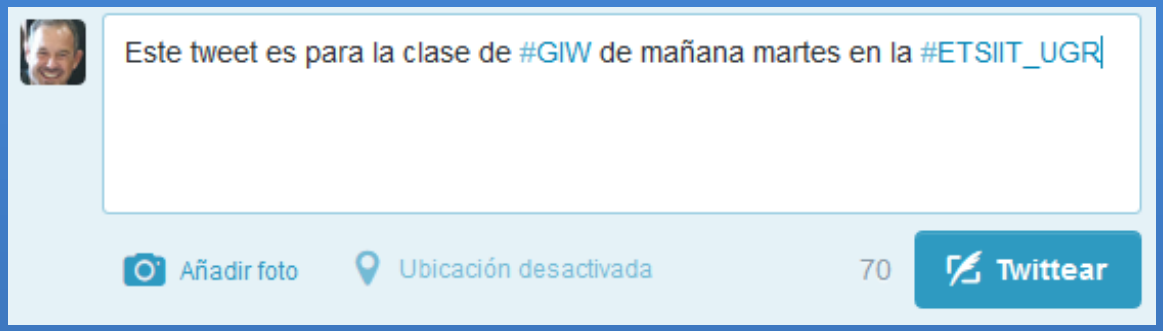
- **Mensajes privados**: Twitter también ofrece la posibilidad de enviar mensajes privados a otro usuario. Esta operación se realizará desde la sección ***Mensajes***



FUNCIONAMIENTO BÁSICO (7)

- **HashTags**: Para identificar temáticas, eventos, lugares, etc. se utilizan etiquetas. Esto facilita las búsquedas y estructura los contenidos. Para introducir una nueva etiqueta, usamos el símbolo #

Ejemplo:



Si alguien busca ese término en el cuadro de búsqueda, los tweets que contengan esa etiqueta aparecerán en los resultados

- **Tendencias (*trending topics*)**: Índice formado por las etiquetas, términos o frases de las que más se está hablando en este momento en Twitter. Se puede elegir el país y la ciudad

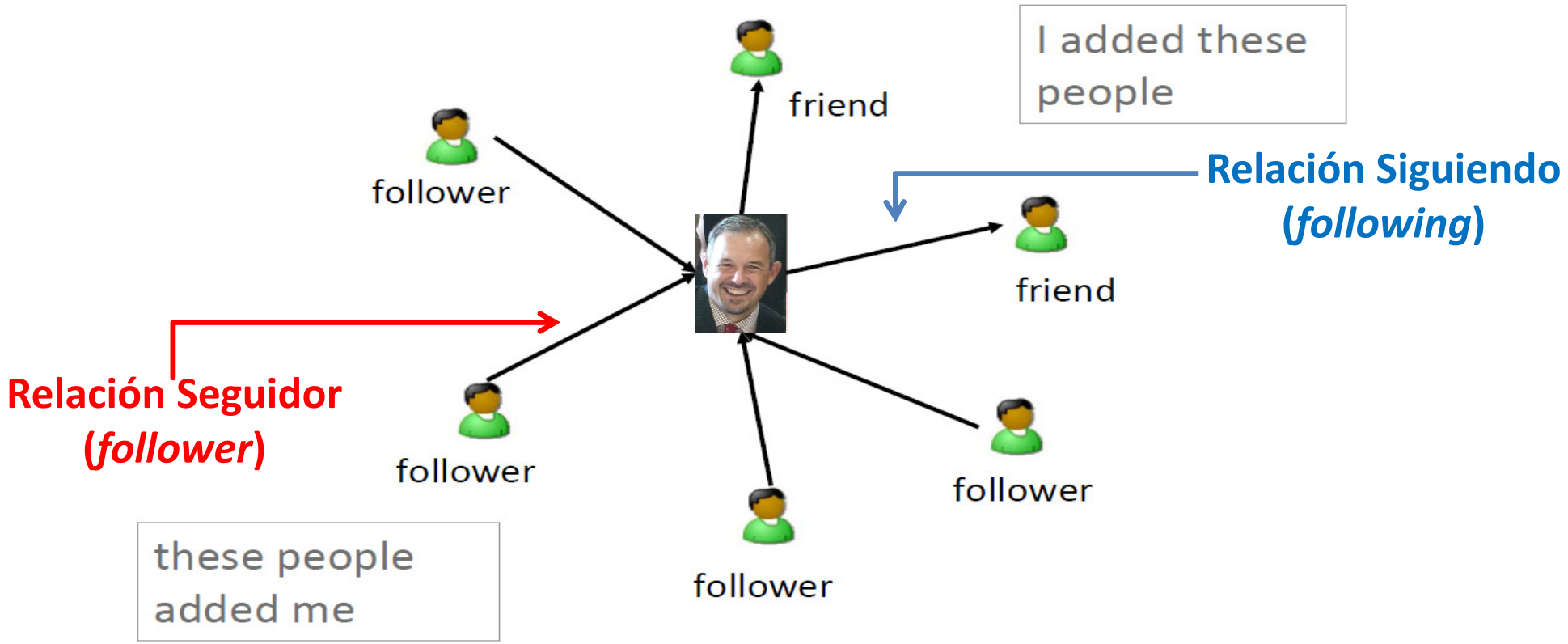
FUNCIONAMIENTO BÁSICO (8)

- **¿Cómo podemos interactuar con los mensajes recibidos?**: Con cualquier tweet podemos realizar tres acciones: marcarlo como **favorito**, **responderlo** o **retwittearlo**:

Las dos últimas provocan un tráfico de tweets a través de la red de contactos

REDES DE TWITTER (1)

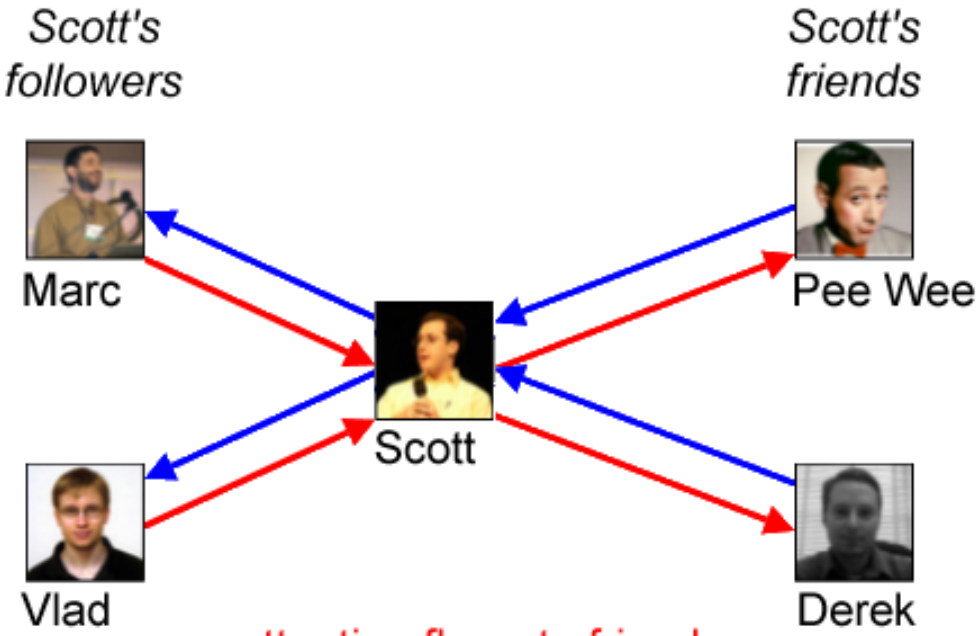
La **red social de amistad (*friend-follower*) de *Twitter*** es un **grafo dirigido**. Los nodos son usuarios y **los enlaces (dirigidos) indican "quién sigue a quién"**. Tu ves lo que publican tus amigos, tus seguidores ven lo que tu publicas



REDES DE TWITTER (2)

A friend-follower network consists of users as nodes and the edges describe *who follows whom*. Each user on Twitter is able to create two types of explicit relationships with another Twitter user. Suppose we have two users: Alice and Bob. If Alice starts following Bob, then we say that Alice is a "follower" of Bob. Here, the decision to initiate the connection is made by Alice. Bob is known as the "friend" of Alice.

Los enlaces dirigidos contenidos en la red de amistad son los **de color rojo**



Ejemplo de red de amistad de Twitter

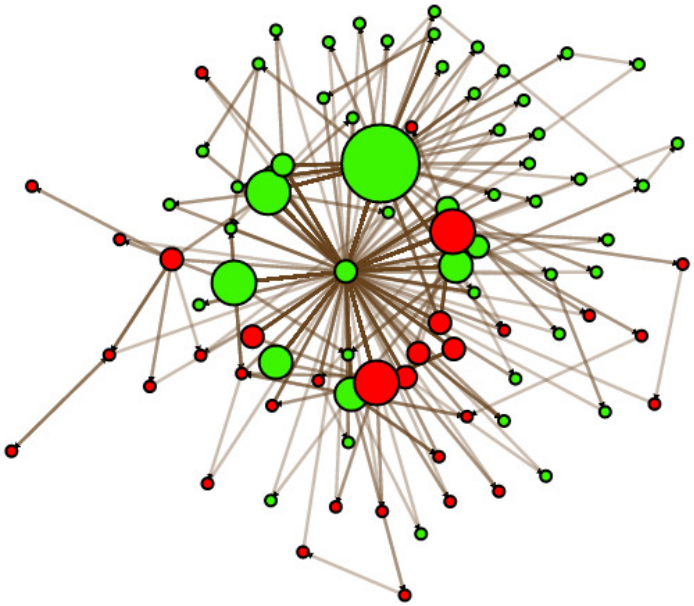
REDES DE TWITTER (3)

Existen otros tipos de redes generables a partir de datos de Twitter:

- **Redes de flujo de información** (*information flow*): Los nodos son usuarios y los enlaces (dirigidos) indican el camino seguido por el tweet ("quién ha retweeteado a quién"). La visualización de los *retweets* refleja el flujo de información en Twitter

- **Redes de co-ocurrencia de *hashtags***: Los nodos son *hashtags* y los enlaces indican que dichos *hashtags* aparecen conjuntamente en el mismo *tweet*. Los enlaces están ponderados, indicando el número de co-ocurrencias

Red de *retweets* de los tópicos "#zuccotti" y "#nypd"



OBTENCIÓN DE DATOS DE TWITTER (1)

Twitter proporciona una API que permite obtener toda la información disponible:

- Información sobre un usuario (datos del perfil)
- Red de amistad (contactos) de un usuario
- Tweets publicados por un usuario
- Resultados de búsqueda en Twitter

El acceso a la API de Twitter sólo puede hacerse vía solicitudes autentificadas. Twitter usa ***Open Authentication***, un proceso en el cual cada solicitud tiene que estar firmada por unas credenciales de usuario de Twitter válidas

El acceso a la API de Twitter está limitado también por un **número específico de solicitudes** en una ventana de tiempo denominada ***rate limit*** (aplicada a nivel de usuario individual y de aplicación). Esta ventana temporal se usa para renovar la cuota de las llamadas a la API periódicamente (cada 15 minutos). P.ej. **sólo se pueden bajar los contactos de 15 usuarios cada 15 minutos**

S. Kumar, F. Morstatter, H. Liu. Twitter Data Analytics. Springer, 2013

OBTENCIÓN DE DATOS DE TWITTER (2)

En principio, el API de Twitter solo proporciona acceso a un **1% de los datos de Twitter**. Hay estudios que analizan cómo de representativa es la muestra:

F. Morstatter, J. Pfeffer, H. Liu, K. Carley. Is the Sample Good Enough? Comparing Data from Twitter's Streaming API with Twitter's Firehose. In International AAAI Conference on Weblogs and Social Media, 2013

Sin embargo, a finales de 2014, Twitter anunció que pasaba a abrir la base de datos completa de tweets desde su creación en 2006:

https://blog.twitter.com/2014/building-a-complete-tweet-index

Los datos se pueden acceder programando aplicaciones y registrándolas en la API o bien usando software existente como **NodeXL**

Engineering Blog

All Twitter Blogs

Building a complete Tweet index

Tuesday, November 18, 2014 | By Yi Zhuang (@yz) 11/18/2014 - 17:35

Tweet

Today, we are pleased to announce that Twitter now indexes every public Tweet since 2006.

Since that first simple Tweet over eight years ago, hundreds of billions of Tweets have captured everyday human experiences and major historical events. Our search engine excelled at surfacing breaking news and events in real time, and our search index infrastructure reflected this strong emphasis on recency. But our long-standing goal has been to let people search through every Tweet ever published.

This new infrastructure enables many use cases, providing comprehensive results for entire TV and sports seasons, conferences (#TEDGlobal), industry discussions (#MobilePayments), places, businesses and long-lived hashtag conversations across topics, such as #JapanEarthquake, #Election2012, #ScotlandDecides, #HongKong, #Ferguson and many more. This change will be rolling out to users over the next few days.

In this post, we describe how we built a search service that efficiently indexes roughly half a trillion documents and serves queries with an average latency of under 100ms.

The most important factors in our design were:

OBTENCIÓN DE DATOS DE TWITTER (3) | Modalidades de Acceso

APIs to access Twitter data can be classified into two types based on their design and access method:

- REST APIs are based on the REST architecture[2] now popularly used for designing web APIs. These APIs use the pull strategy for data retrieval. To collect information a user must explicitly request it.
- Streaming APIs provides a continuous stream of public information from Twitter. These APIs use the push strategy for data retrieval. Once a request for information is made, the Streaming APIs provide a continuous stream of updates with no further input from the user.

[2] http://en.wikipedia.org/wiki/Representational_state_transfer

Las solicitudes al API de Twitter contienen parámetros que pueden incluir hashtags, palabras clave, regiones geográficas e IDs de usuarios

El resultado se almacena en formato ***JavaScript Object Notation*** (JSON)

http://en.wikipedia.org/wiki/JSON

OBTENCIÓN DE DATOS DE TWITTER (4) | Proceso de Autentificación

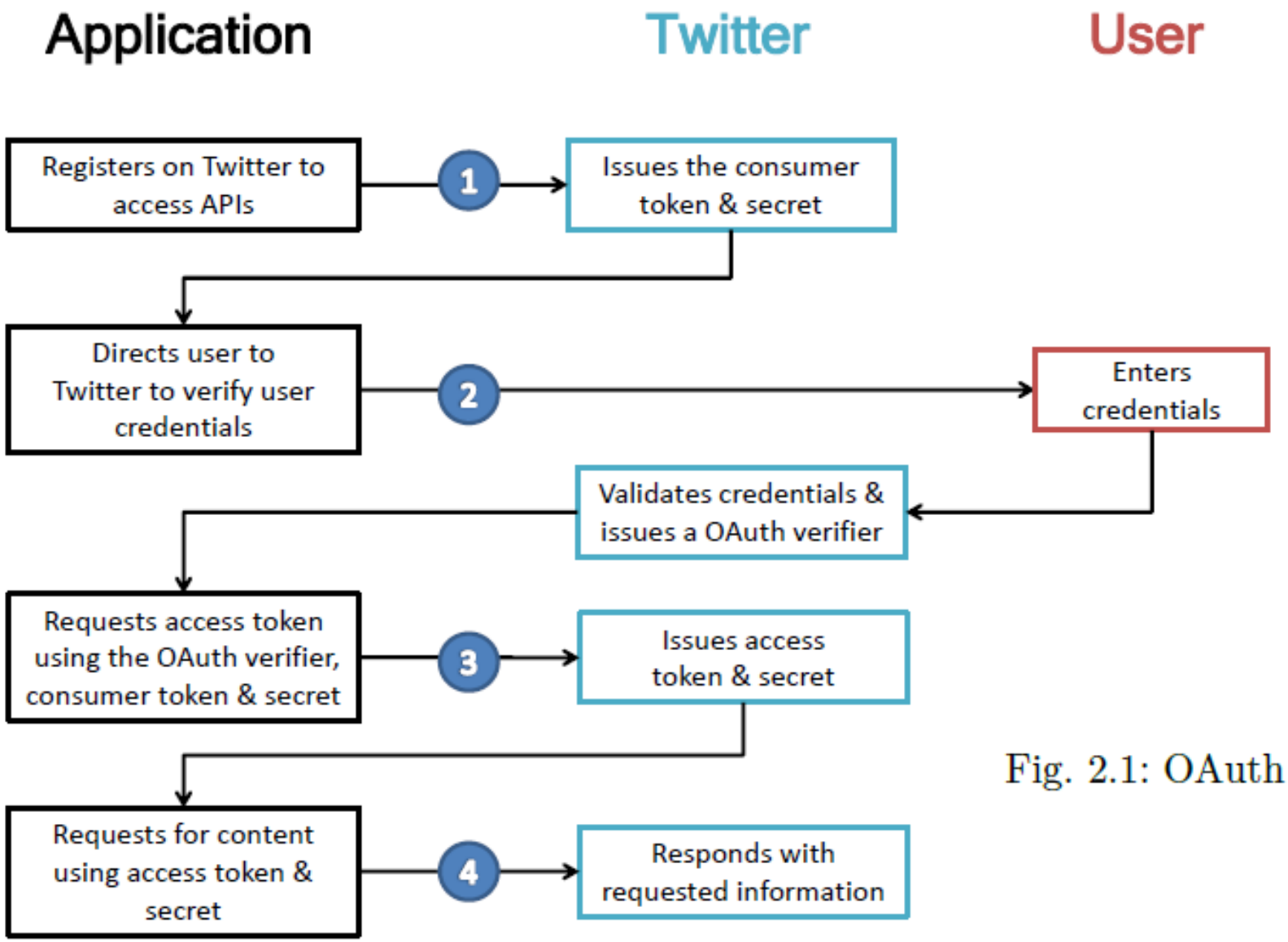
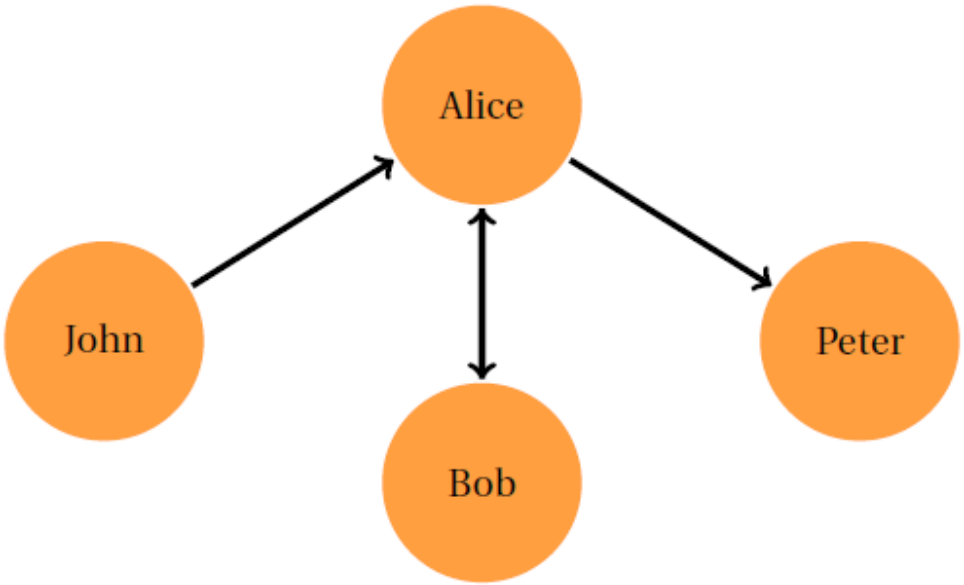


Fig. 2.1: OAuth workflow

OBTENCIÓN DE DATOS DE TWITTER (5) | Red de Amistad (1)



A user's network consists of his connections on Twitter. Twitter is a directed network and there are two types of connections between users. In Figure 2.3, we can observe an example of the nature of these edges. John follows Alice, therefore John is Alice's follower. Alice follows Peter, hence Peter is a friend of Alice.

