UNIVERSIDAD NACIONAL AUTÓNOMA DE NICARAGUA

UNAN – León

UNIVERSIDAD DE ALCALÁ
Escuela Politécnica

TESIS DE MAESTRÍA

DISEÑO Y DESARROLLO DE UNA HERRAMIENTA SOFTWARE PARA EL ANALISIS DE REDES SOCIALES EN TWITTER

Ing. Wilmer Moisés Matamoros Cáceres

Tutor: Dr. David Fernández Barrero

LEÓN, NICARAGUA

Índice de Contenido

Índice de Figuras

Dedico este trabajo a mis padres
Lesbia y Salvador; y a mis hermanos
Luis, Oscar, Maura y Vladimir

Agradecimiento

Antes que a nadie quiero darle las gracias a Dios por haberme permitido llegar hasta donde he llegado, por darme la fortaleza para continuar cuando más lo necesitaba, por ser una fuente de inspiración en los momentos más difíciles.

Quiero también agradecer a mis padres por haberme conducido por el camino que ellos consideraron que era el mejor, y que al final está dando sus frutos.

Quiero agradecer a tres grandes maestros que me apoyaron mucho durante la realización de este trabajo, mi tutor David Fernández, que sin él esto nunca hubiera visto la luz del día; al profesor Francisco Javier Ceballos, por sus consejos y apoyo incondicional para continuar en este trabajo; y a mi director Ricardo Espinoza por su preocupación por que tanto yo como mis compañeros culmináramos este programa de maestría.

No se me puede escapar también agradecer a mis amigos que han sido un gran apoyo para mí, en especial a Osmin y Alex, que siempre mostraron su preocupación por la culminación de este proyecto. Tampoco se me puede escapar dar las gracias a todos mis colegas, que de una u otra manera me han ayudado con este trabajo, a Denis, Julio, Valeria, Aldo, Santiago, Carlos y demás compañeros del departamento. Infinitas gracias a todos.

1. INTRODUCCIÓN

Las redes sociales son un paradigma de aplicación en la Web que se está difundiendo con una espectacular rapidez. A diferencia de los servicios clásicos, el valor de las redes sociales no está tanto en el servicio propiamente dicho, como en las interacciones sociales que se generan alrededor del servicio. De esta manera, el hecho de poder acceder a las fotos de una red de amigos, comentarlas, valorarlas e interaccionar con otras personas cobra una especial relevancia respecto al mero hecho de publicar una foto en Internet. Bajo este paradigma se están desarrollando una extraordinariamente amplia red de servicios que gozan de presencia mundial con decenas de millones de usuarios, como es el caso de Facebook, Twitter o Flickr, sólo por mencionar algunas.

Al margen del indudable interés comercial que las redes sociales suscitan, su estudio está atrayendo un gran interés por parte del mundo académico. Esto se debe a que la posibilidad de acceder a las interacciones realizadas en las redes sociales abre un campo de investigación que los investigadores apenas podían soñar hace unos años. En definitiva, las redes sociales explicitan en el terreno digital una gran cantidad de información del mundo analógico que hasta ahora resultaba casi imposible analizar. Los campos de estudio que abren son apasionantes, análisis de comportamiento electoral, estudio de movimientos Sociales o estudios de mercado.

Un hecho particularmente interesante es la manera en la que las redes sociales encajan dentro de la Teoría de Grafos [15, 16, 17], que tradicionalmente ha sido una herramienta relegada a ámbitos de investigación muy específicos. Las redes sociales, por su propia naturaleza, siguen una estructura de redes, generalmente los nodos están conformados por personas y los enlaces muestran las interacciones entre dichas personas. De manera que la Teoría de Grafos se identifica como una herramienta de estudio de las redes sociales que aporta rigor formal, una teoría sólida y un espectacular campo de acción.

Dentro de todas las redes sociales existentes en la actualidad, probablemente una de las más prometedoras es Twitter. Twitter es una red social especializada en el *microblogging*, es decir, la publicación de mensajes cortos. Al igual que los SMS de la telefonía móvil, las limitaciones de tamaño en los mensajes han sido utilizadas con ingenio por los usuarios, dándole unos usos que superan con creces las expectativas de sus creadores. De tal manera que Twitter se ha convertido en una herramienta de comunicación dinámica, que permite interaccionar con facilidad con personas u organismos. Sobre estas interacciones se han generado aplicaciones de lo más diverso, con un impacto en la sociedad que son objeto de estudio.

Twitter, a diferencia de otras redes sociales, tiene una naturaleza abierta. Este hecho hace de Twitter un objeto de estudio de gran interés, dado que pueden obtenerse grandes volúmenes de información con muy pocas limitaciones, de tal manera que utilizando las

herramientas adecuadas, ofrece un extraordinario campo de estudio. El objetivo principal de este trabajo es mostrar el potencial que tiene Twitter como objeto de estudio y la Teoría de Grafos como herramienta de análisis. De cara a visualizar este potencial se presenta un sencillo análisis de la actividad realizada en Twitter por parte de los jugadores del Real Madrid, y se comprueba si dicha actividad es coherente con el conocimiento existente sobre los mismos.

1.1. Trabajos relacionados

Las redes son estructuras con una larga historia, tanto en el campo de la informática (por ejemplo, redes de computadoras) como en diferentes campos.

El análisis de redes, es por tanto, un tema sobre el que se ha trabajado mucho. Existe una basta colección de herramientas para el análisis de redes en general. Ya que hoy en día el uso de las redes sociales se ha extendido, siendo cada vez más populares, podemos encontrar también un sinnúmero de herramientas para el análisis de redes sociales, entre ellas podemos encontrar: UCINET, CFINDER, SocNetV y Vizster. A continuación vemos algunos detalles de cada una de estas herramientas.

1.1.1. UCINET

UCINET es una herramienta para el análisis de redes sociales que se apoya en una herramienta de análisis de redes en general llamada NETDRAW para la visualización de la red. Fue creada por Steve Borgatti, Martin Everett y Lin Freeman y actualmente es distribuida por Analytic Technologies, teniendo una versión de evaluación de 60 días, teniendo que adquirir después una licencia de uso. Su pagina web (http://www.analytictech.com/ucinet/) ofrece todo tipo de información para poder adquirir una licencia o para descargar la documentación de la aplicación.

Es un sistema bastante completo, e incluso se suele usar como referencia para evaluar otras herramientas. Tiene un sistema de entrada de datos intuitivo y sencillo, al estilo de una hoja de calculo (tipo Excel) e incluso se pueden importar datos en otros formatos como puede ser RAW, DL, Pajek o Excel. También permite exportar los datos en los que se trabaja a esos mismos formatos, aunque para trabajar con ellos se requiere un paso previo de conversión de los mismos al formato propio, y propietario del UCINET, el DL.

En UCINET, una vez cargados los datos, permite realizar múltiples análisis, incluidos grados de centralidad, cercanía, lejanía, cohesión, subgrupos, etc. Los cálculos

sobre los nodos y la red son bastante completos y abarcan casi todas las posibilidades de estudio.

Para mostrar de forma grafica la red que se está analizando, UCINET se apoya en la herramienta NETDRAW. La siguiente imagen muestra la herramienta NETDRAW visualizando un grafo cargado con UCINET.

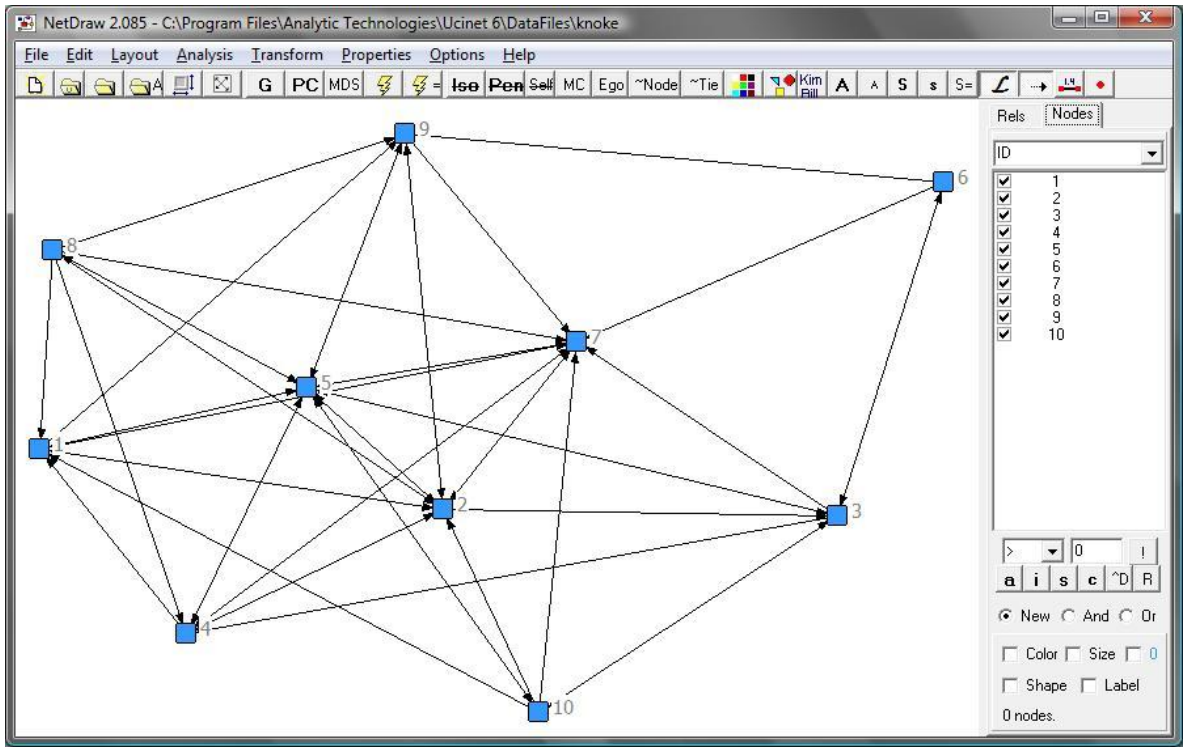


Figura 1.1. Representación de un grafo con NETDRAW cargado con UCINET

1.1.2. CFinder

CFinder es un programa de análisis de redes centrado en la localización de grupos o subconjuntos de actores dentro de los mismos.

La versión que ofrecen es gratuita y válida para todos los sistemas operativos (Windows, Linux y Unix) pero cerrada, aunque, según explican en la pagina web (http://cfinder.org/) se puede pedir permiso para colaborar con ellos añadiendo nuevas funcionalidades.

La entrada de datos al programa es texto plano, aunque sugieren que en el futuro añadirán otros formatos. La salida de sus análisis es la misma, un archivo en texto plano que luego se puede recuperar para trabajar con el.

El software que se ofrece no realiza ningún tipo de análisis de la red más allá de los citados sobre clusters y la calidad de ellos.

La siguiente figura muestra la interfaz principal de la aplicación, donde se ven representados de forma muy clara los posibles grupos existentes.

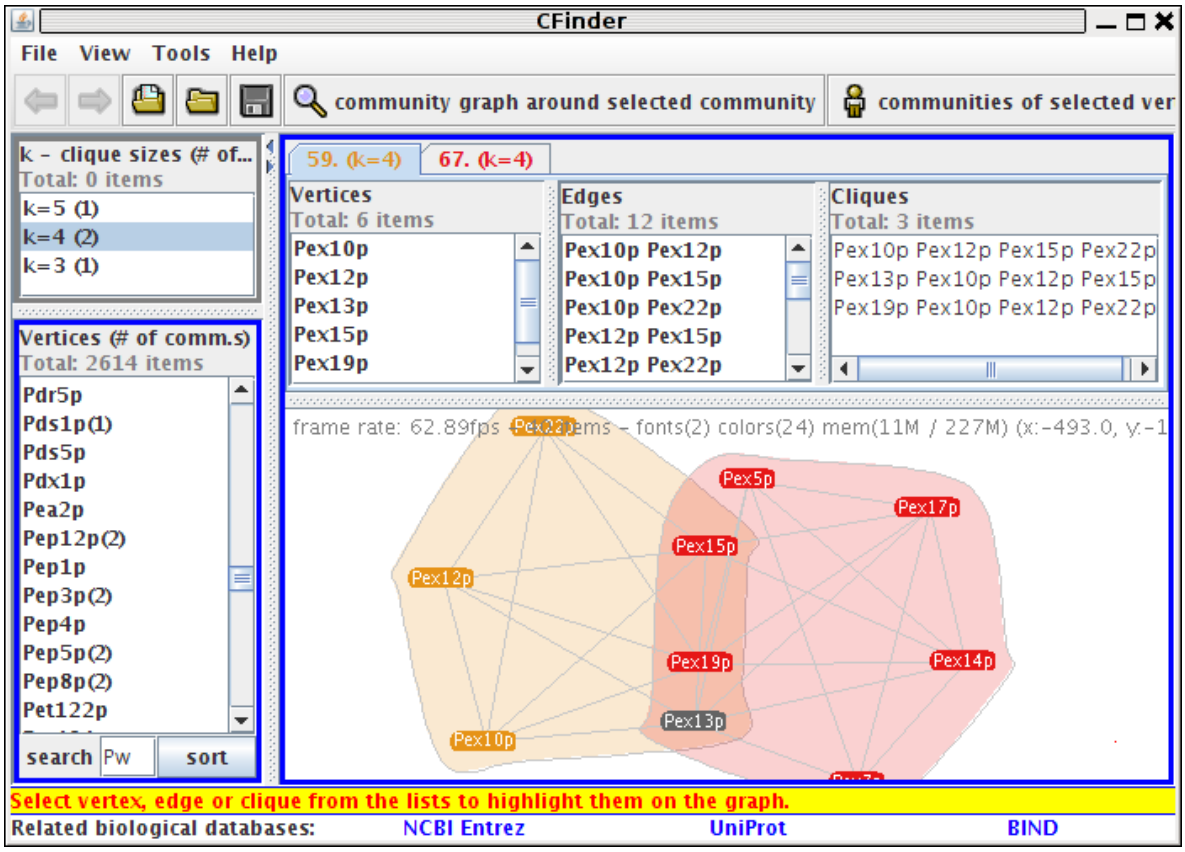


Figura 1.2. Captura de pantalla del programa CFinder

1.1.3. SocNetV

SocNetV es una aplicación gratuita de software libre que ofrece una herramienta flexible y amigable para el análisis de las redes sociales. Está desarrollada usando C++ y utiliza las librerías Qt como soporte para crear el interfaz gráfico.

La aplicación, según los propios autores, está pensada para ser utilizada en Linux, pero permite su uso tanto en Windows como en Mac OSX siempre que se compile en dichas plataformas y se posean las librerías Qt (http://qt.nokia.com/products/appdev)

El SocNetV permite la construcción de un grafo a gusto del usuario gracias a una sencilla interfaz, así como la importación de grafos creados en otras plataformas como pueden ser los GraphViz, GraphML, Adjacency, Pajek o UCINET y su posterior archivo. También da la opción de exportar la imagen del grafo en los formatos más comunes (BMP y PNG entre otros).

La herramienta permite realizar los análisis típicos de la red, como pueden ser el diámetro de la red, la matriz de distancias geodésicas de los nodos o la excentricidad de un nodo. También permite análisis más avanzados como puede ser el grado de entrada (in-degree) o de salida (out-degree), el grado de cercanía (closeness) o el grado de intermediación (betweenness).

La siguiente figura muestra la interfaz de esta herramienta.

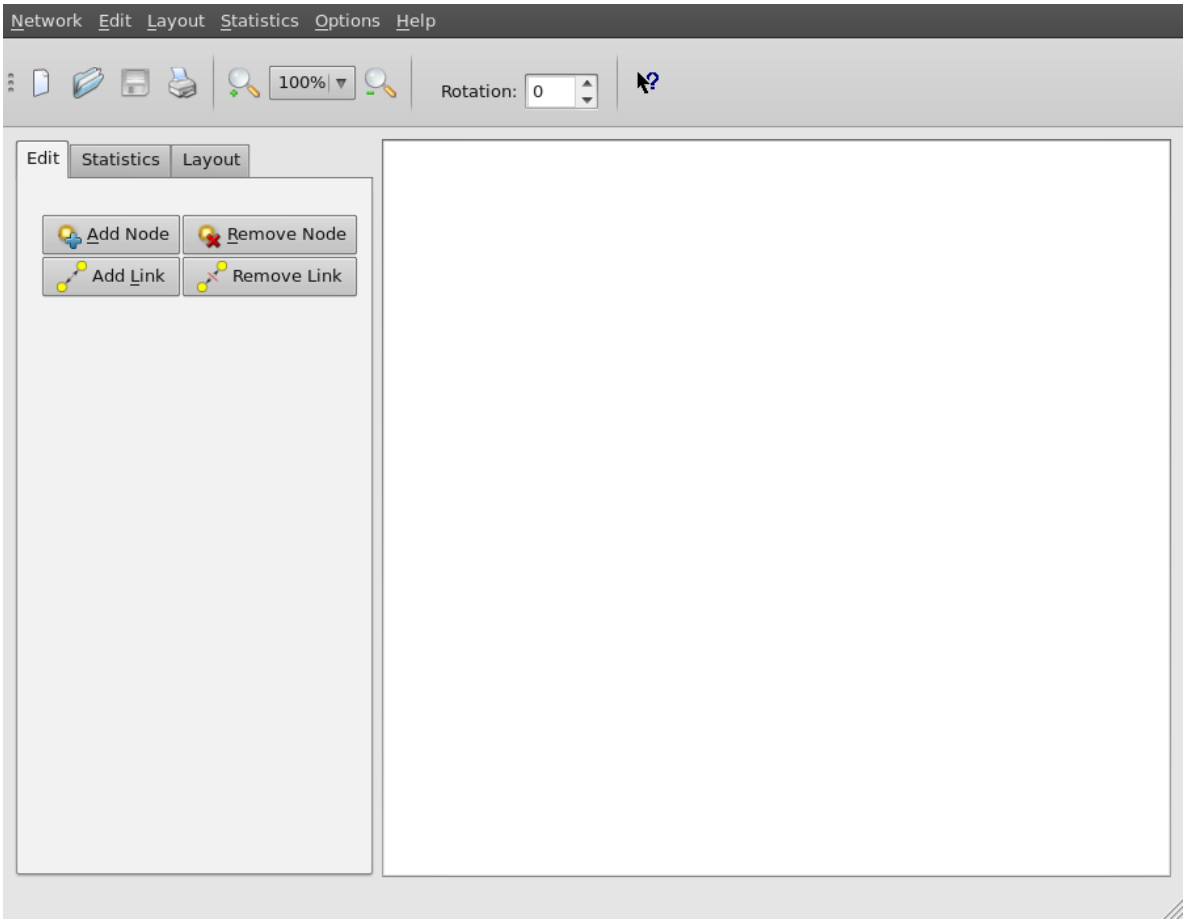


Figura 1.3. Pantallazo de la herramienta SocNetV

1.1.4. Vizster

Vizster, de Jeffrey Heer de la Universidad de Stanford, es una herramienta de visualización de redes sociales que permite la exploración comunidades de las redes que se estudien.

Vizster se ofrece el código fuente de la aplicación bajo la licencia BSD, descargable desde la página web http://sourceforge.net/projects/prefuse/files/vizster/Vizster/vizster.zip/download , requiriendo únicamente tener instalado Java, ya que ofrece un ejecutable y un archivo XML como ejemplo de uso.

La siguiente figura muestra la aplicación con un grafo de una de una red cargada.

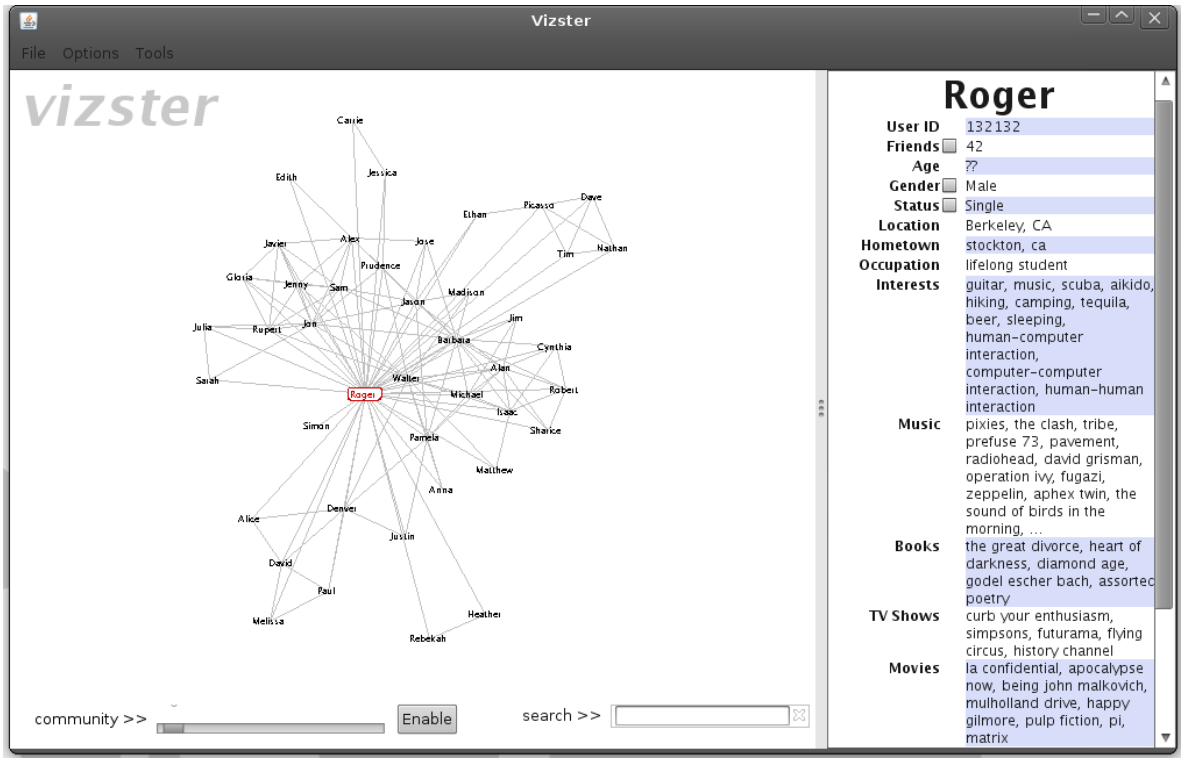


Figura 1.4. Grafo de una red cargada con Vizster.

1.1.5. Gephi

Gephi es un programa de código abierto para la visualización y consulta de grafos. Tiene la particularidad que permite manejar grafos grandes (de miles de nodos) con una muy buena performance, cosa que no es muy común en este tipo de herramientas.

Permite agrupar nodos del grafo, pintarlos de diferentes colores, darle tamaños proporcionales a indicadores, hacer los arcos entre nodos mas gruesos dependiendo de diverso factores, etc.

Muestra muy buena calidad, y permite la importación de archivos en los formatos más comunes para grafos, también permite formatos de archivos separados por comas, haciendo más fácil aún la lectura de dichos grafos.

