MODELOS EN REDES SOCIALES

TRABAJO DE FIN DE GRADO

MÓNICA ANDRÉS MANZANO

TUTORA:
ANA CARPIO RODRÍGUEZ

UNIVERSIDAD COMPLUTENSE
MADRID

Septiembre 2022

GRADO EN INGENIERÍA MATEMÁTICA
DEPARTAMENTO DE MATEMÁTICA APLICADA
FACULTAD DE CIENCIAS MATEMÁTICAS
UNIVERSIDAD COMPLUTENSE DE MADRID

Índice

Resumen

Las redes sociales constituyen una parte de nuestra sociedad, entenderlas puede ayudar a mejorar ciertos procesos y optimizar resultados desde un punto de vista de transmisión de información, económico o social, ya que estas a su vez están compuestas de diferentes comunidades. El presente trabajo se centra en grafos generados aleatoriamente y no dirigidos, analizando las principales características: agrupamiento, asortatividad, medidas centrales, comunidades, etc. Además de cálculos analíticos, el trabajo se presenta con diferentes gráficos que permiten visualizar y comprender mejor los resultados, también se contrasta con diferentes conjuntos de datos.

Abstract

The following project aims to analyze social network models and understand how they work, since social networks are a part of our society, understanding them can help to improve certain processes and optimize results from a point of view of information transmission, economic or social since these, in turn, are composed of different communities. The present TFG focuses on randomly generated and undirected graphs, analyzing the main characteristics: clustering, assortativity, central measures, and communities, ... In addition to analytical calculations, the work is presented with different graphs that allow better visualization and understanding of the results, and it is also contrasted with different datasets.

Palabras clave

Redes sociales, grafo, conexo, centralidad, agrupamiento, asortatividad, clique, comunidad

Keywords

Social networks, graph, connected, centrality, clustering, assortativity, clique, community

1. Introducción

El siguiente trabajo pretende un análisis de los modelos en redes sociales y entender su funcionamiento, para ello hemos generado grafos de modelos conocidos, estudiando las características, comparándolos con data real y analizando el comportamiento.

En primer lugar, comenzaremos recordando los conceptos fundamentales que se van a utilizar para el proyecto, una breve descripción de los artículos en los cuales se publicaron dichos modelos, seguido de simulaciones y análisis, para acabar con un caso práctico.

Las redes sociales son de uso cotidiano, comprenderlas y entender su funcionamiento puede ser muy útil desde un punto de vista informativo, social, económico, etc

La cantidad de opciones es ilimitada, se trata de un tema muy amplio con grandes posibilidades, este trabajo se centra en algoritmos para grafos aleatorios no dirigidos.

El presente proyecto ha sido programado con Python con ayuda de la librería Networkx, en la cual podemos encontrar los principales algoritmos generadores de redes así como los cálculos relacionados con ellos.

En cuanto a la salida de resultados para conseguir una forma más intuitiva, se ha utilizado la librería Pandas, para la visualización Matplotlib, si bien también se podría haber usado Seaborn, Numpy solo se ha usado por simplicidad a la hora de calcular varios grafos con varias probabilidades.

También se ha utilizado Jupyter notebook, Google colaboratory para programar de una forma más dinámica

Los conjuntos de datos han sido obtenidos de fuentes fiables, Stanford University, Kaggle.

2. Conceptos y modelos

2.1 Conceptos utilizados

El entendimiento y manejo de estos conceptos nos permite aplicarlos para poder responder a ciertas preguntas útiles

Recordatorio principales conceptos aplicados a redes sociales

- **Grafo no dirigido:** Un grafo no dirigido es aquel en el que todas sus aristas son bidireccionales. La relación sobre el conjunto de vértices es simétrica. Las aristas se representan como pares no ordenados
- **Grafo dirigido:** Un grafo dirigido es aquel en el que todas sus aristas tienen sentido o dirección. La relación sobre el conjunto de vértices no es simétrica. Las aristas se representan como un par ordenado.
- **Grafo completo:** Un grafo es completo si existen aristas uniendo todos los pares posibles de vértices.
- **Grafo conexo:** Un grafo es conexo si cada par de vértices está conectado por un camino, es decir si para todo par de vértices existe al menos un camino posible.
- **Bucle:** Nodo conectado consigo mismo.
- **Centralidad de grado:** Cociente entre el número de vecinos de un nodo dividido por el número posible de vecinos que podría tener; el número posible es todos menos el mismo si los bluces no están admitidos, y si lo están, sería el número total de nodos (ya que está incluido el mismo)
- **Centralidad de intermediación:** El número de caminos más cortos en un grafo que pasan por un nodo, dividido entre el número de caminos más cortos que existen entre cada par de nodos en un grafo.
- **Coeficiente de agrupamiento:** Mide la interconectividad, mayor nivel, mayor conectividad; el valor máximo puede ser uno.

- **Centralidad de cercanía:** Indica la cercanía de un nodo al resto en la red. Es la inversa de la media del camino más corto entre el vértice y los demás vértices de la red, es decir, uno dividido de la distancia media al resto de vértices.
- **Asortatividad:** Mide la tendencia de nodos de grado similar de conectar, es decir, si nodos con alto grado conectan nodos con alto grado. Los valores oscilan entre -1 y 1, los valores negativos indican que nodos con alto grado conectan nodos con bajo grado.
- **Clique:** Es un conjunto de nodos completamente conectado a los demás nodos del conjunto, es decir todos los nodos se encuentran conectados entre sí (completo y conexo). La clique más sencilla es una simple arista, después un triángulo.
- **Maximal clique:** Clique que cuando intentamos añadir un nodo, deja de ser clique porque ya no están todos los nodos conectados entre sí, deja de ser completo.
- **Comunidad:** Cliques unidos mediante 'puentes'
- **Puentes:** Nodos que sirven de unión entre diferentes cliques

2.2 Modelos de grafos no dirigidos generados aleatoriamente

Toivonen

Este modelo[1] se ha construido con la premisa de que las redes se creen asortativas, es decir, vértices altamente conectados conectan otros vértices altamente conectados, el modelo presentado es un grafo no dirigido, que crece hasta el tamaño deseado, comprendiendo este artículo minuciosamente, se entienden los conceptos principales de las redes y su funcionamiento. De ahí que en todos los modelos simulados y reales, se calculan las características descritas en este artículo. (En este artículo se menciona el algoritmo de Holme-Kim, el cual también simulamos.)

La motivación es entender el comportamiento, para ello se construye una red tan simple como sea posible pero que al mismo tiempo capture la realidad, esto es posible gracias a los siguientes requerimientos:

- Asortatividad
- Alto coeficiente de agrupamiento
- Media de los caminos más cortos crece lentamente con el tamaño de la red
- Amplias distribuciones de grado
- La red debe contener comunidades con densas conexiones internas

Así, el algoritmo consta de dos procesos

1. Adhesión aleatoria
2. Adhesión implícita preferente para los siguientes conexiones de nodos que parten del paso 1 que es aleatoria

Una vez la red ha alcanzado el tamaño deseado se analizan el resultado y compara con data real para poder entender el funcionamiento de las redes.

Distribución uniforme

Con el estudio de este artículo e investigando, surgen otros artículos famosos en el mundo de la ciencia para generar redes con grafos aleatorios:

-Erdös Rényi

-Barabási

-Holme Kim

Erdös Rényi

El modelo de Erdös-Rényi [2] es el modelo de grafo aleatorio más simple. Suponiendo que conocemos el número de nodos y el número de aristas (sin otra información), los nodos se conectan de forma aleatoria. Este proceso puede ocurrir de dos formas:

• L aleatorio pares de nodos se eligen entre el conjunto de todos los existentes pares de nodos.