*Es una **red egocéntrica**, se mira desde la perspectiva del usuario concreto*

Follower = seguidor (usuarios que nos siguen)

Friend = *following* = siguiendo (usuarios a los que seguimos)

OBTENCIÓN DE DATOS DE TWITTER (6)

Red de Amistad (2)

Listing 2.4: Using the Twitter API to fetch the followers of a user

```
public JSONArray GetFollowers(String username) {
        . . .
        // Step 1: Create the API request using the supplied
            username
        URL url = new URL("https://api.twitter.com/1.1/
            followers/list.json?screen_name="+username+"&
            cursor=" + cursor);
        HttpURLConnection huc = (HttpURLConnection) url.
            openConnection();
        huc.setReadTimeout(5000);
        // Step 2: Sign the request using the OAuth Secret
        Consumer.sign(huc);
        huc.connect();
        . . .
        /** Step 3: If the requests have been exhausted,
          * then wait until the quota is renewed
          */
        if(huc.getResponseCode()==429) {
                try {
                        Thread.sleep(this.GetWaitTime("/
                            followers/list"));
                } catch (InterruptedException ex) {
                        Logger.getLogger(RESTApiExample.class
                            .getName()).log(Level.SEVERE,
                            null, ex);
                }
        }
        // Step 4: Retrieve the followers list from Twitter
        bRead = new BufferedReader(new InputStreamReader((
            InputStream) huc.getContent()));
        StringBuilder content = new StringBuilder();
        String temp = "";
        while((temp = bRead.readLine())!=null) {
                content.append(temp);
        }
        try {
                JSONObject jobj = new JSONObject(content.
                    toString());
                // Step 5: Retrieve the token for the next
                    request
                cursor = jobj.getLong("next_cursor");
                JSONArray idlist = jobj.getJSONArray("users")
                    ;
                for(int i=0;i<idlist.length();i++) {
                        followers.put(idlist.getJSONObject(i)
                            );
                }
        . . .
        return followers;
}
```

Source: Chapter2/restapi/RESTApiExample.java

Listing 2.5: Using the Twitter API to fetch the friends of a user

```
public JSONArray GetFriends(String username) {
        . . .
        JSONArray friends = new JSONArray();
        // Step 1: Create the API request using the supplied
            username
        URL url = new URL("https://api.twitter.com/1.1/
            friends/list.json?screen_name="+username+"&cursor
            ="+cursor);
        HttpURLConnection huc = (HttpURLConnection) url.
            openConnection();
        huc.setReadTimeout(5000);
        // Step 2: Sign the request using the OAuth Secret
        Consumer.sign(huc);
        huc.connect();
        . . .
        /** Step 3: If the requests have been exhausted,
          * then wait until the quota is renewed
          */
        if(huc.getResponseCode()==429) {
                try {
                        Thread.sleep(this.GetWaitTime("/
                            friends/list"));
                } catch (InterruptedException ex) {
                        Logger.getLogger(RESTApiExample.class
                            .getName()).log(Level.SEVERE,
                            null, ex);
                }
        }
        // Step 4: Retrieve the friends list from Twitter
        bRead = new BufferedReader(new InputStreamReader((
            InputStream) huc.getContent()));
        . . .
        JSONObject jobj = new JSONObject(content.toString());
        // Step 5: Retrieve the token for the next request
        cursor = jobj.getLong("next_cursor");
        JSONArray userlist = jobj.getJSONArray("users");
        for(int i=0;i<userlist.length();i++) {
                friends.put(userlist.get(i));
        }
        . . .
        return friends;
}
```

Source: Chapter2/restapi/RESTApiExample.java

OBTENCIÓN DE DATOS DE TWITTER (7) | Datos de Usuario

Listing 2.3: A sample Twitter User object

```
{
    "location": "Tempe,AZ",
    "default_profile": true,
    "statuses_count": 1,
    "description": "Twitter Data Analytics is a book for
        practitioners and researchers interested in
        investigating Twitter data.",
    "verified": false,
    "name": "DataAnalytics",
    "created_at": "Tue Mar 12 18:43:47 +0000 2013",
    "followers_count": 1,
    "geo_enabled": false,
    "url": "http://t.co/HnlG9amZzj",
    "time_zone": "Arizona",
    "friends_count": 6,
    "screen_name": "twtanalyticsbk",
   //Other user fields
   . . .
}
```

Note: User information is generally included when Tweets are fetched from Twitter. Although the Streaming API does not have a specific endpoint to retrieve user profile information, it can be obtained from the Tweets fetched using the API.

Las solicitudes de contactos recuperan un vector de objetos de este tipo

OBTENCIÓN DE DATOS DE TWITTER (8) | Tweets de un Usuario

Listing 2.6: An example of Twitter Tweet object

```
{
    "text": "This is the first tweet.",
    "lang": "en",
    "id": 352914247774248960,
    "source": "web",
    "retweet_count": 0,
    "created_at": "Thu Jul 04 22:18:08 +0000 2013",
    //Other Tweet fields
    . . .
    "place": {
        "place_type": "city",
        "name": "Tempe",
        "country_code": "US",
        "url": "https://api.twitter.com/1.1/geo/id/7
            cb7440bcf83d464.json",
        "country": "United States",
        "full_name": "Tempe, AZ",
        //Other place fields
        . . .
    },
    "user": {
        //User Information in the form of Twitter user object
        . . .
    }
}
```

Listing 2.7: Using the Twitter API to fetch the Tweets of a user

```
public JSONArray GetStatuses(String username) {
        . . .
        // Step 1: Create the API request using the supplied
            username
        // Use (max_id-1) to avoid getting redundant Tweets.
        url = new URL("https://api.twitter.com/1.1/statuses/
            user_timeline.json?screen_name=" + username+"&
            include_rts="+include_rts+"&count="+tweetcount+"&
            max_id="+(maxid-1));
        HttpURLConnection huc = (HttpURLConnection) url.
            openConnection();
        huc.setReadTimeout(5000);
        // Step 2: Sign the request using the OAuth Secret
        Consumer.sign(huc);
        /** Step 3: If the requests have been exhausted,
           * then wait until the quota is renewed  */
        . . .
        //Step 4: Retrieve the Tweets from Twitter
        bRead = new BufferedReader(new InputStreamReader((
            InputStream) huc.getInputStream()));
        . . .
        for(int i=0;i<statusarr.length();i++) {
                JSONObject jobj = statusarr.getJSONObject(i);
                statuses.put(jobj);
                // Step 5: Get the id of the oldest Tweet ID
                    as max_id to retrieve the next batch of
                    Tweets
                if(!jobj.isNull("id")) {
                        maxid = jobj.getLong("id");
        . . .
    return statuses;
}
```

Source: Chapter2/restapi/RESTApiExample.java

El API-REST permite recuperar los 3200 tweets más recientes de un usuario, incluyendo retweets

OBTENCIÓN DE DATOS DE TWITTER (9) | Búsquedas

Searching on Twitter is facilitated through the use of parameters. Acceptable parameter values for search include keywords, hashtags, phrases, geographic regions, and usernames or userids. Twitter search is quite powerful and is accessible by the API

Listing 2.10: Searching for Tweets using the REST API

```
public JSONArray GetSearchResults(String query) {
   try {
        // Step 1:
        String URL_PARAM_SEPERATOR = "&";
        StringBuilder url = new StringBuilder();
        url.append("https://api.twitter.com/1.1/search/tweets
            .json?q=");
        //query needs to be encoded
        url.append(URLEncoder.encode(query, "UTF-8"));
        url.append(URL_PARAM_SEPERATOR);
        url.append("count=100");
        URL navurl = new URL(url.toString());
        HttpURLConnection huc = (HttpURLConnection) navurl.
            openConnection();
        huc.setReadTimeout(5000);
        Consumer.sign(huc);
        huc.connect();
        . . .
        // Step 2: Read the retrieved search results
        BufferedReader bRead = new BufferedReader(new
            InputStreamReader((InputStream) huc.
            getInputStream()));
        String temp;
        StringBuilder page = new StringBuilder();
        while( (temp = bRead.readLine())!=null) {
                page.append(temp);
        }
        JSONTokener jsonTokener = new JSONTokener(page.
            toString());
        try{
           JSONObject json = new JSONObject(jsonTokener);
           //Step 4: Extract the Tweet objects as an array
           JSONArray results = json.getJSONArray("statuses");
           return results;
           . . .
}
```

Source: Chapter2/restapi/RESTApiExample.java

The search API takes words as queries and multiple queries can be combined as a comma separated list. Tweets from the previous 10 days can be searched using this API. Requests to the API can be made using the method *GetSearchResults* presented in Listing 2.10. Input to the function is a keyword or a list of keywords in the form of an OR query. The function returns an array of Tweet objects.

OBTENCIÓN DE DATOS DE TWITTER (10) | Localizaciones

Location information on Twitter is available from two different sources:

- Geotagging information: Users can optionally choose to provide location information for the Tweets they publish. This information can be highly accurate if the Tweet was published using a smartphone with GPS capabilities.
- Profile of the user: User location can be extracted from the location field in the user's profile. The information in the location field itself can be extracted using the APIs discussed above.