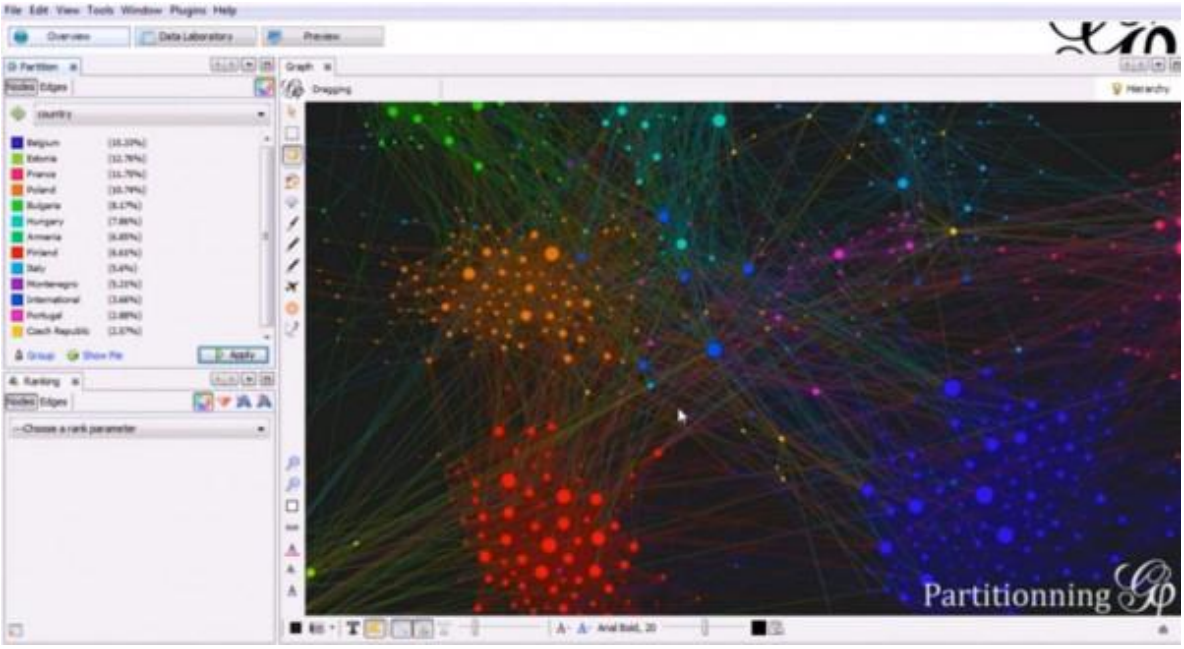


Figura 1.5. Captura de pantalla del programa Gephi.

1.2. Justificación

Puesto que la teoría de redes es una poderosa herramienta matemática para el análisis de diferentes problemas de la vida real, con la presente investigación se pretende plantear el desarrollo de una aplicación que permita hacer análisis de redes; ya que en los últimos años las redes sociales han tomado un gran auge dentro de la sociedad, hemos decidido enfocar el análisis hacia este ámbito, en concreto se pretende realizar una aplicación que permita cargar redes desde la red social Twitter, esto con el fin de aplicar los diferentes conceptos del análisis de redes.

El proyecto nace con la necesidad de estudiar y enfocar la teoría de grafos en un ámbito social, para conocer y dar a conocer como la teoría de redes permite analizar (y a su vez plantear soluciones a) problemas de índole social

1.3. Objetivos

1.3.1. Objetivo general

- Diseñar y desarrollar una herramienta software para la visualización y análisis de la red social Twitter.

1.3.2. Objetivos específicos

- Obtener un grafo dinámico a partir de la información de Twitter, utilizando para ello la API que proporcionan.
- Aplicar conceptos de análisis de redes (teoría de grafos) basándose en la red social Twitter.
- Calcular y mostrar métricas del grafo obtenido, tales como, el número de nodos que la componen, distancia promedio, distancia entre dos nodos, entre otras.
- Mostrar información relativa a los usuarios de Twitter como sus publicaciones, número de seguidores, numero de personas a las que él sigue.
- Realizar un análisis topológico de la red formada por los jugadores del Real Madrid en la red social Twitter.

2. MARCO TEÓRICO

En este capítulo del documento, abordaremos todos aquellos conceptos teóricos relacionados principalmente con el análisis de redes. Empezamos abordando todo lo relacionado con los grafos, tipos de grafos entre otras cosas; a continuación pasaremos a ver cosas sobre análisis de redes o teoría de grafos, también se analizarán las métricas más interesantes que podemos obtener sobre el grafo; el siguiente punto a tratar será las redes sociales y análisis de redes sociales; por último veremos detalles de la red social Twitter y cómo acceder a través de su API.

2.1. Grafos

En matemáticas y ciencias de la computación, un grafo (del griego *grafos*: dibujo, imagen) o gráfica es el principal objeto de estudio de la teoría de grafos.

Los grafos son estructuras discretas que constan de un conjunto de objetos llamados vértices o nodos, unidos por enlaces llamados aristas o arcos, que permiten representar relaciones binarias entre elementos de un conjunto.

Típicamente, un grafo se representa gráficamente como un conjunto de puntos (vértices o nodos) unidos por líneas (aristas).

Formalmente un grafo G se define como un par $G = (V, E)$, donde V es un conjunto finito (que representa a los vértices o nodos) y E es un multiconjunto de pares no ordenados de vértices, denotados por $\{x, y\}$, que definen a las aristas. $V(G)$ representa el conjunto de vértices del grafo G y $E(G)$ el conjunto de aristas del grafo G. Además $v(G)$ y $\varepsilon(G)$ denotan el número de vértices y el número de aristas de G respectivamente. Puesto que E es un multiconjunto es posible que existen pares repetidos, en este caso G tiene lados múltiples. También es posible que algún par no ordenado de E tenga el mismo vértice repetido, en este caso decimos que el lado es un lazo (loop) o bucle. Cuando existen lados múltiples y/o lazos decimos que G es un multigrafo. Si no hay lados múltiples ni lazos decimos que es un grafo simple. Un digrafo G es un par $G = (V,E)$ donde V es un conjunto de vértices y E es un multiconjunto de pares ordenados. Los lados se denotan por pares ordenados, (u, v) denota el lado dirigido que tiene como vértice inicial a u y como vértice terminal a v.

Desde un punto de vista práctico, los grafos permiten estudiar las interrelaciones entre unidades que interactúan unas con otras. Por ejemplo, una red de computadoras puede representarse y estudiarse mediante un grafo, en el cual los vértices representan terminales y las aristas representan conexiones (las cuales, a su vez, pueden ser cables o conexiones inalámbricas).

Prácticamente cualquier problema puede representarse mediante un grafo, y su estudio trasciende a las diversas áreas de las ciencias exactas y las ciencias sociales.

A continuación se describen los diferentes tipos de grafos.

2.1.1. Grafo Simple

Un grafo simple $G = (V, A)$ consta de un conjunto no vacío de vértices V y de un conjunto A de pares no ordenados de elementos distintos de V, a estos pares se les llama aristas. En otras palabras un grafo simple es un grafo en el que existe a lo más una arista que une dos vértices distintos.

2.1.2. Multigrafo

Un multigrafo $G = (V, A)$ consta de un conjunto no vacío de vértices V, un conjunto A de aristas y una función f de A hacia $\{\{u, v\} \mid u, v \in V, u \neq v\}$. Se dice que las aristas a_1 y a_2 son aristas múltiples o paralelas si $f(a_1) = f(a_2)$.

2.1.3. Pseudografos

Un pseudografo $G = (V, A)$ consta de un conjunto V conjunto de V vértices, un conjunto A de aristas y una función f hacia A $\{\{u, v\} \mid u, v \in V\}$. Una arista a es un bucle o lazo, si $f(a) = \{u, u\} = \{u\}$ para algún $u \in V$.

2.1.4. Grafo dirigido

Un grafo dirigido (V, A) consta de un conjunto V de vértices y de un conjunto A de aristas, que son pares ordenados de elementos de V. Utilizamos el par ordenado $\langle u, v \rangle$ para indicar que es una arista dirigida del vértice u al vértice v.

2.1.5. Multigrafos dirigidos

Un multigrafo dirigido $G = (V, A)$ consta de un conjunto V de vértices, un conjunto A de aristas y una función f de A hacia $\{\langle u, v \rangle \mid u, v \in V\}$. Se dice que las aristas a_1 y a_2 son aristas múltiples y paralelas si $f(a_1) = f(a_2)$.

2.1.6. Grafo completo

Un grafo completo es un grafo simple que tiene una arista entre cada par de vértices distintos.

2.2. Conexiones

2.2.1. Caminos

Sea n un entero no negativo y sea G un grafo no dirigido. Un camino de longitud n de u a v en G es una secuencia de n aristas $a_1, a_2, \ldots, a_n$ de G de tal manera que $f(a_1) = \{u, x_1\}, f(a_2) \{x_1, x_2\}, \ldots, f(a_n) = \{x_{n-1}, v\}$. Si el grafo es simple podemos denotar el camino mediante vértices, si es un multigrafo será necesario denotar el camino mediante las aristas, puesto que puede haber ambigüedades.

2.2.2. Circuitos

Un camino de longitud $n > 0$ es un circuito si comienza y termina en el mismo vértice.

2.2.3. Grafos conexos

2.2.3.1. Conexión en grafos no dirigidos

Se dice que un grafo no dirigido es conexo si hay un camino entre cada par de vértices distintos del grafo.

2.2.3.2. Conexión en grafos dirigidos

Se dice que un grafo es *fuertemente conexo* si hay un camino de *a* a *b* y un camino de *b* a *a* para cualesquiera dos vértices *a* y *b* del grafo.

Un grafo es *débilmente conexo* si hay un camino entre cada dos vértices del grafo no dirigido subyacente. Este grafo, es el resultado de ignorar las direcciones de las aristas en un grafo dirigido.

2.3. Grafos ponderados

Llamamos grafos ponderados a los grafos en los que se asigna un número a cada una de las aristas. Este número representa un peso para el recorrido a través de la arista. Este peso podría indicar, por ejemplo, la distancia, el costo monetario o el tiempo invertido, entre otros.

Definimos la longitud de un camino en un grafo ponderado como la suma delos pesos de las aristas de ese camino.

Uno de los problemas más comunes en grafos ponderados es determinar cuál es el *camino más corto* entre dos vértices dados. La solución a este problema tiene aplicaciones directas en muchas áreas, como transporte, manufactura y redes informáticas.

El algoritmo de Dijkstra, también llamado algoritmo de caminos mínimos, es un algoritmo para la determinación del camino más corto dado un vértice origen al resto de vértices en un grafo con pesos en cada arista. Su nombre se refiere a Edsger Dijkstra, quien lo describió por primera vez en 1959.

La idea subyacente en este algoritmo consiste en ir explorando todos los caminos más cortos que parten del vértice origen y que llevan a todos los demás vértices; cuando se obtiene el camino más corto desde el vértice origen, al resto de vértices que componen el grafo, el algoritmo se detiene. El algoritmo es una especialización de la búsqueda de costo uniforme, y como tal, no funciona en grafos con aristas de costo negativo (al elegir siempre el nodo con distancia menor, pueden quedar excluidos de la búsqueda nodos que en próximas iteraciones bajarían el costo general del camino al pasar por una arista con costo negativo).

Dado un grafo *G* ponderado simple y conexo con todos los pesos positivos. Se tiene vértices $v_0, v_1, \ldots\ldots, v_n$, siendo $a = v_0$ el vértice origen y $z = v_n$ el vértice destino. Además tenemos una función de pesos $w(v_i, v_j)$ que determina el peso de la arista que une los vértices v_i y v_j, si dicha arista no existe entonces $w(vi, vj) = \infty$.

El algoritmo incluye un conjunto auxiliar *S* de vértices y una función $L(v)$ que indica la longitud del camino más corto entre *a* y *v*.

- Desde $i = 1$ hast n
  - $L(v_i) = \infty$ [Todos los elementos excepto *a*]
- $L(a) = 0$ [La longitud de *a a a* es 0]
- $S = \Phi$
- Mientras $z \notin S$ hacer
  - u = vértice no en con $L(u)$ mínima
  - $S = S \cup \{u\}$ [agregamos *u* al conjunto]

- Para todos los vértices *v* no en s
    - Si *L(u) + w(u,v) < L(v)* entonces *L(v) = L(u)+w(u,v)* [actualizamos la longitud si fuera menor]
- Al final *L(z)* tiene la longitud del camino más corto entre *a* y *z*.

El Algoritmo de Dijkstra realiza O(*n2*) operaciones (sumas y comparaciones) para determinar la longitud del camino más corto entre dos vértices de un grafo ponderado simple, conexo y no dirigido con *n* vértices.

2.4. Análisis de redes (Teoría de grafos)

El análisis de redes es el área encargada de analizar las redes mediante la teoría de redes (conocida más genéricamente como teoría de grafos). Las redes pueden ser de diversos tipos: social, transporte, eléctrica, biológica, internet, información, epidemiología, etc. Los estudios realizados sobre las redes abarcan sus estructuras tales como en las redes de mundo pequeño, las redes libres de escala, los círculos sociales, medidas de centralidad. Puede ser objeto de estudio la optimización como en el caso de método de la ruta crítica, el PERT (del inglés Program Evaluation & Review Technique). Así como la dinámica de las redes como puede ser el estudio de sistema dinámico secuencial (SDS del inglés Sequential Dynamical System), o de propiedades como la asignación dinámica de flujos.

Cada uno de estos tipos de redes puede ser modelado mediante un grafo, y presentan rasgos topológicos específicos que permiten conocer el tipo de conectividad y el impacto de las actividades que ocurren en dichas redes.

El análisis de redes, puede decirse que conceptualmente es la intersección entre la teoría de grafos y el estudio del conjunto de estadísticas y métricas que se pueden obtener mediante el modelado de las redes.

2.4.1. Redes de Mundo Real

Las redes de mundo real son un tipo de redes que representan diferentes situaciones de la vida cotidiana y que a simple vista no parecen tan complejas de entender, en algunos casos estas redes modelan el comportamiento de la sociedad y la forma en como estas interactúan bajo situaciones reales. Ejemplo de este tipo de redes son las formadas por los alumnos de las escuelas primarias en una determinada ciudad, o el número de enfermos de cáncer en un hospital, o las comunidades de grupos étnicos en el país, entre otros. Sin embargo, las redes de mundo real no sólo se aplican en estructuras sociales, sino que se pueden observar en otro tipo de situaciones como: la propagación de epidemias, las redes de interacción de proteínas en el metabolismo celular, el comportamiento de los animales, las redes de neuronas en los organismos del sistema nervioso, la red de distribución eléctrica, las redes de carreteras, entre otras.

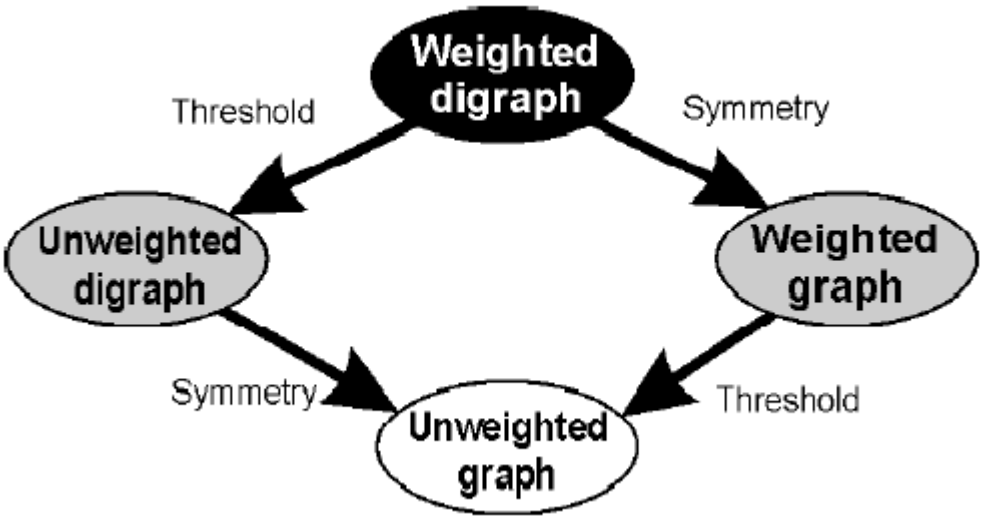


Figura. 2.1. Los 4 tipos principales de redes complejas y sus transformaciones.

2.4.2. Redes complejas

La figura 2.1 muestra los 4 tipos principales de redes complejas que podemos encontrar; éstas son: *digrafos(grafos dirigidos) ponderados*, *digrafos no ponderados*, *grafos ponderados* y *grafos no ponderados*. También se muestran las operaciones que permiten hacer transformaciones entre ellas, que son, la *simetría* que permite transformar un digrafo en un grafo y la *umbralización* que transforma un grafo ponderado en uno no ponderado.

Un *digrafo ponderado* es un grafo en el que el conjunto de vértices tiene una dirección, es decir que podemos identificar en un enlace un nodo como origen y el del otro extremo como el destino, en este caso el enlace se representa por una fleca; también un peso asociado, que indica el costo que tiene ir de un nodo a otro a través de ese enlace.

En cuanto al *dígrafo no ponderado*, podemos decir, que los enlaces tienen una dirección pero no tienen un coste asociado (también se puede decir que su coste o peso es la unidad).

Análogamente, podemos indicar que un *grafo ponderado* es aquel en el que los enlaces no determinan una única dirección, pero si tienen un peso asociado. Y los *grafos no ponderados* son los que ni tienen una dirección ni peso especifico en sus enlaces.

Cabe destacar que todos los tipos de redes pueden ser derivados a partir de un digrafo ponderado aplicando las correspondientes transformaciones.

Las redes de mundo real son más complejas en el sentido que se derivan diferentes características topológicas de la teoría de las redes aleatorias. Un grafo complejo contiene muchos subgrafos diferentes. Muchos sistemas en la naturaleza están construidos por un alto número de conexiones dinámicas (p.ej., redes neuronales y el Internet) y existen diferentes modelos de redes complejas como las *redes aleatorias, redes de mundo pequeño y redes libres de escala* que permiten estudiar el comportamiento de muchas redes del mundo real.

A continuación vemos detalles de cada uno de estos modelos de redes.

2.4.2.1. Redes aleatorias

Este tipo de redes son generadas a partir de agregar nodos de manera aleatoria a un conjunto de datos fijos. Este tipo de redes presentan caminos muy cortos entre nodos y un coeficiente de agrupamiento bajo, además presentan una distribución Binomial o de Poisson como en la figura 2.2. Existen diferentes modelos que han sido propuestos para este tipo de redes como el modelo Erdös-Rényi (ER)[8] y el de Gilbert[6], siendo estos los primeros modelos aplicados a las redes sociales.

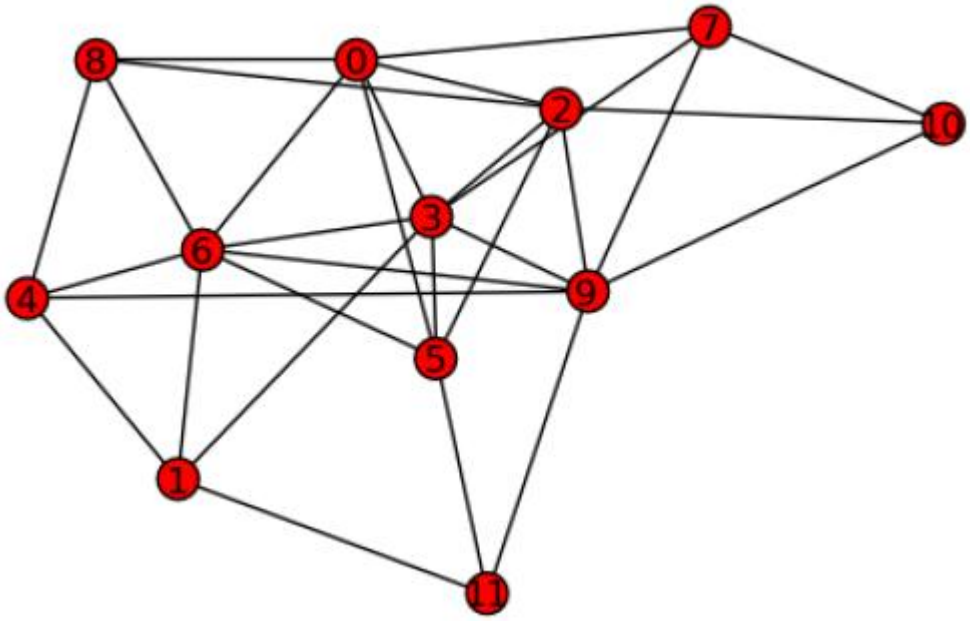


Figura 2.2. Ejemplo de una red aleatoria con 12 vértices y 29 aristas. Las aristas son generadas entre pares de vértices seleccionados uniforme y aleatoriamente. Como resultado se tiene una distribución de grado binomial o de Poisson.

2.4.2.2. Redes de mundo pequeño

Las redes de mundo pequeño son redes que tienen un diámetro pequeño y exhibe un alto agrupamiento. El problema del mundo pequeño fue planteado en 1967 por Stanley Milgram[7], donde textualmente se hace estas dos preguntas:

1. *"Comenzando con dos personas cualquiera en el mundo, ¿Cuál es la probabilidad que ellos se conozcan?"*

2. *"Dados dos personas cualquiera en el mundo, persona X y persona Z, ¿Cuantos enlaces intermedios son necesarios antes que X y Z se conecten"*

Estas interrogantes motivaron a Milgram a generar una serie de experimentos, que consistió básicamente en monitorear a un grupo de personas que interactuaban enviando cartas a desconocidos, los resultados mostraron que la red de la sociedad humana presenta una estructura de mundo pequeño y que los individuos están a *"seis grados de separación"*. Watts [9] describe la teoría de los *seis grados* y plantea que *"el mundo es un pañuelo"*, es decir, cualquiera en el mundo permanece conectado a otra persona a través de otras personas y que el número de personas intermedias es menor a cinco. Watts y Strogatz proponen un modelo basado en el fenómeno de las redes mundo pequeño [12].

Estudios recientes han mostrado que la Web es del tipo mundo pequeño, debido al alto índice de agrupamiento y de las distancias tan cortas que exhiben estos sitios. En un estudio sobre redes criminalistas desarrollado por investigadores de la Universidad de Arizona[10], emplearon la teoría de las redes de mundo pequeño para determinar que individuos dentro de la red formaban parte o podían en algún momento formar parte de un grupo de terroristas o criminales.

2.4.2.3. Redes ley de potencia

Son redes en donde la probabilidad de que un nodo tenga grado *x* es proporcional a $x^{-\alpha}$ donde la constante α, es llamada *coeficiente ley de potencia* y es un valor fijo que debe de satisfacer a $\alpha > 1$. Cuando la probabilidad de alguna variable se distribuye de acuerdo a una ley de potencia, su función de distribución se define como:

$$p(x) = Cx^{-\alpha}$$

donde *p(x)* es la frecuencia (probabilidad) de que la variable tome un valor de *x*, es el exponente de la distribución, *x* es la variable que se requiere analizar y *C* es una constante que depende del tipo de evento.

