UNIVERSIDAD AUTÓNOMA DE MADRID

Escuela Politécnica Superior

Doble Grado en Ingeniería Informática y Matemáticas

TRABAJO FIN DE GRADO

A RANDOM GRAPH MODEL OF SOCIAL NETWORKS
(UN MODELO ALEATORIO DE REDES SOCIALES)

Mario Gómez Alonso

Tutor: Simone Santini

Junio 2017

A RANDOM GRAPH MODEL OF SOCIAL NETWORKS
(UN MODELO ALEATORIO DE REDES SOCIALES)

Autor: Mario Gómez Alonso
Tutor: Simone Santini

Escuela Politécnica Superior
Universidad Autónoma de Madrid

Junio 2017

Resumen

Resumen Las redes sociales pueden ser modeladas como estructuras de grafos cuyos nodos simbolizan usuarios y las aristas representan una conexión directa entre dos de ellos. Algunas características del grafo dependen de la red social que está siendo modelada. Por ejemplo, redes sociales con relación asimétrica (como Twitter) se corresponden con un grafo dirigido mientras que redes como Facebook se ajustan a una estructura de grafo con aristas bidireccionales. Las características estructurales del grafo son importantes para entender cómo fluye la información en las redes sociales, mensajes influyentes se extienden o usuarios populares acaparan una gran cantidad de seguidores, es decir, la viralidad de estos elementos en la redes sociales.

Con el fin de estudiar el comportamiento de las redes sociales, es de gran importancia poder generar grafos de forma aleatoria y automática, compartiendo ciertas características con las redes sociales que existen. Esta generación autónoma (si se ejecuta con precisión) permite una simulación potente y realista de los medios sociales, una herramienta con la que se puede sacar una gran cantidad de información estadística. Hay modelos bien conocidos que replican características macroscópicas importantes de las redes sociales, como la distribución de ejes, el grado medio de los nodos o el diámetro del grafo. Los estudios más recientes han tenido en cuenta el clustering, una característica muy representativa de las redes sociales.

En este trabajo se implementarán diversos algoritmos de generación de grafos, ajustándose a varios modelos que se mencionarán en el cuerpo del documento. Una vez construidos, se obtendrán datos estadísticos (medidas) que serán comparados con datos reales de la red social Twitter. En base a los resultados se extraerán conclusiones, si eran o no esperables, y posibles mejoras a los algoritmos implementados.

Palabras clave Teoría de grafos, Análisis de redes sociales, Generación por procedimientos

Abstract

Abstract Social networks can be modeled as graph structures in which nodes represent users and edges express a direct conection between two of them. Some of the characteristics of this graph depend on the social network that is being modeled. For example, networks with asymmetric relation (such as Twitter) result in directed graphs, while networks such as Facebook fit in a graph structure with bidirectional edges. The structural characteristics of the graph are important to understand how information flows in the social network, how influenctial messages spreads or popular users monopolize lots of followers, that is, the virality of these elements in the social networks.

In order to study the behavior of social networks, it is very important being able to generate random graphs automatically, sharing certain characteristics with the actual network graphs. This autonomous graph generation (if it is executed accurately) allows a powerful and realistic simulation of the social media, a tool wich can be used to draw a large amount of statistical information. There are well known graph models that replicate important macroscopic network characteristics such as the distribution of the fan-out of the nodes, the average number of neighbors per node or the graph diameter. Recent studies have taken into account clustering, a very representative characteristic of the social networks.

In this work, some graph generation algortihms shall be implemented, fiting to several models wich will be menctioned in the document body. Then, statistical data (measures) shall be obtained and compared with actual data of Twitter. Based on the results conclusions will be drawn, whether or not they were expected, and possible improvements will be proposed to the implemented algorithms.

Keywords Graph theory, Social network analysis, Procedural generation

ÍNDICE GENERAL

ÍNDICE DE TABLAS

ÍNDICE DE FIGURAS

GLOSARIO

API *Application Program Interface. Conjunto de rutinas, funciones y procedimientos (o métodos, en la programación orientada a objetos) que ofrece cierta biblioteca para ser utilizado por otro software como una capa de abstracción*. 2, 3, 17, 19, 20, 21, 25, 37

FAV *Favorito. Interacción de Twitter que representa una reacción positiva del usuario ante el tweet al que se le da FAV*. 11

Hashtag *Cadena de caracteres sin espacios y precedida por una almohadilla con el fin de que tanto el sistema como el usuario la identifiquen de forma rápida*. 7

IDE *Integrated Development Environment. Aplicación informática que proporciona servicios integrales para facilitarle al desarrollador o programador el desarrollo de software*. 21, 25

jre *Java Runtime Environment. Conformado por una Máquina Virtual de Java o JVM, un conjunto de bibliotecas Java y otros componentes necesarios para que una aplicación escrita en lenguaje Java pueda ser ejecutada*. 25

RT *Retweet. Interacción de Twitter que en la que un usuario comparte el tweet con todos sus seguidores. Es el modo más efectivo de transmitir los mensajes a través de Twitter*. 11

SO *Sistema Operativo. Conjunto de programas de un sistema informático que gestiona los recursos de hardware y provee servicios a los programas de aplicación de software*. 21, 25

TL *TimeLine. Sección donde se muestran los sucesos incluidos en la misma ordenados cronológicamente*. 1

UML *Unified Modeling Language. Lenguaje gráfico para visualizar, especificar, construir y documentar un sistema*. 19, 21

1 INTRODUCCIÓN

En este capítulo se exponen las circunstancias que motivan la realización de este Trabajo Fin de Grado, planteando los objetivos que este proyecto plantea. También se definirá la estructura que seguirá este documento.

Antes de empezar, resulta oportuno hacer una aclaración terminológica: Siempre que se hable de redes sociales en este documento, se estará refiriendo a aquellas que utilizan internet como medio de difusión. Hay muchos tipos de redes sociales, desde los grupos de amigos a las familias, el estudio de éstas es un campo aparte[9] que aquí no consideramos.

1.1 Motivación

La hegemonía de las redes sociales es una realidad que se ha producido en poco más de quince años. Desde las más significativas (Facebook, Youtube, LinkedIn) hasta aquellas no acaban de despegar del todo, las redes sociales están muy presentes en la vida moderna y comparten unas características fundamentales.

El aspecto que más caracteriza las redes sociales que tratamos aquí es su medio de difusión, internet, elemento imprescindible a la hora de utilizar las redes sociales. Con el paso de los años tener una conexión a internet de alta velocidad ha pasado de ser un privilegio de pocos a ser un elemento de la vida cotidiana de miles de millones de personas y en consecuencia millones de usuarios hacen uso de estas redes sociales.

Una característica fundamental de las redes sociales es la posibilidad de crear conexiones (relaciones) con otros usuarios: Es una interacción habitual o incluso necesaria para tener acceso a la información que la otra parte publique (pudiendo aparecer de forma automática en aquellas aplicaciones que tengan incluida una TimeLine(TL)) y viceversa, además de otras ventajas como comunicación privada o acceso a datos personales.

Es decir, independientemente de la que clase de información se distribuye en la red (mensajes de 140 caracteres, fotos, videos...) la estructura que forman los usuarios y sus relaciones entre ellos es el principal elemento que establece las reglas de difusión de dicha información[2][10]. Es por eso que el estudios del comportamiento y la evolución de las redes sociales se han centrado en analizar las características de esta estructura. El par (actores,relaciones) de una red social es isomorfo al de (nodos,aristas) de los grafos, haciendo que el análisis de redes sociales y la teoría de grafos estén fuertemente relacionados.

Un objetivo de estos estudios es el de obtener un modelo representativo y diseñar un algoritmo a partir del mismo que sea capaz de generar grafos que se

asemejen a la estructura de las redes sociales. Es algo que han conseguido, con mayor o menor éxito, generando modelos y algoritmos como los que proponen Erdös[3], Barabási[1] o Leskovec[7], y que se estudiarán en este trabajo.

Para medir la calidad de estos grafos aleatorios y su correspondencia con la realidad, es necesario compararlos con la estructura de las redes sociales reales usando métricas de descripción de grafos. Con ellas podemos obtener propiedades como el grado medio y cercanía de los nodos, coeficiente de clustering, comunidades, entre otras. Se asume que dos grafos que tienen propiedades topológicas similares equivale a que los dos grafos tienen propiedades métricas semejantes. Esto implica que se puede determinar la precisión de un modelo y su algoritmo aplicando medidas en éste y comparando con grafos generados a partir de datos reales de aquello que está siendo sujeto a modelización.

Con estas herramientas vamos a obtener las medidas de la red social Twitter a partir de una muestra de datos reales significativa y a comparar los resultados con los que se obtengan de los grafos generados de forma automática. Los resultados determinaran en qué grado los modelos se ajustan a la estructura de Twitter, pudiendo variar los parámetros de los algoritmos hasta encontrar los más verosímiles, obteniendo información relevante sobre las características topológicas de la red social.

1.2 Objetivos

El objetivo principal que se plantea para este Trabajo Fin de Grado es el de, a partir de una muestra de datos reales de Twitter, programar una API que mapee la información en un grafo y obtenga sus características topológicas empleando diversas métricas. Estas propiedades serán comparadas con la de los grafos creados a partir de los algoritmos de generación automática, identificando el origen de las diferencias que puedan aparecer. La situación ideal es que esta desigualdad sea mínima, por lo que se deberán variar los parámetros de los algoritmos o incluso modificar estos algoritmos con el fin de obtener los resultados que mejor se ajusten a los datos reales. Se plasmará todo el proceso en diversas gráficas, donde se mostrará la evolución de los grafos autogenerados a medida que modificamos los parámetros. Para llevar a cabo estos objetivos será necesario implementar la obtención de medidas del grafo y los algoritmos de generación automática de grafos que se expondrán en el estado del arte, siendo Java el lenguaje de programación escogido.

Las metas propuestas pueden separarse en la siguiente lista de hitos, que deberán ser cumplidos en el orden que se presentan:

1. Diseñar el modelo de datos del grafo.
2. Diseñar una función que lea el fichero de usuarios y seguidores y lo inyecte al modelo.
3. Definir una interfaz de métricas del grafo e implementarla acorde al modelo de datos.
4. Estudiar los resultados de las medidas aplicadas al grafo de Twitter.

5. Diseñar una interfaz de generación automática de grafos
6. Implementar esta interfaz, ajustándose a los modelos propuestos en el estado del arte y al modelo de datos implementado
7. Aplicar medidas sobre los grafos, comparar con las medidas del grafo inicial.
8. Ajustar los parámetros o modificar el algoritmo hasta obtener las medidas más verosímiles utilizando el punto anterior.
9. Estudiar los resultados definitivos, dando las explicaciones pertinentes.

Los primeros seis hitos entran en la fase de desarrollo e implementación, mientras que los tres últimos pertenecen a la fase de pruebas.

1.3 Estructura del documento

En el siguiente capítulo se revisará el estado del arte. Por un lado se definirán brevemente los conceptos básicos de la teoría de grafos, sus propiedades topológicas y las medidas que se pueden aplicar sobre el mismo. Luego se repasará el estado actual del análisis de redes sociales y veremos tres modelos de grafos aleatorios y sus algoritmos asociados.

Un capítulo estará reservado al estudio de los datos proporcionados para la realización de este Trabajo Fin de Grado y los algoritmos que serán empleados en la construcción de grafos aleatorios, explicando las características de los grafos resultantes y comentando las modificaciones que se han hecho para mejorarlos.

A lo largo del desarollo se diseñará la API, que será la herramienta con la que se realizará el estudio de los grafos, definiendo los módulos e interfaces que deberán implementarse.