Approximately 1% of all Tweets published on Twitter are geolocated. This is a very small portion of the Tweets, and it is often necessary to use the profile information to determine the Tweet's location. This information can be used in different visualizations

Listing 2.12: Translating location string into coordinates

```
public Location TranslateLoc(String loc) {
     if(loc!=null&&!loc.isEmpty()) {
        String encodedLoc="";
        try {
        // Step 1: Encode the location name
                encodedLoc = URLEncoder.encode(loc, "UTF-8");
        . . .
        /** Step 2: Create a get request to MapQuest API with
             the
          * name of the location
          */
        String url= "http://open.mapquestapi.com/nominatim/v1
           /search?q="+encodedLoc+"&format=json";
        String page = ReadHTML(url);
        if(page!=null) {
                try{
                        JSONArray results = new JSONArray(
                            page);
                        if(results.length()>0) {
                                //Step 3: Read and extract
                                    the coordinates of the
                                    location as a JSONObject
                                Location loca = new Location(
                                    results.getJSONObject(0).
                                    getDouble("lat"),results.
                                    getJSONObject(0).
                                    getDouble("lon"));
                                return loca;
        . . .
}
```

Source: Chapter2/location/LocationTranslationExample.java

OBTENCIÓN DE DATOS DE TWITTER (11) | Red de Flujo de Información (1)

On Twitter, information spreads primarily through retweeting. The resulting Tweet is called a retweet. When we visualize retweets we are essentially visualizing the flow of information in the network.

Retweets are marked by the characteristic prefix "RT" followed by the name of the user who originally published the Tweet. For example, consider the following Tweet published by the user John:

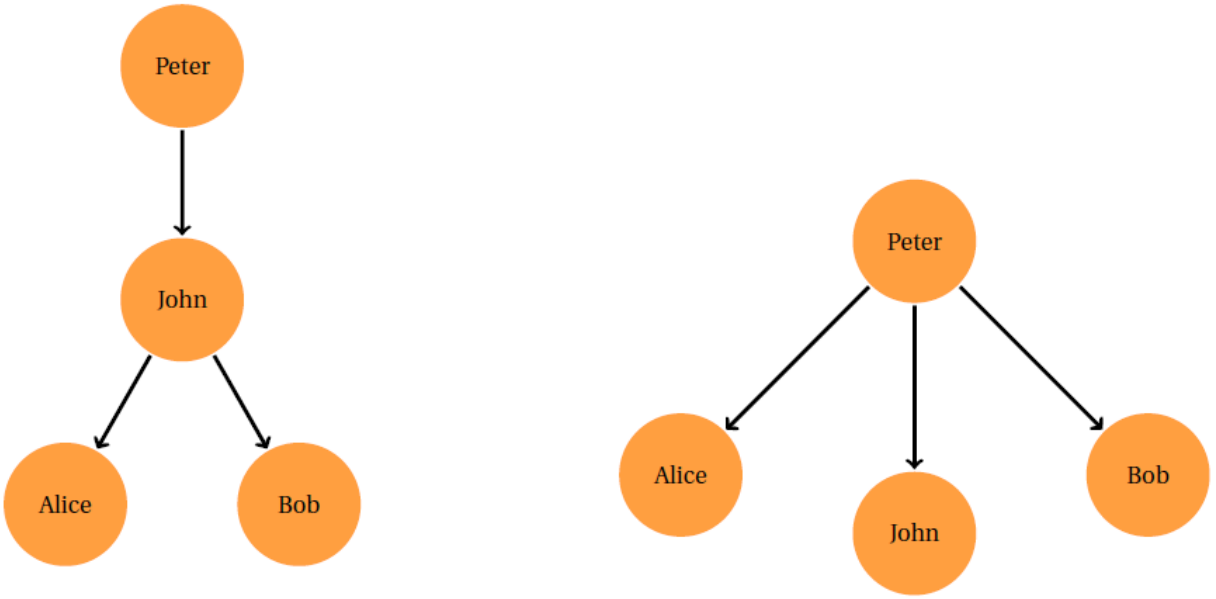
Listing 5.1: Retweet object inside a Tweet object

```
{
      //Other Tweet elements
      . . .
      "created_at": "Thu Mar 14 23:25:03 +0000 2013",
      "text": "RT @Peter: Full-time: Chelsea 3-1 Steaua
          Bucharest. (3-2 on agg) and we're through to the
          quarter-finals. #CFC",
      "retweeted_status": {
         "text": "Full-time: Chelsea 3-1 Steaua Bucharest.
             (3-2 on agg) and we're through to the quarter-
             finals. #CFC",
         "retweeted": true,
          . . .
          //other retweet elements
}
```

Here "Peter" is the name of the user who originally posted the Tweet, which was retweeted by "John". Using this information, an information flow network can be created by connecting information producers with information consumers. When collecting Tweets, one can identify retweets by checking for the presence of the element "retweeted_status" in the JSON response.

Retweet Network Fallacy

An important yet subtle property of this network is that one can only identify the original source of the information and not the intermediate users along the information propagation path. when these Tweets are collected via the Twitter APIs, we only observe the propagation path seen in Figure 5.1b. This means that we cannot identify the full path of propagation, but only the source and the destination.



(a) Actual propagation path (b) Path extracted from Twitter API

Fig. 5.1: Perceived and actual information propagation path of Tweets on Twitter

OBTENCIÓN DE DATOS DE TWITTER (13)

Red de Flujo de Información (3)

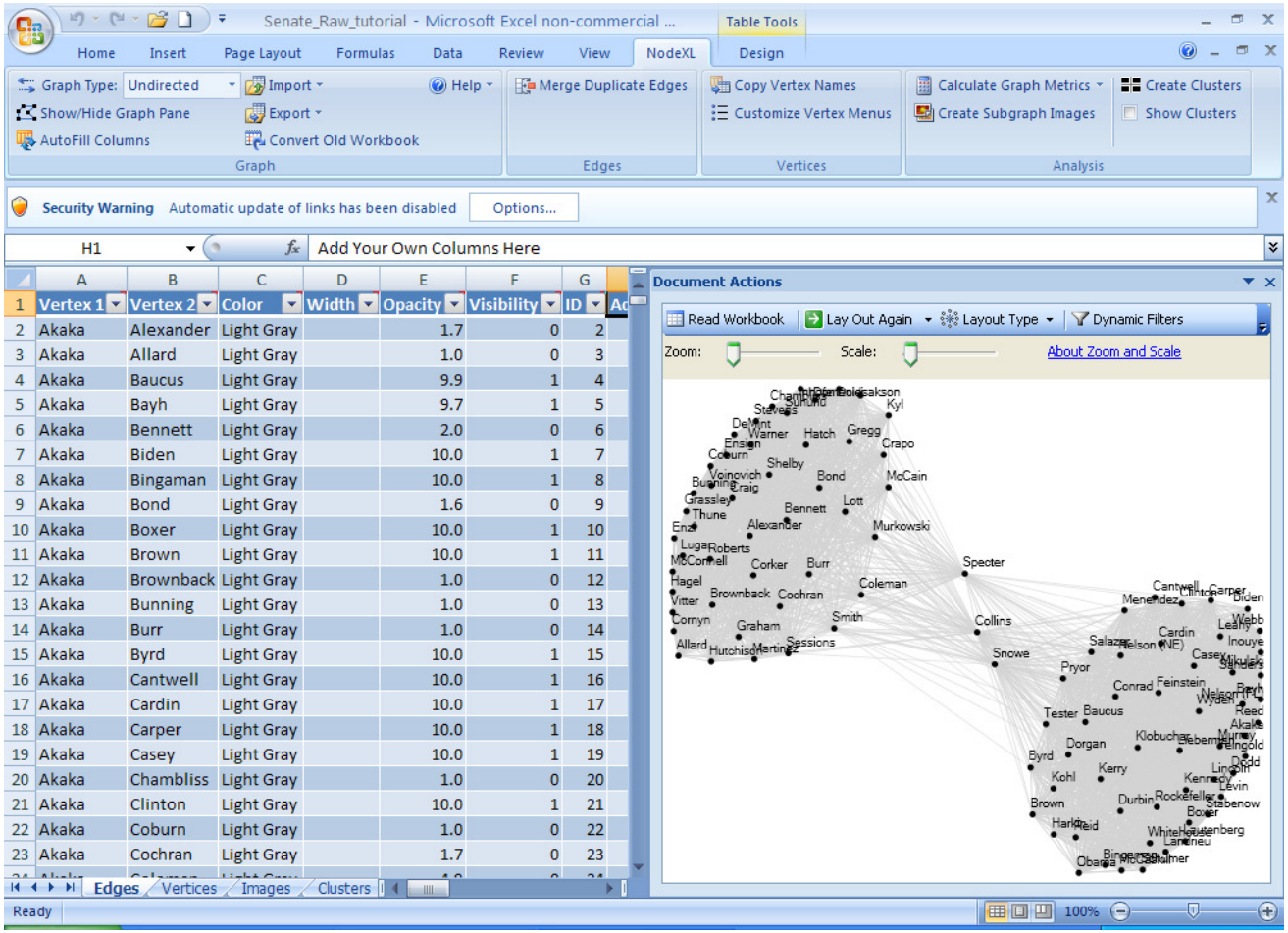
Listing 5.2: Extracting the retweet network

```
public JSONObject ConvertTweetsToDiffusionPath(String
    inFilename,int numNodeClasses, JSONObject hashtags, int
    num_nodes) {
        //Step 1: Read through the file and process Tweets
            matching the topics
            . . .
         //Step 2: Identify the size of the nodes based on
            the number of times they are retweeted
         ArrayList<NetworkNode> nodes = ComputeGroupsSqrt(
            returnnodes, max, min, numNodeClasses);
          . . .
         /** Step 3
            * Prune the network to keep only the top |
               nodes_to_visit| nodes in the network.
         * Recursively visit all top nodes and retain their
            connections.
         */
         for(int k=0;k<nodes_to_visit;k++) {
           NetworkNode nd = nodes.get(k);
           nd.level = 0;
           HashMap<String,NetworkNode> rtnodes =
               GetNextHopConnections(userconnections,nd,new
               HashMap<String,NetworkNode>());
             . . .
      /** Step 4: Compact the nodes of the network by
          removing
         *  all nodes who have never been retweeted
         */
        Set<String> allnodes = prunednodes.keySet();
        //Store the list of retweeted nodes
        ArrayList<NetworkNode> finalnodes = new ArrayList<
            NetworkNode>();
        for(String n:allnodes) {
          . . .
         }
         //Sort in ascending order of the Node ID
         Collections.sort(finalnodes,new NodeIDComparator());
         //Step 5: Reformat the network into  D3 format
         return GetD3Structure(finalnodes);
}
```