2.4.2.4. Redes libres de escala

Las redes libres de escala son una clase de redes ley de potencia, donde un nodo con un alto grado tiende a ser conectado a otro nodo con un alto grado, es decir, el número de enlaces en la red está concentrado en un número pequeño de nodos. Este tipo de red presenta una mejor distribución entre sus enlaces en comparación a las redes aleatorias, ya que en este tipo de redes hay más nodos con pocos enlaces que nodos con un gran número de enlaces, con esto se garantiza que el sistema esté altamente conectado.

Una de las principales diferencias entre las redes libres de escala y las redes aleatorias es la distribución de grado. No todos los nodos en una red tienen el mismo número de enlaces (grado nodal), una red aleatoria genera sus enlaces aleatoriamente y esto provoca que la mayoría de los nodos tengan el mismo grado dado que siguen una distribución de Poisson. La figura 2.3 muestra la distribución en las redes libres de escala y las redes aleatorias, se puede apreciar que las redes aleatorias presentan una homogeneidad en sus enlaces, mientras que en las redes libres de escala no existe tal homogeneidad.

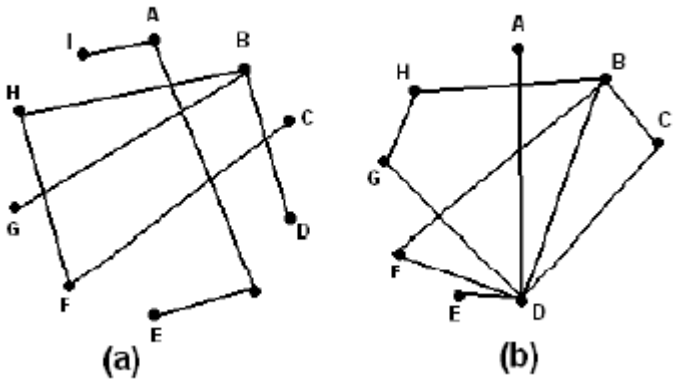


Figura 2.3. Ejemplo de una (a) red aleatoria y una (b) red libres de escala. Nótese que la distribución de enlaces en (a) es homogénea.

La idea principal de este tipo de redes parte del crecimiento de las redes en sistemas complejos con la adición de nuevos nodos y las relaciones formadas en redes existentes y por exhibir un comportamiento de anexo preferencial, es decir, existe una alta probabilidad de que un nuevo nodo se agregue a la red formando un enlace con un nodo que posee un gran número de enlaces. Muchas redes del mundo real presentan un comportamiento de libre de escala (p.ej., la estructura de la red celular y la red de e-mail), otras redes utilizan las propiedades ofrecidas por este tipo de redes para medir su estructura (p.ej., la red formada por las ontologías en la Web semántica).

2.5. Métricas

En este apartado trataremos de describir y analizar las métricas más relevantes que podemos obtener a partir del modelado de un grafo.

2.5.1. Métricas relacionadas con la distancia

En digrafos no ponderados el número de enlaces que unen al nodo *i* y *j* es conocido como *longitud* de la ruta. El camino más corto entre dos nodos es aquel que está compuesto por el menor número de enlaces y es conocido como *distancia geodésica*.

La distancia es una característica muy importante que depende de toda la estructura de la red. A continuación se describe métricas basadas en la distancia de los vértices.

Definiendo la distancia geodésica entre dos vértices de un grafo como d_{ij} podemos definir la distancia geodésica promedio de todo el grafo como:

$$\ell = \frac{1}{N(N-1)} \sum_{i \neq j} d_{ij}.$$

Donde, N es el número de vértices que componen el grafo.

El problema que presenta esta definición es que diverge si se encuentra nodos aislados en la red, es decir nodos que no están conectados al resto de la red. Para evitar este problema, es necesario sólo incluir pares de vértices conectados en la suma. Esto evita la divergencia, pero introduce una distorsión para redes que tienen muchos pares de vértices que no son alcanzables en la red, con lo que muestra un pequeño valor de distancia promedio, por lo que es útil en redes con un elevado número de conexiones.

Latora y Marchioni[8] propusieron una medida estrechamente relacionada que definieron como *eficiencia global* (también conocida como coeficiente de agrupamiento global – ver sección 2.6.2)*:*

$$E = \frac{1}{N(N-1)} \sum_{i \neq j} \frac{1}{d_{ij}},$$

Con esto, en la suma sólo se incluyen aquellos pares de vértices que son alcanzables en la red. Esta métrica cuantifica la eficiencia de la red en cuanto a la

interacción de sus vértices, suponiendo que dicha eficiencia es inversamente proporcional a la distancia. El recíproco de la *eficiencia global* es conocido como *media armónica* de las distancias geodésicas y está dado por:

$$h = \frac{1}{E}.$$

Ya que esta ecuación no presenta el problema de divergencia, es una medida más apropiada para grafos con más de un componente conectado.

2.5.2. Centralidad de redes

La necesidad por entender el comportamiento de los nodos en una red social es algo que hace al concepto de centralidad muy importante en el análisis de redes. Con la centralidad se puede determinar que tanto influye un nodo en comparación a los demás elementos de la red. Los cálculos de centralidad se basan en el uso de las matemáticas y los conceptos de la teoría de grafos.

El tipo de medición de la centralidad se distingue por el tipo de grafo, en el caso de los grafos no dirigidos el sentido de las relaciones no importa, en cambio, los grafos dirigidos si.

En grafos dirigidos, a las relaciones se les conoce como “prestigio”, el cual representa y permite formular las medidas de centralidad de distintas formas.

Existen diferentes tipos de medición de centralidad, los más representativos son: centralidad de grado (*Degree*), centralidad de cercanía (*Closeness*) y la centralidad de intermediación (*Betweenness*), a continuación se detalla de manera más clara este tipo de métricas utilizadas para el análisis de redes.

2.5.2.1. Centralidad de grado

Se define al grado de un nodo como el número de enlaces que posee un nodo, es decir, el número de relaciones que tiene el nodo con los otros nodos. Se dice que un nodo tiene un vecindario el cual está formado por los nodos a su alrededor. El grado es una característica muy importante de un vértice. Basado en esta característica, es posible derivar un conjunto de métricas del grafo. El grado k del vértice i está dado por:

$$k_i = \sum_j a_{ij} = \sum_j a_{ji}.$$

El *grado medio* de un grafo es el promedio de los grados k_i de todos los vértices del grafo. Y se denota por:

$$\langle k \rangle = \frac{1}{N} \sum_i k_i = \frac{1}{N} \sum_{ij} a_{ij}.$$

Suele ser interesante encontrar cual es el máximo grado, esto lo podemos denotado por:

$$k_{\max} = \max_i k_i.$$

Análogamente, podemos obtener el mínimo grado entre todos los vértices.

Para los grafos dirigidos el grado del nodo suele ser dividido en dos formas: el grado de entrada (*in-degree*) y el grado de salida (*out-degree*). El grado de entrada del nodo *i* en un grafo dirigido es el número de nodos N_v en el vecindario que tienen un enlace que se dirigen hacia *i*, el grado de salida del nodo es el número de nodos N_v en el vecindario que poseen enlaces que son dirigidos desde *i*. La figura 2.4 muestra el grado de entrada y el grado de salida para los nodos de un grafo dirigido.

El grado de entrada y salida se define así:

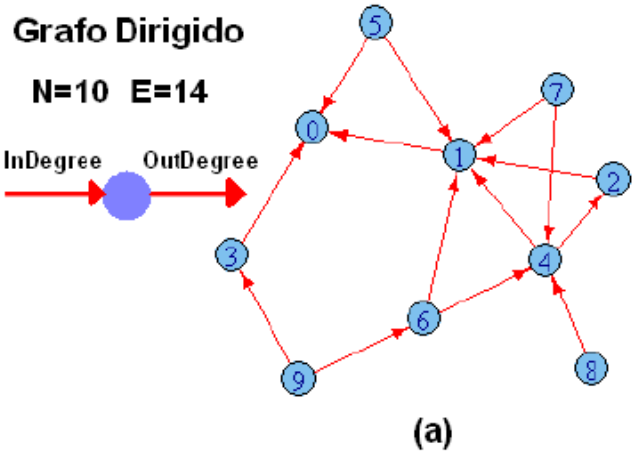
$$\begin{aligned} k_i^{\text{out}} &= \sum_j a_{ij}, \\ k_i^{\text{in}} &= \sum_j a_{ji}. \end{aligned}$$

Cabe destacar que en estos casos el *grado total* se define como la suma de ambos grados:

$$k_i = k_i^{\text{in}} + k_i^{\text{out}}$$

Además, el promedio del grado de salida es igual al promedio de grado de entrada:

$$\langle k^{\text{out}} \rangle = \langle k^{\text{in}} \rangle = \frac{1}{N} \sum_{ij} a_{ij}.$$



Distribución de Enlaces

Nodo	Grado de Entrada (In-degree)	Grado de Salida (Out-degree)	Grado Nodal (Degree)
0	3	0	3
1	5	1	6
2	1	1	2
3	1	1	2
4	3	2	5
5	0	2	2
6	1	2	3
7	0	2	2
8	0	1	1
9	0	2	2

(b)

Figura 2.4. Ejemplo de los diferentes tipos de grados en un grafo dirigido. La figura (a) muestra un grafo dirigido con 10 nodos y 14 aristas y en la tabla (b) se muestra el valor del grado de entrada (in-degree), grado de salida (out-degree) y el grado (degree) para cada uno de los vértices en la red.

2.5.2.1.1. Distribución de grados

La distribución de grados (*P(k)*) expresa la fracción de nodos en una red con el grado *k*, es una propiedad muy importante que podemos medir en redes del mundo real. Para redes dirigidas la distribución de grado de salida P^{out} (k^{out}), la distribución de grado de entrada P^{in} (k^{in}) y la distribución conjunta de grado de entrada y salida $P^{io}(k^{in}, k^{out})$. Esta última distribución da la probabilidad de encontrar un vértice con grado de entrada k^{in} y grado de salida k^{out}. Cabe destacar que en redes ponderadas podemos aplicar este mismo concepto en relación al peso de las aristas.

2.5.2.1.2. Correlación de grados

A menudo es interesante comprobar si existe correlación entre los grados de vértices diferentes. El enfoque más natural es considerar la correlación entre dos vértices conectados directamente por medio de una arista. Esta correlación puede ser expresada por la distribución conjunta de los grados *P(k, k´)*, es decir, la probabilidad de que una arista

cualquiera conecte un vértice con grado *k* con otro vértice de grado *k´*. Otra forma de expresar la dependencia entre los grados de los vértices, es en términos de la probabilidad de que un vecino cualquiera de un nodo con grado *k* tenga un grado *k´*.

$$P(k'|k) = \frac{\langle k \rangle P(k,k')}{kP(k)}.$$

Tenga en cuenta que $\sum_{k'} P(k'|k) = 1$. Para redes no dirigidas, $P(k, k') = P(k', k)$ y $k'P(k|k')P(k') = kP(k'|k)P(k)$. En el caso de redes dirigidas *k* corresponde al grado del vértice saliente (es decir, de donde salen las aristas) y *k´* es el grado del otro vértice, *k* y *k´* pueden ser, el grado de entrada, de salida o el grado total. Para redes ponderadas el peso de las aristas puede utilizarse en lugar de *k*.

2.5.2.2. Centralidad de cercanía

La centralidad de cercanía mide los pasos requeridos para acceder a cada otro vértice de un vértice dado. Esta medida se basa en el uso de la media geodésica entre un nodo y los demás nodos de la red, y se define a la centralidad de un vértice como la inversa de la longitud promedio de los caminos cortos con los demás vértices de la red. Wasserman y Faust [11] define las ecuaciones para el cálculo de la centralidad de cercanía de un vértice para grafos no dirigidos como sigue:

$$C_C(n_i) = \frac{1}{\sum_{j=1}^{g} d(n_i, n_j)} = \frac{1}{MED_{i \neq j}(d(n_i, n_j))}$$

Donde:

- *g* es el número de nodos en la red
- $d(n_i, n_j)$ es el geodésico entre el nodo n_i y el nodo n_j
- *MED* es la media de las distancias

Para grafos dirigidos se planteó la siguiente ecuación:

$$C_C(n_i) = \frac{g-1}{\sum_{j=1}^{g} d(n_i, n_j)} = \frac{g-1}{MED_{i \neq j}(d(n_i, n_j))}$$

2.5.2.3. Centralidad de intermediación

Existen dos tipos de centralidad de intermediación, la primera está basada en la frecuencia en la que un *nodo* aparece en el geodésico entre dos nodos, es decir, las veces en que se presenta entre un nodo con trayectoria mínima. La siguiente ecuación calcula la intermediación (*Betweenness*) de *v* en las trayectorias de *z* y *y*

$$C_I(v) = \sum_{v \neq y \neq z} \frac{d_{yz}(v)}{d_{yz}}$$

Donde:

- d_{yz} es el número de caminos geodésicos que van de *y* a *z*.
- $d_{yz}(v)$ representa el número de caminos geodésicos de *y* a *z* que cruzan por el vértice *v*.

El segundo cálculo de centralidad de intermediación está basado en la importancia que tiene una arista con respecto a una trayectoria mínima, es decir, las veces en que un enlace se presenta en medio de una trayectoria mínima. La ecuación para calcular la intermediación de aristas es la siguiente:

$$C_I(e) = \sum_{x \neq y} \frac{d_{yz}(e)}{d_{yz}}$$

donde:

- d_{yz} es el número de caminos geodésicos que van de y a *z*.
- $d_{yz}(e)$ representa el número de caminos geodésicos de *y* a *z* que cruzan por la arista *e*.

El cálculo de la intermediación está dado en un tiempo de $O(nm + n\ 2log\ n)$ y en espacio $O(n + m)$.

En la figura 2.5 se observan las centralidades de grado, cercanía e intermediación para un grafo dirigido.

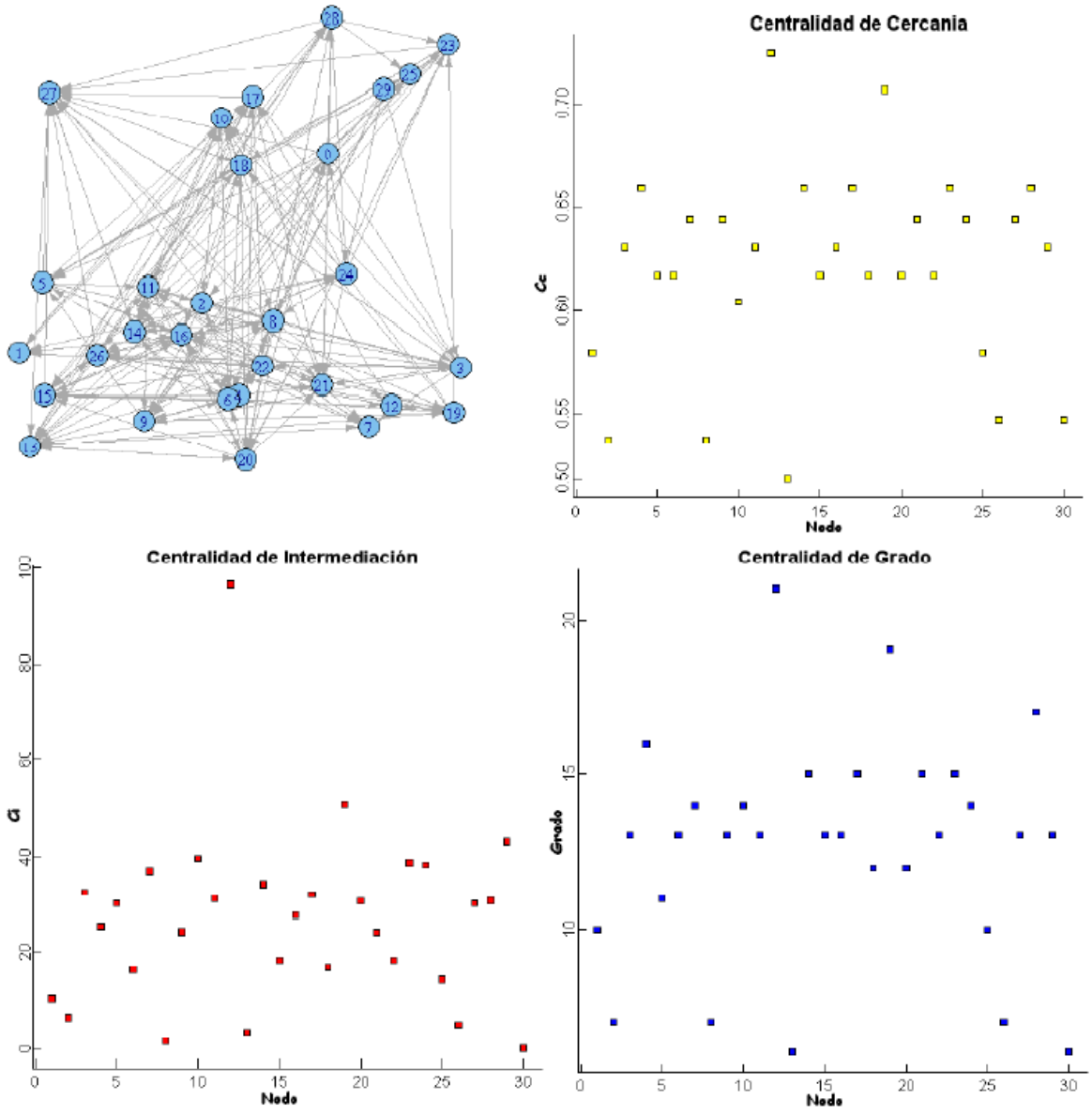


Figura 2.5. Cálculo de las diferentes medidas de centralidad de un grafo dirigido.

2.6. Detección de comunidades

Una de las principales características de los grafos que representan sistemas reales son las comunidades o agrupamientos. Tales agrupaciones o comunidades pueden considerarse como elementos independientes de un grafo, y formalmente como un subcojunto del conjunto inicial. Los métodos para el análisis de redes suelen ser muy costos y más cuando se tienen redes a gran escala, el uso de métodos para detectar comunidades permite reducir en diferentes fracciones a la red original, siempre y cuando se cumplan ciertos criterios sobre como generar las particiones.

2.6.1. Comunidades

Una comunidad es un grupo de vértices y aristas que comparten ciertas propiedades en común e influyen de manera similar con la red. La sociedad ofrece una infinidad de posibles comunidades: familiares, círculos de amistad y de trabajo, ciudades, naciones, entre otras. El problema de detección de comunidades es muy importante para tratar diferentes sistemas reales como son: biológicos, computacionales, económicos, políticos, sociales, etc.

La formalización matemática al problema de la detección de comunidades se le conoce como *agrupamiento de grafos,* el cual tiene sus inicios desde la década de los 70 y que detallaremos en las siguientes secciones.

2.6.2. Agrupamiento de grafos (clustering)

La técnica de *agrupamiento* (o clustering) es la clasificación de objetos en diferentes grupos, o la partición de un conjunto de datos en subconjuntos, de modo tal que cada subconjunto comparte propiedades en común. El objetivo de un agrupamiento consiste en dividir el conjunto de datos en agrupaciones tales que los elementos asignados a un grupo en particular son similares o están conectados en algún sentido.

Existen dos tipos de agrupamientos el global y el local, en el primero se identifican los grupos dentro de la red original de tal manera que los datos en un grupo compartan cierta similitud entre ellos, este tipo de agrupamiento suele ser costoso cuando se maneja mucha información. En el agrupamiento local no es necesario conocer la forma en que se agrupan todos los datos de la red, basta con analizar a un solo elemento y determinar a que grupo pertenece, este tipo de agrupamiento parte de lo microscópico o particular hasta llegar al objetivo.

2.7. Las Redes sociales

Las redes sociales se definen como un conjunto finito de actores (individuos, grupos, organizaciones, comunidades, sociedades, etc) vinculados unos a otros a través de una relación o un conjunto de relaciones sociales. Las redes sociales se apoyan en el Análisis de Redes Sociales (ARS) el cual se centra en tomar las relaciones entre actores como el material sobre el cual se construye y se organiza el comportamiento social de actores. El punto de análisis deja de ser el individuo (egocéntrica) y pasan a serlo las relaciones; proporcionando un conjunto de métodos y técnicas para el estudio formal de las relaciones entre actores.

En las siguientes secciones se presenta una descripción detallada del concepto de redes sociales, las propiedades con las que cuenta, los tipos de modelos de redes, su clasificación en base a su estructura, las técnicas de medición utilizadas y los sistemas que permiten realizar un análisis de estas estructuras.

2.7.1. Origen de las redes sociales

Una red social es una estructura social que está conformada por grupos de individuos, a los cuales se les llama nodos y los cuales están conectados mediante algún tipo de relación (p.ej., amistad, negocios, parentesco, etc). Este tipo de estructuras son frecuentemente utilizadas para modelar una situación social. En 1930 aparece la sociometría como una manera de formalizar las ciencias sociales, donde se utilizaba la estadística para estudiar poblaciones y a la teoría de grafos servía para modelar la relación entre personas.

En años recientes la teoría basada en la estructura de las redes sociales fue retomada, debido al alto número de sistemas en la Web que permiten generar comunidades virtuales que pueden ser representadas como una estructura de red social.

2.7.2. Análisis de redes sociales

El Análisis de Redes Sociales (ARS1) nace en los años 70 con la fundación de la *International Network for Social Network Analysis* (INSNA2), la cual es una asociación profesional para investigadores interesados en el análisis de redes sociales aplicada en diferentes áreas del conocimiento.

El análisis de redes es una aproximación intelectual amplia para identificar las estructuras sociales que emergen de las diversas formas de relación, pero también es un conjunto específico de métodos y técnicas. El ARS se ha desarrollado como herramienta de medición y análisis de las estructuras sociales que surgen de las relaciones entre actores sociales diversos (individuos, organizaciones, naciones, etc).

2.7.3. Propiedades de las redes sociales

Las redes sociales son representadas mediante grafos, y utiliza técnicas de la teoría de grafos para estudiar la estructura de las redes sociales. Sin embargo, existen diferencias en la forma en como se aplican los distintos conceptos de la teoría de grafos sobre el ARS. Las redes sociales adoptan muchas de las propiedades de las redes sociales, en esta sección

estudiamos las diferentes propiedades que presentan las redes sociales en base a su forma, distribución y similaridad entre los conjuntos de nodos y relaciones que existen en la red.

2.7.3.1. Distancia en las redes

La *distancia* es una medición de la teoría de grafos que permite definir propiedades más complejas sobre la posición de los individuos y sobre la estructura de la red. Una trayectoria es el número de enlaces que existen entre dos nodos en el grafo. La distancia geodésica es el número mínimo de enlaces que llevan de un nodo a otro (trayectoria mínima), esta medida es muy utilizada en el ARS, ya que nos permite obtener el camino más eficiente entre dos actores, sin embargo, para el caso de las redes sociales el camino más corto no siempre es el buscado y dependerá del tipo de red social para determinar el tipo de distancia a utilizar.