• Las uniones se producen con probabilidad p

La distribución de grado es binomial, posible de aproximar con una Poisson[6]

Barabási-Albert

Este modelo[4] al igual que el de Toivonen, pretende mediante una forma sencilla recrear el comportamiento de las redes reales, también se cree que nodos con un alto grado conectan nodos con alto grado, el proceso de adhesión preferente consta de dos parámetros, el número de aristas a unir en cada paso, y el numero inicial de nodos, así el proceso es el siguiente:

- Empezamos con un grafo conexo de n nodos
- En cada paso añadimos un nuevo nodo y m aristas (menor o igual que n nodos) conectándolo con m nodos elegidos aleatoriamente proporcional a su grado.

La distribución de grado creada por la adhesión preferente es una Ley de potencia que no depende de los parámetros m y n, y es estacionaria en tiempo; El grado de los nodos si crece con el tiempo, antes se crea el nodo más grande el grado. El camino medio más corto de este modelo es pequeño.[6]

Existe una versión en Networkx de este algoritmo llamada dual, en la cual es posible elegir m1 y m2, con probabilidad p y (1- p) respectivamente.

Holme-Kim

El último modelo[3] consiste en una extensión de Barabási-Albert, en la cual se incorpora un nuevo paso: formación de tríadas con el que se pretende conseguir un mayor nivel de agrupación [6]

3. Desarrollo del trabajo

Una vez comprendido cómo funcionan estos algoritmos generadores de grafos aleatorios, procedemos a simularlos con la ayuda de la librería Networkx, generamos los modelos comparándolos con un conjunto de datos de Facebook (Stanford University).

Los resultados obtenidos sorprenden, ya que aún tuneando los modelos para intentar aproximarnos al conjunto de datos, entonces el número de aristas difiere considerablemente, debido al tamaño del grafo, el tiempo computacional es elevado sin conseguir los resultados esperados, esto hace pensar que el coeficiente de agrupamiento del conjunto de datos es demasiado elevado, puesto que no es posible conseguir una

similitud general en todos las mediciones, es decir, si el coeficiente de agrupamiento se aproxima mucho al del conjunto de datos, los otros resultados difieren y viceversa.

Para comprobar si es así, se utiliza otro conjunto de datos con más información, en el cual el coeficiente de agrupamiento es considerablemente inferior, y buscando información al respecto se llega a la conclusión que efectivamente 0.65 no está adecuado a la realidad, esto ocurre porque los datos son ego lo cual se explicara más adelante, como máximo en Facebook el coeficiente de agrupamiento es alrededor de 0.3.

Pequeños conjuntos de datos de kaggle demuestran que es así, incluso existen coeficientes de agrupamiento extremadamente bajos

Otro punto a destacar, cuando se escribieron estos artículos, como en el de Toivonen cita explícitamente ‘no es posible que un individuo conozca miles de usuarios’, entonces no existía el fenómeno broadcaster, y las relaciones de amistad eran recíprocas. Hoy en día la mayoría de usuarios tiene conexiones con otros usuarios, que realmente no se conocen, y también existen redes dirigidas que no implican una relación mutua

Debido a lo computacionalmente costoso que resulta tunear estos modelos para un gran número de nodos (aun con la ayuda de potentes notebooks con GPU/TPU), el siguiente paso en este proyecto es utilizar los conceptos y aplicarlos a muestras de data muy pequeñas, para eso utilizamos los conjuntos de datos de la librería Networkx

3.1 Datasets de Networkx

	Karate_Club	Davis_Women	Florentine	Miserables
Nodos	34	32	15	77
Aristas	78	89	20	254
Media centralidad de grado	0.139037	0.179435	0.190476	0.086808
Media centralidad de intermediación	0.044006	0.043548	0.114286	0.021882
Promedio Clustering	0.570638	0.0	0.16	0.573137
Promedio camino mas corto	2.4082	2.306452	2.485714	2.641148
Triángulos	34	32	15	77
Tamaño mayor clique	5	2	3	10
Pearson	-0.475613	-0.336998	-0.374838	-0.165225
Asortatividad	-0.475613	-0.336998	-0.374838	-0.165225

Dichos conjuntos de datos poseen un tamaño reducido, lo que permite que el tiempo computacional sea ínfimo, se observa que los modelos simulados se ajustan bastante a la información de los conjuntos de datos; Simulamos Holme-Kim con diferentes probabilidades para cada uno de los cuatro conjuntos de datos y otros modelos dependiendo de los datos y con ayuda de los gráficos vemos cuan aproximados son.

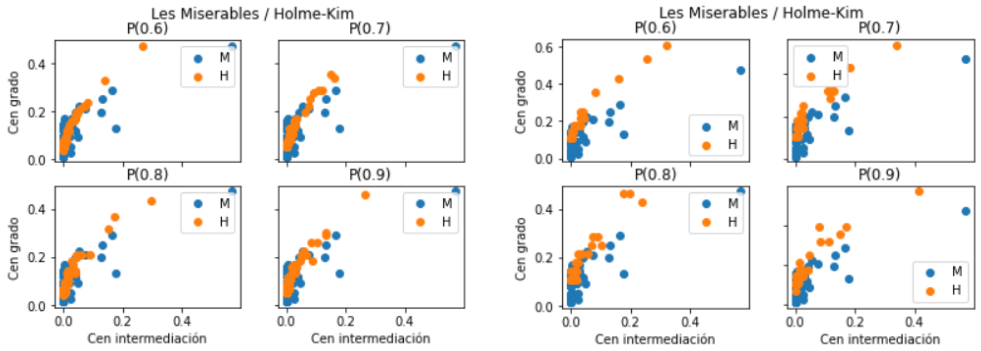
Karate club

- Simulamos Holme-Kim utilizando n = 34, m = 2, para acercarnos lo máximo posible al número de aristas del dataset, también variamos las probabilidades de 0.1 a 0.9 con paso 0.1, también simulamos con n = 29, m = 3; para conseguir el número de aristas.

	Nodos	Aristas	Media centralidad de grado	Media centralidad de intermediación	Promedio Clustering	Promedio camino mas corto	Triángulos	Tamaño mayor clique	Pearson
HK p(0.1)	34	64	0.114082	0.046179	0.176101	2.477718	34	3	-0.324928
HK p(0.2)	34	64	0.114082	0.048685	0.311269	2.557932	34	3	-0.203688
HK p(0.3)	34	64	0.114082	0.047293	0.311230	2.513369	34	3	-0.235541
HK p(0.4)	34	64	0.114082	0.052418	0.281046	2.677362	34	3	-0.172086
HK p(0.5)	34	64	0.114082	0.049131	0.257113	2.572193	34	3	-0.228161
HK p(0.6)	34	64	0.114082	0.046457	0.479896	2.486631	34	3	-0.302063
HK p(0.7)	34	64	0.114082	0.048852	0.489069	2.563280	34	3	-0.286971
HK p(0.8)	34	64	0.114082	0.048852	0.539559	2.563280	34	3	-0.331391
HK p(0.9)	34	64	0.114082	0.051693	0.742694	2.654189	34	3	-0.293761
Karate_Club	34	78	0.139037	0.044006	0.570638	2.408200	34	5	-0.475613

	Nodos	Aristas	Media centralidad de grado	Media centralidad de intermediación	Promedio Clustering	Promedio camino mas corto	Triángulos	Tamaño mayor clique	Pearson	Asortatividad
HK p(0.1)	29	77	0.189655	0.038862	0.338221	2.049261	29	4	-0.272363	-0.272363
HK p(0.2)	29	78	0.192118	0.037128	0.424782	2.002463	29	4	-0.281643	-0.281643
HK p(0.3)	29	78	0.192118	0.038405	0.346004	2.036946	29	4	-0.223857	-0.223857
HK p(0.4)	29	78	0.192118	0.040960	0.397787	2.105911	29	4	-0.149305	-0.149305
HK p(0.5)	29	77	0.189655	0.042054	0.538952	2.135468	29	4	-0.201720	-0.201720
HK p(0.6)	29	78	0.192118	0.037949	0.549859	2.024631	29	4	-0.327565	-0.327565
HK p(0.7)	29	78	0.192118	0.042510	0.544400	2.147783	29	4	-0.189944	-0.189944
HK p(0.8)	29	78	0.192118	0.042967	0.529870	2.160099	29	4	-0.201851	-0.201851
HK p(0.9)	29	77	0.189655	0.043423	0.617581	2.172414	29	4	-0.240239	-0.240239
Karate_Club	34	78	0.139037	0.044006	0.570638	2.408200	34	5	-0.475613	-0.475613