Source: Chapter5/network/CreateD3Network.java

NodeXL (1)

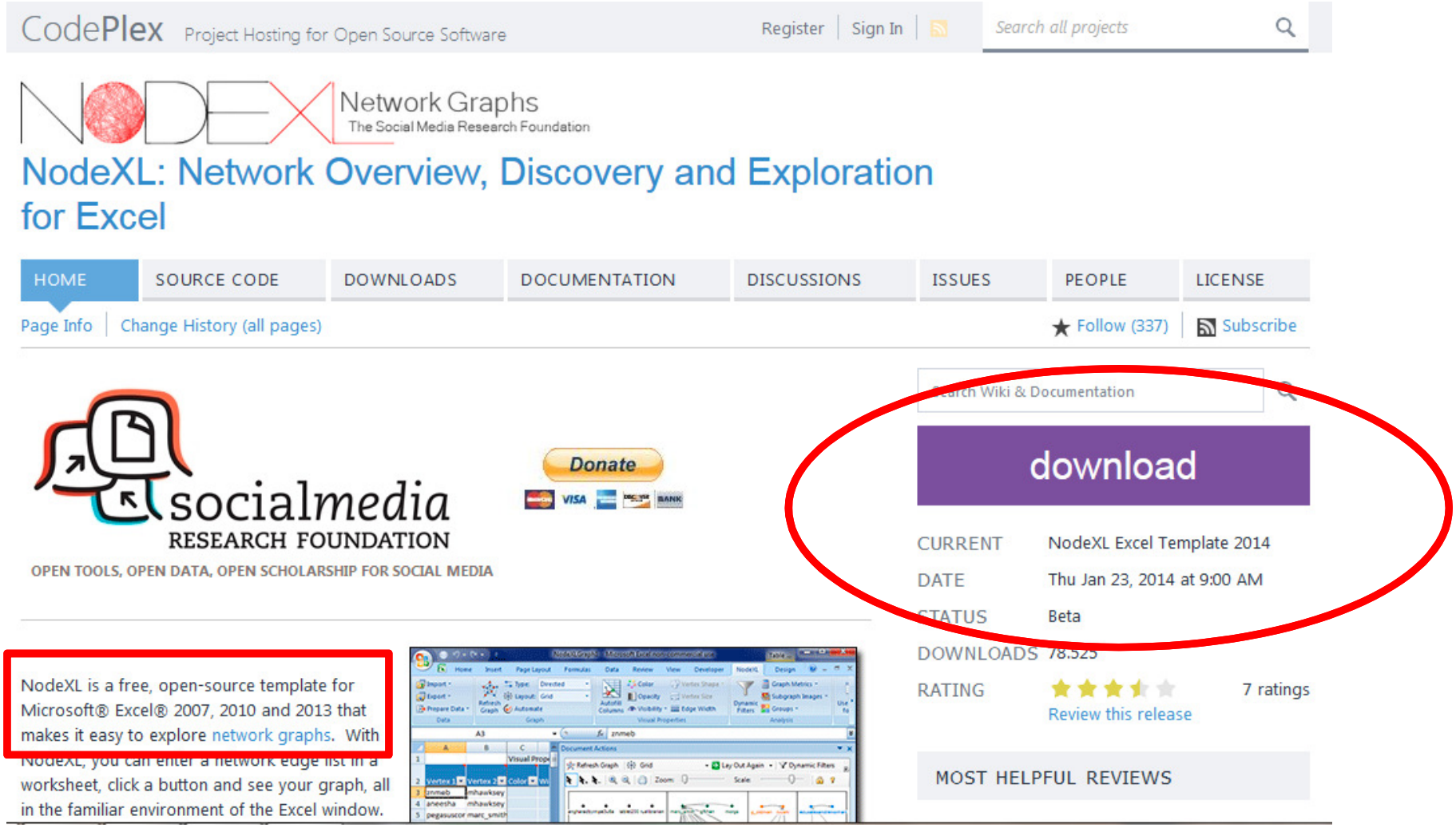
Free/Open Social Network Analysis add-in for Excel 2007/2010 makes network theory (and especially social network analysis from social media) as easy as a pie chart, with integrated analysis of social media sources



	A	B	C	D	E	F	G
1	Vertex 1	Vertex 2	Color	Width	Opacity	Visibility	ID
2	Akaka	Alexander	Light Gray		1.7	0	2
3	Akaka	Allard	Light Gray		1.0	0	3
4	Akaka	Baucus	Light Gray		9.9	1	4
5	Akaka	Bayh	Light Gray		9.7	1	5
6	Akaka	Bennett	Light Gray		2.0	0	6
7	Akaka	Biden	Light Gray		10.0	1	7
8	Akaka	Bingaman	Light Gray		10.0	1	8
9	Akaka	Bond	Light Gray		1.6	0	9
10	Akaka	Boxer	Light Gray		10.0	1	10
11	Akaka	Brown	Light Gray		10.0	1	11
12	Akaka	Brownback	Light Gray		1.0	0	12
13	Akaka	Bunning	Light Gray		1.0	0	13
14	Akaka	Burr	Light Gray		1.0	0	14
15	Akaka	Byrd	Light Gray		10.0	1	15
16	Akaka	Cantwell	Light Gray		10.0	1	16
17	Akaka	Cardin	Light Gray		10.0	1	17
18	Akaka	Carper	Light Gray		10.0	1	18
19	Akaka	Casey	Light Gray		10.0	1	19
20	Akaka	Chambliss	Light Gray		1.0	0	20
21	Akaka	Clinton	Light Gray		10.0	1	21
22	Akaka	Coburn	Light Gray		1.0	0	22
23	Akaka	Cochran	Light Gray		1.7	0	23

NodeXL (2)

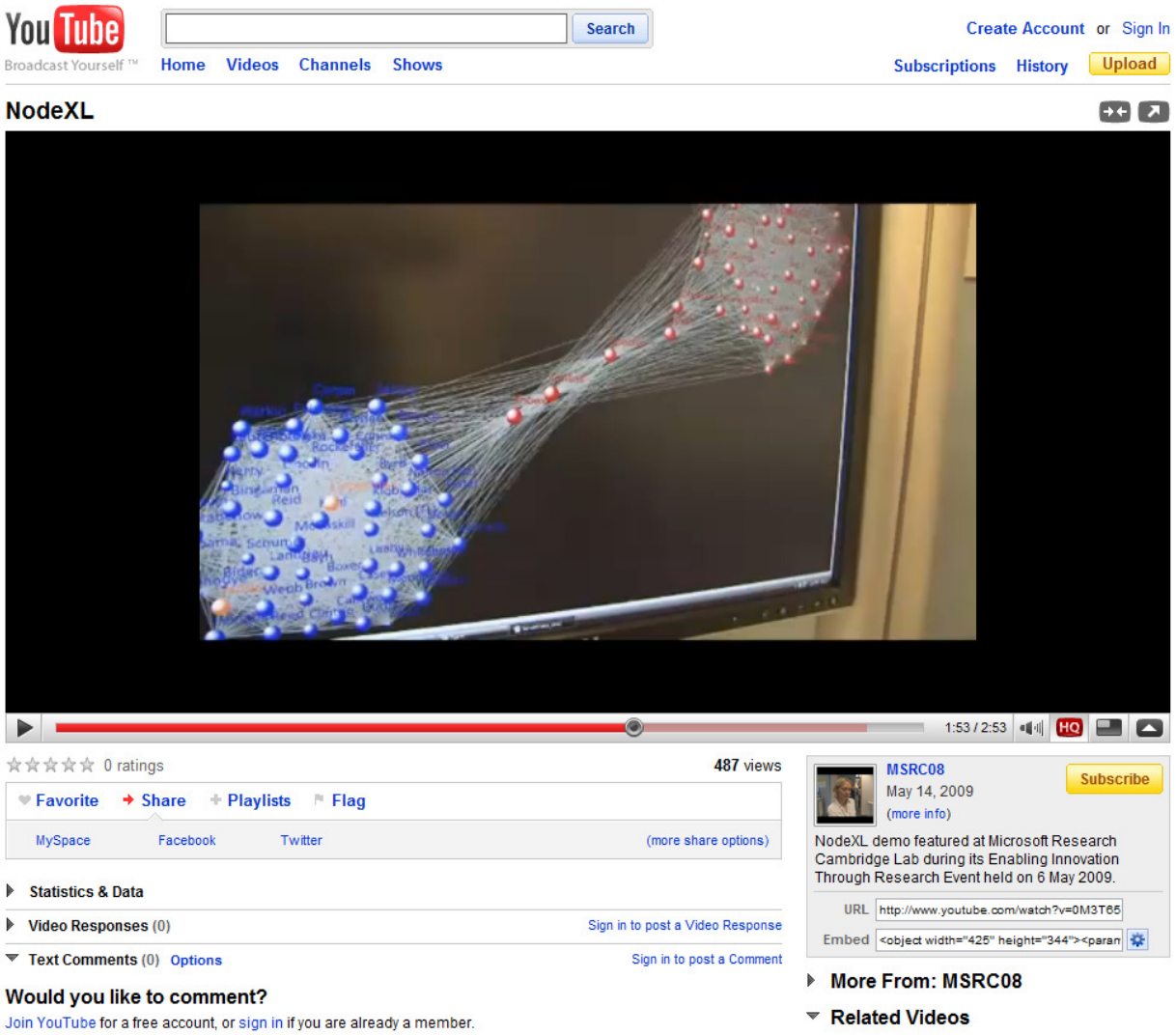
http://nodexl.codeplex.com/



NodeXL (3)