La aplicación será la herramienta utilizada para obtener los datos necesarios en la fase de pruebas. Al inicio de esta sección se mostrarán gráficas sobre los datos reales de Twitter y los subgrafos generados a partir de los algoritmos que se implementen, comentando algunas de sus características más representativas. Entonces se iniciará una rutina de comparación y perfeccionamiento de los parámetros de los algoritmos con el fin de obtener las medidas más verosímiles con respecto a los datos reales. Cuando los algoritmos estén bien configurados se hará una comparación final de éstos y los datos reales, obteniendo conclusiones acerca de la efectividad de cada modelo respecto de la red social Twitter.

Por último, en el capítulo 6 se expondrán las conclusiones del trabajo y se discutirán las líneas de trabajo futuro que surgen a partir del proyecto realizado.

ESTADO DEL ARTE

2

En este capítulo estudiaremos la teoría subyacente al Trabajo Fin de Grado. Esta sección está organizada en dos apartados, ya que hay dos ramas distintas que confluyen en este proyecto. En primer lugar repasaremos brevemente la teoría básica de los grafos y sus métricas, después analizaremos los principales modelos de grafos aleatorios que se han propuesto como modelos de redes sociales.

2.1 Teoría de grafos

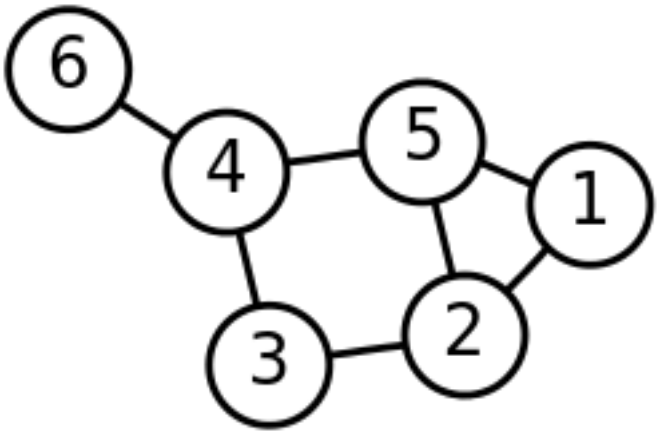

Figura 2.1: Representación gráfica de un grafo de seis nodos (círculos numerados) y siete aristas (lineas). Podemos suponer que es un grafo no dirigido en base a que los ejes carecen de orientación (representado por flechas)

Un grafo[8] es una estructura $G = (N, A)$ formada por un conjunto N de nodos y un conjunto $A \subseteq NxN$ de aristas. El par $a, b \in N, (a, b) \in A$ es una arista desde a(su origen) hasta b(su destino). En otras palabras, el grafo representa un conjunto abstracto de elementos y relaciones binarias sobre los mismos.

Hay multitud de tipos de grafos dependiendo de las propiedades que se le den a las aristas, siendo dos los más representativos:

- Grafo no dirigido: Es aquél en el que las aristas no tienen orientación, es decir, $(a, b) \in A \rightarrow (b, a) \in A$. En este caso una arista se puede considerar como un conjunto $\{a, b\}$
- Grafo dirigido: Se da cuando el grafo no cumple la propiedad de grafo no dirigido.

Sobre $G(N, A)$, el concepto de camino de a a b se define como

$$\pi(a, b) \equiv \{[n_1, n_2, ..., n_k]/n_i \in N_{\forall i=1,2,...,k}, n_1 = a, n_k = b, (n_i, n_i + 1) \in A_{\forall i=1,2,...,k-1}\}$$

Una secuencia de nodos conectados por aristas. De este modo, podemos decir que a y b están conectados sin la necesidad de que exista la arista (a, b) o (b, a) en A

En base a los caminos del grafo, podemos distinguir dos tipos:

- Grafo débilmente conexo: Es aquél en el que todo par de nodos está conectado en un sentido, es decir, $a \neq b \rightarrow \exists\pi(a, b) \vee \pi(b, a)\forall_{a,b\in N}$.
- Grafo fuertemente conexo: Es aquél en el que todo par de nodos está conectado en ambos sentidos, es decir, $a \neq b \rightarrow \exists\pi(a, b) \wedge \pi(b, a)\forall_{a,b\in N}$.

  Esta distinción sólo se tiene en cuenta cuando el grafo es dirigido. En el caso de grafos no dirigidos, la reciprocidad de las aristas hace que ambas definiciones de grafo conexo sean equivalentes.

- Grafo no conexo: Se da cuando el grafo no cumple ninguna propiedad de grafo conexo.

2.1.1 Métricas del grafo

Se pueden aplicar sobre los grafos una cantidad innumerable de medidas[4] que evalúan las propiedades topológicas del grafo. Estas características influyen directamente en la mecánica de los procesos ejecutados sobre el grafo, la creación y transmisión de información en el caso de las redes sociales. Estas métricas tienen dos categorías: de conectividad y de distancia.

Las métricas de distancia tienen como principal elemento de conteo los saltos o aristas que distan entre dos o más nodos en base a unos criterios. Generalmente se aplican sobre la componente conexa más grande del grafo. Estas son las medidas de distancia más habituales:

- Cercanía: Es la media del camino más corto de un nodo respecto a los demás. Esta medida indica el grado de participación del nodo en un grafo, cuanto menor es el valor de su cercanía más rápido llega y transmite la información al resto de nodos.
- Excentricidad/Diámetro: La excentricidad se aplica sobre los nodos y es el máximo de la distancia de los caminos del nodo. El diámetro se aplica sobre el grafo y es la excentricidad máxima de entre todos sus nodos. Esta medida se relaciona con el tiempo necesario para que una pieza de información se propague en todo el grafo.

Por otra parte tenemos las métricas de conectividad, que miden el grado de cohesión de los nodos del grafo. De entre todas estas medidas hay tres que son las más significativas:

- Grado de un nodo: Define la cantidad de aristas que posee un nodo. En caso de grafos dirigidos podemos diferenciar entre el grado de salida y de entrada.

- Distribución en grado k del nodo: Dado un k, es la probabilidad de que un nodo escogido al azar del grafo tenga grado k.
- Coeficiente de clustering: Medido de forma local en el nodo, es la proporción de vecinos del nodo que son vecinos entre sí. Dicho de otra forma, mide la cantidad de triángulos cerrados del nodo respecto del total[1]. El clustering es un indicador potente de localidad, utilizado para detectar comunidades.
- Mayor componente conexa: Es el subconjunto con la máxima cantidad de nodos tales que forman un grafo conexo. En grafos dirigidos se distingue entre componente fuertemente y débilmente conexa. En las redes sociales esta componente conexa está formada por alrededor del 70 %-90 % de los nodos.

Hay infinidad de métricas que pueden aplicarse sobre el grafo para obtener propiedades muy concretas. Las medidas mencionadas son las que tienen mayor uso, ya que obtienen las propiedades más relevantes de la estructura topológica del grafo.

2.2 Análisis de redes sociales

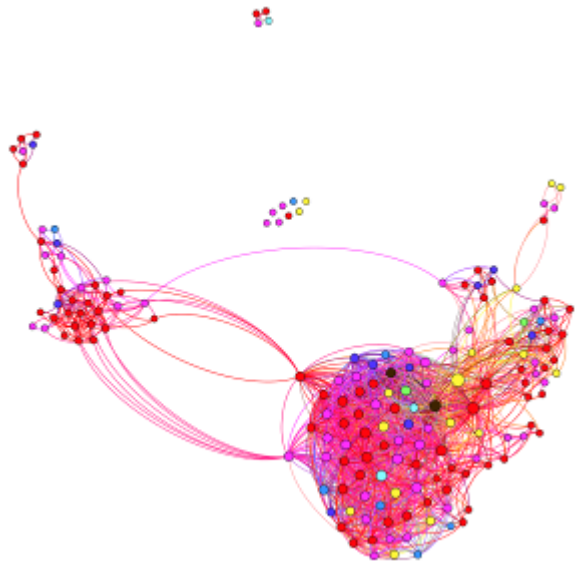

Figura 2.2: Representación gráfica de una red social. Se puede observar cómo los nodos tienden a agruparse en comunidades.

El análisis de una red social es el proceso de estudiar estas estructuras a través de la teoría de grafos (2.1)[6]. Las redes sociales suelen estar formadas por varios grafos paralelos (usuarios, interacciones, menciones, entre otros) cuyos nodos y aristas tienen datos adicionales tales como el tiempo, nombres, Hashtag, imágenes, datos personales, etc.[5] En base a esto se diseñan algoritmos y métricas propias que manejan estos datos.

[1] Si el nodo tiene n vecinos, hay $\binom{n}{2}$ combinaciones posibles.

Otra de la incógnitas que pretenden resolver estos estudios es el proceso de formación de la topología de una red social, intentando definir un modelo y un algoritmo que defina la evolución de estas redes. Una vez más, de entre todos los modelos que se han propuesto se tratarán tres . Estos modelos serán analizados y modificados en el capítulo 3, implementados en el capítulo 4 y puestos a prueba en la fase de pruebas(5).

2.2.1 Modelo de Erdös-Rényi

Fue propuesto por Paul Erdös y Alfréd Rényi en 1959 y es uno de los más antiguos que existen. Este modelo propone que todos los grafos de n nodos y M aristas son equiprobables. Existen dos variantes de este modelo:

- Se define con n nodos y con $M \leq \binom{n}{2}$[2] aristas. El grafo $G(n, M)$ es escogido aleatoriamente entre todos los grafos que se pueden construir con n nodos y M aristas.
- Se define con n nodos y con p, siendo p la probabilidad con la que se añade una arista al grafo. El grafo $G(n, p)$ se construye añadiendo todos los nodos y posteriormente todas las aristas, cada una con probabilidad p.

La relación entre n y p define la conectividad del grafo, pero el resto de propiedades quedan en manos del azar. El modelo no fue diseñado para simular redes sociales (ni siquiera existían cuando este modelo fue propuesto), no obstante se podría intentar ajustar a la fase temprana de una red social con sólo un puñado de usuarios.

2.2.2 Modelo de Barabási-Albert

Diseñado por Albert-László Barabási y Réka Albert en 1999, es un algoritmo que genera grafos cuyos nodos tienen un grado que sigue una distribución de tipo ley potencial ($y = ax^{-k}$) y sus aristas tienen conexión preferencial. El algoritmo sigue un proceso iterativo de añadir nodos al grafo y conectar el nuevo vértice a nodos ya existentes del grafo, siendo más probable conectarse a los que ya tienen muchas conexiones.

La red $BA(n, m_0, d)$ comienza con un conjunto inicial de $m_0 \geq 2$ nodos, cada uno con grado mayor o igual a uno. Por comodidad, muchos algoritmos que implementan el modelo de Barabasi comienzan con un grafo completo como conjunto inicial. Desde ese momento y hasta que el grafo alcanza el tamaño de n nodos el algoritmo va añadiendo uno a uno nodos y creando $d < m_0$[3] aristas nuevas que lo conectan al grafo.

Está demostrado que este modelo sigue una distribución de grado de tipo ley potencial y la conexión preferencial se ajusta mejor que el modelo anterior con respecto a las redes sociales. Sin embargo, no tiene en cuenta el coeficiente de clustering.