- Analíticamente en las imágenes superiores podemos observar que el primer modelo con probabilidad 0.8 parece el mejor aproximado(para esos parámetros), y el segundo el que tiene p(0.7) visualizamos diagramas de dispersión para apreciar la similitud entre los modelos simulados y el real



- En los gráficos de dispersión, sin embargo, parece que el primero con p(0.9) es más aproximado, en el caso del segundo p(0.8)

- Si representamos gráficamente el grafo, el más similar al modelo original es el primer modelo con probabilidad 0.9 a pesar de que la diferencia de asortatividad es grande entre ambos (para todas las probabilidades), gráficamente son muy similares

Karate Club

Holme-Kim (n = 34, m =2, p = 0.9)

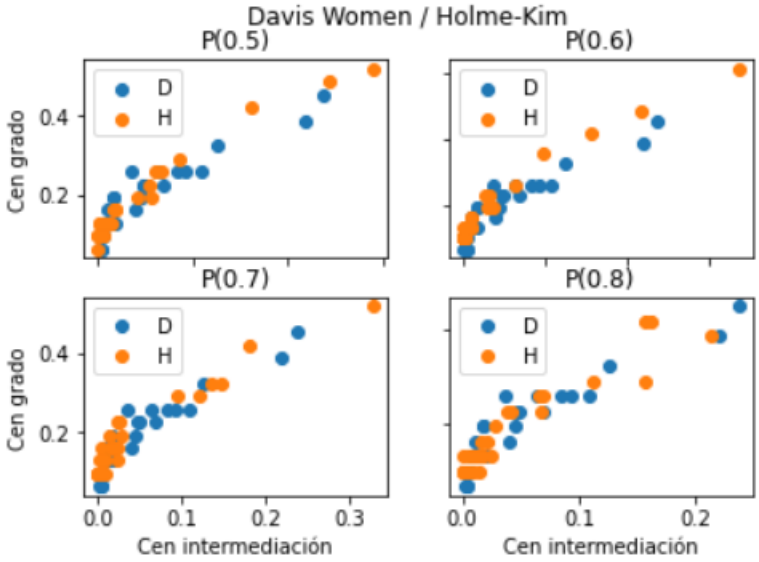
*El resultado del resto de modelos no resultan apropiados para este conjunto en particular, debido a la cifra baja de aristas en relación al número de nodos

Davis Women

- Para este conjunto utilizamos n = 32, m = 3, así podemos acercarnos lo máximo posible al número de aristas, de nuevo, simulamos Holme-Kim con diferentes probabilidades de 0.1 a 0.9 con paso 0.1

	Nodos	Aristas	Media centralidad de grado	Media centralidad de intermediación	Promedio Clustering	Promedio camino mas corto	Triángulos	Tamaño mayor clique	Pearson
HK p(0.1)	32	86	0.173387	0.036694	0.308184	2.100806	32	4	-0.276903
HK p(0.2)	32	87	0.175403	0.034879	0.411715	2.046371	32	4	-0.278403
HK p(0.3)	32	87	0.175403	0.036425	0.335311	2.092742	32	4	-0.223916
HK p(0.4)	32	87	0.175403	0.038306	0.384151	2.149194	32	4	-0.179251
HK p(0.5)	32	86	0.173387	0.038105	0.514169	2.143145	32	4	-0.266081
HK p(0.6)	32	87	0.175403	0.034879	0.525613	2.046371	32	4	-0.330933
HK p(0.7)	32	87	0.175403	0.039651	0.538371	2.189516	32	4	-0.197158
HK p(0.8)	32	87	0.175403	0.039919	0.517199	2.197581	32	4	-0.202816
HK p(0.9)	32	86	0.173387	0.041801	0.609721	2.254032	32	4	-0.231008
Davis_Women	32	89	0.179435	0.043548	0.000000	2.306452	32	2	-0.336998

- Analíticamente en la imagen superior podemos apreciar que la mayor similitud entre las simulaciones y el conjunto real, corresponde al de probabilidad 0.6, gráficamente la dispersión parece que el de probabilidad 0.5



- Así pues, si visualizamos los diferentes grafos parece que el más adecuado es el de probabilidad 0.6

Davis Women

Holme-Kim (n = 32, m =3, p = 0.6)

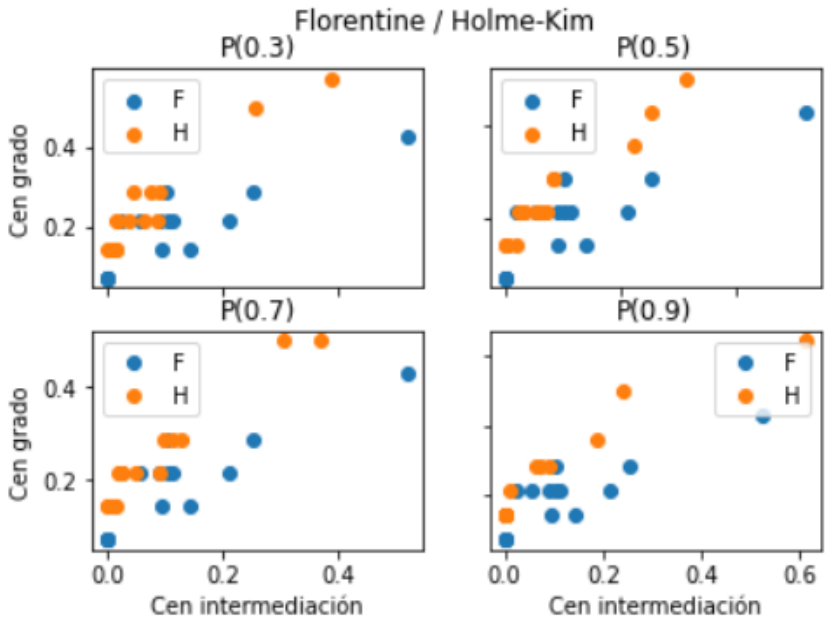
*De igual manera, los otros algoritmos presentaban menos similitud.

Florentine

- De nuevo ajustamos las variables para obtener un número similar de aristas y nodos al conjunto original, intentándonos acercar lo máximo posible, utilizamos n = 15, m = 2, ya que si ponemos m = 1 el resultado difiere de los datos originales (todas las conexiones van a tener solo un vecino), simulamos de nuevo como en los modelos anteriores Holme-Kim con diferentes probabilidades de 0.1 a 0.9 con paso 0.1

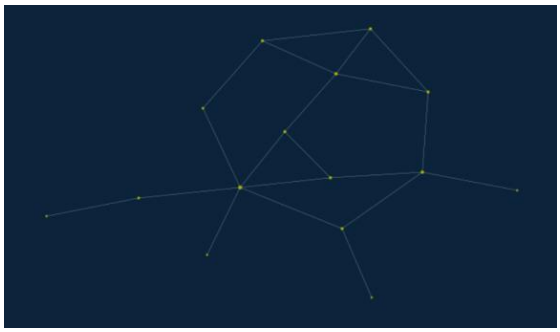
	Nodos	Aristas	Media centralidad de grado	Media centralidad de intermediación	Promedio Clustering	Promedio camino mas corto	Triángulos	Tamaño mayor clique	Pearson
HK p(0.1)	15	26	0.247619	0.083516	0.240635	2.085714	15	3	-0.244172
HK p(0.2)	15	26	0.247619	0.087179	0.361587	2.133333	15	3	-0.277883
HK p(0.3)	15	26	0.247619	0.075458	0.197619	1.980952	15	3	-0.317934
HK p(0.4)	15	26	0.247619	0.084249	0.332698	2.095238	15	3	-0.143322
HK p(0.5)	15	26	0.247619	0.084249	0.332698	2.095238	15	3	-0.143322
HK p(0.6)	15	26	0.247619	0.084249	0.339683	2.095238	15	3	-0.181302
HK p(0.7)	15	26	0.247619	0.084982	0.328571	2.104762	15	3	-0.144366
HK p(0.8)	15	26	0.247619	0.085714	0.365397	2.114286	15	3	-0.127425
HK p(0.9)	15	26	0.247619	0.084982	0.736455	2.104762	15	3	-0.354913
Florentine	15	20	0.190476	0.114286	0.160000	2.485714	15	3	-0.374838