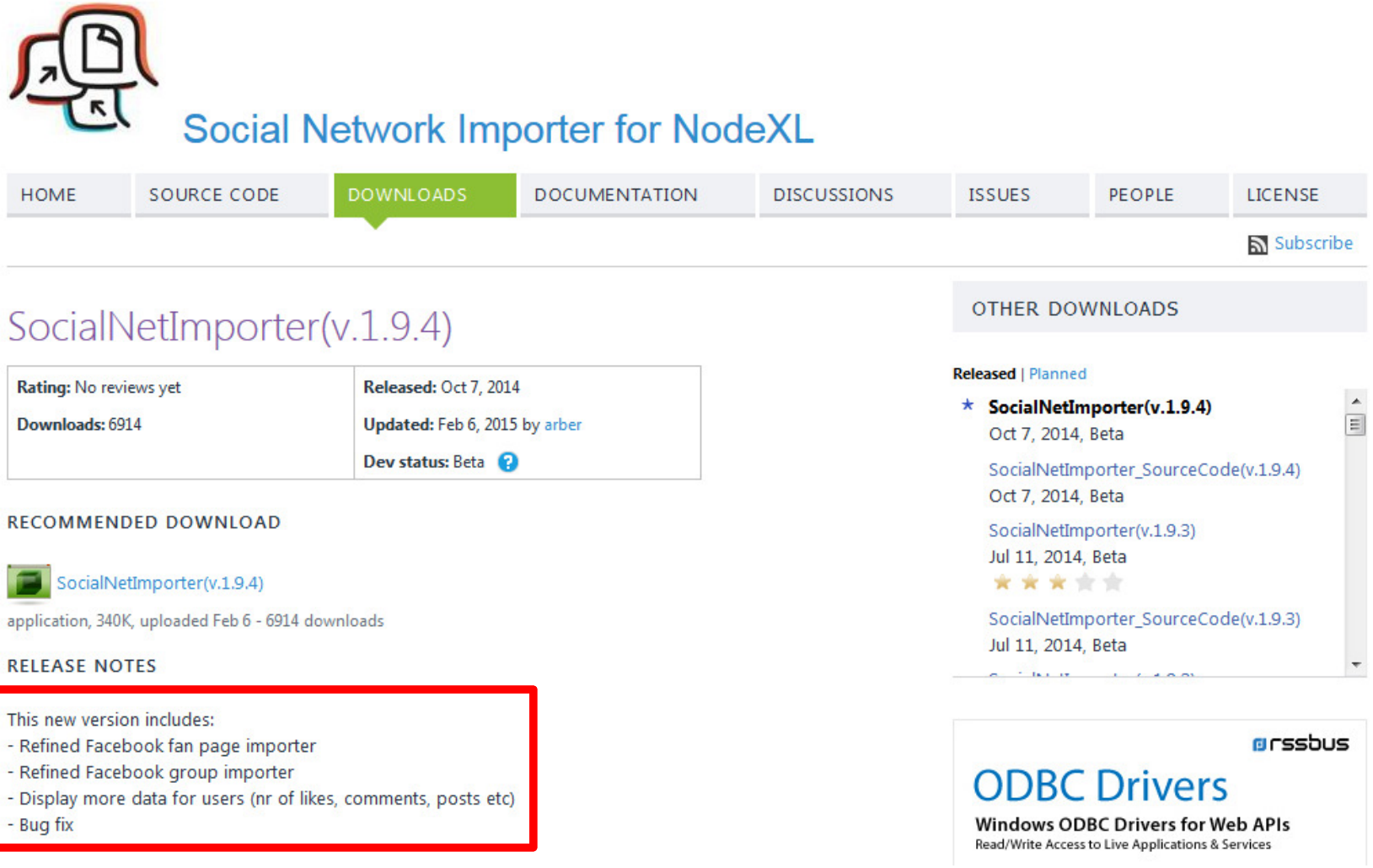
NodeXL Video

http://www.youtube.com/watch?v=0M3T65Iw3Ac



NodeXL (4)

http://socialnetimporter.codeplex.com/releases/view/134889



Social Network Importer for NodeXL

HOME | SOURCE CODE | DOWNLOADS | DOCUMENTATION | DISCUSSIONS | ISSUES | PEOPLE | LICENSE

Subscribe

SocialNetImporter(v.1.9.4)

Rating: No reviews yet	**Released:** Oct 7, 2014
Downloads: 6914	**Updated:** Feb 6, 2015 by arber
	Dev status: Beta

RECOMMENDED DOWNLOAD

SocialNetImporter(v.1.9.4)

application, 340K, uploaded Feb 6 - 6914 downloads

RELEASE NOTES

This new version includes:
- Refined Facebook fan page importer
- Refined Facebook group importer
- Display more data for users (nr of likes, comments, posts etc)
- Bug fix

OTHER DOWNLOADS

Released | Planned

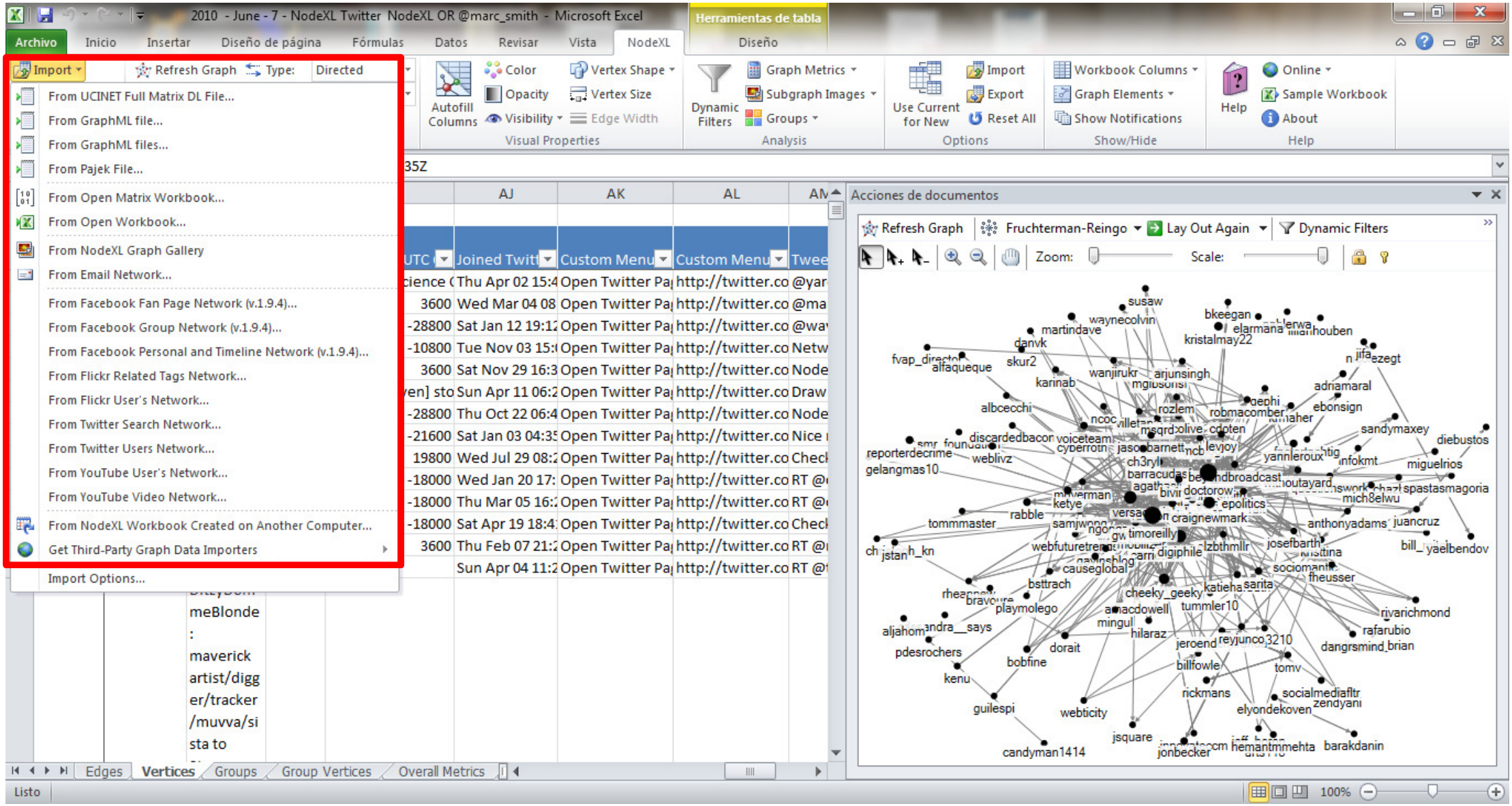
★ **SocialNetImporter(v.1.9.4)**
Oct 7, 2014, Beta