[2] $2 * \binom{n}{2}$ en caso de grafos dirigidos

[3] Cuando $d \geq m_0$ los primeros $d - m_0 + 1$ nodos se conectarían por completo al conjunto inicial, lo que se traduciría en un conjunto inicial de $d + 1$ nodos

2.2.3 Modelo de Leskovec

Este modelo fue propuesto por Jure Leskovec, Lars Backstrom, Ravi Kumar y Andrew Tomkins en 2008 una vez analizadas una cantidad enorme de datos reales de redes sociales del momento. Con los datos recopilados definieron un algoritmo que emula con más exactitud que el modelo de Barabási (2.2.2) el proceso de evolución de una red social. Explicar detalladamente el modelo acapararía más espacio del conveniente para esta sección, mas el proceso de elección a través de máxima verosimilitud es recomendable de leer. Mostramos un resumen del algoritmo $Lesk(n, N(t), \lambda, \alpha, \beta)$:

1. La creación de nodos sigue una función $N(t)$ no definida, pues varía para cada red social.
2. Cada nodo tiene un tiempo de vida siguiendo una distribución exponencial de parámetro λ.
3. El nodo se conecta al grafo (primera arista) siguiendo la ley de conexión preferencial.
4. Cada vez que crea una arista, el nodo permanece dormido un tiempo que sigue la siguiente distribución: $(1/Z)\delta^{-\alpha}exp(-\beta d\delta)$, siendo d el grado actual del nodo.
5. Cuando el nodo despierta, crea una conexión con nodos que tiene a distancia dos, cerrando triángulos. Este nodo lo elige de forma aleatoria entre los vecinos de sus vecinos.
6. Si el tiempo de vida del nodo no ha terminado se vuelve al paso 4.

En el mismo artículo se estudian posibles variantes modificando el comportamiento del paso 5. En el siguiente capítulo seguiremos esa indicación para perfeccionar el algoritmo.

2.3 Conclusiones del capítulo

Por una parte tenemos la teoría de grafos y varias medidas relevantes, las cuales pueden ser utilizadas para obtener el grado de conectividad del grafo y la distancia media entre los nodos.

Por otro lado tenemos tres modelos de generación automática de grafos, de fecha y complejidad ascendente, las cuales crean estructuras en función de una serie de parámetros.

Para este trabajo tenemos acceso a una cantidad considerable de datos de la red social Twitter. Se puede generar un grafo a través de esta información, comparar la eficacia de los algoritmos a través de las medidas y determinar si son suficientemente exactos para esta red social.

ESTUDIO DE LOS DATOS Y ALGORITMOS 3

Este capítulo se centra en el estudio de los datos que se van a tomar para comparar los grafos generados de forma aleatoria y en el análisis más ajuste de los algoritmos de generación automática de grafos. Aquí es donde verdaderamente empieza el trabajo: Es donde se utilizan los conocimientos expuestos en los capítulos anteriores para analizar y mejorar los algoritmos comentados.

En este capítulo se explican los tres algoritmos escogidos para este Trabajo Fin de grado de forma detallada. De este modo se detectarán los puntos de mejora de los mismos, generando de este modo nuestros propios algoritmos que serán probados en el capítulo 5. Allí quedará demostrado como las modificaciones propuestas en este capítulo presentan mejores resultados que los originales.

3.1 Datos

Para la realización de este trabajo se ha tenido acceso a varios ficheros, tomado de la red social Twitter en un instante de tiempo determinado. Los datos fueron recopilados de la red social, obteniendo información de algo más de diez mil usuarios. Esto representa nuestra base de referencia o población, con la cual contrastaremos los resultados obtenidos de los grafos generados de forma aleatoria. Esta población estará sujeta a un proceso de muestreo aleatorio, tal y como se verá en el capítulo siguiente.

Estos datos se dividen en cuatro ficheros:

- Tweets: Lista de tweets realizados por los usuarios, con su id, autor, mensaje, timestamp de creación, número de favoritos(FAV) y retweets(RT). En principio es un tipo de dato que no arroja información respecto a los temas que trata este trabajo. Este fichero hace referencia a la información que genera la red social, no cómo se distribuye. No obstante, el tiempo de creación de tweets puede resultar útil para hacer una aproximación al tiempo de llegada de los usuarios al grafo basándonos en el momento que hizo su primer tweet[1].
- Seguidores: En este fichero viene la lista de "seguimientos" de Twitter, cada entrada define el usuario seguidor y el usuario seguido. Visto desde la perspectiva de la teoría de grafos (2.1), este fichero está definiendo el conjunto A de aristas de Twitter. Este fichero es el que más importancia tiene para la realización de este trabajo ya que proporciona la estructura del grafo de Twitter con la que se va a contrastar todas las medidas obtenidas en los grafos generados aleatoriamente.

[1]Una aproximación bastante razonable si tenemos en cuenta que el tutorial de la aplicación invita al usuario a enviar su primer tweet.

- Timestamp de interacciones. Recoge el instante de tiempo en el que un usuario interaccionó con otro y el tipo de interacción que se produjo[2]. Aunque este fichero también puede ser usado para generar un grafo de interacciones no entra dentro de las competencias de este trabajo.
- Cantidad de interacciones. Es un fichero parecido al anterior, donde se agrupan todas las interacciones que se producen entre dos usuarios por el tipo de interacción, mostrando el conteo del grupo. Este fichero da lugar a un tipo de grafo en el que las aristas tienen asignado un peso, ampliamente utilizado en los algoritmos de búsqueda de caminos. De nuevo, el grafo generado a partir de estos datos no va a ser aplicado en este trabajo.

3.2 Algoritmos

En esta sección estudiaremos en detalle los algoritmos correspondientes a los modelos propuestos en el estado del arte (2.2) y, si procede, se aplicarán cambios con el fin de mejorarlos, siempre que éstos no hacen que los grafos generados se salgan del modelo al que pertenecen.

3.2.1 Algoritmo de Erdös-Rényi

El modelo de Erdös[3](2.2.1) tiene una antigüedad y aplicaciones que no se corresponden con las redes sociales y, básicamente, enuncia que todos los grafos tienen la misma probabilidad de aparición. Si bien este modelo queda muy lejos de la estructura de las redes sociales, puede ser interesante hacer comparaciones con grafos de muy pocos nodos donde el número de aristas es muy reducido. A partir de un determinado número de aristas este modelo sólo es útil para llegar a la conclusión de que la evolución de la estructura de las redes sociales no es producto del azar. Es una conclusión trivial, pero es necesario hacer una demostración experimental.

Dado que es una concepción puramente teórica, la primera variante del modelo tiene un algoritmo que no es concebible en cuanto a coste computacional. Este algoritmo $Erdos_1(n, M)$ tiene como parámetros de entrada:

- n: Número de nodos.
- M: Número de aristas.

Y tiene el siguiente funcionamiento:

1. El algoritmo construye todos los grafos de n nodos y M aristas.
2. El algoritmo escoge uno de los grafos generados en el paso 1 aleatoriamente.

Por otro lado, la segunda variante del modelo tiene un algoritmo que es más razonable. $Erdos_2(n, p)$ posee los siguientes parámetros:

[2]Mención, RT, FAV o respuesta.

- n: Número de nodos.
- p: Probabilidad de que la arista esté en el grafo.

Y este procedimiento

1. El algoritmo construye un grafo vacío de n nodos.
2. Para cada par de nodos, el algoritmo le asigna una arista con probabilidad p.

Este algoritmo tampoco resulta exacto ya que, por producto del azar, la invocación de éste puede dar grafos con un número de aristas distinto.Ya que la primera concepción del modelo es la más concreta, se ha tomado ese como referencia en este trabajo. Ha sido necesaria una reinterpretación del algoritmo para que pueda aplicarse de forma secuencial y eficiente.

El algoritmo $Erdos_3(n, M)$ tiene los mismos parámetros que el primero, pero ahora sigue un procedimiento distinto:

1. El algoritmo construye un grafo vacío de n nodos.
2. Si el grafo contiene M aristas termina su ejecución
3. Si no, se añade una arista del grafo aleatoriamente y se vuelve al paso 2

De este modo el algoritmo construye un grafo que sigue el Modelo de Erdös de forma eficiente. Se ajusta al modelo ya que, al ser todas las aristas igual de susceptibles de ser añadidas, todos los grafos de los nodos y aristas especificados tienen la misma probabilidad de ser construidos en el proceso.

3.2.2 Algoritmo de Barabási-Albert

El modelo de Barabási[1](2.2.2) fue planteado para aportar una aproximación a la evolución de las redes sociales en las que los nodos se añaden en el grafo secuencialmente y las aristas del nuevo nodo tienen como objetivo aquellos nodos con muchas conexiones con mayor probabilidad que los que tienen pocas aristas conectadas.

Antes de definir el algoritmo es fundamental conocer el proceso de conexión preferencial, también conocido como "ventaja acumulativa". En el modelo de Barabási, la probabilidad de que un nodo i sea escogido para una conexión es:

$$p_i = \frac{d_i}{\sum_{j \in N} d_j}$$

Siendo d_k el grado del nodo k.

Esto tiene unas implicaciones interesantes, ya que este algoritmo hace uso de las métricas de teoría de grafos, realizando mediciones en cada paso e influyendo así el estado del mismo en su siguiente instancia. Es una técnica que ha sido

empleada en otros modelos, también en el de Leskovec, por lo que es conveniente familiarizarnos con este proceso.

El algoritmo de Barabasi, $Barabasi(n, m0, d)$, tiene los siguientes parámetros de entrada:

- n: Número de nodos.
- $m0$: Número de nodos del conjunto inicial.
- d: Número de aristas que se conectan al nuevo nodo.

Y construye un grafo aleatorio siguiendo las reglas básicas que se expusieron en el estado del arte con el siguiente proceso:

1. Construye un grafo completo de $m0$ nodos.
2. Si el grafo contiene n nodos termina su ejecución.
3. Si no, añade un nuevo nodo en el grafo.
4. El algoritmo añade d nuevas aristas que conectan el nuevo nodo con el grafo siguiendo el proceso de conexión preferencial y vuelve al paso 2.

Lamentablemente este algoritmo estaba pensado para la generación de grafos no dirigidos, donde el grado de entrada del nodo es el mismo que el grado de salida. La métrica del grado de salida del grafo resultante no contendrá información alguna, ya que todos los nodos tendrán d aristas salientes. Es un problema que no puede atajarse sin hacer que el algoritmo modificado se salga del modelo, por lo que sólo se tendrá en cuenta el grado total a la hora de generar las gráficas.

Otro problema que ha sido detectado es que este algoritmo ha sido creado sin tener en cuenta el clustering que se produce en las redes sociales. En el caso extremo, cuando el algoritmo está configurado con $d = 1$, el grafo resultante tiene una estructura en forma de árbol, donde no se producen cierre de ciclos, haciendo que el índice de clustering se concentre en el 0 para valores altos de n. De nuevo, no es posible modificar el algoritmo sin que se salga del modelo.

3.2.3 Algoritmo de Leskovec-Backstorm-Kumar-Tomkins

El modelo propuesto por Leskovec[7](2.2.3) es uno de los más recientes que se han propuesto. No obstante, el modelo carece de un algoritmo bien definido, pues se limita a enumerar los procesos que sigue el modelo. La implementación de este algoritmo es compleja ya que el proceso de creación de nodos y aristas siguen un orden marcado por los tiempos en que son asignados, los cuales se obtienen de forma aleatoria. Por tanto, es necesario idear un sistema que acumule los sucesos y los ejecute de forma ordenada, al igual que almacenar un registro de nodos y sus tiempos de vida para determinar cuándo éstos dejan de generar nuevas aristas.