- Parece que una probabilidad baja de unión es la que presenta menos dispersión de datos y la asortatividad (Pearson) más cercana al conjunto de datos original

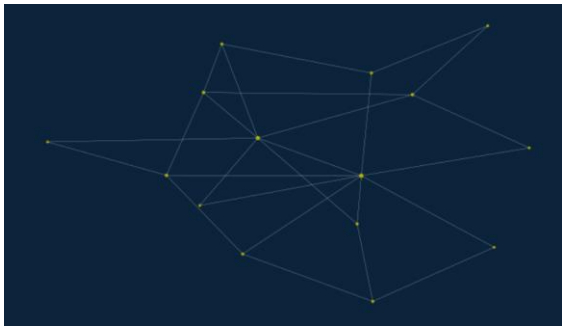


- Visualizamos diferentes grafos con diferentes probabilidades, gráficamente parece que el más adecuado es el cual que aproximaba el valor de Pearson mejor, cabe destacar que en este conjunto de datos hay nodos que solo tienen un vecino, lo que dificulta aproximarlo con este modelo, se intentó con m = 1 viendo que había muchos nodos que no tenían conexiones, pero el resultado no era aproximado a la realidad.

Florentine



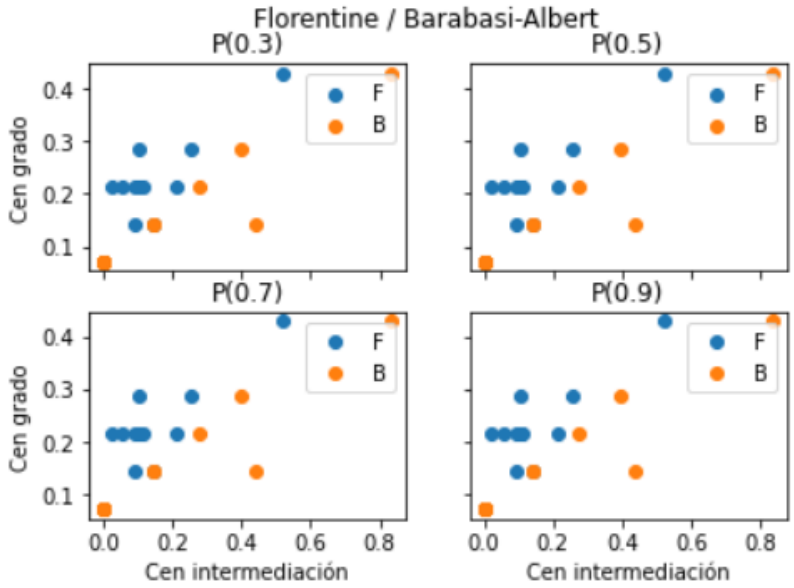
Holme-Kim (n = 15, m =2, p = 0.3)



- Buscamos pues otro algoritmo de Networkx más aproximado a este tipo de casos, parece que el que mejor se adapta es dual_barabasi_albert DUAL, el cual habíamos comentado con anterioridad la posibilidad de elegir dos parámetros de m, es decir el número de aristas que conectan, esto puede resultar muy útil en este tipo de grafos con nodos con bajo grado (es decir pocos vecinos) lo simulamos con parámetros n = 15, m1 = 1, m2 = 2, en términos de adhesión esto implica que la probabilidad de conexión entre dos nodos con 1 arista es p y con 2 aristas es 1 – p

	Nodos	Aristas	Media centralidad de grado	Media centralidad de intermediación	Promedio Clustering	Promedio camino mas corto	Triángulos	Tamaño mayor clique	Pearson
AB p(0.1)	15	26	0.247619	0.074725	0.679630	1.971429	15	3	-0.411139
AB p(0.2)	15	25	0.238095	0.078388	0.600265	2.019048	15	3	-0.366626
AB p(0.3)	15	20	0.190476	0.106960	0.318413	2.390476	15	3	-0.266491
AB p(0.4)	15	16	0.152381	0.140659	0.082222	2.828571	15	3	-0.409962
AB p(0.5)	15	16	0.152381	0.140659	0.082222	2.828571	15	3	-0.409962
AB p(0.6)	15	18	0.171429	0.116484	0.153016	2.514286	15	3	-0.265423
AB p(0.7)	15	17	0.161905	0.122344	0.077778	2.590476	15	3	-0.257848
AB p(0.8)	15	16	0.152381	0.129670	0.000000	2.685714	15	2	-0.543478
AB p(0.9)	15	14	0.133333	0.148718	0.000000	2.933333	15	2	-0.482117
Florentine	15	20	0.190476	0.114286	0.160000	2.485714	15	3	-0.374838

- Analizando la dispersión vemos que, aunque sea menos preciso, es más exacto.

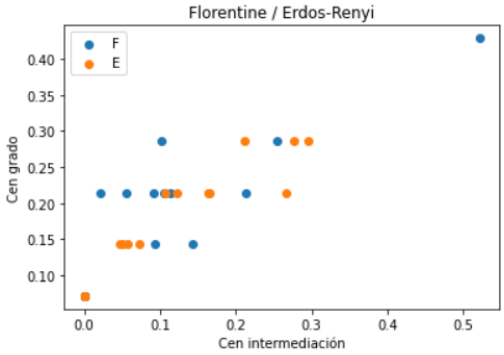


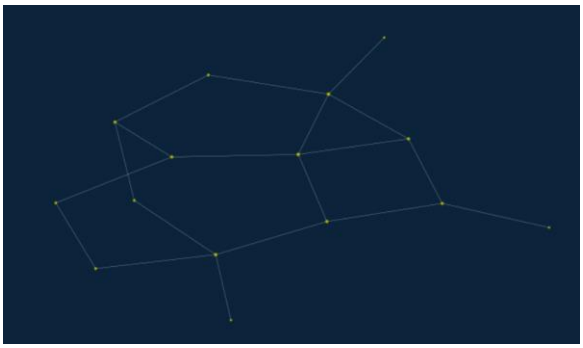
- Visualizando el grafo, resulta obvio que este modelo se aproxima mucho mejor al conjunto de datos

Florentine Barabasi (n = 15, m1 = 1, m2 = 2, p = 0.3)

- Por último se decide probar con Erdos-Renyi (la versión del algoritmo en la cual se especifica el número de nodos y la probabilidad de unión), a pesar de que generalmente produce grafos altamente conectados, la versatilidad de este algoritmo hace que si elegimos una probabilidad muy baja de unión podemos

también conseguir resultados similares a Albert-Barabasi dual, como comentario añadir que este algoritmo es de los años 60, lo cual resulta muy llamativo ya que el resultado es de admiración por cómo puede replicar tan bien un un conjunto de datos;
p = 0.11





- Concluimos para este conjunto de datos que Erdos-Renyi con probabilidad 0.11 es el algoritmo que mejor se aproxima, seguido de Albert-Barabasi Dual