SocialNetImporter_SourceCode(v.1.9.4)
Oct 7, 2014, Beta

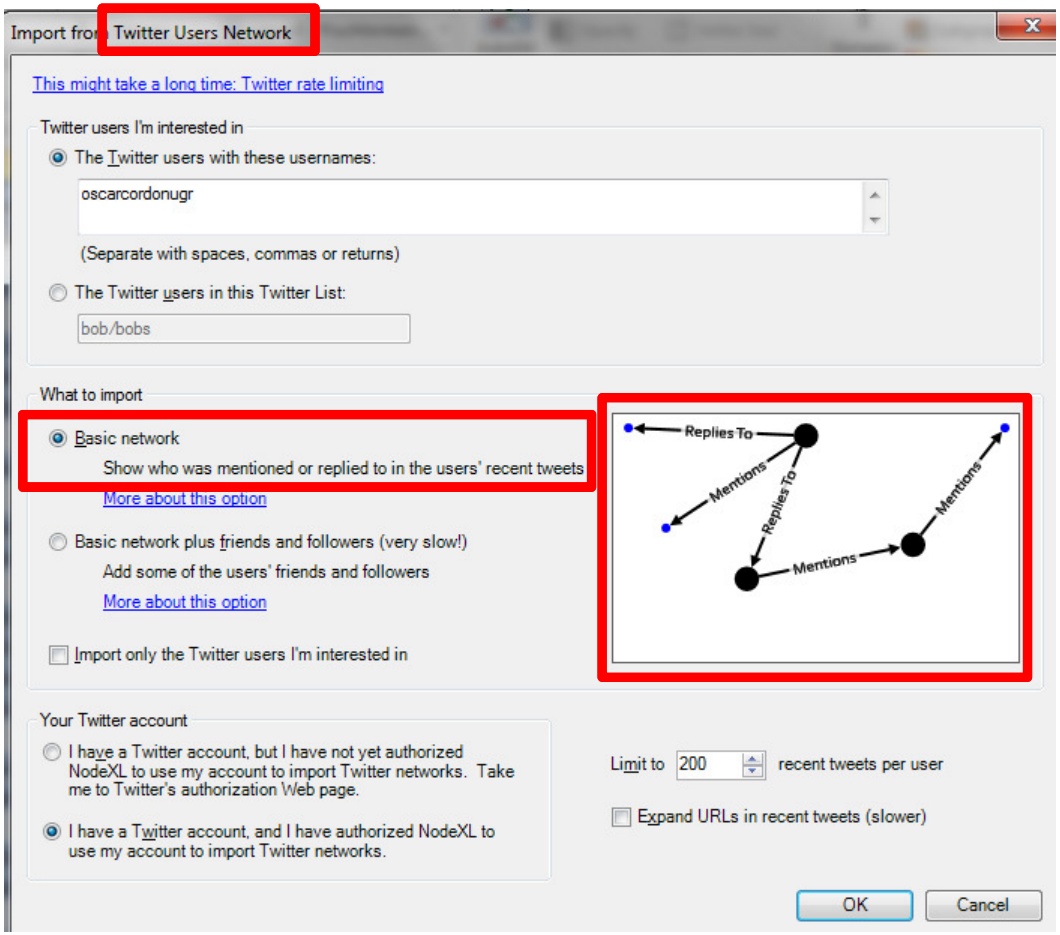
SocialNetImporter(v.1.9.3)
Jul 11, 2014, Beta

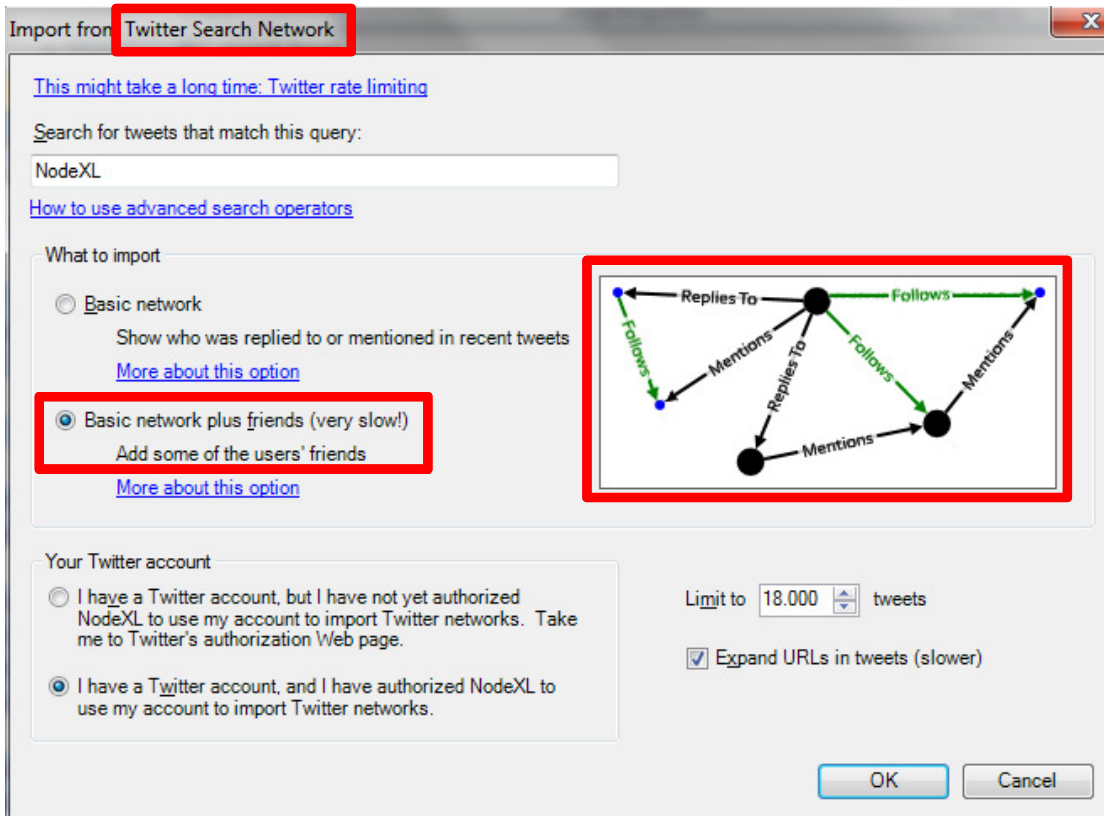
SocialNetImporter_SourceCode(v.1.9.3)
Jul 11, 2014, Beta

NodeXL (5)


2010 - June - 7 - NodeXL Twitter NodeXL OR @marc_smith - Microsoft Excel
Herramientas de tabla
Archivo
Inicio
Insertar
Diseño de página
Fórmulas
Datos
Revisar
Vista
NodeXL
Diseño
Import
Refresh Graph
Type:
Directed
From UCINET Full Matrix DL File...
From GraphML file...
From GraphML files...
From Pajek File...
From Open Matrix Workbook...
From Open Workbook...
From NodeXL Graph Gallery
From Email Network...
From Facebook Fan Page Network (v.1.9.4)...
From Facebook Group Network (v.1.9.4)...
From Facebook Personal and Timeline Network (v.1.9.4)...
From Flickr Related Tags Network...
From Flickr User's Network...
From Twitter Search Network...
From Twitter Users Network...
From YouTube User's Network...
From YouTube Video Network...
From NodeXL Workbook Created on Another Computer...
Get Third-Party Graph Data Importers
Import Options...
Autofill Columns
Color
Opacity
Visibility
Vertex Shape
Vertex Size
Edge Width
Visual Properties
Dynamic Filters
Graph Metrics
Subgraph Images
Groups
Analysis
Use Current for New
Import
Export
Reset All
Options
Workbook Columns
Graph Elements
Show Notifications
Show/Hide
Help
Online
Sample Workbook
About
Help
AJ
AK
AL
Joined Twitt
Custom Menu
Custom Menu
Twee
Acciones de documentos
Refresh Graph
Fruchterman-Reingo
Lay Out Again
Dynamic Filters
Zoom:
Scale:
Edges
Vertices
Groups
Group Vertices
Overall Metrics
Listo
100%

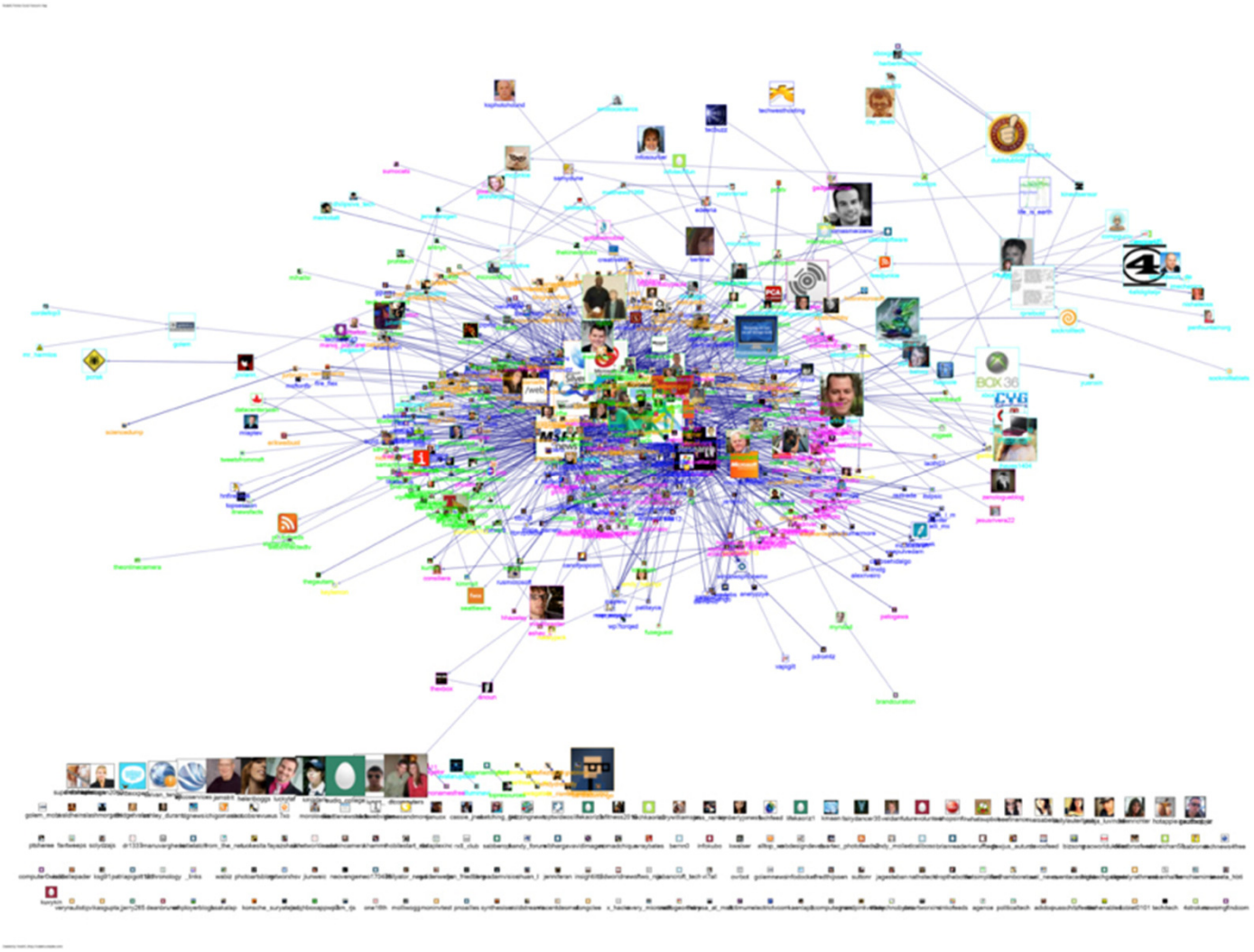
NodeXL (6) Estructura de las Redes de Twitter