El *diámetro* de un grafo está definido como el máximo geodésico de todos los vértices dentro de un grafo. El diámetro representa el tamaño del grafo y permite saber que tan grande es.

2.7.3.2. Tipos de interacción

La forma en que interactúan los nodos dentro de un grafo permite establecer un nivel de conexión entre nodos y sus relaciones. A este fenómeno se le llama conectividad y permite reducir las trayectorias entre dos nodos. En un grafo con conectividad alta se puede ir de un nodo a otro con trayectorias más cortas.

La reciprocidad en las redes sociales es un fenómeno presentado en grafos dirigidos, también llamado simetría de vínculos. Este fenómeno se presenta cuando en un grafo existe un enlace que va de A hacia B y uno que va de B hacia A. La simetría de vínculos en un grafo reduce el diámetro de la red e incrementa la conectividad de la red.

2.7.3.3. Coeficiente de agrupamiento

El coeficiente de agrupamiento es una métrica de similaridad, podemos destacar dos coeficientes de agrupamiento:

- El *coeficiente de agrupamiento local,* que mide el nivel de agrupamiento o interconexión de un vértice con sus vecinos, donde el vecindario de un nodo i está formado por el número de nodos que están adyacentes al nodo i, es decir, aquellos nodos con los que se mantiene una relación.

- El *coeficiente de agrupamiento global,* que indica el nivel de agrupamiento de los nodos con respecto a la red total.

2.7.3.4. Cliques

Un clique en la teoría de grafos es un conjunto de vértices en el cual para cada vértice existe una arista que los conecta. Un clique es un subgrafo en el cual cada vértice está conectado a cada otro vértice del grafo, es decir, el subgrafo puede ser considerado como un grafo completo. En la siguiente figura se muestra un grafo completo, si es un subgrafo de una red se dice que es un clique de tamaño 6.

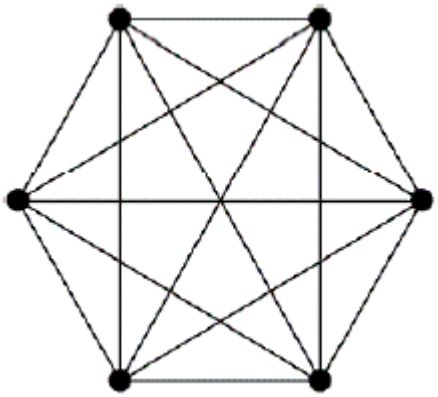


Figura 2.6. Ejemplo de un clique de tamaño 6

2.7.4. Clasificación de Redes Sociales

Actualmente no existe una clasificación que permita especificar los tipos de redes sociales, ya que dependiendo del área de investigación, una red social puede ser clasificada según el enfoque del estudio, pero es necesario contar con una clasificación que permita distinguir a las nuevas redes sociales sin caer en ambigüedades. A continuación proponemos una clasificación de los tipos más comunes de redes sociales.

- **Redes sociales basadas en su tamaño:** Este tipo de redes depende del diámetro de la red, es decir, la distancia mayor entre dos actores en la red. No existe una medida exacta para poder determinar cuando una red social es grande o pequeña, por lo que dicho valor dependerá de lo que se quiera representar con la información.

  - **Redes a pequeña escala:** Este tipo de redes pueden ser analizadas por la mayoría de los sistemas para visualización de redes, y generalmente se

utiliza al conjunto de datos completo para su análisis. Un ejemplo de este tipo de redes sociales es la red formada a partir de la colaboración de investigadores, donde los investigadores son los nodos de la red y los enlaces están formados por los artículos en los que uno o más investigadores participan.

- **Redes a gran escala:** Estas redes no pueden ser analizadas fácilmente y en muchas ocasiones suelen realizarse las mediciones en base a una porción representativa de la red. Aún no existe sistema alguno que permita visualizar este tipo de redes de manera completa. Un ejemplo de estas redes es la formada por los sistemas de correo electrónico, donde las cuentas de usuario son los actores en la red y un enlace es representado por un usuario en la lista de contactos, donde resultaría difícil tratar de representar una red de todas las cuentas de correo que interactúan en la Web.

- **Redes basadas en la evolución:** Este tipo de redes depende de los cambios que sufre la red a través del tiempo. Estas redes pueden ser de cualquier tamaño y tener distintas formas.

  - **Redes Estáticas:** Este tipo de redes no sufre ningún tipo de alteración cuando son sujetas a estudio, por lo que mantienen la misma estructura desde el momento en que son analizadas hasta el final de su estudio.

  - **Redes Dinámicas:** Estas redes sufren cambios en su estructura debido a la incorporación y/o eliminación de nuevos actores y las relaciones entre ellos. Existen muchas redes de este tipo pero la más importante es la propia Web, ya que se encuentra en constante cambio y su tamaño aumenta cada vez más al pasar el tiempo. Diferentes fenómenos se presentan en este tipo de redes y muchos trabajos han sido propuestos como son la predicción de vínculos y los modelos de crecimiento.

- **Redes basadas en su origen:** Este tipo de redes depende de su fuente de datos de origen. Muchas de estas redes pueden representar comunidades virtuales y/o del mundo real.

  - **Redes fuera de línea (Off-line):** Son aquellas en las que las relaciones sociales son establecidas sin la intervención de un medio electrónico, es decir, la administración y conocimiento de las relaciones recae exclusivamente en el conocimiento del individuo sin ayuda de un sistema software que le permita llevar la gestión de contactos. Un ejemplo de este tipo de redes fue la red generada para el caso del presunto suicidio del

científico británico David Kelly [13], dicha red fue generada a partir de documentos del gobierno sobre el caso.

- **Redes en línea (On-line):** Son redes que dependen altamente de medios electrónicos y se mantienen ligadas a los cambios en la tecnología de los sistemas. Ejemplo de estas redes son Facebook, Twitter, Orkut, entre otras.

- **Redes basadas en su topología:** Este tipo de redes depende de la complejidad de la red.

  - **Redes Simples:** Este tipo de redes son estructuras sencillas y pueden ser fácilmente analizadas con conceptos básicos de la teoría de grafos.

  - **Redes Complejas:** El estudio de este tipo de redes está basada en el estudio empírico de las redes de mundo real, se dice que son complejas por que presentan propiedades no triviales (p.ej., calcular el coeficiente de agrupamiento y la centralidad de la red). Ejemplo de este tipo de redes son las redes aleatorias, las de mundo pequeño y las libres de escala.

2.7.5. Redes Sociales en línea

Recientemente las redes sociales en línea han adquirido gran popularidad y se encuentran dentro los sitios más importantes en la Web. Sin embargo, este tipo de redes se ha convertido en uno de los principales generadores de tráfico en Internet, donde investigaciones recientes de la empresa CISCO revelan que más del 30% del tráfico en las redes es generado por este tipo de sistemas y se pronostica que para el año 2013 el tráfico generado por estas redes aumentará en un 500%.

Según datos estadísticos la cantidad de personas en el mundo es aproximadamente 6.8 mil millones, donde países como China con 1.3 mil millones, India con 1.1 mil millones y Estados Unidos con 295 millones de habitantes son los países con mayor densidad de población. Por otro lado sitios como Facebook, Twitter y Youtube promedian 280 millones de usuarios cada uno. En base a esto podemos decir que las redes sociales en línea tienen condiciones para ser consideradas como redes de mundo real, aunque este tipo de relaciones son impersonales y poco confiables.

2.7.6. Representación de las Redes Sociales

En una red social cada nodo, también llamado actor o vértice, puede ser representado como un individuo o grupo de individuos. Una arista, también llamada relación o vínculo, conecta a dos nodos y representa el enlace entre dos individuos en una red social. Las redes pueden tener pocos o muchos actores y uno o más tipos de relaciones entre pares de actores. El ARS usa dos tipos de herramientas de las matemáticas para representar información de las relaciones entre actores sociales, los grafos y las matrices, una razón para usar métodos formales para representar redes sociales es que la representación matemática permite utilizar computadoras para dicho análisis.

El ARS usa un tipo de representación gráfica de estas estructuras que consiste de puntos (o nodos) para representar actores y líneas (o aristas) para representar lazos o relaciones.

Sin embargo, cuando los sociólogos empezaron con el análisis de redes tomaron esta representación y la llamaron sociograma. Matemáticamente los sociogramas son conocidos como grafos dirigidos. En la siguiente figura se aprecia una red social como un grafo dirigido o sociograma.

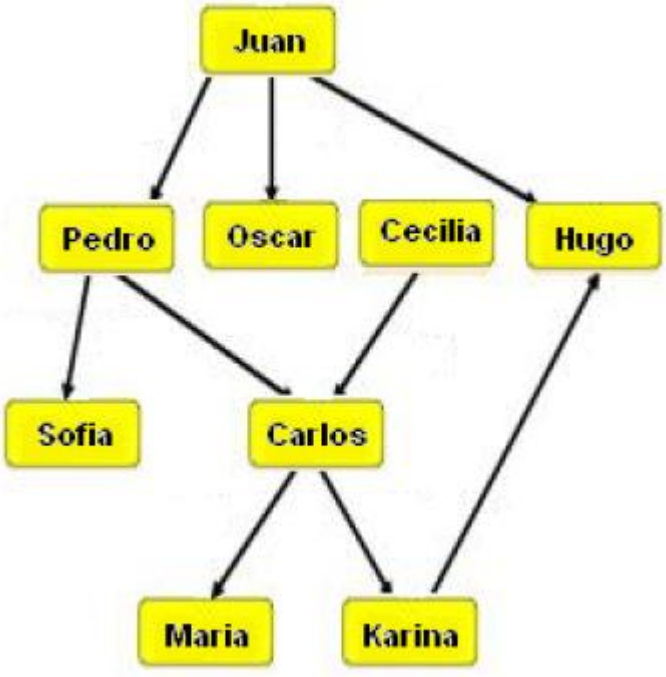


Figura. 2.7. Ejemplo de una red social como un grafo dirigido o sociograma.

2.7.7. Estructura de las Redes Sociales

La estructura de una red social está basada en individuos u organizaciones, que están conectadas por una o más relaciones, tales como, amistad, contactos profesionales, parentesco, entre otros. Las redes sociales en términos de la teoría de grafos representa una estructura basada en grafos complejos. Las redes sociales pueden ser representadas como grafos dirigidos, no dirigidos, bipartidos y completos.

2.7.7.1. El componente gigante de las redes sociales

En el año 2000 se publica un artículo [14] sobre las características de laWeb, entre sus principales descubrimientos encontraron que un número pequeño de nodos presentaba alto agrupamiento, a este fenómeno le llamaron Componente Fuertemente Conectado (CFC) o Componente Gigante (CG) y también demostraron que existía la presencia de más de uno. Esto permitió diferenciar ampliamente a las redes aleatorias de las redes de mundo real, ya que las aleatorias presentan una distribución homogénea entre sus nodos, es decir, no hay presencia de CFC's y los nodos presentan una distribución de grado similar.

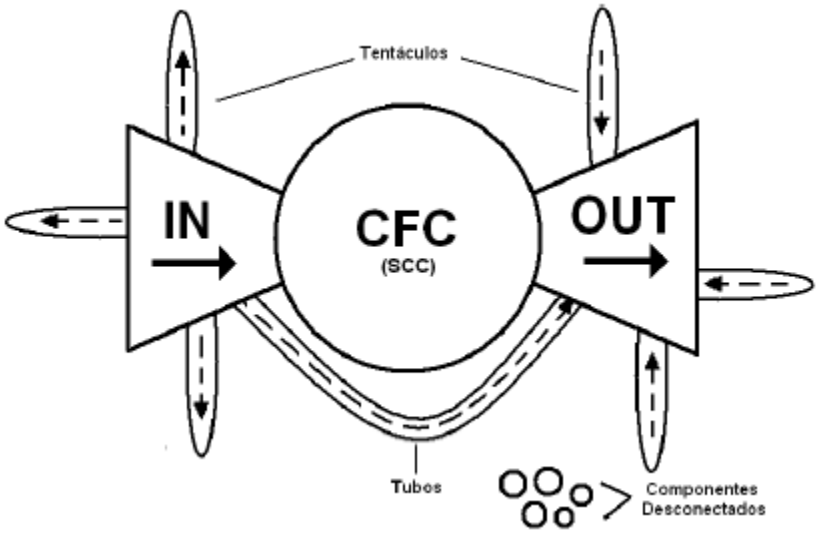


Figura. 2.8. Conectividad de la Web, donde el CFC es el núcleo de la red y el conjunto de nodos en IN son los nodos que entran en el CFC y el conjunto de nodos OUT son los que salen del CFC. Los tentáculos y los tubos representan a los nodos externos al núcleo CFC.

Un CFC es definido como un conjunto de nodos tal que para cualquier par de nodos *u* y *v* en el conjunto hay un camino entre *u* a *v*. La figura 2.8 muestra la conectividad de la

Web, donde el CFC es el núcleo de la red y el conjunto de nodos en IN son los nodos que entran en el CFC y el OUT es el conjunto de nodos que salen del CFC, los tentáculos representan a los nodos que conectan a un CFC con otros tipo de formaciones, ya sea otro CFC o islas de nodos.

2.8. La red social twitter

Twitter es un microblogging, con sede en San Francisco (California), con filiales en San Antonio (Texas) y Boston (Massachusetts) en Estados Unidos. Twitter, Inc. fue creado originalmente en California, pero está bajo la jurisdicción de Delaware desde 2007. Desde que Jack Dorsey lo creó en marzo de 2006, y lo lanzó en julio del mismo año, la red ha ganado popularidad mundialmente y se estima que tiene más de 200 millones de usuarios, generando 65 millones de tweets al día y maneja más de 800.000 peticiones de búsqueda diarias. Ha sido apodado como el "SMS de Internet".

La red permite mandar mensajes de texto plano de corta longitud, con un máximo de 140 caracteres, llamados tweets, que se muestran en la página principal del usuario. Los usuarios pueden suscribirse a los tweets de otros usuarios – a esto se le llama "seguir" y a los suscriptores se les llaman "seguidores" o tweeps ('Twitter' + 'peeps'). Por defecto, los mensajes son públicos, pudiendo difundirse privadamente mostrándolos únicamente a seguidores. Los usuarios pueden twittear desde la web del servicio, desde aplicaciones oficiales externas (como para smartphones), o mediante el Servicio de mensajes cortos (SMS) disponible en ciertos países. Si bien el servicio es gratis, acceder a él vía SMS comporta soportar tarifas fijadas por el proveedor de telefonía móvil.

Twitter dispone de una API pública que se describe en el siguiente subapartado y permite acceder a su información de una forma práctica y poca costosa.

2.9. La API de Twitter

La API de Twitter es relativamente potente y extensa, además de estar abundantemente documentada. Sus principales inconvenientes son las limitaciones que impone Twitter sobre su uso. La API está compuesta por tres módulos diferentes: dos que siguen el paradigma de programación REST (REST API y Search API) y otro más reciente basado en “streaming”. Esta división se debe a motivos históricos, ya que todos los módulos han sido desarrollados de forma independiente y más tarde integrados, o aún en proceso, bajo una única API.

Los métodos de la REST de Twitter permiten a los desarrolladores acceder al núcleo de los datos del servicio de red social. Esto significa que cualquier acción posible a través

del portal web, puede realizarse también mediante la API. Adicionalmente la Search API ofrece a los desarrolladores métodos para interactuar con Twitter Search e información relativa a las tendencias en Twitter. Por su parte, la API de "streming" proporciona un elevado volumen de acceso a los "tweets" en tiempo prácticamente real, y de forma filtrada. Desde el punto de vista del desarrollador la división de Twitter en tres módulos es transparente, excepto por las limitaciones de uso de cada módulo y sus formatos de representación de la información.

Actualmente el proceso de integración de los módulos continúa y se están comenzando a aplicar medidas como la unificación de las limitaciones de uso de cada uno de ellos. Es un proceso dinámico y en constante avance, que puede seguirse a través del perfil de Twitter *twitterapi*.

3. ANÁLISIS DEL SISTEMA

En este capitulo se discuten las principales cuestiones relacionadas con el análisis para el desarrollo de una aplicación que permita cargar un grafo a partir de una red de amigos en Twitter, empezamos por definir que tipo de grafo es el que se debe de cargar, luego se plasman todas las funciones que debe cumplir el sistema y se muestran los diferentes casos de uso.

3.1. Análisis del grafo

Vista la particularidad de Twitter respecto a la forma de gestionar las relaciones entre las entidades, en las que un usuario sigue o lo siguen otros usuarios, vamos a obtener un *grafo dirigido no ponderado*. La figura 3.1 muestra un grafo obtenido a partir del usuario *@wilmata*

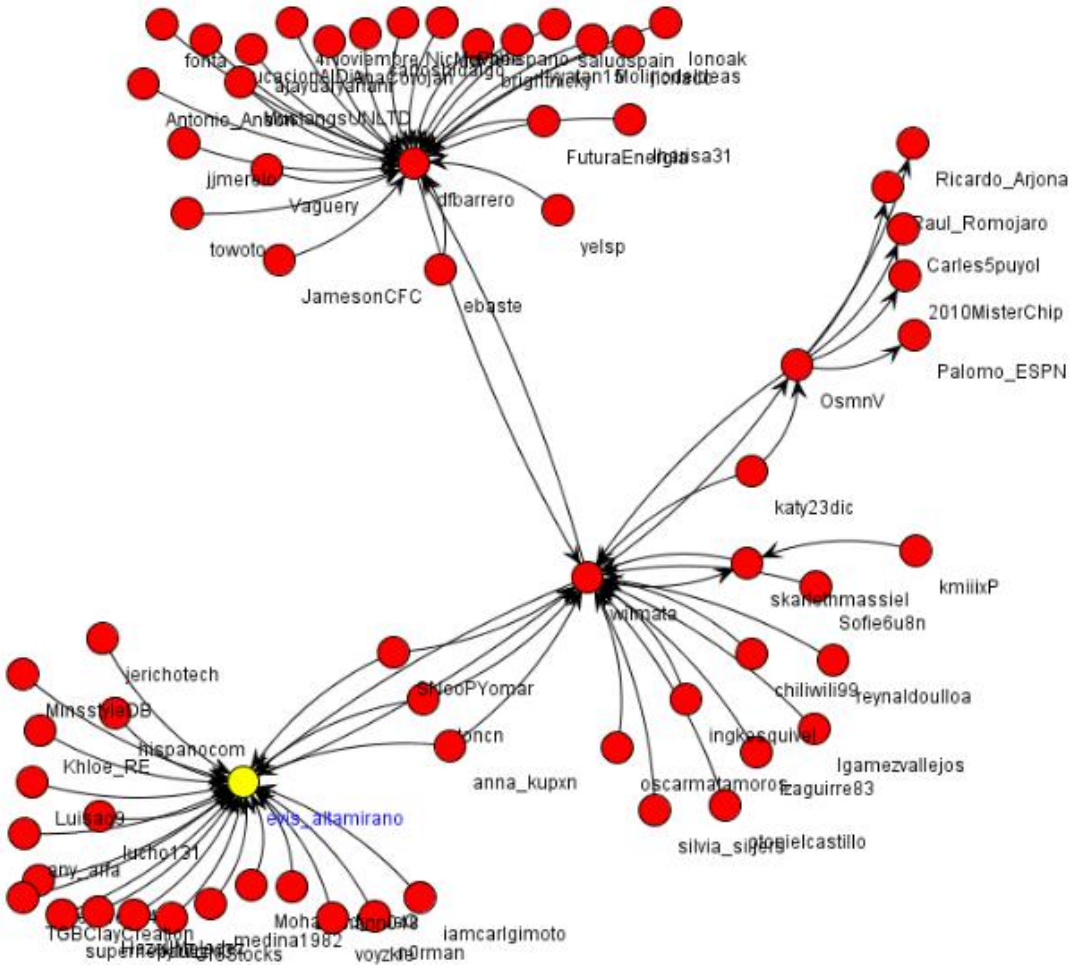


Figura 3.1. Grafo cargado de Twitter a partir del usuario @wilmata

3.2. Funciones del sistema

A continuación se listan todas las funciones que debe cumplir el sistema:

- **Abrir:** esta función permite abrir y cargar un grafo desde un archivo previamente guardado en disco.

- **Guardar y guardar como:** estas funciones permiten guardar un grafo en un archivo con formato XML.

- **Agregar usuario al grafo:** esta función permite buscar un usuario en Twitter y añadirlo al grafo.

- **Generar grafo a partir de un usuario:** esta función permite generar el grafo a partir de uno de los nodos cargados en el grafo, recibe como parámetros el nivel de profundidad con el que se va cargar a los usuarios y el número de usuarios (follores y followings) por cada nivel.

- **Mostrar métricas del grafo:** esta es una de las funciones más importantes de la aplicación, por medio de la cual se mostrará un diálogo al usuario con información relevante a las métricas del grafo. Las métricas a mostrar son las siguientes:

  - El número de vértices
  - El número de enlaces
  - Distancia promedio de la red
  - Diámetro de la red
  - Eficiencia global
  - Media armónica
  - Degree medio
  - InDegree medio
  - OutDegree medio

  También se podrá mostrar información detallada por cada uno de los vértices en forma tabular.

- **Mostrar métricas de un vértice:** esta es otra función muy útil, puesto que con esta función se mostrará al usuario métricas relacionadas con uno de los vértices. Las métricas a mostrar son:

Métricas a partir del grafo cargado

- La distancia promedio para llegar al vértice desde cualquier otro vértice.
- La distancia promedio para alcanzar otro vértice partiendo de dicho vértice.
- Centralidad de grado
- Centralidad de grado de entrada
- Centralidad de grado de salida
- Centralidad de cercanía
- Centralidad de intermediación
- Coeficiente de agrupamiento

Métricas a partir de la información del usuario en Twitter

- El número de followers
- El número de followings
- La cantidad de Tweets publicados
- La cantidad de Tweets marcados como favoritos

- **Obtener distancia:** permite al usuario obtener la distancia que hay desde un nodo a otro a partir del grafo cagado.

- **Obtener Followings:** permite cargar al grafo una cantidad de followings de un determinado usuario representado por un nodo.

- **Obtener Followers:** permite cargar al grafo una cantidad de followers de un determinado usuario representado por un nodo.

- **Ver perfil de un usuario:** muestra en un dialogo información del perfil de Twitter del usuario seleccionado, tales como:

  - Nombre de usuario de Twitter
  - Nombre real
  - Su ubicación
  - El número de Tweets publicados y el contenido de los Tweets más recientes publicados
  - El número de followers y followings

- **Eliminar vértice:** permite borrar un vértice del grafo y sus respectivas relaciones o enlaces.

- **Eliminar enlace:** permite borrar del grafo una relación o enlace que une dos vértices en el grafo.