Les Miserables

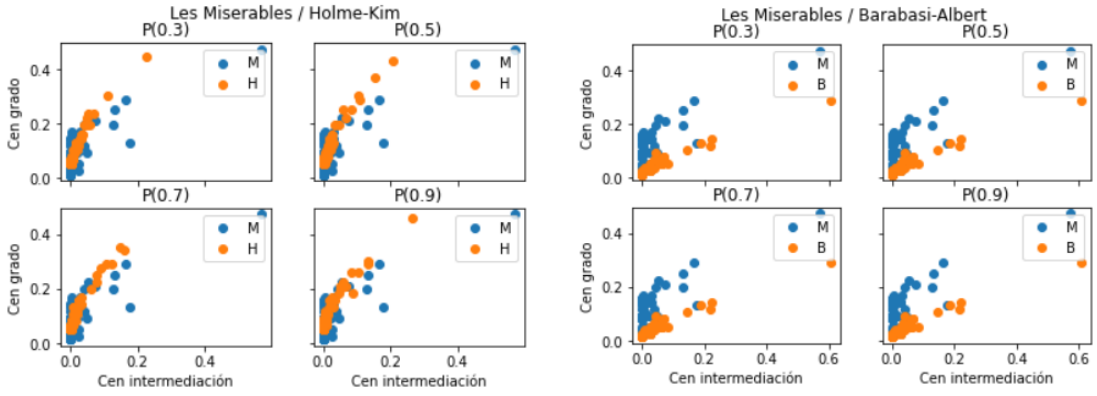
- El análisis del modelo anterior induce a pensar que para este conjunto de datos (debido a algunos usuarios poco conectados, que tienen un solo vecino) el modelo simulado que mejor se va a ajustar es el de Barabasi-Albert DUAL, pero de igual manera observando el alto nivel de agrupamiento del conjunto de datos, sabiendo que Holme-Kim reproduce altos niveles, podemos intuir que puede aproximarse al conjunto. Así pues simulamos ambos.

- Holme-Kim con n = 77, m = 4 (de nuevo elegimos parámetros que produzcan un resultado lo más cercano posible al número de aristas)

- Barabasi-Albert con n = 77, m1 = 1, m2 = 4, elegimos m1 = 1 para intentar replicar los nodos que solo tienen un vecino, por eso este algoritmo aproxima mejor este tipo de data cuando existen algunos nodos con grado bajo, ya que con Holme-Kim si elegimos 1 es para todas las conexiones, Barabasi-Albert DUAL nos permite elegir dos parámetros de adhesión

	Nodos	Aristas	Media centralidad de grado	Media centralidad de intermediación	Promedio Clustering	Promedio camino mas corto	Triángulos	Tamaño mayor clique	Pearson
HK p(0.1)	77	291	0.099453	0.017416	0.213102	2.306220	77	4	-0.120464
HK p(0.2)	77	288	0.098428	0.017448	0.238704	2.308612	77	4	-0.132289
HK p(0.3)	77	290	0.099111	0.016828	0.254901	2.262133	77	5	-0.183255
HK p(0.4)	77	290	0.099111	0.016555	0.308070	2.241627	77	4	-0.234602
HK p(0.5)	77	289	0.098770	0.017002	0.380470	2.275120	77	5	-0.209982
HK p(0.6)	77	289	0.098770	0.017266	0.368544	2.294942	77	5	-0.163517
HK p(0.7)	77	289	0.098770	0.017535	0.422901	2.315106	77	5	-0.187342
HK p(0.8)	77	288	0.098428	0.018278	0.470546	2.370813	77	5	-0.141555
HK p(0.9)	77	292	0.099795	0.018059	0.520322	2.354409	77	5	-0.148474
Les Miserables	77	254	0.086808	0.021882	0.573137	2.641148	77	10	-0.165225

	Nodos	Aristas	Media centralidad de grado	Media centralidad de intermediación	Promedio Clustering	Promedio camino mas corto	Triángulos	Tamaño mayor clique	Pearson
BA p(0.1)	77	262	0.089542	0.017940	0.173461	2.345523	77	4	-0.208310
BA p(0.2)	77	241	0.082365	0.020114	0.168519	2.508544	77	4	-0.155483
BA p(0.3)	77	220	0.075188	0.021681	0.119309	2.626111	77	4	-0.111340
BA p(0.4)	77	199	0.068011	0.023299	0.150713	2.747437	77	4	-0.132031
BA p(0.5)	77	184	0.062884	0.023500	0.109966	2.762474	77	4	-0.180599
BA p(0.6)	77	166	0.056733	0.022871	0.092903	2.715311	77	4	-0.218084
BA p(0.7)	77	136	0.046480	0.030030	0.043551	3.252221	77	3	-0.183752
BA p(0.8)	77	133	0.045455	0.030226	0.083996	3.266917	77	4	-0.162415
BA p(0.9)	77	100	0.034176	0.031374	0.030980	3.353042	77	4	-0.229486
Les Miserables	77	254	0.086808	0.021882	0.573137	2.641148	77	10	-0.165225

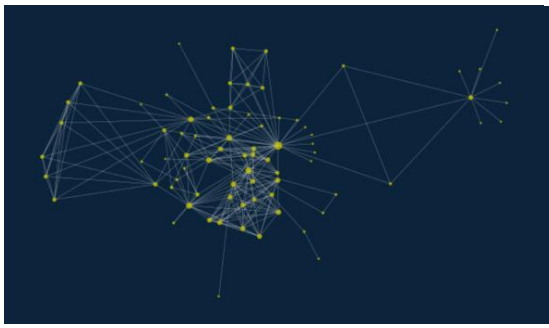


- Analíticamente la simulación que más se aproxima al conjunto de datos es Holme-Kim, estando la mayoría de parámetros muy bien aproximados, incluido el alto coeficiente de agrupamiento difícil de obtener con Albert-Barabasi, en cuanto a la dispersión podemos apreciar también que Holme-Kim satisface mejor el conjunto de datos.

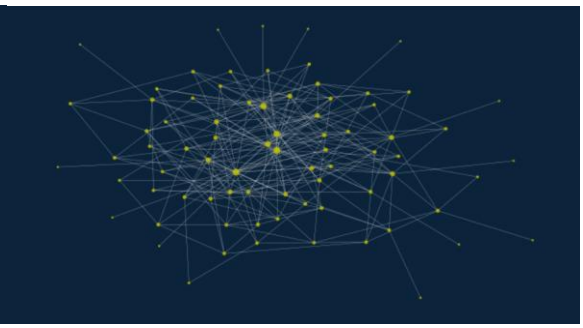
Les Miserables

Holme-Kim con n = 77, m = 4

Les Miserables



Barabasi-Albert DUAL n = 77, m1 = 1, m2 = 4



3.2 Data Musae Facebook Kaggle[11]

A continuación, procedemos a analizar un conjunto de datos de gran volumen, con un número de nodos = 22470; Debido a la cifra tan elevada para analizar con una sola máquina, ciertas operaciones tales como encontrar caminos mínimos y cliques resultan muy lentas, a veces interrumpidas debido a la larga duración; Intentamos encontrar un modelo y parámetros que más se ajusten

- Empezamos simulando Holme-Kim, ya que produce grafos con alto coeficiente de agrupamiento, observamos que si aproximamos el coeficiente de agrupamiento, el número de aristas difiere en gran medida del data set, y si aproximamos las aristas el agrupamiento es de un nivel bastante inferior en comparación con el conjunto de datos(tal y como podemos observar en la salida de datos)

```
{'name': 'Holme_Kim3',
 'Nodos': 22470,
 'Aristas': 44936,
 'Media centralidad de grado': 0.00017800720858682754,
 'Media centralidad de intermediación': 0.00020010759089093935,
 'Promedio Clustering': 0.3661039175847572,
 'Asortatividad': -0.03541011128813503}
```

- Procedemos a simular extended_Barabasi_Albert Graph, este modelo sigue la idea de Toivonen, no es idéntico, pero si análogo, ambos se basan en adhesión aleatoria previa, seguida de implícita regida por varias probabilidades; el tiempo de ejecución de este modelo es extremadamente largo para una sola máquina que termina interrumpiéndose (+10h), y tunearlo resulta muy complicado. Ejecutando en el ordenador sólo el modelo, conseguimos varias simulaciones (generando el grafo 5h, los cálculos básicos 2h). Observamos que este algoritmo para generar grafos es computacionalmente pesado, ya que los cálculos requieren de tiempo, pero sólo generar el grafo en otros modelos el tiempo es considerablemente más reducido.