Teniendo en cuenta todas las consideraciones anteriores, una primera versión del algoritmo es generada $Lesk_1(n, K, \lambda, \alpha, \beta)$, el cual sigue al pie de la letra todos los procesos que se especifican en el modelo. El algoritmo cuenta con los siguientes campos:

- n: Número de nodos.
- K: Parámetro de la función que calcula el tiempo de espera entre creación de nodos.
- λ: Parámetro de la distribución exponencial que define el tiempo de vida de los nodos.
- α: Parámetro de la distribución potencia-exponencial que define el tiempo de inactividad de un nodo entre creación de aristas.
- β: Parámetro de la distribución potencia-exponencial que define el tiempo de inactividad de un nodo entre creación de aristas.

El algoritmo se divide en cuatro rutinas: La principal, la de gestión de sucesos, la de creación de aristas y la de creación de nodos. Se explicará uno a uno la algoritmia de cada proceso:

Generación de nodo:

1. El algoritmo crea un nuevo nodo y lo añade al grafo.
2. Fija su tiempo esperado de vida, siguiendo la distribución $exp(\lambda)$
3. Crea una arista que conecta el nuevo nodo con el grafo en base a conexión preferencial.
4. Calcula el tiempo que el nodo pasará inactivo, este tiempo seguirá la distribución $(1/Z)\delta^{-\alpha}exp(-\beta d\delta)$.
5. Si el tiempo obtenido es mayor al tiempo de vida obtenido en el paso 2, se salta al paso 7.
6. Si no, añade una tarea de generación de arista en la lista de sucesos[3].
7. Si el grafo tiene n nodos, termina la ejecución.
8. Si no, calcula el tiempo estimado de creación del nuevo nodo. El tiempo obtenido se calcula a través de la función $t_{new} = \sqrt{n + 1/K}$ siendo n el numero de nodos del grafo y K una constante.
9. El algoritmo crea una nueva tarea de generación de nodo en la lista de sucesos[3].

Generación de arista:

[3]Este suceso se ordenará en la lista en función del tiempo obtenido.

1. El nodo origen es el nodo que creó la tarea.
2. El nodo destino se escoge de entre los nodos de distancia dos, para crear una arista que cierre un triángulo.
3. Calcula el tiempo que el nodo pasará inactivo, este tiempo seguirá la distribución $(1/Z)\delta^{-\alpha}exp(-\beta d\delta)$.
4. Si el tiempo obtenido es mayor al tiempo de vida obtenido, termina su ejecución.
5. Si no, añade una tarea de generación de arista en la lista de sucesos[3].

Gestión de la lista de sucesos:

1. El constructor coge (elimina) el primer suceso de la lista de sucesos.
2. Llama a la rutina dependiendo del suceso, es decir, a generación de nodo o a generación de arista.
3. El constructor comprueba que la lista no está vacía, en ese caso vuelve al paso 1.

Proceso principal:

1. El algoritmo inicia una tarea de creación de nodo en la lista de sucesos.
2. Inicia la gestión de la lista de sucesos

La constante adición, ejecución y eliminación de sucesos de la lista ordenada cronológicamente simula el proceso de evolución de una red social siguiendo las propiedades enunciadas en la sección 2.2.3 de forma secuencial, simulando un desarrollo en tiempo continuo.

Sin embargo, la ejecución de este algoritmo genera grafos con demasiado índice de clustering, pues siempre intenta generar aristas que cierren triángulos.

3.2.4 Modificación del algoritmo de Leskovec

Para corregir ese funcionamiento se ha añadido un nuevo parámetro y se ha ajustado el proceso de generación de aristas, obteniendo la segunda versión del algoritmo. $Lesk_2(n, K, p, \lambda, \alpha, \beta)$ tiene un nuevo campo:

- P probabilidad con la que crea una arista de cierre de triángulo. $(1 - P)$ es la probabilidad de elegir una arista de conexión preferencial.

Y ha modificado la rutina de generación de aristas:

1. El nodo origen es el nodo que creó la tarea.

2. Con probabilidad (P) el nodo destino se escoge de entre los nodos de distancia dos, para crear una arista que cierre un triángulo. Con probabilidad $(1 - P)$ el nodo destino se escoge a través de conexión preferencial.

3. Calcula el tiempo que el nodo pasará inactivo, este tiempo seguirá la distribución $(1/Z)\delta^{-\alpha}exp(-\beta d\delta)$.

4. Si el tiempo obtenido es mayor al tiempo de vida obtenido, termina su ejecución.

5. Si no, añade una tarea de generación de arista en la lista de sucesos[3].

3.3 Conclusiones del capítulo

Hemos hecho un breve estudio sobre los datos disponibles para este Trabajo Fin de Grado, identificando el conjunto que es aplicable al estudio que estamos realizando. Por otro lado hemos descrito y analizado los algoritmos de generación aleatoria de grafos que van a ser empleados, realizando algunas modificaciones para ajustarlos a las redes sociales. En este punto del trabajo, es el momento de definir la API que aplicará estos algoritmos de generación y realizará medidas sobre sus productos con el fin de hacer las comparaciones con los datos reales.

4 Diseño y Desarrollo

A lo largo de este capítulo se especificará el diseño de la API que procesará los datos de Twitter, generará automáticamente los grafos de acuerdo a los algoritmos planteados en el capítulo anterior (3) y aplicará sobre ellos las medidas, guardando sus resultados en sendos ficheros para su posterior uso en el capítulo 5.

4.1 Diseño

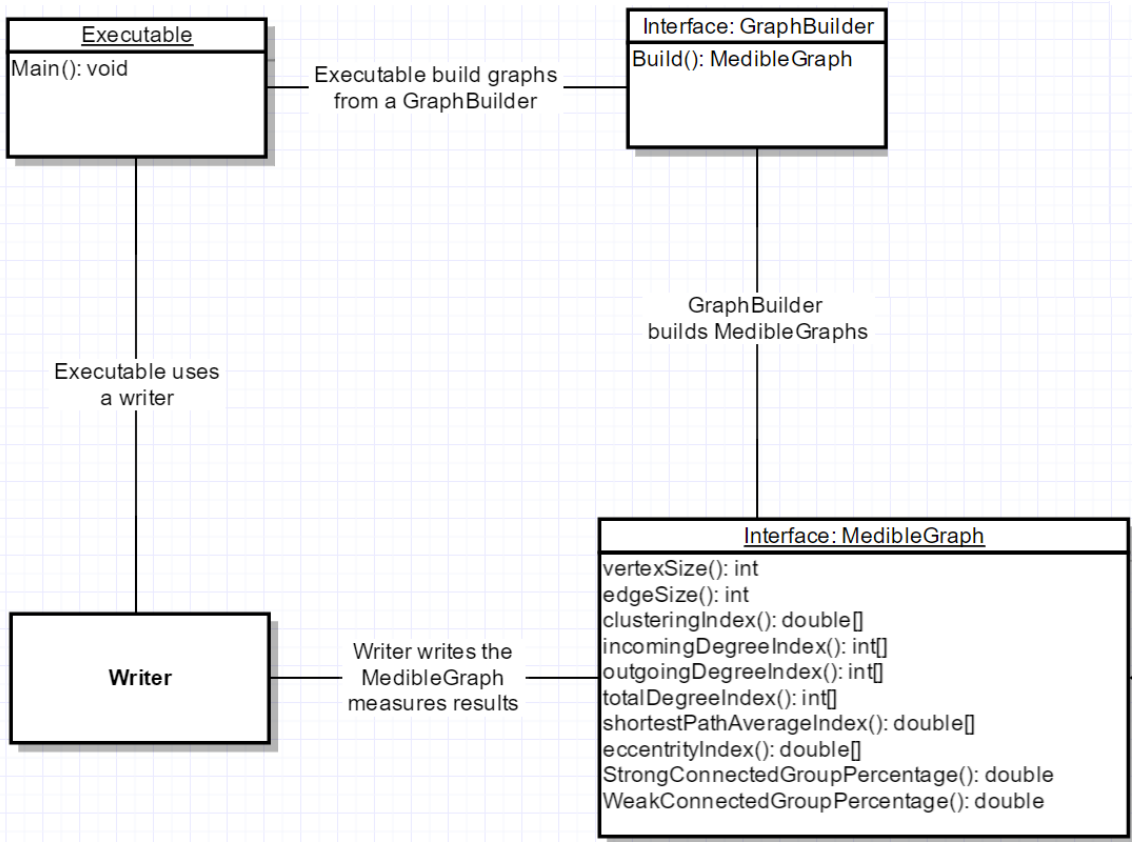

Figura 4.1: Primer diagrama UML de la aplicación. Presentado en su máxima abstracción: la API genera grafos de forma aleatoria con el GraphBuilder y escribe sus medidas a través del Writer.

Tal y como se presenta en la Figura 4.1 la API puede entenderse como un programa que construye grafos medibles y guarda el resultado de las medidas a través de un módulo de escritura (Writer). Sin profundizar más en el diseño o lenguaje de programación se han definido cuatro módulos principales.

4.1.1 Módulo ejecutable

Es el módulo que se ejecuta al arrancar el programa y el encargado de gestionar y comunicar el resto de interfaces y módulos. Su lógica (método main) sigue la siguiente estructura:

1. Llamada al módulo GraphBuilder
2. Llamada al módulo Writer, pasando por parámetro la estructura MedibleGraph obtenida en el paso 1,

Este módulo será modificado múltiples veces a lo largo de la fase de pruebas(5) para hacer las llamadas a todos los tipos de GraphBuilder que se vayan implementando.

4.1.2 Módulo de escritura

Este módulo Writer recibe como parámetro un grafo medible (MedibleGraph), persistiendo todos los resultados a las llamadas de medidas en algún formato. Dado que la API sólo se utiliza para obtener estos datos y no hay nada que deba ser securalizado, lo más lógico y cómodo es que se persista en ficheros de texto plano.

Los datos obtenidos se procesarán y almacenarán con un formato simple y entendible pues un programa externo (GNUPlot) leerá esos ficheros para generar gráficas.

4.1.3 Interfaz de Medidas

La interfaz MedibleGraph contiene toda la lógica correspondiente a las medidas del grafo, haciendo que cualquier módulo que implemente esta interfaz sea manejable por el módulo de escritura (4.1.2). Los métodos que se han definido son los que se especificaron previamente en el estado del arte (2.1.1), añadiendo medidas generales del grafo (número de nodos y aristas) y las variantes necesarias de las medidas de grado y componente conexa que deben aplicarse en caso de grafos dirigidos.

Con estas aclaraciones, la interfaz define los siguientes métodos:

- vertexSize: Devuelve el número de nodos del grafo.
- edgeSize: Devuelve el número de aristas del grafo.
- clusteringIndex: Devuelve un vector de números entre el 0 y el 1 correspondiente al clustering local de cada nodo del grafo y finalmente el clustering global del grafo[1]. Cualquier número fuera del rango especificado corresponde a un clustering no definido[2].
- incoming/outgoing/totalDegreeIndex: Devuelve un vector de enteros positivos o cero correspondiente al grado de entrada, salida o total de un

[1] Esto quiere decir que la dimensión del vector es igual al número de nodos del grafo mas uno.
[2] Este suceso se da cuando el nodo tiene menos de dos vecinos

nodo, respectivamente. Él último nodo corresponde con el grado medio del grafo[1].

- shortestPathAverageIndex: Devuelve un vector de números positivos correspondiente a la cercanía media de cada nodo de la componente conexa más grande, siendo el último número la media general del grafo[1,3].
- eccentrityIndex: Devuelve un vector de enteros positivos correspondiente a la excentricidad de cada nodo de la componente conexa más grande, el último entero corresponde con el diámetro del grafo[1,3]. Cabe la posibilidad de que esta medida devuelva 0, esto implica que la componente conexa está compuesta por un nodo o ninguno.
- strong/weakConnectedGroupPercentage: Devuelve un número entre 0 y 1, que es la proporción de nodos del grafo que pertenecen a la componente fuertemente conexa o débilmente más grande, respectivamente.