3.3. Casos de uso

Ya que todos los requisitos de nuestro sistema serán funcionales se ha decidido mostrar estos requisitos directamente a través de los casos de uso. En el proyecto se han detectado un total de 13 casos de usos que se muestran en la siguiente figura:

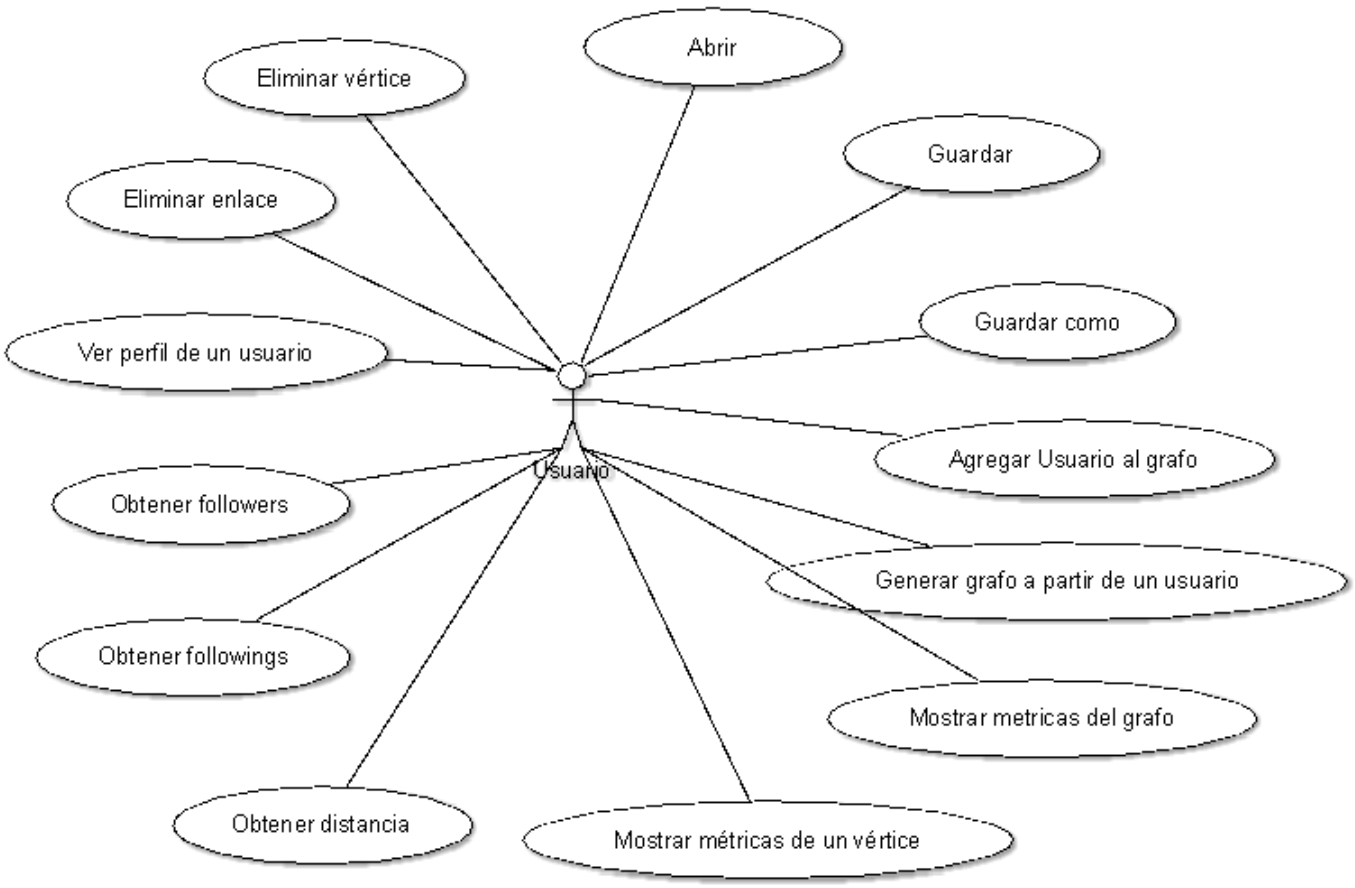


Figura 3.2. Casos de uso

3.3.1. Descripción de casos de uso

Caso de uso 1: Abrir

Nombre:	Abrir
Descripción:	Este caso de uso refleja los pasos que se llevan a cabo para que un usuario pueda abrir un grafo previamente almacenado en disco.
Requerimiento	
Precondición	Se debe estar ejecutando el programa
Flujo Normal:	
Usuario	**Sistema**
1 El usuario debe seleccionar la orden "Open…" del menú "File" **3** Selecciona el archivo que desea abrir.	2 Muestra el diálogo abrir. 4 Carga el grafo almacenado en el archivo seleccionado.
Excepciones:	En caso de error en la carga del archivo se informa al usuario con un mensaje.
Poscondición:	Después de ejecutarse este caso de uso, se tendrá cargado un grafo que que fue previamente almacenado en disco, siempre y cuando nunca se produzca algún error.
Puntos de extensión:	

Caso de uso 2: Guardar

Nombre:	Guardar
Descripción:	Este caso de uso refleja los pasos que se deben llevar a cabo para que un usuario pueda guardar un grafo cargado en el sistema.
Requerimiento	
Precondición	Se debe tener un grafo cargado en el sistema.
Flujo Normal:	
Usuario	**Sistema**
1 El usuario debe seleccionar la orden “Save” del menú “File”	2 Si el grafo ya fue guardado anteriormente se procede a guardar, si no se muestra un diálogo para que el usuario elija dónde y con qué nombre guardará el grafo.
Excepciones:	En caso de error al guardar el archivo se informa al usuario con un mensaje.
Poscondición:	Después de ejecutarse este caso de uso, el grafo ha sido guardado.
Puntos de extensión:	

Caso de uso 3: Guardar como

Nombre:	Guardar como
Descripción:	Este caso de uso refleja los pasos que se deben llevar a cabo para que un usuario pueda guardar un grafo cargado en el sistema indicando siempre dónde y con qué nombre se guardará
Requerimiento	
Precondición	Se debe tener un grafo cargado en el sistema.
Flujo Normal:	
Usuario	**Sistema**
1 El usuario debe seleccionar la orden "Save as…" del menú "File" **3** Selecciona la ubicación y el nombre del archivo con que se guardara el grafo	2 Muestra un diálogo para que el usuario elija donde y con que nombre guardará el grafo. 4 Guarda el grafo en la ubicación y con el nombre especificado.
Excepciones:	En caso de error al guardar el archivo se informa al usuario con un mensaje.
Poscondición:	Después de ejecutarse este caso de uso, el grafo ha sido guardado.
Excepciones:	
Puntos de extensión:	

Caso de uso 4: Agregar usuario al grafo

Nombre:	Agregar usuario al grafo
Descripción:	Este caso de uso refleja los pasos que se deben llevar a cabo para agregar un nodo al grafo dado el nombre de un usuario de Twitter
Requerimiento	
Precondición	Se debe tener un grafo cargado en el sistema.
Flujo Normal:	
Usuario	**Sistema**
1 El usuario debe seleccionar la orden "Add user…" del menú "Graph" **3** Escribe el nombre del usuario	2 Solicita el nombre del usuario a agregar al grafo. 4 Verifica que el usuario no esté presente en el grafo. 5 Busca al usuario en la red social Twitter y si lo encuentra lo añade al grafo. 6 Por último busca relaciones con los demás usuarios que ya están en el grafo.
Excepciones:	Pueden ocurrir las siguientes excepciones: 1. No halla conexión a internet 2. El servidor de Twitter no esté disponible
Poscondición:	
Puntos de extensión:	

Caso de uso 5: Generar grafo a partir de un usuario

Nombre:	Generar grafo a partir de un usuario
Descripción:	Este caso de uso refleja los pasos que se deben llevar a cabo para generar un subgrafo a partir de un usuario ya existente en el grafo
Requerimiento	
Precondición	Se debe tener un grafo cargado en el sistema.
Flujo Normal:	
Usuario	**Sistema**
1 El usuario debe hacer click derecho en el nodo del usuario del que desea obtener su grafo; selecciona la orden "Generate Graph"	2 Muestra un diálogo al usuario solicitando cuantos niveles se desean obtener, y cuantas personas por nivel. 3 Genera el subgrafo a partir de la información proporcionada.
Excepciones:	Pueden ocurrir las siguientes excepciones: 1. No halla conexión a internet. 2. El servidor de Twitter no esté disponible. 3. Se agoten el número de peticiones.
Poscondición:	
Puntos de extensión:	

Caso de uso 6: Mostrar métricas del grafo

Nombre:	Mostrar métricas del grafo.
Descripción:	Este caso de uso refleja los pasos que se deben llevar a cabo para mostrar las distintas métricas relacionadas con el grafo.
Requerimiento	
Precondición	Se debe tener un grafo cargado en el sistema.
Flujo Normal:	
Usuario	**Sistema**
1 El usuario debe seleccionar la orden "Metrics…" del menú "Analysis"	2 El sistema muestra las diferentes métricas relacionadas con el grafo; tales como: el número de nodos y enlaces, distancia promedio entre los nodos, diámetro de la red, coeficiente de agrupamiento global (eficiencia global), media armónica, grado medio de los nodos, grado medio de entrada y grado medio de salida.
Excepciones:	
Poscondición:	
Puntos de extensión:	

Caso de uso 7: Mostrar métricas de un vértice

Nombre:	Mostrar métricas de un vértice.
Descripción:	Este caso de uso refleja los pasos que se deben llevar a cabo para mostrar las distintas métricas relacionadas con un vértice.
Requerimiento	
Precondición	Se debe tener un grafo cargado en el sistema.
Flujo Normal:	
Usuario	**Sistema**
1 El usuario debe hacer click derecho en el nodo del usuario del que desea obtener sus métricas; selecciona la orden "Metrics…"	2 El sistema muestra las diferentes métricas relacionadas con el vértice; tales como: distancia promedio de entrada y salida, grado del nodo, grado de entrada y salida, centralidad de cercanía, centralidad de intermediación, coeficiente de agrupamiento local, tmbien de mostrar información de Twitter, como el número de followings, followers, Tweets y favoritos.
Excepciones:	
Poscondición:	
Puntos de extensión:	

Caso de uso 8: Obtener distancia

Nombre:	Obtener distancia.
Descripción:	Este caso de uso refleja los pasos que se deben llevar a cabo para calcular la distancia que hay desde un nodo a otro.
Requerimiento	
Precondición	Se debe tener un grafo cargado en el sistema.
Flujo Normal:	
Usuario	**Sistema**
1 El usuario debe hacer clic derecho sobre el nodo desde donde queremos obtener la distancia. Selecciona la opción "Get distance from here…" **2** El usuario debe hacer clic derecho sobre el nodo hasta donde queremos obtener la distancia. Selecciona la opción "Get distance to here…"	3 El sistema obtiene la ruta que va desde el primer nodo seleccionado hasta el segundo, muestra mediante un mensaje la distancia que hay entre ellos así como ruta que atraviesa. 4 Selecciona los nodos y enlaces que forman parte de la ruta calculada.
Excepciones:	
Poscondición:	
Puntos de extensión:	

Caso de uso 9: Obtener followings

Nombre:	Obtener followings.
Descripción:	Este caso de uso refleja los pasos que se deben llevar a cabo para obtener followings a partir de un nodo seleccionado.
Requerimiento	
Precondición	Se debe tener un grafo cargado en el sistema.
Flujo Normal:	
Usuario	**Sistema**
1 El usuario debe hacer click derecho en el nodo del usuario del que desea obtener sus followings; selecciona la orden "Generate Next Followings…"	2 Obtiene followings del usuario seleccionado y los añade al grafo.
Excepciones:	Pueden ocurrir las siguientes excepciones: 1. No halla conexión a internet 2. El servidor de Twitter no esté disponible 3. Se agoten el número de peticiones.
Poscondición:	
Puntos de extensión:	

Caso de uso 10: Obtener followers

Nombre:	Obtener followers.
Descripción:	Este caso de uso refleja los pasos que se deben llevar a cabo para obtener followers a partir de un nodo seleccionado.
Requerimiento	
Precondición	Se debe tener un grafo cargado en el sistema.
Flujo Normal:	
Usuario	**Sistema**
1 El usuario debe hacer click derecho en el nodo del usuario del que desea obtener sus followers; selecciona la orden "Generate Next Followers…"	2 Obtiene followers del usuario seleccionado y los añade al grafo.
Excepciones:	Pueden ocurrir las siguientes excepciones: 1. No halla conexión a internet 2. El servidor de Twitter no esté disponible 3. Se agoten el número de peticiones.
Poscondición:	
Puntos de extensión:	

Caso de uso 11: Ver perfil de un usuario

Nombre:	Ver perfil de un usuario.
Descripción:	Este caso de uso refleja los pasos que se deben llevar a cabo para el perfil de un usuario a través de uno de los nodos del grafo.
Requerimiento	
Precondición	Se debe tener un grafo cargado en el sistema.
Flujo Normal:	
Usuario	**Sistema**
1 El usuario debe hacer click derecho en el nodo del usuario del que desea ver su perfil; selecciona la orden "View Twitter profile…"	2 Muestra un diálogo con información relevante al usuario; tales como: número de tweets, número de followings y followers, últimas publicaciones entre otras.
Excepciones:	Pueden ocurrir las siguientes excepciones: 1. No halla conexión a internet 2. El servidor de Twitter no esté disponible 3. Se agoten el número de peticiones.
Poscondición:	
Puntos de extensión:	

Caso de uso 12: Eliminar vértice

Nombre:	Eliminar vértice.
Descripción:	Este caso de uso refleja los pasos que se deben llevar a cabo para eliminar un vértice del grafo
Requerimiento	
Precondición	Se debe tener un grafo cargado en el sistema.
Flujo Normal:	
Usuario	**Sistema**
1 El usuario debe hacer click derecho en el nodo del usuario que desea eliminar; selecciona la orden "Delete vertex"	2 Elimina el vértice seleccionado y sus enlaces.
Excepciones:	
Poscondición:	
Puntos de extensión:	

Caso de uso 13: Eliminar enlace

Nombre:	Eliminar enlace.
Descripción:	Este caso de uso refleja los pasos que se deben llevar a cabo para eliminar un enlace del grafo.
Requerimiento	
Precondición	Se debe tener un grafo cargado en el sistema.
Flujo Normal:	
Usuario	**Sistema**
2 El usuario debe hacer click derecho en el enlace que desea eliminar; selecciona la orden "Delete Edge"	3 Elimina el enlace seleccionado.
Excepciones:	
Poscondición:	
Puntos de extensión:	

4. DISEÑO DEL SISTEMA

En cuanto al diseño de la aplicación podemos decir que está dividida en 3 grandes partes o capas de software, la primera podemos identificarla como *capa de acceso a Twitter*, que proporciona mecanismos para obtener la información necesaria desde los servidores de Twitter; la segunda *capa de gestión del grafo*, consta de una serie de clases que permiten la creación y manipulación de grafos, así también el calculo de las diferentes métricas que podemos obtener; por último, tenemos la *capa de interfaz de usuario*, que consta de una serie de formularios, diálogos y vistas o controles con los que el usuario podrá interactuar con la aplicación.

Cada una de estas capas de software está compuesta por diferentes paquetes, que a su vez contienen clases, interfaces, enumerados que describiremos en los siguientes apartados.

4.1. Capa de acceso a Twitter

La capa de acceso a Twitter está compuesta únicamente por un paquete llamado *CTwitter*, que a su vez contiene una clase con el mismo nombre.

La clase *CTwitter* permite hacer peticiones de datos a Twitter, se apoya en la librería *Twitter4J* que a su vez, utiliza la API http de Twitter para realizar las peticiones. Esta clase sólo tiene un dato miembro, que representa una instancia de Twitter en nombre de quien se pedirán los datos; y como funciones miembros podemos destacar las siguientes:

- **getFollowings:** obtiene *followings* de un usuario dado, recibe dos parámetros, uno es una cadena que indica el nombre del usuario al que se deben obtener los followings; y el otro el número de followings que se desean obtener (si no se especifica se obtienen todos los followings).

- **getFollowers:** obtiene *followers* de un usuario dado, recibe dos parámetros, uno es una cadena que indica el nombre del usuario al que se deben obtener los followings; y el otro el número de followers que se desean obtener (si no se especifica se obtienen todos los followers).

- **getUserByName:** obtiene una instancia de usuario de Twitter dado su nombre de usuario.

- **searchUsers:** permite realizar búsquedas de usuarios dado un patrón pasado como parámetro.

- **getRecentUserTimeLineSinceId:** este método permite obtener el TimeLine de un usuario, a partir de una publicación dada por un ID.

- **getUserTimeLine:** a través de este método se obtiene el TimeLine de un usuario.

- **existFriendShip:** este método nos permite conocer si existe relación de amistad entre dos usuarios, en concreto indica si un usuario sigue a otro.

La figura 4.1 muestra el diagrama de clases del paquete *CTwitter*.

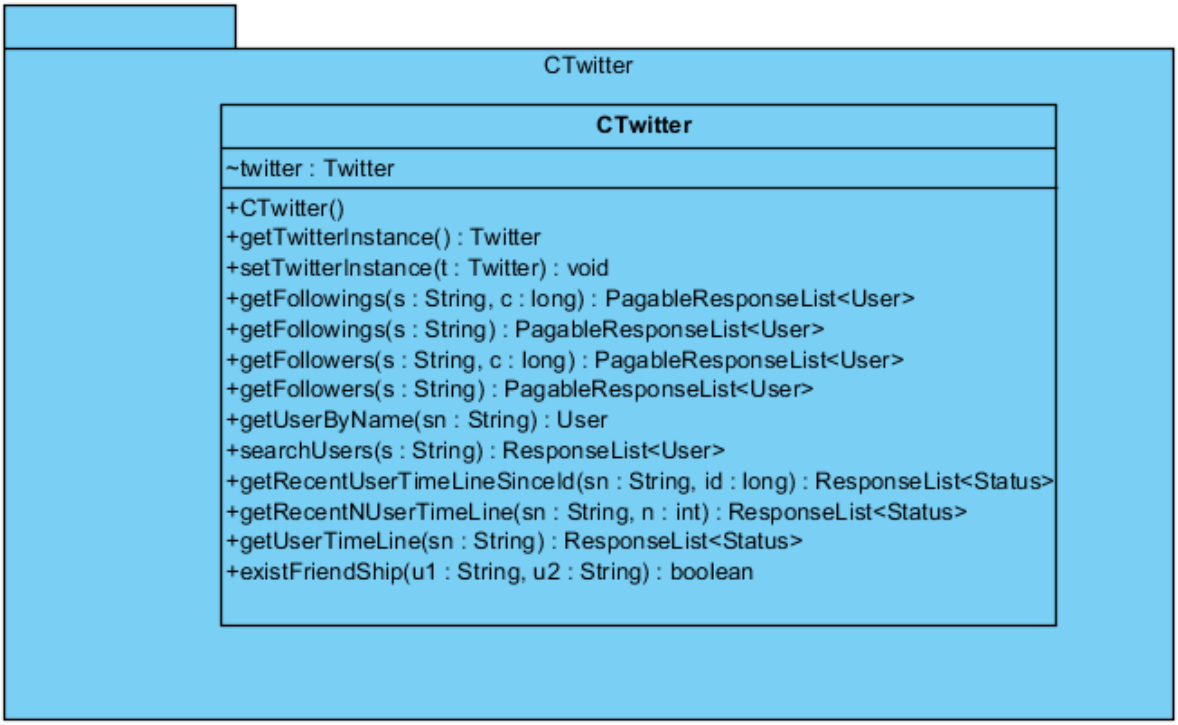


Figura 4.1. Diagrama de clases del paquete *CTwitter*

4.2. Capa de gestión del grafo

Esta capa de software, se ha diseñado para llevar a cabo toda la gestión del grafo, desde la creación, visualización, hasta el cálculo de métricas. Está compuesta por un paquete llamado *TwitterGraph*, que a su vez incluye otros paquetes, *TwitterGraph.Metrics*, *TwitterGraph.Events, TwitterGraph.MouseMenu*.

De estos paquetes los más interesantes son los dos primeros (*TwitterGraph* y *TwitterGraph.Metrics*), y son los que pasaremos a analizar. Por los otros dos paquetes, está bien mencionar que *TwitterGraph.MouseMenu*, contiene una serie de clase e interfaces que permiten la creación de menús contextual para los componentes del grafo (vértices y aristas). En cuanto al paquete *TwitterGraph.Events*, es un paquete que gestiona la generación de dos eventos, el añadir y eliminar usuarios que pueden ser implementados desde otras clases donde se utilice un grafo *TwitterGraph*.

4.2.1. Paquete *TwitterGraph*

Además de los otros paquetes que mencionamos, este contiene una clase nombrada *TwitterGraph*, que es la clase donde se modela el grafo que debe ser cargado desde Twitter. Dos clases *TwitterVertex* y *TwitterEdge* que modelan los vértices y aristas, respectivamente, del grafo. Por último, contiene una clase *TwitterGraphml* que implementa métodos para cargar y guardar grafos en disco en formato XML. La figura 4.2 muestra el diagrama de clases correspondiente al paquete *TwitterGraph*

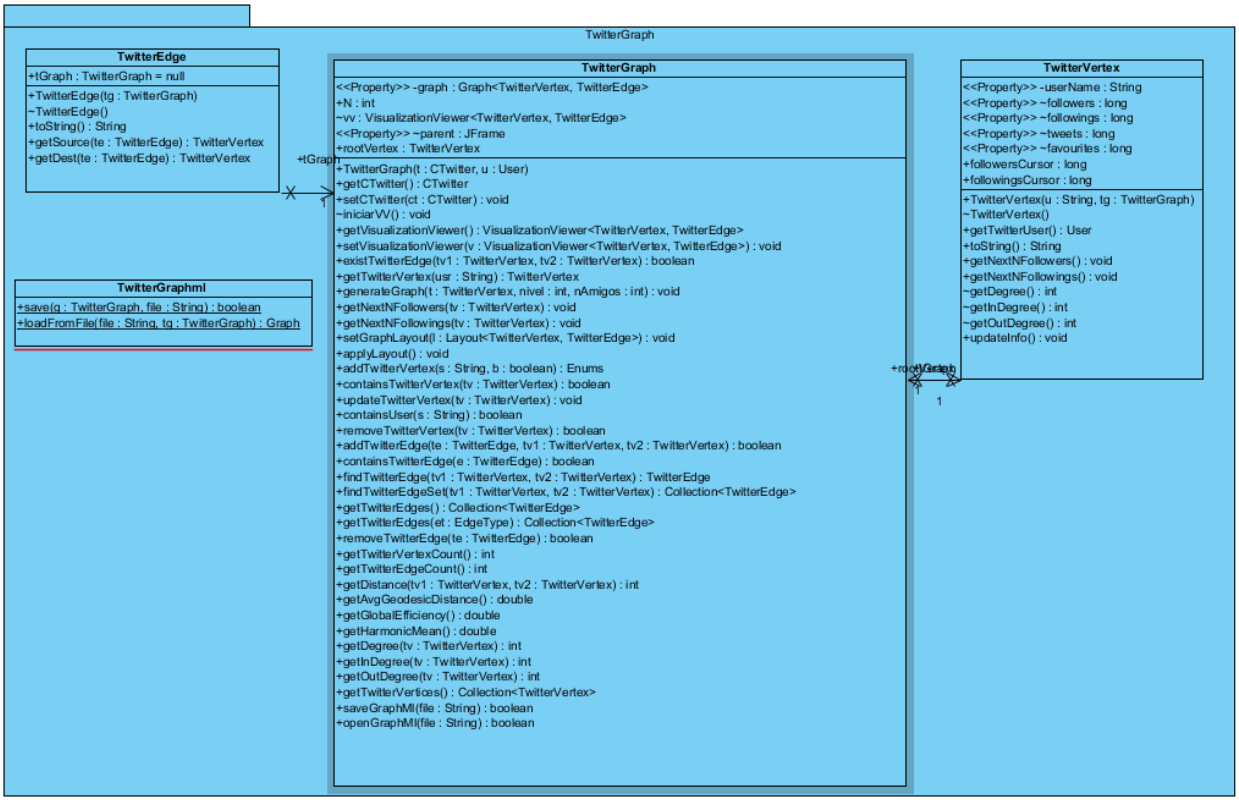


Figura 4.2. Diagrama de clases del paquete *TwitterGraph*

4.2.1.1. La clase *TwitterGraph*

La clase *TwitterGraph* es la encargada de modelar el grafo, a través de esta clase, podemos cargar usuarios desde Twitter, también permite borrar elementos del grafo cargado. Esta clase se apoya en la librería Jung para el modelado del grafo. Podemos destacar los siguientes datos miembros:

- **graph:** representa el grafo propiamente dicho, es de tipo *Graph* que viene incluido en la librería JUNG. Este grafo esta compuesto por nodos instanciados a partir de la clase *TwitterVertex* y aristas por la clase *TwitterEdge*.