- Observamos que el modelo que mejor podría replicar el comportamiento del conjunto de datos es Holme-Kim; comprobamos que en este tipo de datos en concreto(ego, con alto nivel de agrupamiento) el modelo que mejor funciona es

Holme-Kim con m = 8 y probabilidad 0.5, como hemos mencionado anteriormente, conseguir alto agrupamiento y numero de aristas simultáneamente no es posible.

name	Nodos	Aristas	Media centralidad de grado	Media centralidad de intermediación	Promedio Clustering	Asortatividad
Musae_Facebook	22470	171002	0.000677	0.000177	0.359738	0.085058
extended_barabasi_albert_3_025_025	22470	99774	0.000395	0.000134	0.004005	0.023231
extended_barabasi_albert_5_03_04	22470	224095	0.000888	0.000100	0.011090	0.015672
Holme_Kim_8_05	22470	179613	0.000712	0.000102	0.115522	-0.065789
Holme_Kim3	22470	44936	0.000178	0.000200	0.366104	-0.035410

-Aclaración: No es posible coger una muestra aleatoria para reducir el tamaño, ya que el grafo deja de ser conexo-

El tamaño del conjunto de datos es considerable para poder realizar cálculos con una sola máquina (para reducir el tiempo en gran medida, consiguiendo un tiempo permitible, mostrado en la imagen superior, hemos calculado solo las principales centralidades, asortatividad y coeficiente de agrupamiento, eliminando así el resto de cálculos), procedemos a buscar un conjunto de datos más reducido.

3.3 Data Facebook Stanford[10]

Stanford University posee la colección más amplia de conjuntos de datos de redes, es notable el alto coeficiente de agrupamiento comparado con otros conjuntos de datos analizados, esto es debido a que se trata de una ego red, es decir, una porción de una red social formada por un individuo dado, denominado ego, y las otras personas con las que tiene una relación social, denominada alters. En la página web podemos ver los resultados con egoNet

Dataset statistics	
Nodes	4039
Edges	88234
Nodes in largest WCC	4039 (1.000)
Edges in largest WCC	88234 (1.000)
Nodes in largest SCC	4039 (1.000)
Edges in largest SCC	88234 (1.000)
Average clustering coefficient	0.6055
Number of triangles	1612010
Fraction of closed triangles	0.2647
Diameter (longest shortest path)	8
90-percentile effective diameter	4.7

Descargamos el conjunto de datos y procedemos a analizarlo, a pesar de ser de un tamaño considerablemente mas reducido que el anterior, de nuevo para analizar los caminos más cortos, cliques, comunidades, computacionalmente es complicado; intentamos encontrar algún modelo de los estudiados que se aproxime a los valores del conjunto de datos, pero al ser el coeficiente de agrupamiento tan elevado, resulta complicado; si conseguimos aproximar el coeficiente de agrupamiento, el número de aristas se vuelve muy elevado en comparación con el conjunto de datos; ciertos modelos encontrados que intentan simular este conjunto de datos no tienen en cuenta el número de aristas, de ahí que parezca ser más aproximado.

En la tabla podemos observar que si aproximamos el número de aristas(lo máximo posible para usar los mismos parámetros de entrada y ver el comportamiento) y las centralidades, el coeficiente de agrupamiento y la asortatividad se alejan de los valores deseados, también se aprecia como Holme-Kim produce altos niveles de agrupamiento respecto a los otros modelos, usando los mismos parámetros de entrada.

	Erdos	Barabasi	HolmeKim	Facebook
Nodos	4.039000e+03	4039.000000	4039.000000	4039.000000
Aristas	5.710087e+06	76380.000000	76085.000000	88234.000000
Media centralidad de grado	7.002168e-01	0.009366	0.009330	0.010820
Media centralidad de intermediación	7.425889e-05	0.000393	0.000371	0.000667
Promedio Clustering	7.002217e-01	0.034021	0.105039	0.605547
Pearson coef	-5.911084e-04	-0.007184	-0.093918	0.063577

- Observamos que el modelo que mejor podría replicar el comportamiento del conjunto de datos es Holme-Kim, lo simulamos con diferentes parámetros de entrada, y conseguimos aproximar el número de aristas y otros valores, pero el alto nivel de clustering no es posible de replicar

	holmekim_21_08	holmekim_23_08	Facebook
Nodos	4039.000000	4039.000000	22470.000000
Aristas	84155.000000	92120.000000	99774.000000
Media centralidad de grado	0.010320	0.011296	0.000395
Media centralidad de intermediación	0.000376	0.000362	0.000134
Promedio Clustering	0.100318	0.096351	0.004005
Pearson coef	-0.060808	-0.064584	0.023231

4. Caso práctico

HUAWEI-FACEBOOK (KAGGLE DATASET[11])

Procedemos a analizar un conjunto de datos con Google Colaboratory Pro

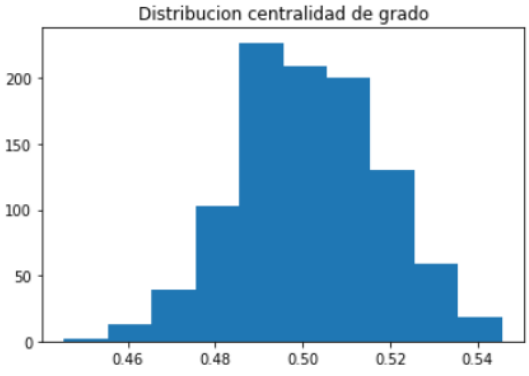
¿Cómo se caracteriza la red?

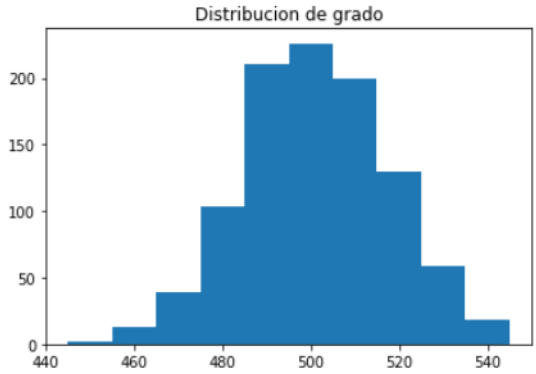
-Número de nodos y aristas: (1000, 250315)

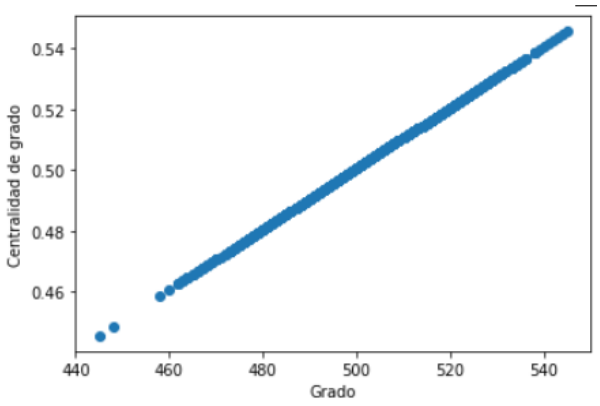
-Bucles: No tiene

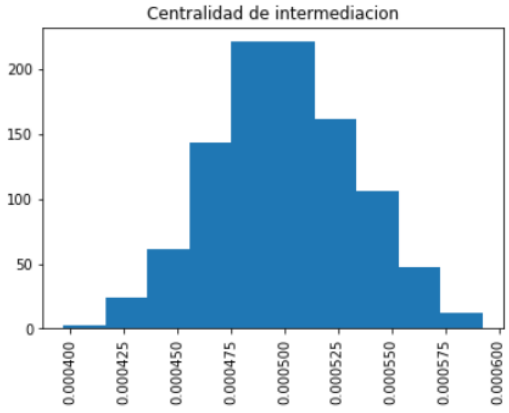
-Grado de los nodos: Min = 445, Mean = 500.63, Max = 545

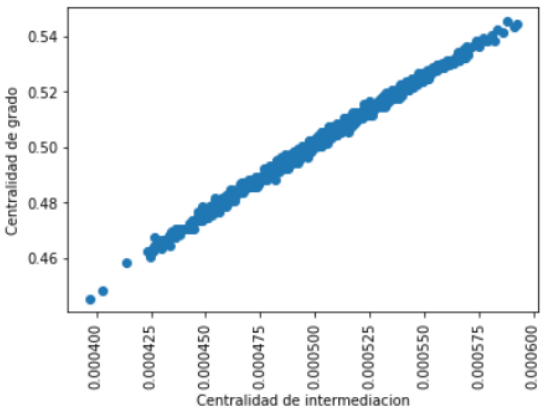
Gráficos para entender los datos











¿Como encontramos usuarios importantes(nodos)?