4.1.4 Interfaz de construcción de grafos medibles

La interfaz GraphBuilder contiene la lógica necesaria para la construcción de los grafos medibles, permitiendo que cualquier módulo que implemente la interfaz pueda ser utilizado por el módulo ejecutable. Cada implementación de grafo medible tendrá varios constructores, mínimo uno por cada modelo especificado en el estado del arte (2.2.1).

4.2 Desarrollo

Esta sección tratará la implementación del diseño planteado. En ésta se especificará el entorno y el lenguaje de programación escogido, así como las implementaciones a las interfaces definidas anteriormente (4.1).

4.2.1 Lenguaje y entorno de programación. Diagrama de clases

El IDE escogido para el desarrollo de la aplicación ha sido Eclipse Neon, usando Java como lenguaje de programación. Adicionalmente se ha utilizado Maven para gestionar el proyecto ya que permite descargar e inyectar las dependencias de librerías externas por medio de un único fichero de configuración. Estas herramientas han permitido codificar la API de forma rápida y sencilla, permitiendo la reproducción del mismo entorno de desarrollo en la mayoría de equipos ya que eclipse tiene soporte para un amplio abanico de SO.

Una vez definido el lenguaje y entorno de desarrollo se ha definido la implementación de las interfaces tal y como se muestra en el diagrama 4.2. Las clases especificadas de este diagrama más la implementación de los módulos ejecutable y de escritura del primer UML forman la totalidad de la API.

4.2.2 Implementación de la interfaz de medidas

Esta implementación lleva a cabo dos objetivos relacionados: Diseñar la estructura de datos del grafo y la lógica que permite aplicar las medidas sobre la estructura definida.

[3]Suponiendo que el resto de nodos que no pertenecen a la componente conexa más grande poseen unos valores de cercanía despreciables.

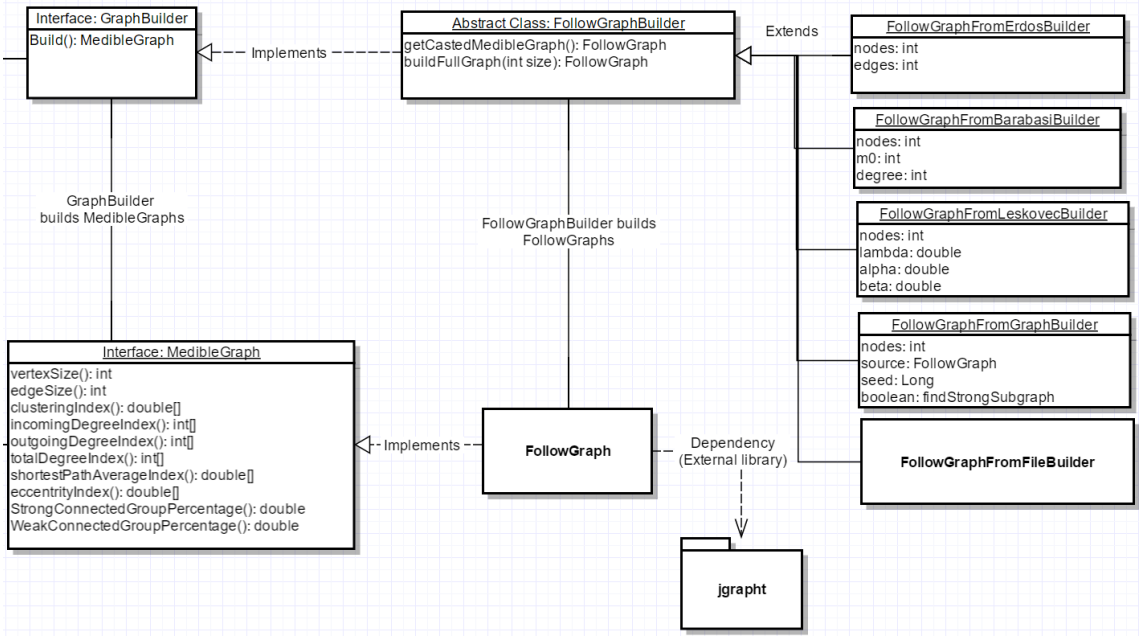

Figura 4.2: Segundo diagrama UML de la aplicación. Aquí se definen las implementaciones de las interfaces especificadas en el primer UML (Figura 4.1). Se ha escogido Java como implementación ya que posee librerías de gestión de grafos muy completas.

Se ha utilizado la librería jgrapht para este primer objetivo, contiene todos los métodos necesarios para la creación de grafos, gestión y acceso a nodos y aristas. También contiene interfaces para el cálculo de los caminos más cortos y búsqueda de vecinos. El programador sólo debe hacer uso de los métodos disponibles para gestionar la topología del grafo y obtener los datos que necesite.

La lógica de las medidas ha sido implementada siguiendo la algoritmia definida en la teoría de grafos[4] utilizando los métodos que proporciona jgrapht.

Por último, este Trabajo Fin de Grado tiene acceso a datos de seguidores, interacciones y mensajes. Los datos que se mapearán a un grafo y se compararán con los generados automáticamente serán los procedentes del fichero de seguidores, que serán almacenados en la clase FollowGraph, un grafo dirigido cuyos nodos están identificados por un número entero.

4.2.3 Implementación de la interfaz de construcción

Para construir objetos de la clase FollowGraph se ha definido una clase abstracta FollowGraphBuilder que implementará la interfaz GraphBuilder, la cual proporciona dos métodos que comparten todas las clases que hereden de ellas.

- buildFullGraph: Método de construcción de grafos completos.
- getCastedMedibleGraph(): Sigue la misma lógica que el método de interfaz build, el objeto que devuelve está "casteado" a clase FollowGraph. Utilizado principalmente para hacer pruebas con los objetos devueltos.

Se han implementado cinco herencias de la clase abstracta.

- FollowGraphFromFileBuilder: Construye el grafo a partir de un fichero, más concretamente el fichero que contiene datos reales sobre usuarios y seguidores de twitter. Cada linea del fichero corresponde a una arista, en caso de que los nodos de la arista no estén definidos los añade al grafo. El algoritmo del constructor lee todo el fichero, añadiendo nodos y aristas hasta que alcanza el final del mismo.
- FollowGraphFromGraphBuilder: Construye un subgrafo de uno ya existente, seleccionando nodos al azar y replicando las aristas correspondientes a esta estructura reducida. Esta clase se implementó por la utilidad que supone tener un constructor capaz de obtener muestreos aleatorios d euna población.
- FollowGraphFromErdosBuilder: Construye un grafo siguiendo el algoritmo del modelo de Erdös, tal y como se especifica en la sección 3.2.1.
- FollowGraphFromBarabasiBuilder: Construye un grafo siguiendo el algoritmo del modelo de Barabási, tal y como se especifica en la sección 3.2.2.
- FollowGraphFromLeskoveciBuilder: Construye un grafo siguiendo el algoritmo del modelo de Leskovec, tal y como se especifica en la sección 3.2.3.

FASE DE PRUEBAS

5

En este capítulo se hará uso de la API especificada en el capítulo anterior para analizar y comparar la eficacia de los grafos generados por el programa. Este contraste se realizará en base a diversas gráficas, explicando las diferencias que puedan aparecer en las mismas.

En primer lugar se detallará el entorno usado en la ejecución de la aplicación y el programa externo encargado de generar las gráficas a partir de las medidas, almacenadas en ficheros de texto plano. Entonces se mostrarán las gráficas del grafo obtenido a partir de datos reales de Twitter, exponiendo brevemente los datos de las medidas más relevantes. A partir de entonces se hará un estudio individual de cada algoritmo de generación automática de grafos, comparando los valores obtenidos en sus medidas con los del grafo de Twitter, detectando las diferencias más significativas y razonando el motivo de estos resultados.

5.1 Entorno de puebas

5.1.1 Sistema Hardware

	PC
Procesador	Intel Core i7-5820K 3,30 GHz, 6 núcleos
RAM	16 GB
SO	Windows 10 Home 64bits

Tabla 5.1: Especificaciones técnicas del ordenador de pruebas.

El entorno de pruebas consiste en un ordenador personal, cuyas especificaciones se detallan en la tabla 5.1. Éste será el encargado de el programa de generación y medición de grafos, guardando los resultados en ficheros de texto. El código java se ejecuta en un jre 1.8 incorporado en el IDE de Eclipse Neon.

Dada la simplicidad del programa este entorno de pruebas podría replicarse en entornos cuyas especificaciones sean mucho menores. Los tiempos de cálculo, salvando las medidas de cercanía y excentricidad que implican cálculo de caminos, son muy bajos si se manejan grafos que superen las varias decenas de miles de nodos.

El cálculo de caminos, al ser un problema del tipo NP-Completo, requiere un esfuerzo computacional muy alto. A partir de grafos de mil nodos las medidas de excentricidad y cercanía requieren tiempos que pueden alcanzar varias horas. Es por eso que el código contempla la opción de ejecutar sólo las medidas "ligeras" a fin de que el programa pueda trabajar a distintas velocidades.

5.1.2 Formato y representación gráfica de las medidas

En la fase final de la ejecución, el programa accede al módulo de escritura para persistir todos los cálculos realizados en ficheros de texto, uno por cada medida que se aplique. El formato de salida de los ficheros es el siguiente[1]:

- Para medidas de incoming/outgoing/totalDegreeIndex: Estructurado en forma de tabla con tres columnas: número de aristas[2], número de nodos y proporción de nodos.
- Para el índice de clustering: Está estructurado como una tabla de veinte filas y tres columnas La razón de que tenga veinte filas fijas es que se agrupa el rango de los valores de cluster (que va de 0 a 1) en veinte bandas de longitud 0.05. Las columnas tienen la siguiente definición: Banda de clustering (al que se le asigna un valor múltiplo de 0.05), número de nodos y proporción de nodos.
- Para la cercanía media: sigue el mismo esquema de agrupación que el fichero de clustering. No obstante el número de bandas depende del camino corto más largo[3].
- Para el índice de excentricidad: sigue el mismo esquema que en las medidas de grado, constando de tres columnas: Excentricidad, número de nodos y proporción de nodos.

También generará un resumen de las medidas del grafo (número de nodos, aristas, clustering medio, proporción de nodos en la mayor componente conexa, etcétera)

Se ha usado gnuplot para tratar los datos persistidos. Ofrece una interfaz rápida y fácil de usar con la que puedes leer y representar gráficamente tablas de datos, añadiendo toda la información necesaria (como nombres de gráficas o valores de los ejes) y exportando el resultado en formato png.

5.2 Grafo de Twitter

5.2.1 Configuración de parámetros

Ya que el algoritmo del constructor se limita a leer los datos de un fichero de texto, no hay consideraciones respecto al funcionamiento del mismo. El constructor siempre genera el mismo grafo pues carece de parámetros que sean configurables.

Por cuestiones de eficiencia no es viable ejecutar en este grafo las medidas de excentricidad y camino más corto medio.

[1] Nota: Filas separadas por saltos de línea y columnas separadas por espacio.
[2] Entrantes, salientes y totales, respectivamente.
[3] El dominio de las bandas $\in [1, d(g)]$ siendo $d(g)$ el diámetro del grafo.