- **rootVertex:** representa un nodo raíz del grafo.

- **VisalizationViewer**: este es un objeto que permite la visualización gráfica del grafo, y que debe ser añadido a algotro contenedor, dentro de un dialogo o formulario.

- **twitter:** instancia de *CTwitter* con la que se realizan las peticiones

De sus métodos miembros podemos destacar los siguientes:

- **set(get)Graph:** permiten obtener y establecer el grafo que se desea mantener, o que se mantiene.

- **set(get)VisualizationViewer:** permite obtener y establecer el objeto VisualizationViewer.

- **existTwitterEdge:** este método permite saber si existe una arista que vaya desde un nodo a otro, recibe como parámetro ambos nodos.

- **getTwitterVertex:** permite obtener una instancia de un nodo dado el nombre del usuario al que representa el nodo.

- **generateGraph:** permite obtener el grafo de un usuario representado por un nodo, recibe como parámetro dos valores enteros que indican el nivel de profundidad con que se va a generar el grafo y el número de personas (followings y followers) por nivel.

- **getNextNFollowers:** permite obtener followers de un nodo dado.

- **getNextFollowings:** permite obtener followings de un nodo dado.

- **setGraphLayout:** establece un layout con que se mostrará el grafo.

- **applyLayout:** aplica el layout actual al grafo.

- **addTwitterVertex:** permite añadir un usuario de Twitter al grafo. Es un método sobrecargado que soporta diferentes variantes, por ejemplo, podamos solo pasar el nombre del usuario y añade al usuario y las relaciones que los usuarios ya cargados en el grafo; podemos pasar el nombre del usuario y un booleano que indica si queremos que se carguen las relaciones con los otros usuarios. También podemos pasarle una instancia de *TwitterVertex* para que se cargue al grafo.

- **constainTwitterVertex:** este método permite saber si una determinada instancia de *TwitterVertex* pertenece al grafo.

- **updateTwitterVertex:** a través de este método podemos actualizar la información de un usuario, esto incluye la información relacionada con él (numero de foolowings, followers, tweets y tweets marcados como favorito) y las relaciones que se mantienen con los otros nodos del grafo.

- **containUser:** este método da información si un determinado usuario (dado su nombre de usuario) está cargado en el grafo o no.

- **removeTwitterVertex:** permite eliminar del grafo un nodo dado.

- **getTwitterVertices:** obtiene el conjunto de nodos del grafo.

- **addTwitterEdge:** este método permite añadir aristas al grafo.

- **findTwitterEdge:** permite encontrar en el grafo una arista que vaya directamente desde un nodo dado a otro.

- **getTwitterEdges:** permite obtener todo el conjunto de aristas del grafo.

- **removeTwitterEdge:** elimina una arista del grafo.

- **getTwitterVertexCount:** obtiene el número de nodos en el grafo.

- **getTwitterEdgeCount:** obtiene el número de aristas en el grafo.
- **getDistance:** obtiene la distancia geodésica que hay desde un nodo a otro en el grafo.
- **getAvgGeodesicDistance:** obtiene el promedio de la distancia geodésica de todos los nodos.
- **getGlobalEfficiency:** obtiene la eficiencia global.
- **getHarmonicMean:** calcula la media armónica.
- **getDegree:** obtiene el grado de un nodo.
- **getInDegree:** obtiene el grado de entrada de un nodo.
- **getOutDegree:** obtiene el grado de salida de un nodo
- **saveGraphML:** este método guarda el grafo en un archivo en formato GraphML, se apoya en la clase *TwitterGraphml* (ver 4.2.1.3).
- **openGraphML:** permite cargar un grafo almacenado en disco en formato GraphML. Se apoya en la calse *TwitterGraphml* (ver 4.2.1.3).

4.2.1.2. Las clases *TwitterVertex* y *TwitterEdge*

Para el modelado de los nodos y las aristas podemos crear clases personalizadas, para ello, hemos creado la clase *TwitterVertex*, para representar un nodo en el grafo; y la clase *TwitterEdge* para las aristas.

4.2.1.2.1. La clase *TwitterVertex*

Como se mencionó antes, la clase *TwitterVertex* se utiliza para instanciar nodos de un grafo, cada instancia de esta clase va contener información relacionada con un usuario de Twitter que es representado por un nodo de dicho grafo. Esta información la tenemos a través de los siguientes datos miembros:

- **userName:** campo que contiene el nombre del usuario de Twitter al que representa el nodo.

- **followers:** es un long que representa el número de seguidores en Twitter del usuario

- **followings:** representa la cantidad de personas a las que el usuario sigue en Twitter.

- **tweets:** este campo indica la cantidad de publicaciones realizadas por el usuario.

- **favourites:** este valor representa la cantidad de tweets o publicaciones que el usuario ha marcado como favoritos.

- **TwitterGraph:** mantiene una referencia al grafo al que pertenece el nodo.

Además de estos campos miembros, podemos destacar los siguientes métodos (algunos son redundantes de los que se expusieron en la clase *TwitterGraph*):

- **set(get)UserName:** establece (obtiene) el valor del campo *userName*.

- **getTwitterUser:** obtiene una instancia de usuario de Twitter a partir del nombre del usuario

- **toString:** método utilizado para convertir el nodo en cadena de texto, en realidad sólo se devuelve el nombre del usuario. Este método es utilizado para asignar la etiqueta del nodo en el grafo.

- **updateInfo:** actualiza la información del usuario con la información real en Twitter, sería una especie de sincronización.

Esta clase también contiene los métodos: *getDegree, getOutDegree, getInDegree* vistos en el apartado 4.2.1.1. Además, contiene los métodos get y set para cada uno de los siguientes datos miembros: *followers, followings, tweets y favourites.*

4.2.1.2.2. La clase *TwitterEdge*

Una instancia de la clase *TwitterEdge* representa una arista dentro del grafo, puesto que estas aristas no tienen un peso asignado, por ahora no se almacena ningún tipo de información relacionada con el enlace. Esta clase solo contiene los siguientes métodos:

- **getSource:** este método permite obtener el nodo (*TwitterVertex*) que representa el origen de la arista. Recuerde que se trata de un grafo dirigido.

- **getDest:** este otro método permite obtener el nodo (*TwitterVertex*) que representa el destino de la arista.

- **toString:** sobrecarga del método *toString*, para convertir una arista a cadena, en este caso se devuelve una cadena vacía. Este método se suele utilizar para asignar una etiqueta a las aristas en el grafo.

4.2.1.3. La clase *TwitterGraphml*

Esta clase implementada métodos para guardar y cargar grafos en formato XML, más conocido como *GraphML*. Estos métodos son:

- **save:** este método permite guardar un grafo (*TwitterGraph*) pasado como parámetro, en un archivo en disco cuyo nombre se pasa por parámetro.

- **loadFromFile:** este método permite cargar un grafo guardado en un archivo en un objeto *TwitterGraph.*

4.2.2. El paquete *TwitterGraph.Metrics*

Este paquete está destinado para la obtención de métricas basadas en un grafo (*TwitterGraph*), como vimos en la sección 4.2.1.1, la clase *TwitterGraph* ya contiene el cálculo de ciertas métricas, pero en este paquete se implementan clases que intentan ser más eficientes en cuanto al cálculo de dichas métricas.

Este paquete consta de dos clases para el cálculo de las métricas, *GraphMetrics* y *VertexMetrics*, que detallamos a continuación. En la siguiente figura se muestra su correspondiente diagrama de clases.

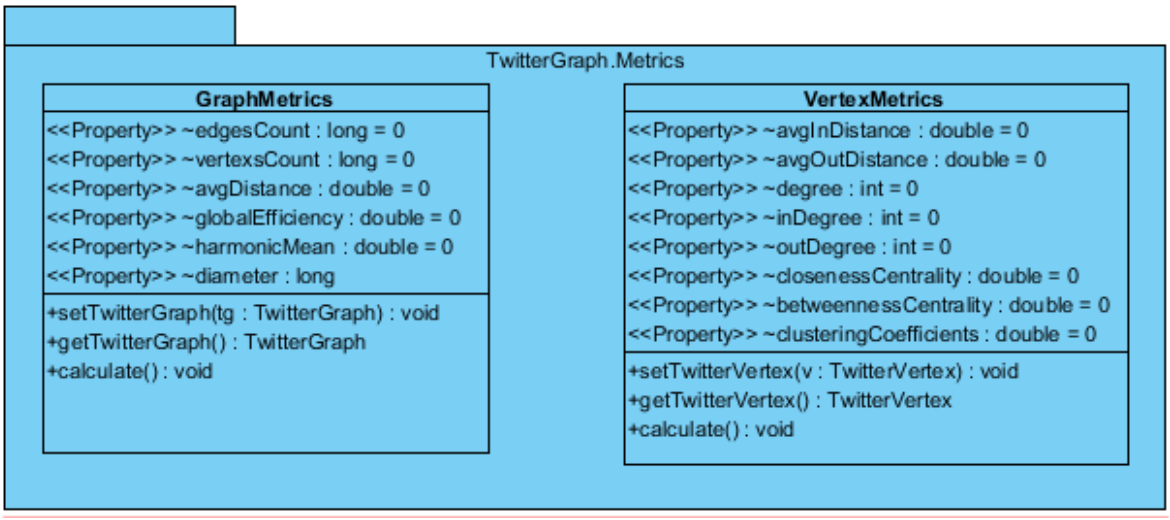


Figura 4.3. Diagrama de clases del paquete *TwitterGraph.Metrics*

4.2.2.1. La clase *GraphMetrics*

Esta clase está destinada para el calculo de las diferentes métricas que podemos obtener del grafo en general, para ello, se declararon los siguientes datos miembros, cada uno de los cuales representa una métrica:

- **edgesCount:** representa el número de aristas que posee un grafo de twitter.
- **vertexsCount:** devuelve el número de nodos que contiene el grafo.
- **avgDistance:** este campo representa la distancia promedio que hay entre los distintos nodos en el grafo.
- **globalEfficiency:** a través de este campo obtenemos el valor de la eficiencia global de la red. (ver 2.5.1)
- **harmonicMean:** representa la media armónica del grafo. (ver 2.5.1)
- **diameter:** este campo almacena el valor del diámetro de la red, es decir, la distancia más larga que hay entre dos nodos del grafo.
- **tGraph:** además se tiene este campo que es de tipo *TwitterGraph* y almacena la instancia del grafo para el cual se van a calcular las métricas.

Para realizar el cálculo de estas métricas, basta con invocar al método ***calculate***, el cual obtiene el valor de cada una de las métricas y las almacena en su respectivo dato miembro; para acceder a cada métrica disponemos de métodos *get* para cada una.

4.2.2.2. La clase *VertexMetrics*

La clase *VertexMetrics* es análoga a la clase *GraphMetrics*, es utilizada para obtener las métricas a partir de un nodo dado del grafo. Contiene los siguientes datos miembros:

- **avgInDistance:** representa la distancia promedio que hay desde cualquier otro nodo.
- **avgOutDistance:** representa el promedio de la distancia que hay desde un nodo a otro nodo de la red.
- **degree:** grado del nodo.
- **inDegree:** grado de entrada del nodo.
- **outDegree:** grado de salida del nodo.
- **closenessCentrality:** centralidad de cercanía (ver 2.5.2.2)
- **betweennessCentrality:** centralidad de intermediación (ver 2.5.2.3)
- **clusteringCoefficients:** representa el coeficiente de agrupamiento para el nodo dado.
- **TwitterVertex:** además de las diferentes métricas, contiene un dato miembro de tipo *TwitterVertex* en el cual se debe asignar (por medio del método *setTwitterVertex*) la referencia a la instancia del nodo al cual se desea obtener sus métricas.

Para calcular las métricas, se tiene el método ***calculate***, que calcula cada una de las métricas y las asigna a su correspondiente dato miembro, luego, para acceder a las métricas basta con invocar al método *get* correspondiente a la métrica que se desea conocer.

4.3. La capa de interfaz de usuario

La capa de interfaz de usuario es la capa de software donde se define la interfaz con la que el usuario interactuará con la aplicación. Esta compuesta por los paquetes *Dialogs* y *TwitterViews.*

4.3.1. El paquete *Dialogs*

El paquete *Dialogs* consta de una serie de clases que representan los formularios y diálogos que se muestran al usuario. Estas clases son:

- **frmInicio:** representa el formulario de inicio, o la ventana principal del programa, en donde se muestra el grafo, menús y diferentes opciones al usuario para utilizar adecuadamente la aplicación.

- **dlgGraphMetrics:** este es un dialogo donde se muestran las diferentes métricas obtenidas a través del grafo.

- **dlgVertexMetrics:** este dialogo muestra las métricas obtenidas a partir de un nodo del grafo. También muestra información de su cuenta de Twitter como el numero de followings y followers, tweets publicados y la cantidad de tweets marcados como favoritos.

- **dlgVertexDetails:** este dialogo consta en si de una tabla donde se muestran todos los nodos del grafo con sus respectivas métricas, muy útil para hacer comparaciones y conocer valores máximos y minimos dentro de la red.

- **dlgGenerateGraph:** es un dialogo que se muestra para solicitar los detalles de como se debe generar un grafo a partir de un nodo dado. En concreto, se solicita al usuario la cantidad de niveles a generar y la cantidad de personas por nivel que se deben cargar.

La figura 4.4 muestra el diagrama de clases del paquete *Dialogs*

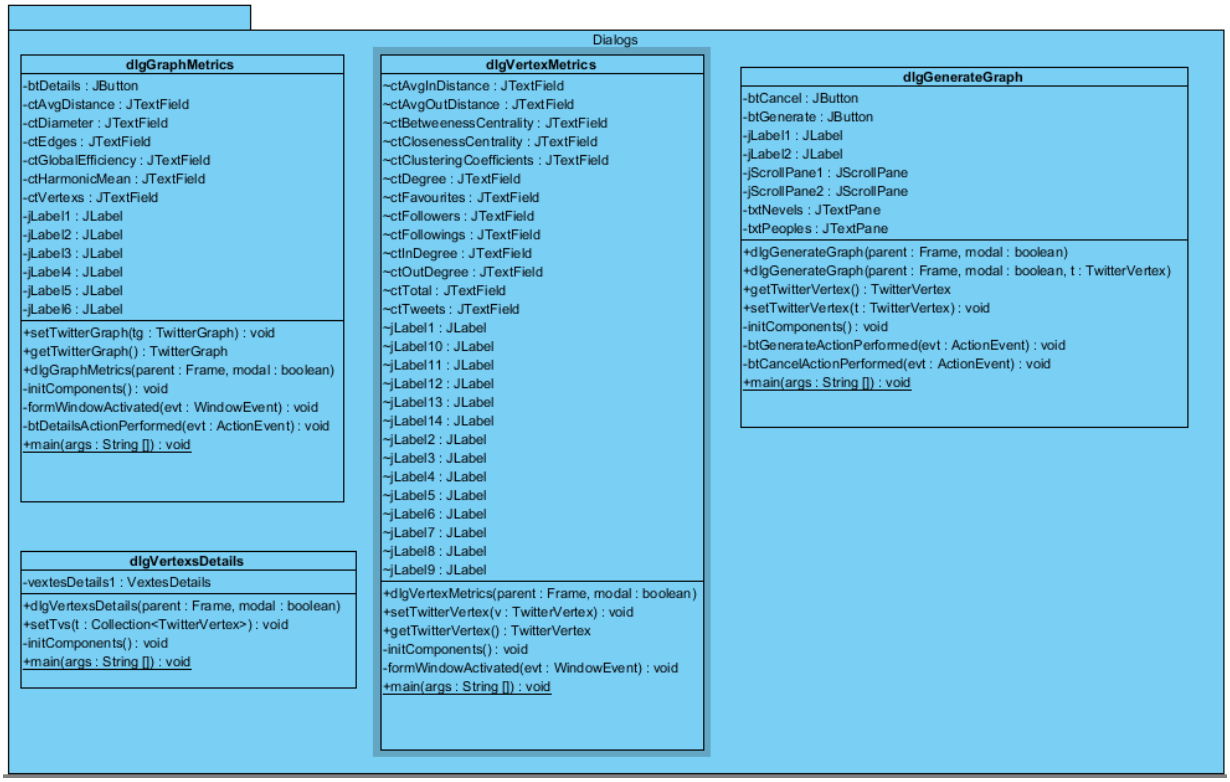


Figura 4.4. Diagrama de clases del paquete *Dialogs*.

4.3.2. El paquete *TwitterViews*

Este paquete define una serie de vistas o controles que permiten visualizar información de un usuario en Twitter. Consta de las siguientes clases que representan controles:

- **StatusView:** es una vista que permite ver una publicación o Tweet.
- **TimeLineView:** muestra el TimeLine, en concreto consta de una colección de *StatusView* que se muestran al usuario en forma de TimeLine.
- **userView:** muestra información relativa a un usuario.
- **usersView:** esta vista se utiliza para cargar una lista de usuarios, por ejemplo, los followers o followings de un determinado usuario.
- **dlgProfile:** este es un dialogo que muestra el perfil de un usuario de Twitter.

Su correspondiente diagrama de clases se muestra en la figura 4.5

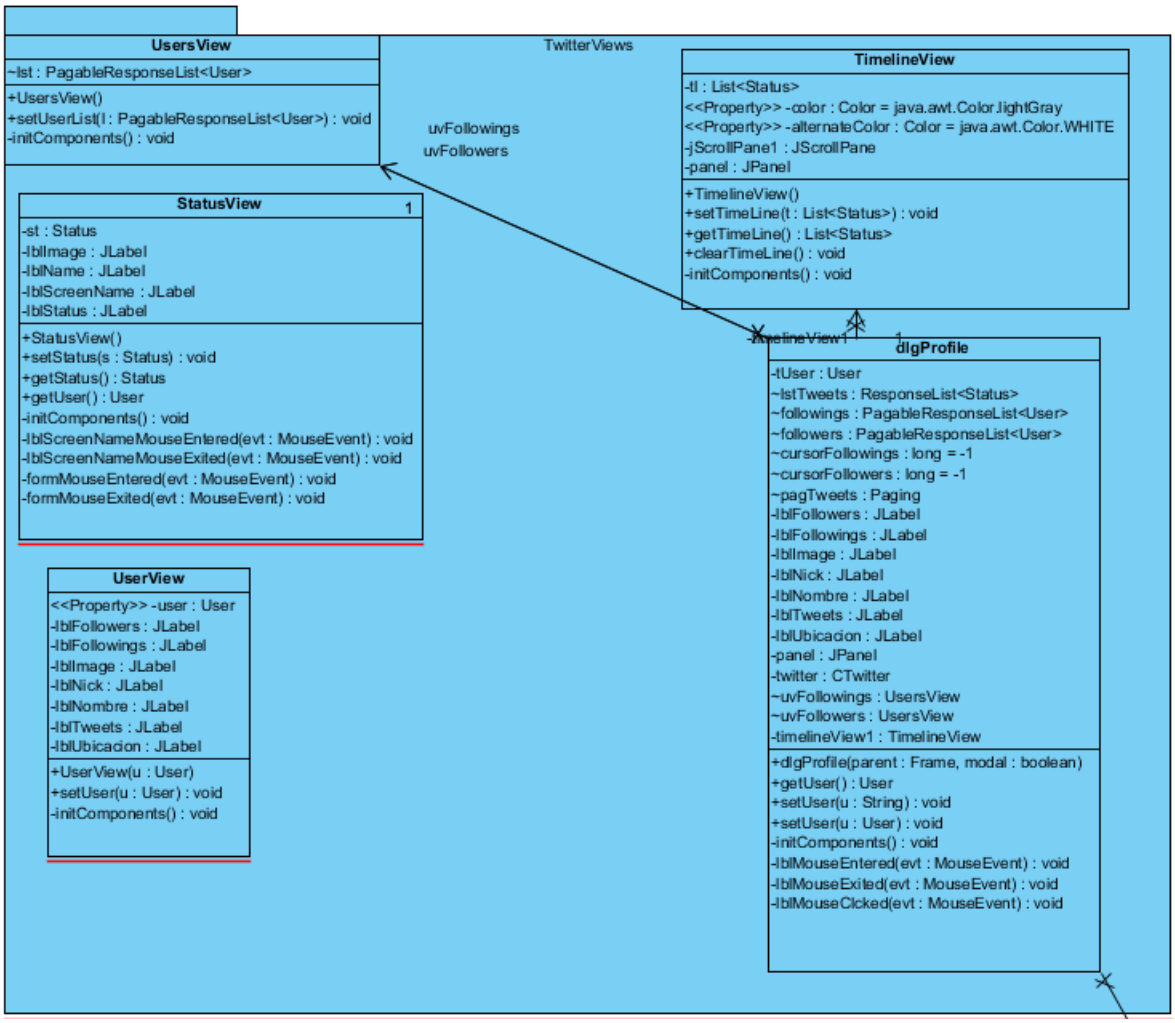


Figura 4.5. Diagrama de clases del paquete *TwitterViews*

5. IMPLEMENTACIÓN DEL SISTEMA

En este apartado se describirán las distintas plataformas y lenguajes de programación utilizadas para el desarrollo de la aplicación, así como el entorno de desarrollo utilizado.

5.1. Sistema Operativo

Tanto para el desarrollo de la aplicación como para la implementación de la misma se utilizó el sistema operativo Windows 7 Ultimate, que corre sobre un ordenador con las siguientes características:

- Marca: DELL
- Modelo: Vostro 1520
- Memoria Ram de 2GB
- Disco Duro de 150GB
- Procesador Intel Core 2 Duo a 2.10Ghz

5.2. Lenguaje de programación, entorno de desarrollo, y otras librerías

En este apartado se describirán el lenguaje de programación que se utilizó (Java), el entorno de desarrollo que se seleccionó (Netbeans), y algunas librerías o bibliotecas como *Twitter4J* que permite la interacción con la API de Twitter; o *JUNG* que es una librería que nos permite crear y manipular grafos.