Usuarios más conectados al resto de usuarios:

Usuarios con centralidad de grado más alta: ‘Dililah’

¿Como encuentro puentes o puntos de transmisión de información?

Usuarios con centralidad de intermediación más alta: ‘Homer’

Captura los cuellos de botella en un gráfico en lugar de nodos altamente conectados,

Detección de Puentes entre diferentes comunidades, nodos cruciales por el que se comunica y fluye información en diferentes subredes

¿Como detectamos comunidades?

-Encontrar cliques, es decir subgrafos completamente conectados dentro de un grafo más grande

-Encontrar las uniones de esos cliques (Puentes)

Dichos puentes son extremadamente útiles para el estudio de redes.

K-clique método, siendo k el tamaño deseado.

-Tamaño de la clique más grande

-Función que dado el grafo y el tamaño nos devuelve el número de cliques de ese tamaño

¿Como encontrar las comunidades de colaboradores más grandes?

Max clique, clique que no puede ser extendida añadiendo otro nodo, ya que deja de ser clique porque el nodo añadido no está completamente conectado al resto de nodos

Subgrafos

-Nodos importantes y sus vecinos: Puede resultar útil generar los subgrafos correspondientes a los nodos de interés, función implementada con este objetivo, en el caso de una muestra tan pequeña y cogiendo los dos usuarios más destacados, el subgrafo resultante tiene el mismo tamaño que el original.

-Subgrafo más grande conexo: Otra buena forma de comprender y analizar el conjunto de datos, numerosas aplicaciones como construir sistemas de recomendación

¿Como construimos un sistema de recomendación?

Triángulos abiertos: Existen 1000 triángulos abiertos

Función que dado un grafo y un nodo devuelve True si el nodo está en algún triangulo abierto, false en caso contrario

¿Como encontramos caminos?

Existen numerosos algoritmos, cabe destacar:

-Algoritmo BFS: encuentra el camino más corto entre dos nodos, se desarrolló en los 50 para encontrar el camino más corto en un laberinto

-Adaptaciones al conocido Dijkstra

Las aplicaciones son diversas, optimización, modelización, etc.

Por último intentemos generar un grafo similar al del conjunto de datos, para ello simularemos Holme-Kim y Erdos-Renyi

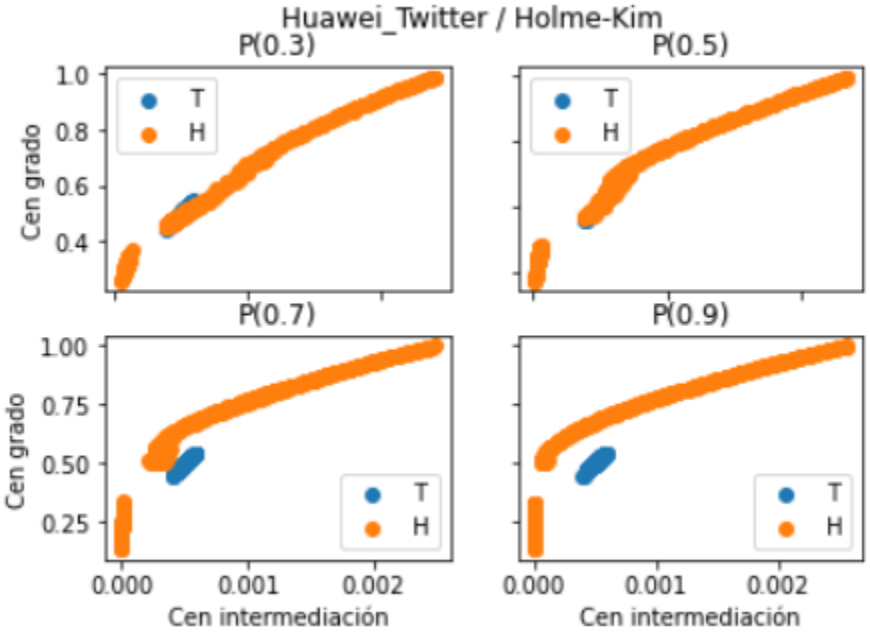
Holme-Kim

- Empezamos con Holme-Kim, con diferentes inputs, recordemos que con el conjunto de datos de Stanford University el problema era conseguir un alto coeficiente de agrupamiento sin elevar el tamaño de las aristas, aquí tenemos el problema de que al aproximar la mayoría de resultados la asortatividad se aleja del

valor del conjunto de datos, en el resto de los parámetros analiticamente parece que Holme-Kim se adapta bien; no obstante la diferencia de asortividad es muy considerable

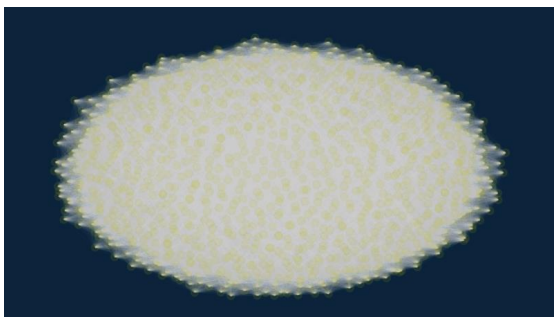
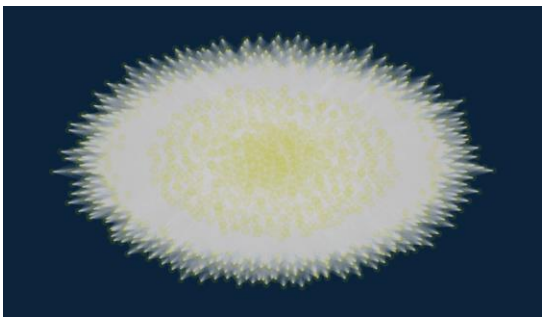
name	Nodos	Aristas	Media centralidad de grado	Media centralidad de intermediación	Promedio Clustering	Asortatividad
Twitter_Huawei	1000	250315	0.501131	0.000500	0.501210	-0.004439
Holme_Kim	1000	203867	0.408142	0.000593	0.622285	-0.236012
Holme_Kim2	1000	239080	0.478639	0.000522	0.704025	-0.293524
Holme_Kim3	1000	244554	0.489598	0.000511	0.684867	-0.347380

- Graficamente podemos observar efectivamente esta gran diferencia de asortatividad entre los valores del conjunto de datos y los simulados.



- También en visualización del grafo podemos apreciar que este modelo no aproxima bien el conjunto de datos.