5.2.2 Propiedades generales

- Número de nodos: 10029
- Número de aristas: 560227
- Tamaño de la componente fuertemente conexa: 97.71 %
- Tamaño de la componente débilmente conexa: 100.0 %

Los valores obtenidos en las componentes conexas son especialmente altos, con la muy improbable situación de que la componente débilmente conexa sea la totalidad del grafo. Esto tiene una explicación plausible: Los datos han sido recogidos por algún algoritmo que recorría la lista de seguidores de los usuarios de forma recursiva, el cual se ha truncado en algún momento. No hay ningún nodo con grado 0 ya que el grafo ha sido creado a partir del fichero de seguidores (aristas).

5.2.3 Grado de los nodos

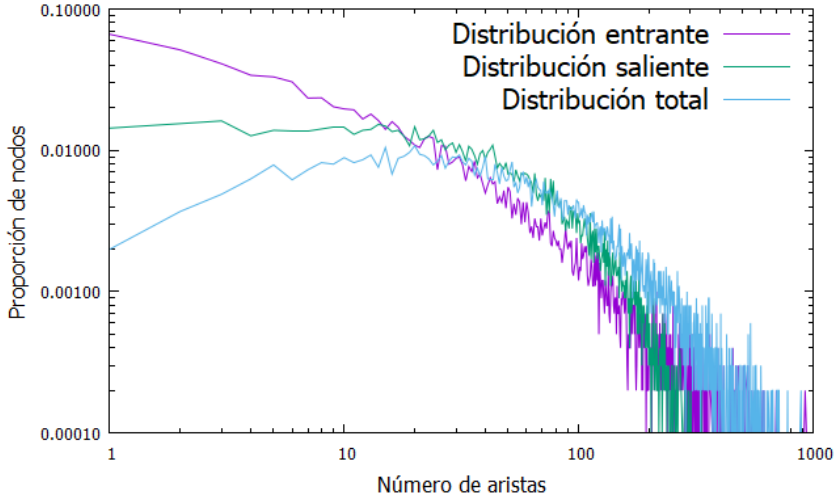

Figura 5.1: Gráfica de la distribución en grado de los nodos de Twitter a escala logarítmica.

La figura 5.1 muestra el comportamiento del grado de los nodos, el cual parece seguir una función de distribución de tipo exponencial para las aristas entrantes. Tanto el índice de aristas salientes como de las totales tienen un comportamiento distinto al inicio, pero luego acaban alineándose con el índice de aristas entrantes.

El grado medio del grafo (numero de aristas por nodo) es de 55.

5.2.4 Clustering

En la figura 5.2 se puede observar como la proporción de nodos alcanza su máximo (alrededor del 16 %) con índices de clustering de entre el 15 y el 20 %. La gráfica podría estar siguiendo una funcion de distribución de Poisson o Chi cuadrado, pues la gráfica presenta asimetría por la derecha.

El clustering medio del grafo es del 27.4 %

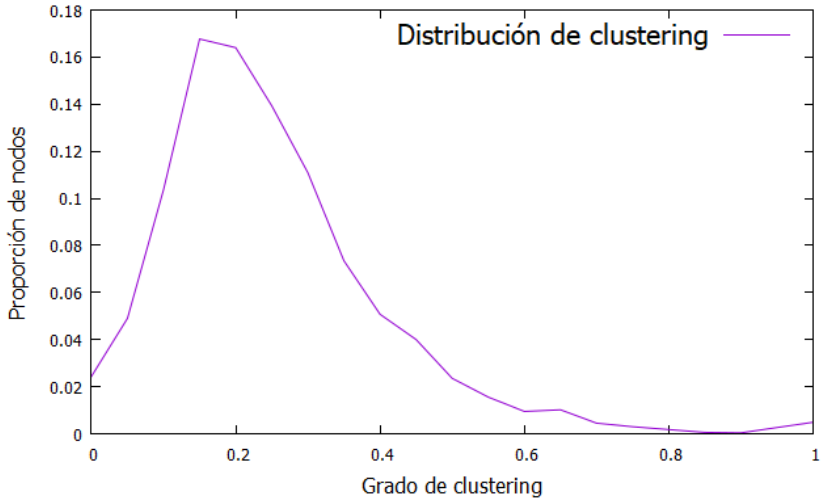

Figura 5.2: Gráfica de la distribución del índice de clustering de Twitter.

5.3 Subgrafos de Twitter

5.3.1 Configuración de parámetros

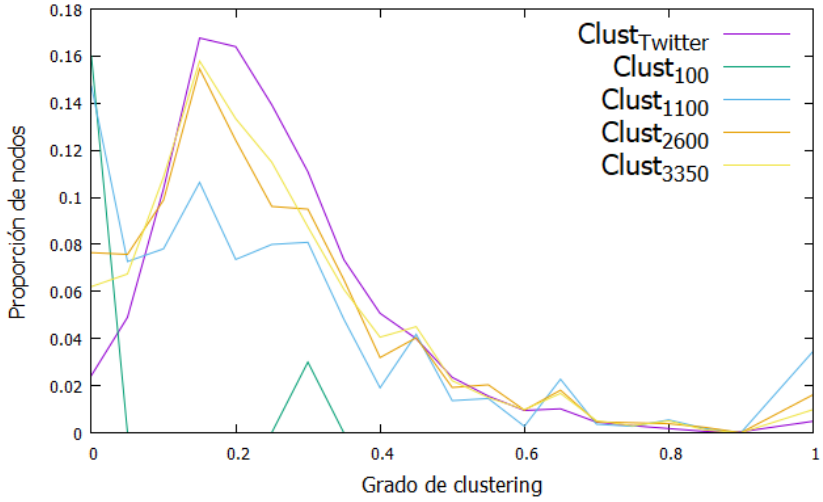

Figura 5.3: Evolución del índice de clustering del subgrafo en función del número de nodos.

El algoritmo de generación de subgrafos es configurable a través del número de nodos. Ya que toma los nodos del grafo inicial de forma aleatoria y las aristas que aplican al nuevo grafo podemos observar cómo las medidas se van ajustando a las del grafo original a medida que aumentamos el número de nodos (Figura 5.3).

Subgrafos de tamaño menor de 500 no tienen ningún valor estadístico, sin embargo, a partir de 1000 nodos empieza a apreciarse un ajuste considerable. Llega un momento en el que la mejora de precisión en relación al tamaño es despreciable. Llegados a ese punto resulta conveniente truncar la búsqueda, ya que una muestra con buena relación precisión/tamaño permitirá hacer comparaciones verosímiles de las medidas complejas (excentricidad y cercanía) sin que suponga una enorme carga computacional, pudiendo extrapolar las conclusiones obtenidas del muestreo a toda la población.

5.3.2 Propiedades generales

Generando y analizando un subgrafo de 2500 nodos tenemos las siguientes propiedades generales:

- Número de nodos: 2500
- Número de aristas: 33537
- Tamaño de la componente fuertemente conexa: 77.8 %
- Tamaño de la componente débilmente conexa: 98.12 %

La componente fuertemente conexa tiene una proporción que se ajusta mejor a la media que tienen las redes sociales, entre el 60 y el 80 %.

5.3.3 Grado de los nodos

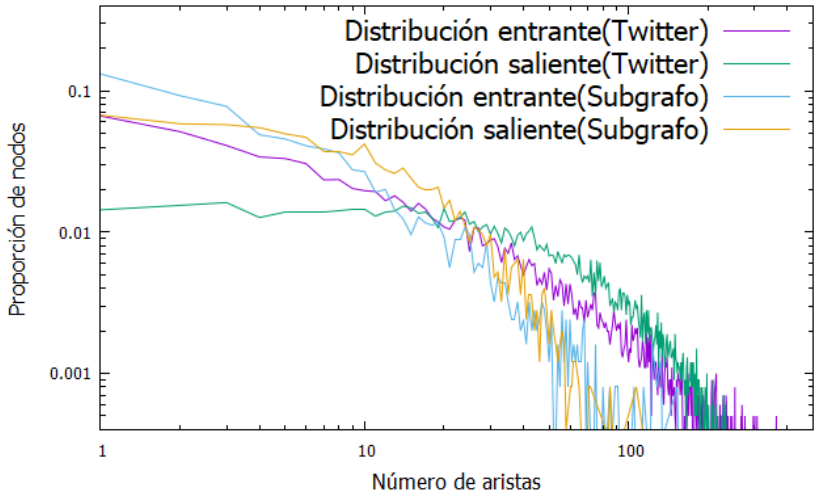

Figura 5.4: Gráfica comparativa de la distribución de grado del grafo original y el subgrafo, en escala logarítmica.

Se puede apreciar como las distribuciones de grado son similares a las del grafo original. En las redes sociales existen usuarios que no tienen seguidores, situación que no se contempla en la muestra original, la cual todos los nodos tienen grado 1 o más. Por otra parte, el índice de aristas salientes del subgrafo se ha vuelto muy parecido a el de aristas entrantes.

Esta gráfica nos hace plantearnos la pregunta de si es posible que la muestra refleje mejor la realidad que el grafo original respecto a la distribución en grado de los nodos. Posiblemente sea así.

El grado medio del grafo es de 13.

5.3.4 Clustering

Una vez más, el subgrafo parece comportarse bien respecto al índice de clustering, siguiendo la tendencia que habíamos observado en la figura 5.3. El clustering medio del subgrafo es del 26.67 %, un 0.8 % menor que el del grafo original.

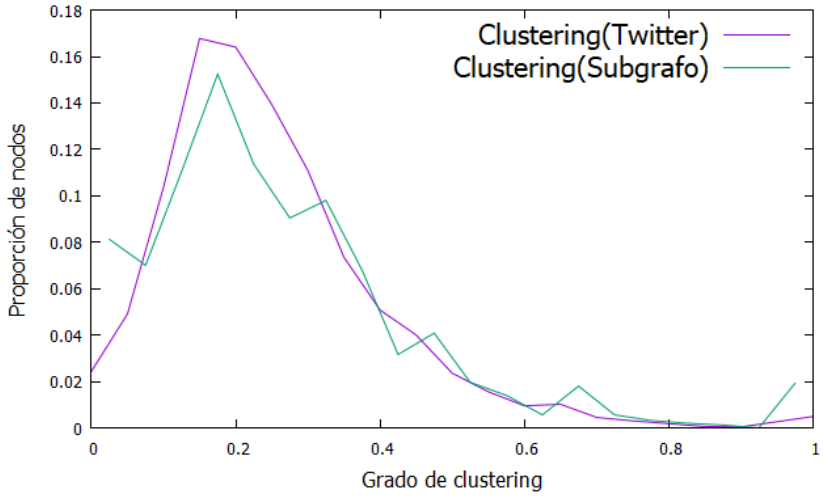

Figura 5.5: Gráfica comparativa del clustering del grafo original y el subgrafo.

5.3.5 Medidas complejas

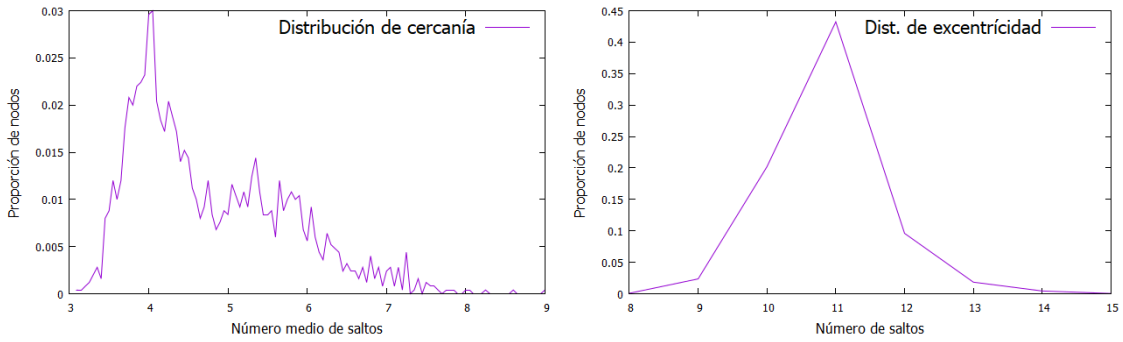

Figura 5.6: A la izquierda el índice de cercanía del subgrafo. A la derecha el índice de excentricidad.