NodeXL (7) Ejemplo de Red de Búsqueda (1)

Red de Twitter para “Microsoft Research”
ANTES



NodeXL (8) Ejemplo de Red de Búsqueda (2)

Red de Twitter para “Microsoft Research”
DESPUÉS

Group-In-a-Box Layout for Multi-faceted Analysis of Communities

Eduarda Mendes Rodrigues[*], Natasa Milic-Frayling[†], Marc Smith[‡], Ben Shneiderman[§], Derek Hansen[§]

[*]Dept. of Informatics Engineering, Faculty of Engineering, University of Porto, Portugal
eduardamr@acm.org
[†]Microsoft Research, Cambridge, UK
natasanf@microsoft.com
[‡]Connected Action Consulting Group, Belmont, California, USA
marc@connectedaction.net
[§]Dept. of Computer Science & Human-Computer Interaction Lab
University of Maryland, College Park, Maryland, USA
ben@cs.umd.edu
[§]College of Information Studies & Center for the Advanced Study of Communities and Information
University of Maryland, College Park, Maryland, USA
dlhansen@umd.edu

Abstract—Communities in social networks emerge from interactions among individuals and can be analyzed through a combination of clustering and graph layout algorithms. These approaches result in 2D or 3D visualizations of clustered graphs, with groups of vertices representing individuals that form a community. However, in many instances the vertices have attributes that divide individuals into distinct categories such as gender, profession, geographic location, and similar. It is often important to investigate what categories of individuals comprise each community and vice-versa, how the community structures associate the individuals from the same category. Currently, there are no effective methods for analyzing both the community structure and the category-based partitions of social graphs. We propose *Group-In-a-Box* (GIB), a meta-layout for clustered graphs that enables multi-faceted analysis of networks. It uses the treemap space filling technique to display each graph cluster or category group within its own box, sized according to the number of vertices therein. GIB optimizes visualization of the network sub-graphs, providing a semantic substrate for category-based and cluster-based partitions of social graphs. We illustrate the application of GIB to multi-faceted analysis of real social networks and discuss desirable properties of GIB using synthetic datasets.

Keywords—network visualization; group-in-a-box; layout; meta-layout; force-directed; communities; clustering; semantic substrates.

I. Introduction

Network structures appear in many contexts, from biological systems to communications networks. With the recent proliferation of social media services such as Twitter and Facebook, public awareness and usage of social network data have increased. This led to a growing need for comprehensible visualizations of complex networks that enable exploratory analysis.

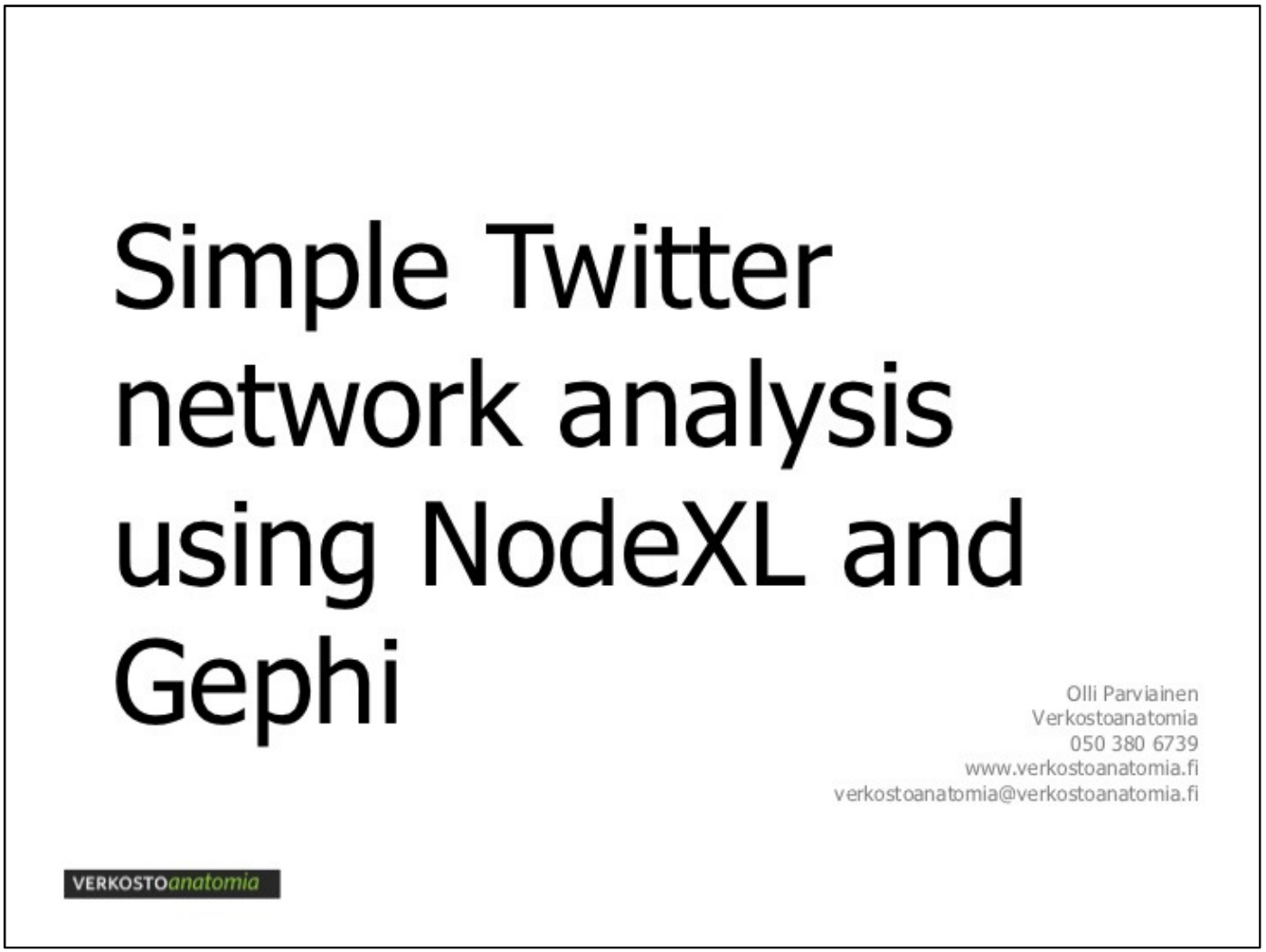
One particularly important aspect of social network analysis is the detection of *communities*, i.e., sub-groups of individuals or entities that exhibit tight interconnectivity among the wider population. For example, Twitter users who regularly retweet each other's messages may form cohesive groups within the Twitter social network. In a network visualization they would appear as clusters or sub-graphs, often colored distinctly or represented by a different vertex shape in order to convey their group identity.

In addition to the clusters that emerge from the network structure, individuals in a social network are often divided into categories that reflect specific attributes. For example, members of the Twitter community may be categorized based on the number of followers they have, the location from which they tweet, or the date they joined Twitter. Such attributes may be useful to gain further insights about the community. Thus, in addition to detecting communities based on the network structure, it is important to enable analysis of the social graph along various attributes of the individuals. At the same time, none of the widely adopted layouts for visualizing clustered graphs accommodate such multi-faceted analysis.

Our research fills that gap by extending the work on network visualization with semantic substrates [20]. We introduce *Group-In-a-Box* (GIB), a meta-layout for clustered graphs that enables multi-faceted analysis of networks. GIB uses the treemap space filling technique to create graphs for individual categories, thus providing a semantic substrate for laying out each group. Furthermore, we demonstrate the use of the method to analyze communities, i.e., graph clusters by assigning them specific spatial regions, applying layout algorithms to reveal the local cluster structure, and observing the categories that their members fall into. In both cases, we arrive at well separated sub-graphs that clearly reveal the

NodeXL y Gephi

http://es.slideshare.net/Verkostoanatomia/visualize-your-twitter-network



Referencias y Agradecimientos

Para diseñar los materiales de este tema, he hecho uso de material desarrollado por expertos en el área disponible en Internet:

- "Guía Básica de Twitter". CGB Informática SL: www.esla.com/cgb/_cgb/Guía%20Básica%20Twitter.ppt

- M.A. Smith. "Charting Collections of Connections in Social Media: Creating Maps & Measures with NodeXL". Social Media Research Foundation: http://es.slideshare.net/Marc_A_Smith/2013-nodexl-social-media-network-analysis