5.2.1. Java

Java es un lenguaje de programación orientado a objetos, desarrollado por Sun Microsystems a principios de los años 90. El lenguaje en sí mismo toma mucha de su sintaxis de C y C++, pero tiene un modelo de objetos más simple y elimina herramientas de bajo nivel, que suelen inducir a muchos errores, como la manipulación directa de punteros o memoria. La memoria es gestionada mediante un recolector de basura.

Las aplicaciones Java están típicamente compiladas en un *bytecode*, aunque la compilación en código máquina nativo también es posible. En el tiempo de ejecución, el *bytecode* es normalmente interpretado o compilado a código nativo para la ejecución, aunque la ejecución directa por hardware del *bytecode* por un procesador Java también es posible.

La implementación original y de referencia del compilador, la máquina virtual y las bibliotecas de clases de Java fueron desarrolladas por Sun Microsystems en 1995. Desde

entonces, Sun ha controlado las especificaciones, el desarrollo y evolución del lenguaje a través del Java Community Process, si bien otros han desarrollado también implementaciones alternativas de estas tecnologías de Sun, algunas incluso bajo licencias de software libre.

Entre diciembre de 2006 y mayo de 2007, Sun Microsystems liberó la mayor parte de sus tecnologías Java bajo la licencia GNU GPL, de acuerdo con las especificaciones del Java Community Process, de tal forma que prácticamente todo el Java de Sun es ahora software libre (aunque la biblioteca de clases de Sun que se requiere para ejecutar los programas Java aún no lo es).

5.2.2. Netbeans

NetBeans es un entorno de desarrollo integrado libre, hecho principalmente para el lenguaje de programación Java. Existe además un número importante de módulos para extenderlo. NetBeans IDE es un producto libre y gratuito sin restricciones de uso.

NetBeans es un proyecto de código abierto de gran éxito con una gran base de usuarios, una comunidad en constante crecimiento, y con cerca de 100 socios en todo el mundo. Sun MicroSystems fundó el proyecto de código abierto NetBeans en junio de 2000 y continúa siendo el patrocinador principal de los proyectos.

La plataforma NetBeans permite que las aplicaciones sean desarrolladas a partir de un conjunto de componentes de software llamados módulos. Un módulo es un archivo Java que contiene clases de java escritas para interactuar con las APIs de NetBeans y un archivo especial (manifest file) que lo identifica como módulo. Las aplicaciones construidas a partir de módulos pueden ser extendidas agregándole nuevos módulos. Debido a que los módulos pueden ser desarrollados independientemente, las aplicaciones basadas en la plataforma NetBeans pueden ser extendidas fácilmente por otros desarrolladores de software.

5.2.3. Twitter for Java (Twitter4j)

Uno de los principales problemas a la hora de extraer información de Twitter es la complejidad del proceso de interacción con el servicio de red social Twitter. La API de Twitter atiende peticiones HTTP, y devuelve las respuestas en formato XML que ha de ser interpretado. Afortunamente, existen numerosos “bindigns” ya desaroollados que se ocupan de simplificar este proceso. En particular, los más populares en lenguaje Java son: T*witter4J, Java-Twitter, JTwitter* y *TwitterStream4J.*

De todas ellas, *Twitter4J* es la más completa en cuanto a funcionalidades, estabilidad y frecuencia de actualizaciones. Soporta la mayoría de los métodos de los módulos de la API de Twitter (REST API, Search API y "streaming"), funciona como una biblioteca Java (.jar) y no posee dependencias externas. Además, ofrece una extensa documentación disponible en la página web del proyecto y una lista de correo muy activa, en la que los desarrolladores exponen sus problemas y sugerencias al resto de colegas. Por todos estos motivos, *Twitter4J* fue seleccionado para uso en el proyecto.

El tedioso proceso de realizar peticiones HTTP a la API de Twitter para recuperar la información en formato XML, es reemplazado por *Twitter4J* por el uso de simples llamadas a métodos de objetos Java. Las peticiones HTTP son gestionadas internamente y toda la información recuperada es procesada y cargada en lista de objetos para facilitar su acceso y ordenación. Por otra parte, *Twitter4J* también es capaz de tratar posibles excepciones en el proceso de recuperación de la información de Twitter, mediante mensajes de error.

El funcionamiento de *Twitter4J* puede dividirse en varias fases, que se ilustran en la siguiente figura:

Figura 5.1. Funcionamiento de *Twitter4J*

Twitter4J ha sido desarrollado y es periódicamente actualizado por el desarrollador japonés de alias *yusukey*. Se trata de un proyecto de código abierto gratuito, distribuido con

licencia BSD y disponible para ser utilizado con fines tanto comerciales como no comerciales. Por otra parte, el proyecto admite donaciones económicas altruistas como forma de financiación extraordinaria.

La versión de *Twitter4J* utilizada en el proyecto ha sido la 2.2.2.

5.2.4. JUNG

JUNG (Java Universal Network/Graph) es un framework OpenSource para el modelado y visualización de grafos escrito en Java , bajo licencia BSD. El framework viene con una serie de algoritmos de diseño (layouts) integrados, así como algoritmos de análisis, como la agrupación de grafos y métricas de centralidad de nodo.

La arquitectura de JUNG está diseñada para representar una gran variedad de entidades y sus relaciones, tales como, grafos dirigidos y no dirigidos, multigrafos, grafos con aristas paralelas e hypergrafos. Proporciona un mecanismo de metadatos para asignar propiedades y sus valores tanto a los grafos, nodos y sus correspondientes enlaces. JUNG también facilita la creación de herramientas de análisis para conjuntos de datos complejos en las que se pueden examinar las relaciones existentes entre los diferentes nodos, así como los metadatos asociados a cada entidad o relación. JUNG incluye la implementación de una gran variedad de algoritmos de teoría de grafos, minería de datos y análisis de redes sociales, tales como, *agrupamiento de grafos* (clustering), *descomposición de grafos, optimización, generación de grafos aleatorios, análisis de estadísticas, cálculos de distancias, flujos* y un variedad métricas muy importantes.

JUNG incluye un framework de visualización que hace muy fácil la elaboración de herramientas para la exploración interactiva de datos de la red. Los usuarios pueden utilizar diferentes layouts ya incluídos en el framework, o pueden pueden crear sus propios diseños personalizados. Además, los mecanismos de filtrado que se proporcionan permiten a los usuarios centrar su atención, o sus algoritmos, en ciertas partes de la grafo.

La figura 4.1 muestra un grafo creado con el framework de JUNG.

6. ANÁLISIS DE LA RED COMPUESTA POR LOS JUGADORES DEL REAL MADRID EN TWITTER

La base principal para un estudio sobre el análisis de redes sociales es la información, dicha información es necesaria para realizar mediciones, comparaciones y pruebas. Donde el tamaño de la información y como ésta se genera son factores fundamentales para lograr un buen estudio.

Como caso de estudio, vamos a seleccionar la red formada por los jugadores del primer equipo de futbol del Real Madrid C. F. que tienen cuenta en la red social Twitter; a partir de dicha red vamos a tomar la información y métricas relevantes para mostrar datos interesantes.

6.1. Metodología del estudio

Para el desarrollo de este análisis se llevaron a cabo 3 etapas que se ilustran en la figura 6.1 que son: muestreo, obtención de un grafo representativo y análisis de los resultados. A continuación detallamos cada una de estas etapas.

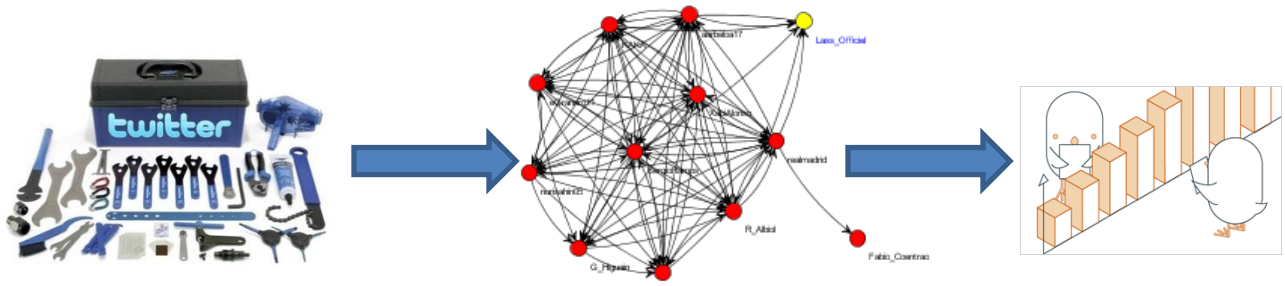


Figura 6.1. Etapas realizadas en el análisis de la red formada por los jugadores del Real Madrid C. F. en Twitter

Muestreo

Esta primera etapa consiste en investigar qué jugadores del Real Madrid poseen cuentas oficiales en la red social Twitter.

De los 23 jugadores que forman parte de la primera plantilla de futbol del Real Madrid, 11 poseen una cuenta oficial en Twitter (Cristiano Ronaldo, Kaká, Álvaro Arbeloa, Xabi Alonso, Sergio Ramos, Esteban Granero, Raúl Albiol, Gonzalo Higuaín, Lassana Diarrá, Fabio Coentrao y Nuri Sahin). Por tanto, esta será nuestra muestra para formar el grafo, además añadimos la cuenta oficial del club (@realmadrid). La tabla 6.1 muestra las respectivas cuentas.

Además de cargar, el nombre de usuario de cada jugador, también se cargará información relevante a cada uno de ellos como el número de followings/followers, número de publicaciones o tweets y el número de publicaciones marcadas como favoritas. Todos estos datos fueron obtenidos del día 25 de mayo de 2012, y con ellos haremos el análisis correspondiente.

Jugador	**ID Twitter**
Real Madrid	@realmadrid
Lassana Diarra	@Lass_Officiel
Alvaro Arbeloa	@aarbeloa17
Raúl Albiol	@R_Albiol
Nuri Sahin	@nurisahin05
Kaká	@KAKA
Sergio Ramos	@SergioRamos
Cristiano Ronaldo	@Cristiano
Fabio Coentrao	@Fabio_Coentrao
Esteban Granero	@eGranero11
Xabi Alonso	@XabiAlonso
Gonzalo Higuaín	@G_Higuain

Tabla 6.1. Cuentas de Twitter consideradas en el estudio

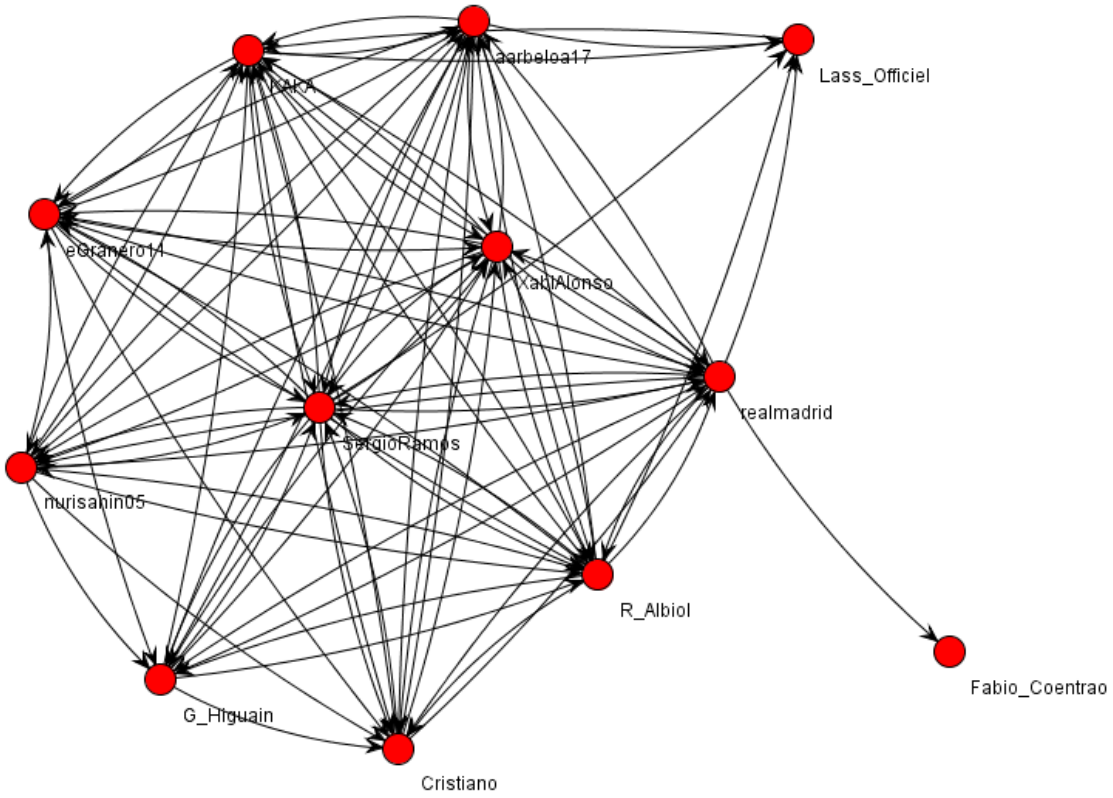


Figura 6.2. Grafo obtenido a partir de las cuentas de Twitter de los jugadores del Real Madrid, también la cuenta oficial del club.

Grafo representativo

Una vez obtenida las cuentas de Twitter de cada uno de los jugadores pasaríamos a formar el grafo de acuerdo a las relaciones de amistad que se tengan entre ellos, se incluye también la cuenta del club. En la figura 6.2 se muestra el grafo obtenido. Como vemos el grafo posee un total de 12 vértices, que representan cada una de las cuentas en Twitter; y un total de 89 enlaces que representan las relaciones de amistad (o seguimiento) entre cada uno de ellos.

Análisis de los resultados

Por último en esta etapa se realizará una serie de análisis de la información generada a partir del grafo obtenido. Empezaremos haciendo un análisis sobre las métricas generales del grafo, luego algunas propiedades de modelos de red que cumple el grafo y por último veremos un pequeño análisis con la información obtenida propiamente dicha de Twitter.

6.2. Análisis de la red de los jugadores del Real Madrid

A continuación mostramos el análisis realizado a partir de las diferentes métricas obtenidas con el grafo generado. Cabe mencionar, que para realizar este pequeño análisis, se cargó la red formada únicamente por los jugadores de la primera plantilla de futbol del Real Madrid, no se incluyen a sus seguidores, esto por dos grandes razones, la primera, por la gran cantidad de seguidores que tienen estos jugadores, que sería humanamente imposible de interpretar; y la segunda por la limitante impuesta por la API de Twitter en la cantidad de peticiones por hora.

Puesto que la red social Twitter está en constante actividad, la información y estadísticas de sus usuarios cambia constantemente, por ello, seleccionaremos como muestra los datos tomados el día 25 de mayo de 2012.

Una vez seleccionada la muestra pasamos a construir el grafo de acuerdo a las relaciones que tienen los jugadores del Real Madrid entre sí y con la cuenta oficial del club. La figura 6.2 muestra el grafo obtenido.

A continuación pasamos a ver una serie de análisis a partir de la obtención de métricas del grafo. Empezamos viendo un análisis general sobre el grafo, luego vemos algunas propiedades de modelos de red que presenta el grafo y por último vemos un análisis relacionado con la información propiamente dicha de la red social Twitter, en la que veremos un breve estudio sobre la popularidad y actividad de los jugadores del Real Madrid en Twitter.

6.2.1. Análisis general del grafo

Como se mencionó anteriormente, el grafo consta de un total de 12 vértices y 89 aristas, lo que en promedio nos daría alrededor de 7 aristas por cada vértice lo que nos hace ver claramente que no todos los jugadores se siguen entre sí. La tabla 6.1 muestra las métricas generales del grafo.

Como observamos en la tabla 6.2, la distancia promedio para alcanzar un nodo desde cualquier otro es de 1.74, esto es muy bajo porque como sabemos estamos sobre una red pequeña en la que todos los actores se conocen entre sí.

Métrica	Valor
Vértices	12
Aristas	89
Distancia promedio	1.74
Diámetro de la red	3
Eficiencia global (coeficiente de agrupamiento global)	0.79
Media armónica	1.26
Grado medio de los nodos	14.83
Grado medio de entrada	7.41
Grado medio de salida	7.41

Tabla 6.2. Métricas generales obtenidas del grafo.

El *diámetro de la red*, que es la distancia más larga entre un nodo y otro, es de 3 y es el que va desde Lass Diarra hasta Fabio Coentrao, esto es así porque como vemos, a Coentrao el único que lo sigue es la cuenta del club, y Diarrá no sigue a la cuenta del club.

La *eficiencia global* (ver 2.5.1) de la red es de 0.79, esto indica que la red tiene un grado de interconectividad de 79%, a esto también se le conoce como coeficiente de agrupamiento global, y la media armónica (que es el recíproco de la eficiencia global) tiene un valor de 1.26. También observamos el promedio de grados (14.83) por nodo, promedio de grado de entrada (7.41) y de grado de salida (7.41), el promedio de grado de entrada y salida siempre será el mismo valor.

En la figura 6.3 podemos ver el comportamiento de otras propiedades del grafo como son, la centralidad de cercanía, la centralidad de intermediación y el coeficiente de agrupamiento local para cada uno de los nodos que componen la red.

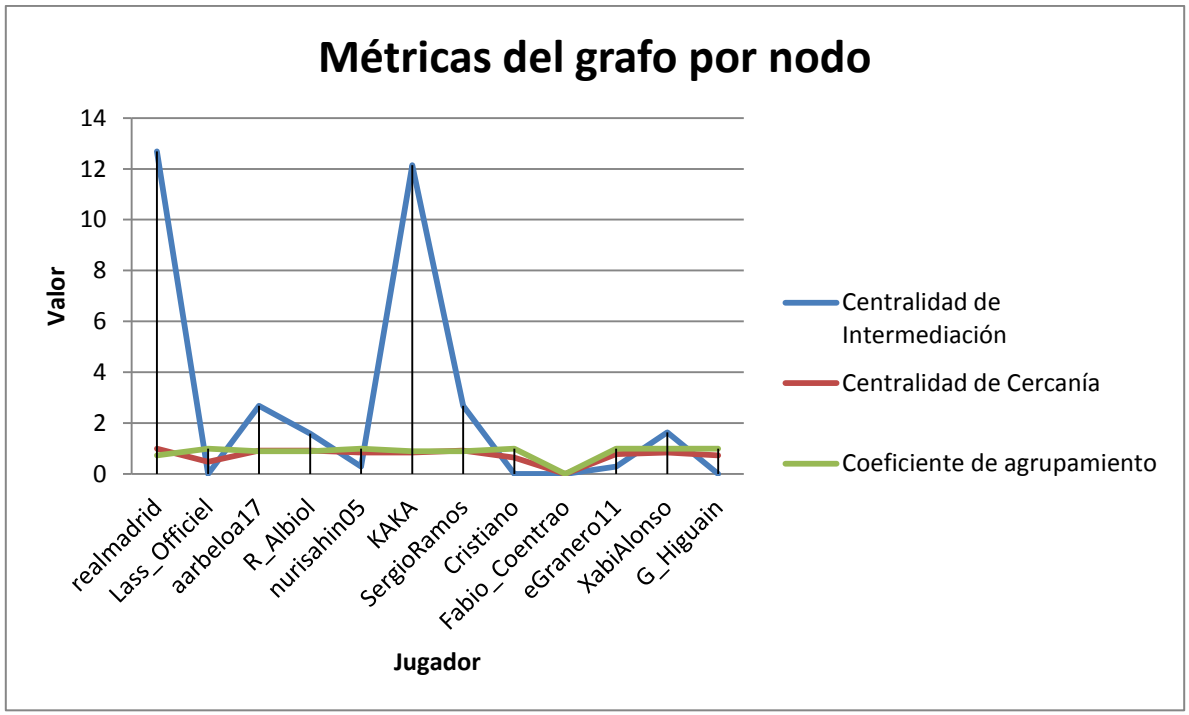


Figura 6.3. Métricas (centralidad de intermediación, centralidad de cercanía y coeficiente de agrupamiento local) del grafo para cada nodo.

6.2.1.1. Modelo de la red

En este apartado se discuten algunas propiedades de modelos redes que cumple la red formada por los jugadores del Real Madrid en Twitter.

Propiedades del mundo pequeño

Según el modelo de mundo pequeño de Watts-Strogatz (ver sección 2.4.2.2) para que una red cumpla con los requerimientos mínimos de estas redes debe cumplir lo siguiente:

- La trayectoria promedio de todos los vértices en la red debe ser pequeña (según la teoría de los seis grados debe ser menor a 6). La figura 6.4 se muestran estos promedios.
- El coeficiente de agrupamiento global de la red debe ser alto.
- Un diámetro de red pequeño.

Podemos destacar que la red cumple con estas tres características, ya que podemos ver que la *distancia promedio* dentro de toda la red es 1.74, en la figura 6.5 se muestra la distancia promedio de entrada y salida para cada jugador. El *coeficiente de agrupamiento*

que es lo mismo que la *eficiencia global* (ver sección 2.5.1) tiene un valor de 0.79 que a como vimos antes, esto indica que la red tiene un grado de interconexión del 79%. Por ultimo, vimos que el diámetro de la red es apenas de 3, esto es la distancia más larga que hay dentro de la red (Dirra -> Coentrao).

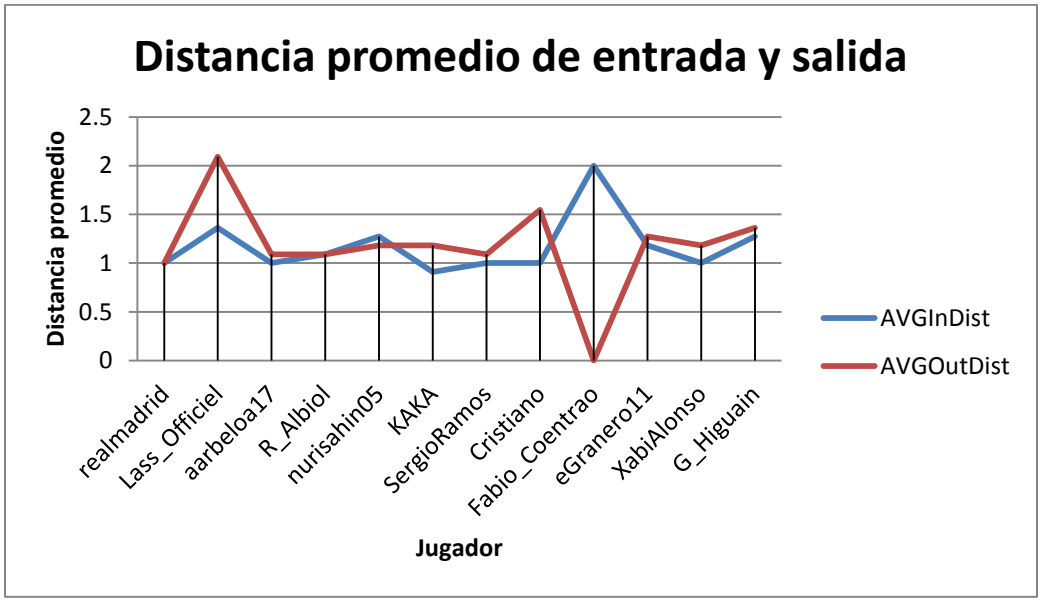


Figura 6.4. Distancia promedio de entrada y salida para cada jugador.