La métrica de cercanía (Figura 5.6, izquierda) muestra como los nodos tienen, de media, distancias cercanas al 4 (3-4-5) respecto al resto de nodos de la componente conexa. Esa proporción se reduce drásticamente a partir de 6, haciendo el resto de valores muy inusuales.

Por otro lado, la excentricidad de los nodos se concentra en los valores 10 y 11, no siendo nunca menor que 8 o mayor que 15.

El valor medio de cercanía del grafo es 4.79 y el radio del grafo es 15.

5.4 Modelo de Erdös-Rényi

5.4.1 Configuración de parámetros

Este algoritmo necesita como parámetros de entrada el número de nodos y aristas. Si queremos comparar las medidas obtenidas con los datos reales de Twitter, la única forma de configurar estos parámetros es, para un número determinado de nodos, ejecutar el algoritmo anterior y obtener en número de aristas de la muestra generada.

Al aplicar las métricas a grafos obtenidos a partir del algoritmo de Erdös los resultados son, literalmente, nulos. Vamos a comentar brevemente los resultados de cada medida

5.4.2 Propiedades generales

Ya que no se ha encontrado una parametrización óptima, no tiene sentido que hablemos de las propiedades del grafo. Aún así, todos los que se han generado han tenido un numero de nodos y aristas idéntico a la de la muestra[4], una componente conexa minúscula y un grado medio de clustering igual a 0. La aleatoriedad de la generación de estos grafos hacen que, como era de esperar, no sea capaz de crear una estructura que tenga las propiedades de una red social.

5.4.3 Grado de los nodos

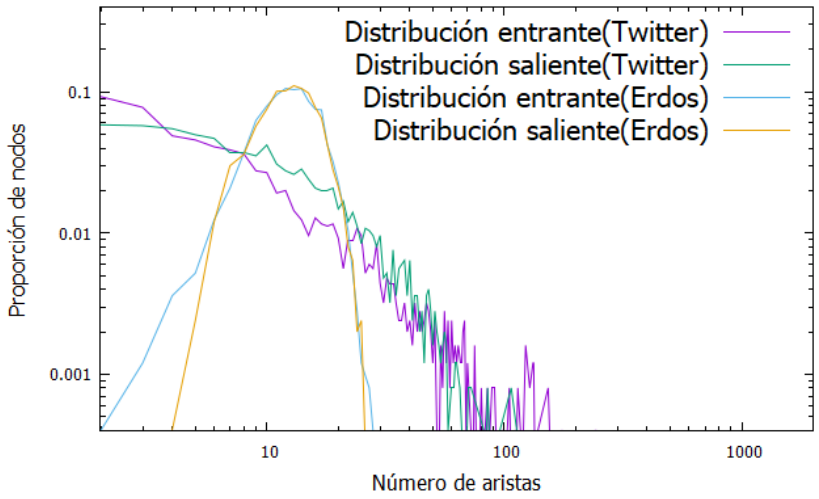

Figura 5.7: Gráfica comparativa de la distribución de grado de una muestra de Twitter de 2500 nodos y el grafo generado con el algoritmo de Erdos en escala logarítmica.

Tal y como se aprecia en la figura 5.7, la distribución de grado que tienen los grafos creados por este algoritmo no siguen una distribución de ley potencial, concluyendo en que ambos grafos no poseen una distribución de aristas equivalentes.

5.4.4 Conclusiones

Debido a la inexistencia de coeficiente de clustering ni de componentes conexas, no es posible sacar gráficas referentes al clustering, cercanía o excentrecidad. Por tanto podemos afirmar que, como era de esperar, el proceso de evolución de la red social de Twitter no sigue un modelo aleatorio.

5.5 Modelo de Barabási-Albert

5.5.1 Configuración de parámetros

El algoritmo de barabasi se configura a través del número de nodos y del grado medio del grafo. Ya que la muestra escogida de Twitter es la conformada

[4]Por como se ha definido el proceso de parametrización.

de 2500 nodos, se ha configurado el algoritmo de Barabasi para que genere algoritmos de 2500 nodos y con un grado medio de 13, que es el grado medio obtenido de la muestra.

Esta configuración da lugar a grafos que tienen un buen comportamiento en la distribución en grado total[5].

5.5.2 Propiedades generales

Generando y analizando un grafo de Barabási de 2500 nodos tenemos las siguientes propiedades generales:

- Número de nodos: 2500
- Número de aristas: 32500 = 2500×13
- Tamaño de la componente fuertemente conexa: 0.56 % = 14 / 2500
- Tamaño de la componente débilmente conexa: 100 %

Las propiedades generales del grafo resultante son fijas por cómo está diseñado el algoritmo.

5.5.3 Grado de los nodos

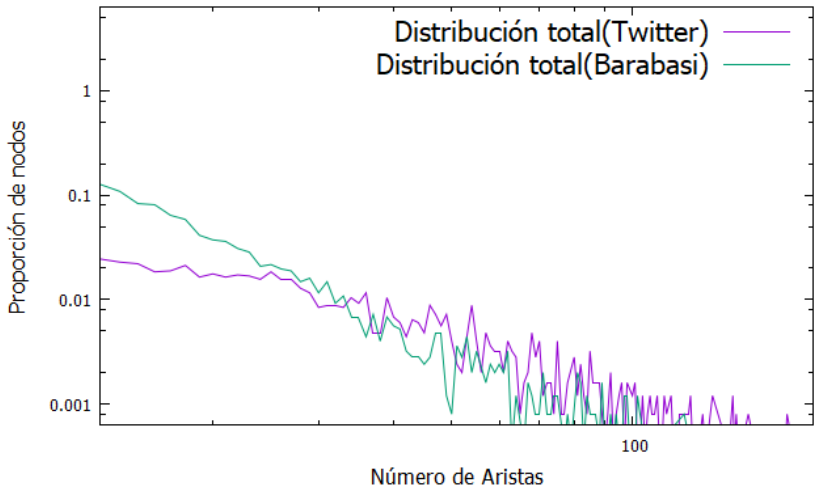

Figura 5.8: Gráfica comparativa de la distribución de grado de una muestra de Twitter de 2500 nodos y el grafo generado con el algoritmo de Barabási, en escala logarítmica.

En la figura 5.8 se puede observar cómo la distribución de grado del grafos generados por el algoritmo de Barabási sigue una función similar a la del grafo de la muestra de Twitter, ambos siguiendo una una ley potencial. Podemos observar como el grafo de barabasi tiene una concentración de nodos grado bajo anómalamente alta, pero esa concentración se equilibra a medida que aumenta el grado, equiparándose a la gráfica de la muestra. Esto demuestra la buena aproximación que tiene el algoritmo para esta medida siempre que se elijan buenos parámetros.

[5]Recordemos que en la sección 3.2.2 se indicó que este algoritmo no está pensado para grafos dirigidos ya que crea grafos con grado de salida constante

En base a estos resultados podemos sacar conclusiones en el otro sentido: El grafo de Twitter sigue una regla de conexión preferencial similar a la que se ha definido en el algoritmo de Barabási.

Grado medio de los nodos: 13.

5.5.4 Clustering

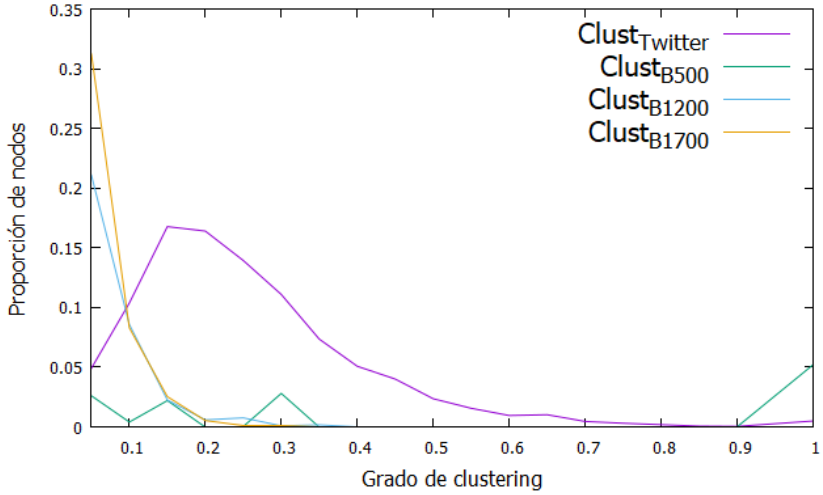

Figura 5.9: Gráfica comparativa del clustering del grafo original y varios grafos de Barabasi de distinto tamaño.

Como puede observarse en la figura 5.9, los grafos de Barabasi no reflejan la realidad del clustering de las redes sociales. La forma en la que se asignan las aristas no permite que se produzcan conexiones de cierre de triángulo, haciendo que el clustering de los nodos se concentre en torno al 0.

5.6 Modelo de Leskovec modificado

5.6.1 Configuración de parámetros

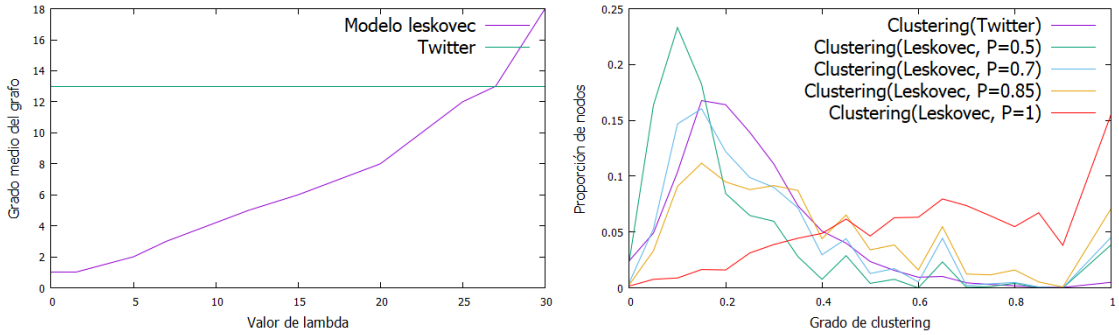

Figura 5.10: A la izquierda el grado medio del grafo Leskovec en función del parámetro Lambda. A la derecha el índice clustering en función del parámetro P.

El algoritmo del modelo de Leskovec es configurable por multitud de parámetros, de los cuales se han congelado en primera instancia los siguientes: Número de nodos(2500) , k(2), alfa(0.8) y beta(0.02). De este modo tenemos dos parámetros configurables:

- Lambda: Es el parámetro que define el tiempo de vida medio de los nodos. La vida media de los nodos es importante, ya que repercute directamente en el grado medio del grafo resultante.
- P: Es la probabilidad con la que el algoritmo elige una estrategia de cierre de triángulo en la generación de aristas. Este parámetro repercute en el índice de clustering, cuanta más probabilidad haya de cerrar triángulos más proporción de nodos se concentrarán en valores de clustering cercanos a 1.

Se ha reflejado la variación de estos parámetros y su repercursión en el grado medio y el clústering en la figura 5.10. En primer lugar se ha parametrizado lambda para que el grado medio del grafo fuera similar al de una muestra de Twitter del mismo número de nodos.

Con Lambda configurado se procede a configurar el parámetro P. Puede observarse cómo el clústering de los grafos generados se asemeja al del grafo de Twitter para valores de P menores que 0.85. Cuando P tiene valores cercanos a 1 la gráfica se deforma ya que la mayoría de nodos tienen un grado de clustering demasiado alto. Es importate tener en cuenta que modificando el valor de P podemos pasar de un grafo de Barabási (con valores de P cercanos a cero) a uno de Leskovec (con valores cercanos a uno).

En base a la gráfica el valor óptimo de P se da con valores cercanos a 0.7, con lo que podemos suponer que los usuarios de Twitter tienden a seguir a seguidores de sus seguidores en un 70 % de las ocasiones y el 30 % son seguimientos en base a la popularidad del usuario al que va seguir.

5.6.2 Propiedades generales

Al algoritmo de Leskovec, configurado con los parámetros especificados en la sección anterior, tiene las siguientes características generales.

- Número de nodos: 2500
- Número de aristas: 36349
- Tamaño de la componente fuertemente conexa: 87.4 %
- Tamaño de la componente débilmente conexa: 100 %

La componente fuertemente conexa del grafo tiene un valor verosímil con respecto al porcentaje de la componente conexa de Twitter. Esto confirma el potencial del algoritmo, el cual es capaz de simular con bastante precisión el proceso de evolución de una red social a lo largo del tiempo.

5.6.3 Grado de los nodos

Como se aprecia en la figura 5.11, la distribución de grado de los grafos generados por el algoritmo de Leskovec es prácticamente idéntica a la obtenidos de la muestra de Twitter, respaldando todos los datos obtenidos previamente.

El grado medio del grafo de Leskovec es de 13

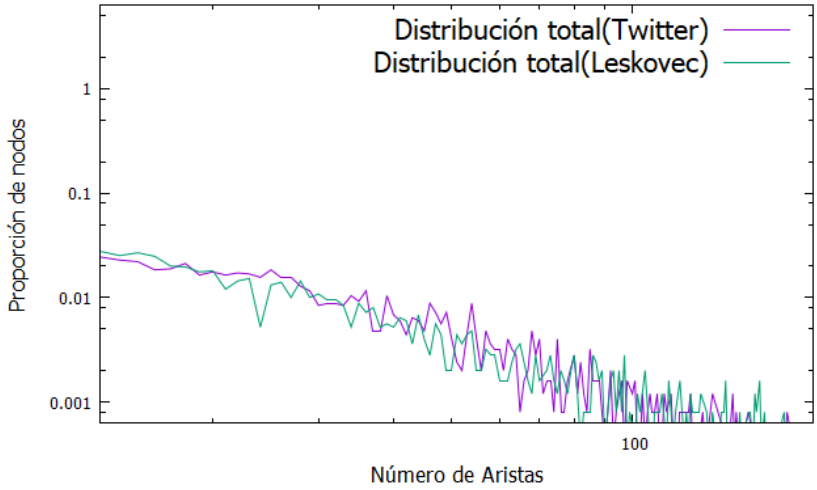

Figura 5.11: Gráfica comparativa de la distribución de grado de una muestra de Twitter de 2500 nodos y el grafo generado con el algoritmo de Leskovec, en escala logarítmica.

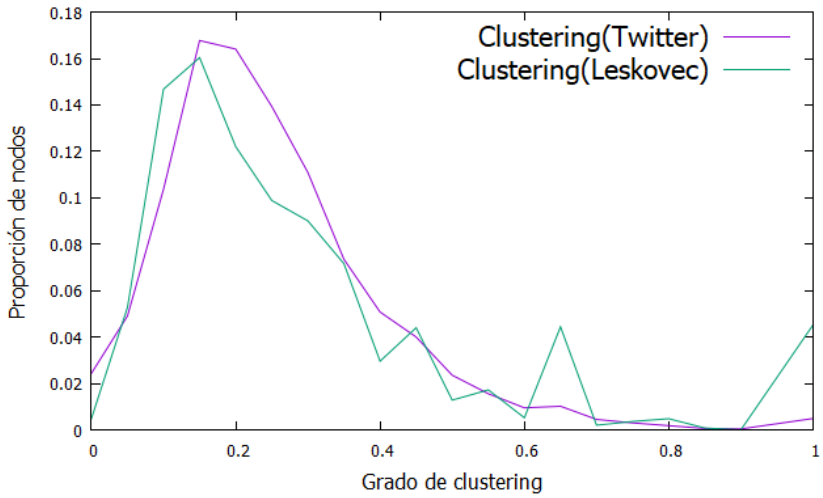

Figura 5.12: Gráfica comparativa del clustering del grafo original el grafo del modelo Leskovec.

5.6.4 Clustering

El clustering medio del grafo de Leskovec es del 31 %, un 4 % superior al del grafo de Twitter, una diferencia bastante aceptable teniendo en cuenta que ninguno de los otros algoritmos ha podido superar el 10 % de clustering medio.

6 Conclusiones

En este Trabajo de Fin de Grado se ha realizado un estudio comparativo de tres modelos de generación aleatoria de grafos y un grafo generado a partir de datos reales de Twitter, haciendo uso de diversas métricas de teoría de grafos que analizan las propiedades topológicas de estas estructuras.

Para ello, en primer lugar, ha sido necesario estudiar métricas y algoritmos con el fin de implementarlas en la fase de desarrollo. En el caso del algoritmo de Erdös y Leskovec, algunas mejoras se han llevado a cabo para obtener mejores resultados durante las pruebas.

Durante el desarrollo se ha especificado el diseño de la API para las pruebas, definiendo sus módulos, interfaces y sus implementaciones. Es una aplicación cuya dificultad de programación reside en la codificación de los algoritmos. El algoritmo de Leskovec modificado fue el que presentó mayor complejidad, pero los grafos resultantes tienen las mejores medidas.

Finalmente, en la fase de pruebas, se han hecho numerosas ejecuciones del programa con el fin de obtener datos. Estos datos han sido representados gráficamente a través de gnuplot, generando diversas gráficas comparativas de los algoritmos respecto a una muestra de tamaño similar de los datos reales. Todos los algoritmos, de una forma u otra, han aportado información para este Trabajo Fin de grado.

El algoritmo de Erdös genera grafos simples y puramente aleatorios. No tiene en cuenta medidas básicas como la distribución de grado o el índice de clustering. Los datos, inútiles en un principio, han servido para demostrar que Twitter no es una red social que sigue un proceso de evolución aleatorio.

El algoritmo de Barabási es el primero en desentrañar la evolución del grafo de Twitter. Aunque sus valores de clustering no son positivos, la distribución en grado de sus aristas son muy similares a las que presenta Twitter. Con esto, podemos suponer que las aristas de Twitter se crean siguiendo el principio de conexión preferencial.

El algoritmo de Leskovec tiene buenos resultados si se comparan con los algoritmos anteriores. Sin embargo llevaba a que el clustering fuera demasiado alto La modificación que se ha propuesto en este Trabajo Fin de grado permite modular ese coeficiente de clústering. Este último es el que ha dado los resultados más concluyentes. Una vez configurados los parámetros, los valores de clustering y su distribución de grado de las aristas encajan casi a la perfección con las del grafo de Twitter. Estos resultados son bidireccionales: El modelo de Leskovec simula con gran exactitud la evolución de la red social Twitter, por un lado, y también hemos descubierto propiedades del grafo de Twitter que no eran visibles

a través de las métricas definidas: El tiempo de vida de los nodos y la forma que tienen los usuarios de seguir a otros.

La realización de este Trabajo Fin de Grado ha producido en mí una serie de valoraciones personales:

Con respecto a los resultados finales del trabajo, no esperaba obtener conclusiones tan gratamente positivas durante la fase de pruebas. Especialmente el modelo de Leskovec, ya que presenta una algoritmia compleja con una gran cantidad de parámetros. Hasta que no procedí a comparar los resultados en base a estos parámetros no me di cuenta de la potencia con la que funcionaba el algoritmo, el cual sólo necesitaba ser configurado para dar grafos muy aproximados de la red social de Twitter.

Hacer el estudio del estado del arte de la teoría de grafos me ha ayudado a ampliar mis conocimientos sobre los grafos. En la carrera se ha hecho hincapié en el peso de aristas en el cálculo del camino más corto, algunas métricas aplican estos conocimientos para establecer el grado de cercania de los nodos. La distribución de grado y el clustering me han enseñado a ver la estructura de los grafos desde otra perspectiva, donde se valora más la centralidad de los nodos y su cantidad de aristas en vez de la cercanía de los nodos.

Finalmente, como consumidor diario de diversos tipos de redes sociales, ampliar el conocimiento que tenía sobre estas redes ha resultado ser gratificante. El modelo de leskovec me ha hecho ver que el estudio de las redes, aunque aún le queda recorrido, ha llegado a un punto interesante. Actualmente se tiene la capacidad de obtener grafos semejantes a los de las redes sociales mediante la configuración de un puñado de parámetros. Con el grafo construido se pueden aplicar simulaciones bastante verosímiles de la red social que se haya modelizado.

6.1 Trabajo futuro

Este trabajo ha estudiado el cuál es la estructura de la red social de Twitter, sin embargo, no se ha profundizado en cómo se mueve la información a través de ese grafo[6]. Hay multitud de datos que este Trabajo Fin de Grado no ha utilizado: Los tweets y las interacciones. Eso genera un tipo nuevo de grafo que, si se relaciona con los grafos obtenidos en este trabajo, es posible que se puedan sacar conclusiones interesantes.

Por otra parte, la librería de java jgrapht da acceso a la construcción de grafos con listeners. Si se plantea bien, podría ser una buena continuación el implementar grafos con listeners. Estos pueden permitir obtener información interesante como la velocidad de propagación o el cálculo de comunidades.

Además un Trabajo de Fin de Grado similar a este podría plantearse. Esta vez comparando los algoritmos con datos reales de una red social no dirigida, como por ejemplo Facebook. Ésta posee nodos que no necesariamente pueden ser usuarios: Páginas, grupos y eventos representarían un tipo especial de nodo.

BIBLIOGRAFÍA

[1] R. Barabási, Albert-László; Albert. Emergence of scaling in random networks. *Science*, 286:509–512, Ocotber 1999.

[2] F. Bonchi. Influence propagation in social networks: A data mining perspective. *IEEE Intelligent Informatics Bulletin*, 12(1):8–16, 2011.

[3] A. Erdos, P.; Rényi. On random graphs. i. *Publicationes Mathematicae*, 6:290–297, 1959.

[4] J. M. H. P. V. Mieghem. Classification of graph metrics. *Delft University of Technology tech report*, 2011.

[5] M. E. Newman, D. J. Watts, and S. H. Strogatz. Random graph models of social networks. *Proceedings of the National Academy of Sciences*, 99(suppl 1):2566–2572, 2002.

[6] J. Sanz-Cruzado Puig. Contact recommendation: Effercts on the evolution of social networks. Master's thesis, Escuela Politécnica Superior; Universidad Autónoma de Madrid, 2017.

[7] J. L. L. B. K. Tomkins. Microscopic evolution of social networks. In *KDD'08*, pages 462–470, August 2008.

[8] R. J. Trudeau. *Introduction to Graph Theory*. Dover publications, 1993.

[9] K. Wasserman, Stanley; Faust. Social network analysis in the social and behavioral sciences. *Cambridge University Press.*, pages 1–27, 1994.

[10] H. Zhu, B. A. Huberman, and Y. Luon. To switch or not to switch: Understanding social influence in online choices. In *ACM SIGCHI Conference on Human Factors in Computing Systems (CHI 2012)*, Austin, TX, USA, May 2012.