Propiedad libre de escala

El modelo de crecimiento más utilizado en las redes sociales es el de *Barabasi-Albert* (BA), en el que la formación de nuevas relaciones está determinado por un *anexo preferencial*. Para probar que una red sigue un anexo preferencial se debe de mostrar la distribución de grado de los nodos de la red.

El grado de un nodo representa el número de contactos con los que interactúa el usuario, a su vez indica que tan importante o activo es el usuario dentro de la red. La distribución de grado de una red representa la ocurrencia de los usuarios al agregarse al sistema (*anexo preferencial*). La figura 6.5(a) muestra el grado por cada nodo, y la figura 6.5(b) la distribución de dichos grados. Como se vio en la sección 2.4.24, un *anexo preferencial* se da cuando un nuevo nodo se agrega a un nodo con un alto número de enlaces, y vemos que en nuestra red si un nuevo jugador se añade a la red existe una alta probabilidad de que se una a un nodo con un alto número de enlaces.

En el caso de redes dirigidas (tal es el caso de nuestra red) se suele utilizar los términos *creación preferencial* y de *recepción preferencial*, para denotar los mecanismos

utilizados para la creación de enlaces en base a su grado de salida y en base a su grado de entrada para un usuario en la red. En la figura 6.5(c) y (d) se muestra la distribución de grado de entrada y de salida, respectivamente. Estos datos muestran que tanto la creación preferencial como la recepción preferencial siguen un *anexo preferencial*.

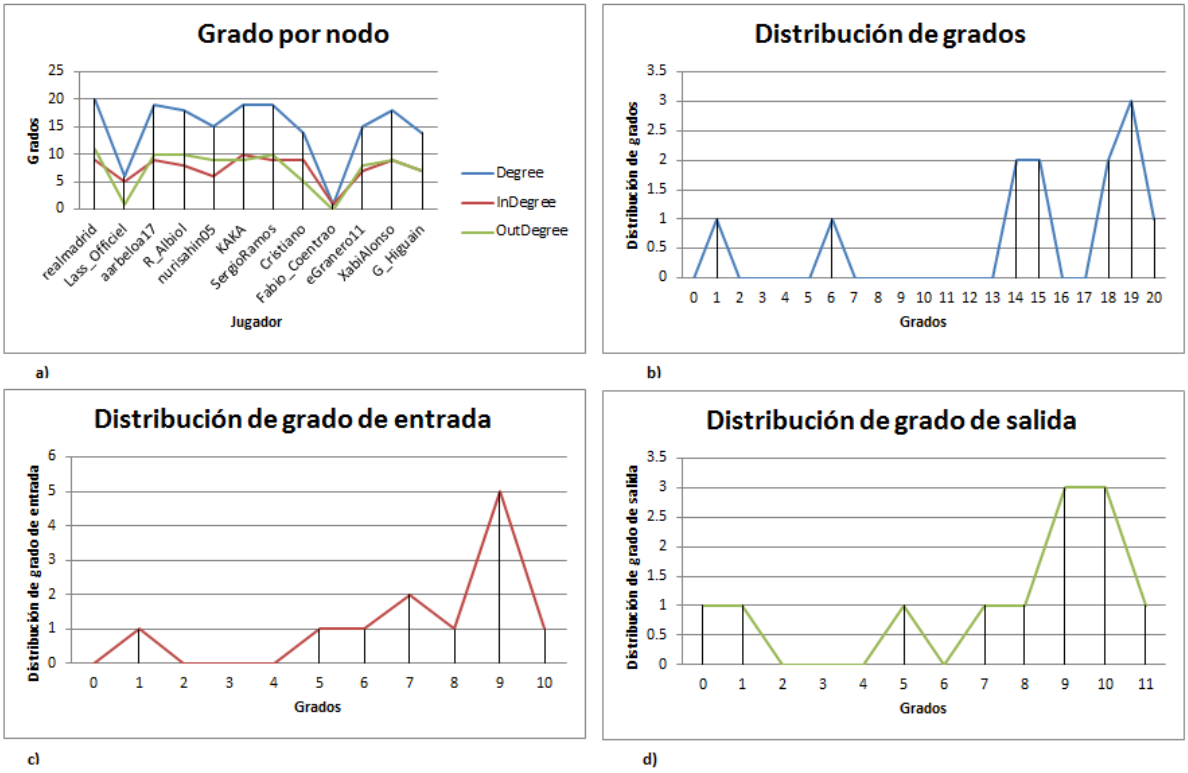


Figura 6.5. a) Grados por nodo. b) Distribución de grados. c) Distribución de grado de entrada. d) Distribución de grado de salida.

6.2.2. Análisis de la información de Twitter

En esta sección vamos a hacer un pequeño análisis de la información obtenida propiamente de Twitter; en concreto con dicha información vamos a ver que jugadores son los más populares dentro de la red social, así como también, determinar quienes son los más activos. Para este análisis solo tomamos en cuenta los datos de los jugadores, eliminando los datos que corresponden a la cuenta del club (@realmadrid) Todos los datos usados para este análisis fueron tomados el día 25 de mayo de 2012. En la figura 6.6 se muestra en resumen la informacion obtenida de Twitter por cada jugador (Followings, Followers, Tweets, Favourites). En base a estos datos se harán los estudios de popularidad y actividad en Twitter.

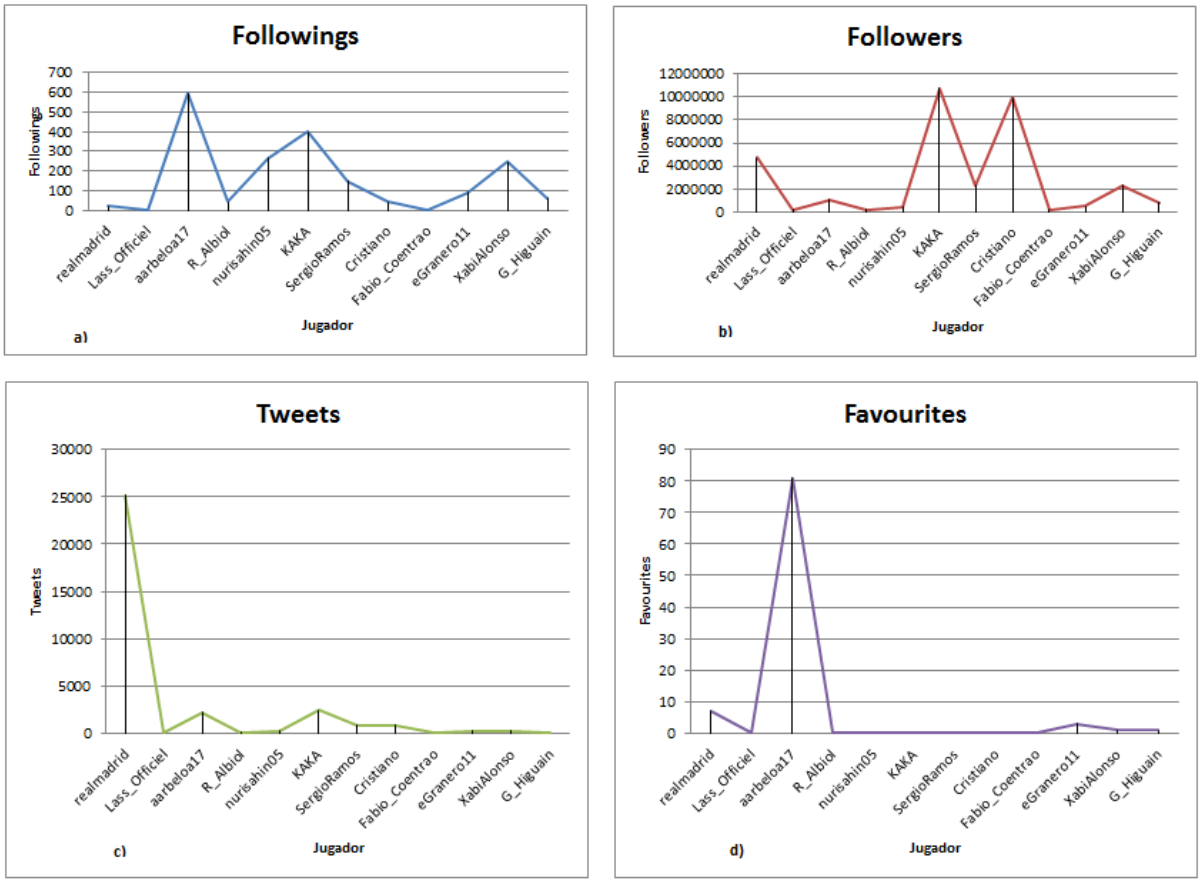


Figura 6.6. a) Cantidad de followings por jugador. b) Cantidad de followers por jugador. c) Cantidad de Tweets por jugador. d) Cantidad de favoritos por jugador.

6.2.2.1. Estudio de popularidad de los jugadores

Una manera para determinar quién es más popular dentro de la red social Twitter es ver la cantidad de seguidores que tienen. En la figura 6.7(b) se muestra la cantidad de seguidores que tienen cada uno de los jugadores en el que podemos destacar al brasileño Ricardo Kaká como el mas seguido por los usuarios en Twitter con un total 10,780,233, le sigue muy de cerca el portugués Cristiano Ronaldo con 9,915,545 y de lejos completa el pódium Xabi Alonso con un total de 2,326,622.

Por otro lado, podemos destacar el caso de Fabio Coentrao que no es muy querido dentro de los seguidores del Real Madrid, y esto también se refleja en Twitter, donde es el jugador con menos seguidores. Es un jugador bastante criticado por la prensa y los aficionados, y al parecer tampoco es muy querido en el seno de la plantilla (por palabras mismas de él, en la concentración con su selección para la Eurocopa 2012, afirmó que “en

la selección todos me dan confianza"), esto también se refleja en el grafo (figura 6.2) donde podemos ver que ninguno de sus compañeros de equipo le sigue (tampoco él sigue a sus compañeros). A pesar de todo ello, es un jugador que cuenta con la plena confianza de su entrenador y compatriota portugués José Mourinho.

6.2.2.2. Estudio de actividad de los jugadores en Twitter

Para determinar que jugador es el más activo dentro de la red social Twitter, podemos basarnos en la cantidad de publicaciones hechas. En la figura 6.7(c) podemos apreciar que quien más publica en Twitter es Kaká con un total de 2,453 tweets, completan el pódium Arbeloa con 2,179 y Sergio Ramos con 824 (Cristiano Ronaldo muy cerca con 822).

Pero no sólo por los tweets se puede determinar quién es el más activo, podemos también ver los usuarios a los que ellos siguen, esto claramente será un reflejo de que tanto interés tiene sobre las diferentes temáticas o usuarios que se encuentran en la red social; en este apartado quien tiene más followings es Álvaro Arbeloa con un total de 596 usuarios que sigue, largo quedan los 398 de Kaká y 264 de Nuri Sahin (Xabi Alonso muy cerca con 246).

Otra forma, bastante interesante para determinar quien aprovecha más todas las opciones que ofrece Twitter es medir el número de tweets marcados como favoritos, esta opción la ofrece twitter, pero no es comúnmente utilizada entre sus usuarios, así que midiendo este dato podremos determinar qué jugador saca más partido o provecho a la red social. En la figura 6.7(d) podemos observar que el jugador que más tweets ha marcado como favoritos es Álvaro Arbeloa quien tiene un total de 81 favoritos, mientras que Esteban Granero apenas tiene 3, y Xabi Alonso y Gonzalo Higuain tienen sólo 1; y el resto de jugadores no ha marcado ninguno como favorito.

Vistos estos datos, podemos concluir que de los jugadores del Real Madrid el más Twittero es el defensa español Álvaro Arbeloa. Y quienes siguen a los jugadores del Real Madrid en Twitter pueden dar fe de ello, puesto que es uno de los que más interactúa con los usuarios, esa es la percepción, y estos datos terminan demostrándolo.

6.2.2.3. Agrupamiento por demarcaciones

Podemos hacer un agrupamiento de los jugadores por la demarcación que cubren en el terreno de juego. En este caso 4 jugadores son defensas (Álvaro Arbeloa, Sergio Ramos, Raúl Albiol y Fabio Coentrao), otros 4 son mediocampistas (Xabi Alonso, Esteban

Granero, Lass Diarrá y Nuri Sahin) y 3 se desempeñan como delanteros (Kaká, Cristiano Ronaldo y Gonzalo Higuaín).

En la figura 6.7(a) podemos observar que los jugadores más seguidos por los usuarios en Twitter son los delanteros. En la figura 6.7(b) podemos ver el comportamiento de las otras tres métricas (followings, tweets y favoritos).

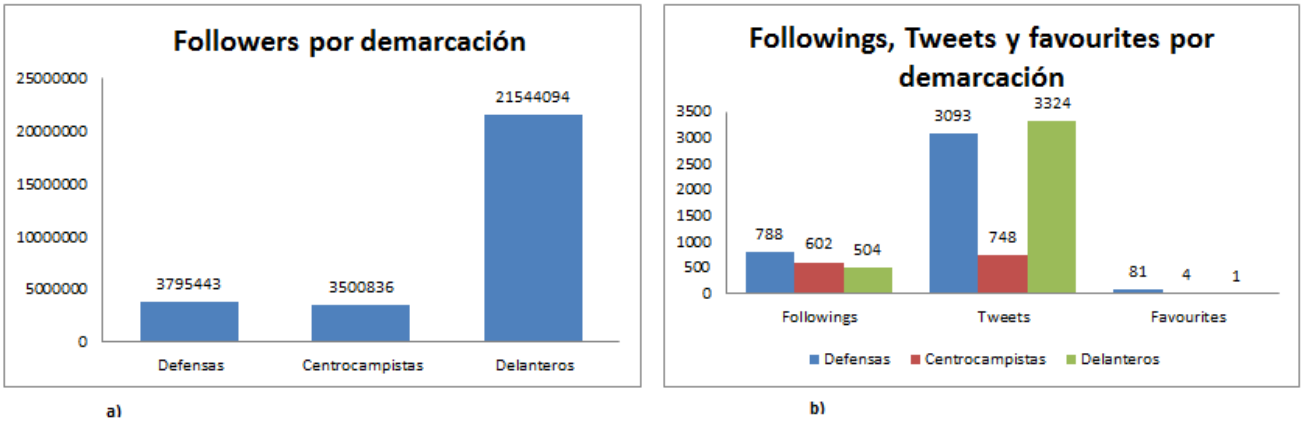


Figura 6.7. a) Followers por demarcación. b) Followings, tweets y favoritos por demarcación.

6.2.3. Discusión

Para realizar este análisis hemos seleccionado las cuentas oficiales de Twitter de los jugadores del primer equipo de futbol del Real Madrid junto con la cuenta del club, a partir de esto hemos creado un grafo representativo acerca de las relaciones de estos jugadores en la red social mencionada.

A partir del grafo obtenido, hemos calculado métricas con las que se han hecho los distintos análisis. Entre las métricas que calculamos tenemos el numero de vértices y aristas del grafo, la distancia promedio en el grafo (distancia promedio de entrada y salida a cada nodo), el diámetro de la red, la distribución de grados (también de entrada y salida).

Una vez analizados estos datos, pudimos observar que el grafo cumple con propiedades de redes de mundo pequeño, puesto que cumple con las 3 condiciones que deben cumplir este tipo de redes (distancia promedio entre los nodos bajo, coeficiente de agrupamiento global alto, diámetro de red pequeño).

Por último, se hizo un pequeño estudio sobre la popularidad y actividad de los jugadores en la red social, en la que obtuvimos como resultado que el jugador más seguido es el brasileño Kaká, seguido muy de cerca Cristiano Ronaldo; a su vez se pudo constatar

que los jugadores más seguidos son los delanteros. En cuanto a la actividad, pudimos comprobar que el jugador más activo es Álvaro Arbeloa, es quien más utiliza las diferentes opciones de Twitter, es uno de los que más tweets publica, es el que sigue más usuarios y el que más tweets ha marcado como favoritos.

Dentro del análisis se hizo un pequeño agrupamiento en relación a las demarcaciones que cubren cada uno de los jugadores, esto para determinar, que tipos de jugadores son los favoritos de los seguidores. Este agrupamiento, se ha hecho de una forma rustica, sin emplear ningún algoritmo de agrupamiento de grafos, para futuros trabajos sería interesante, aplicar algún algoritmo de agrupamiento, para encontrar diferentes grupos dentro de una red. También, resultaría interesante aplicar un estudio sobre el contenido de las publicaciones de los jugadores para, entre otras cosas, conocer que temas o tendencias se suelen debatir, hacer un estudio sobre que jugador es el que más interactúa con sus seguidores, entre otras.

7. CONCLUSIONES Y FUTUROS TRABAJOS

A través, del presente trabajo se ha desarrollado una herramienta software básica para el análisis de redes sociales en Twitter, para ello se ha hecho estudio sobre el análisis de redes (más conocido como teoría de grafos), profundizando sobre su aplicabilidad en redes sociales.

Para la realización de este proyecto, nos hemos auxiliado de varias herramientas que son muy poderosas, en el caso de JUNG, es una librería que permite el modelado de grafos, entre otras cosas permite: crear un grafo, visualizar grafos, obtener métricas del grafo y aplicar diversos algoritmos aplicables a los conceptos de redes. Otra herramienta en la que apoyamos este proyecto, es la librería Twitter4J, que son un conjunto de clases que permiten el acceso a la API de Twitter de una forma sencilla y eficaz, ahorrando trabajo al momento de analizar los documentos XML que dicha API proporciona.

Por último, hemos planteado un caso de uso de nuestra herramienta de análisis de redes sociales. Este caso de uso consistió en cargar la red formada por los jugadores del Real Madrid para obtener diferentes estadísticas. Tras el análisis de las estadísticas obtuvimos como resultados, que el jugador más seguido es Ricardo Kaká y el jugador más activo en la red social es Álvaro Arbeloa. También, pudimos observar que los jugadores más seguidos son aquellos que juegan como delanteros.

En conclusión podemos decir que tanto el análisis de redes, como su aplicabilidad a las redes sociales, son herramientas poderosas que permiten el análisis de diversos problemas dentro de una sociedad, tales como, epidemiología, tendencias electorales, así como también se puede utilizar para análisis de estudio de mercadeo para conocer las tendencias y los gustos de la sociedad para el lanzamiento de productos. Tras los diferentes análisis, se pueden plantear posibles soluciones a estos problemas.

Siguiendo esta línea de investigación podemos plantear como futuros trabajos:

- Análisis del contenido de las publicaciones de los usuarios, para conocer tendencias y temas sobre los que se debaten.
- Mantener históricos sobre como evoluciona la red.
- Hacer seguimientos a individuos destacados dentro de la red.
- Aplicar algoritmos de agrupamiento para la detección de comunidades dentro de la red.
- Extender el uso de la aplicación a otras redes sociales, tales como, Facebook, Flickr, Youtube, Hi5, entre otras.

8. BIBLIOGRAFÍA

[1] L. da F. Costa, F. A. Rodrigues, G. Travieso and P.R. Villas Boas, *Characterization of Complex Networks: A Survey of measurements*, Paper, 84pags, Feb. 2008.

[2] Réka Albert and Albert-Lá szló Barabási, *Statistical mechanics of complex networks*, Review of Modern Physics, Vol. 74, 51 pags, Ene. 2002.

[3] Joshua O´Madadhain, Danyel Fisher, Padhraic Smyth, Scott White, Yan-Biao Boey, *Analysis and Visualization of Network Data Using JUNG*, Journal of statistical Software, 35 pags.

[4] Alberto Díaz Angulo, *CICLOPE VISUALIZATOR: Herramienta para el análisis y visualización de redes sociales.* TFC Universidad Politécnica de Madrid, Facultad de infórmtica, 117 pags, Mar. 2010.

[5] Cristian Paolo Mejia Olivares, *Análisis de redes sociales a gran escala*, TFC Centro De Investigación Y De Estudios Avanzados del Instituto Politécnico Nacional Departamento De Computación, 117 pags, Feb. 2010.

[6] E. N. Gilbert. Random graphs. *The Annals of Statistics*, 30(30):1141–1144, 1959.

[7] S. Milgram. *The small world problem.* Psychology Today, 2(1):60–67, 1967.

[8] Mark E. J. Newman, Albert L. Barabási, and Duncan J.Watts, editors. *The Structure and Dynamics of Networks*. Princeton University Press, 2006.

[9] Duncan J. Watts. Six degrees: *The science of a connected age. WW Norton & Company*, 2003.

[10] Jennifer Xu and Hsinchun Chen. *The topology of dark networks*. Commun. ACM, 51(10):58–65, 2008.

[11] Stanley Wasserman, Katherine Faust, and Dawn Iacobucci. *Social Network Analysis: Methods and Applications (Structural Analysis in the Social Sciences)*. Cambridge University Press, Nov. 1994.

[12] Duncan J. Watts and Steven H. Strogatz. *Collective dynamics of 'small-world' networks*. Nature, 393(6684):440–442, Jun. 1998.

[13] Christo Wilson, Bryce Boe, Alessandra Sala, Krishna P. Puttaswamy, and Ben Y. Zhao. *User interactions in social networks and their implications. In Proceedings of the Fourth ACM European conference on Computer Systems (EuroSys)*, pages 205–218, New York, NY, USA, 2009. ACM Press.

[14] Andrei Broder, Ravi Kumar, Farzin Maghoul, Prabhakar Raghavan, Sridhar Rajagopalan, Raymie Stata, Andrew Tomkins, and Janet Wiener. *Graph structure in the web: experiments and models.* In Proceedings of the Ninth International World-Wide Web Conference (WWW9, Amsterdam, May 15-19, 2000-Best Paper). Foretec Seminars, Inc. (of CD-ROM), Reston, VA, 2000.

[15] Hayes, B. (2000). Graph Theory in Practice: Part I. American Scientist, 88(1), 9-13.

[16] Hayes, Brian. (2000). Graph Theory in Practice: Part II. American Scientist, 88(2), 104-109.

[17] Luciano, Rodrigues, F. A., Travieso, G., & Boas, V. P. R. (2006). Characterization of complex networks: A survey of measurements. Advances in Physics, 56(1), 167-242. Taylor & Francis.

REFERENCIAS EN INTERNET

- http://es.wikipedia.org/wiki/Red_social
- http://es.wikipedia.org/wiki/Analisis_de_redes
- http://es.wikipedia.org/wiki/Teoria_de_grafos
- http://en.wikipedia.org/wiki/JUNG