Huawei-Twitter Holme-Kim n = 1000, m = 500, p = 0.3

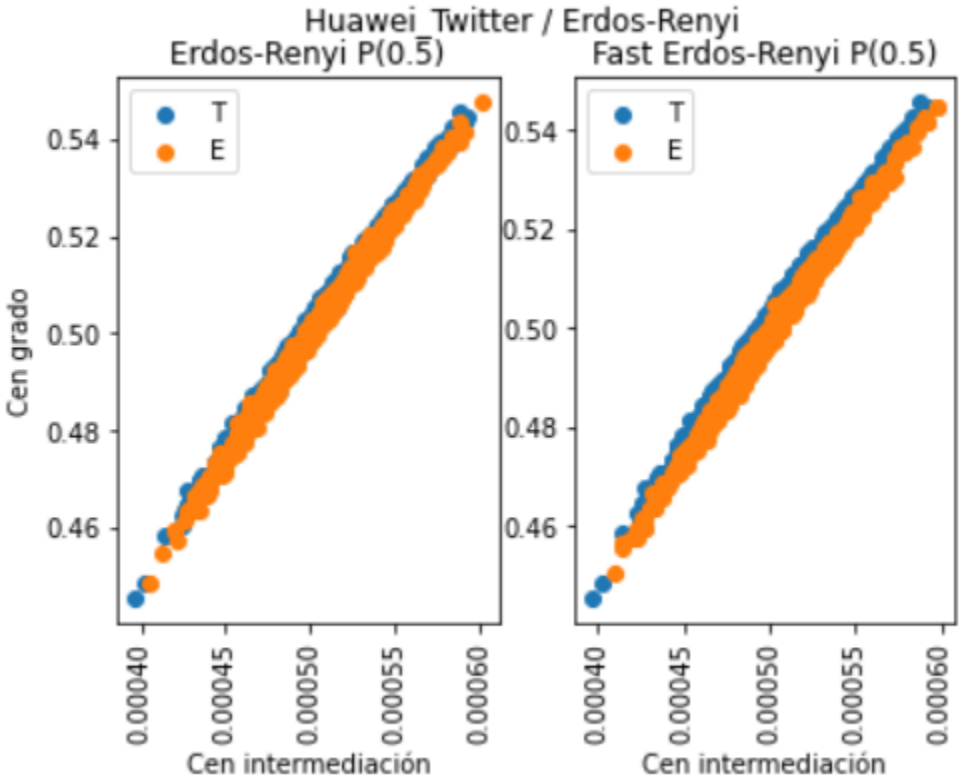
Erdos-Renyi

- Debido a la versatilidad demostrada por este algoritmo y su gran grado de eficiencia, procedemos a simularlo, teniendo en cuenta que el grafo del conjunto de datos esta muy bien conectado y que este algoritmo puede producir grafos con las mismas características, utilizamos dos versiones, Erdos-Renyi y fast-Erdos-Renyi, ambos con diferentes probabilidades.

	name	Nodos	Aristas	Media centralidad de grado	Media centralidad de intermediación	Promedio Clustering	Asortatividad
0	Huawei_Twitter	1000	250315	0.501131	0.000500	0.501210	-0.004439
1	Erdos_Renyi_P(0.5)	1000	249782	0.500064	0.000501	0.500038	-0.000335
2	Erdos_Renyi_P(0.55)	1000	274454	0.549457	0.000451	0.549444	-0.000865
3	Erdos_Renyi_P(0.6)	1000	299420	0.599439	0.000401	0.599459	-0.001416
4	Erdos_Renyi_P(0.7)	1000	349161	0.699021	0.000302	0.699027	-0.001501
5	Fast_Erdos_Renyi_P(0.5)	1000	249360	0.499219	0.000502	0.499225	-0.003933
6	Fast_Erdos_Renyi_P(0.51)	1000	254414	0.509337	0.000492	0.509195	-0.002684
7	Fast_Erdos_Renyi_P(0.49)	1000	244302	0.489093	0.000512	0.489048	-0.004686

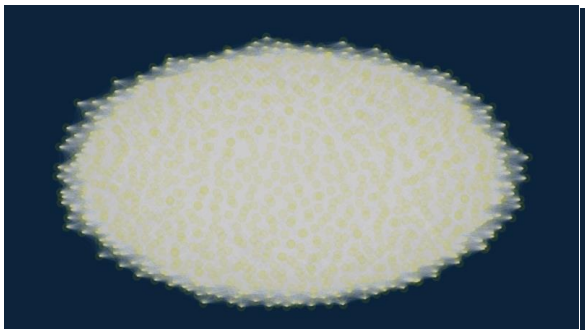
- En la imagen superior observamos como una vez más, este algoritmo produce resultados extremadamente cercanos al conjunto de datos, demostrando de nuevo la posibilidad de adaptarse a diferentes grafos ofreciendo resultados tan próximos

- Podemos apreciar la similitud entre la simulación y el conjunto de datos observando el gráfico de dispersión

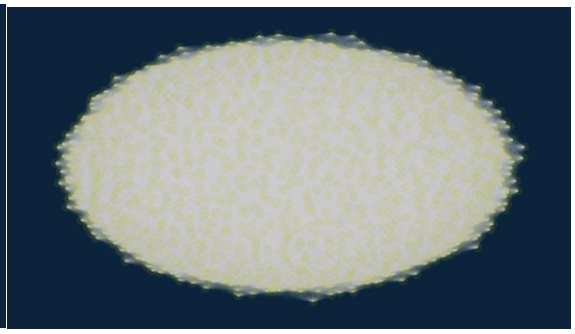


- Visualizando los grafos, es obvio cuan aproximada es la simulación a los datos, pareciendo el mismo conjunto.

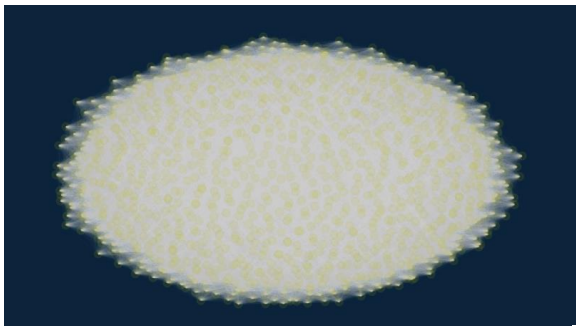
Huawei-Twitter Erdos-Renyi p(0.5)



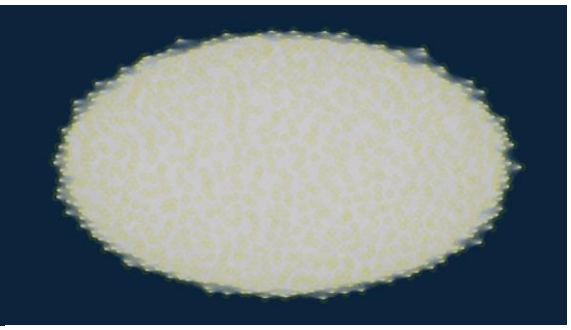
Huawei-Twitter



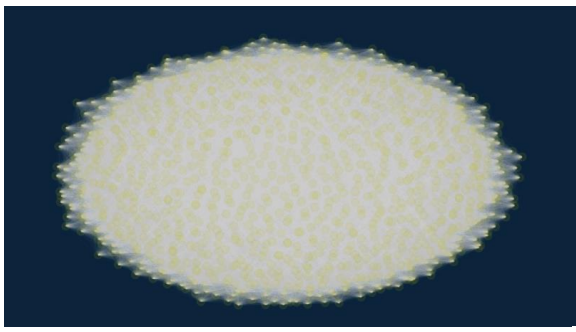
Fast Erdos-Renyi p(0.5)

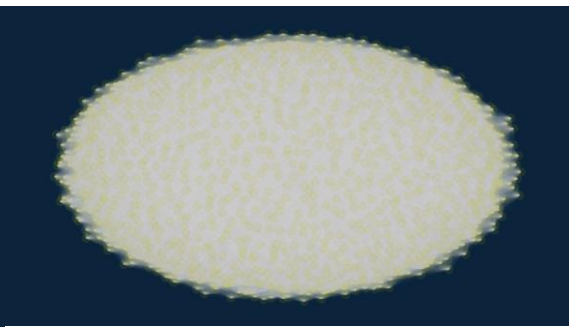


Huawei-Twitter



Fast Erdos-Renyi p(0.49)





- Como podemos observar el modelo de Erdos-Renyi se ajusta perfectamente al conjunto de datos.

5. Conclusiones

Sobre los modelos

En un breve resumen, las simulaciones invitan a pensar:

- Ante un gran clustering en el conjunto de datos: Holme-Kim, también si el número de aristas es bajo en relación a los nodos pero bien conectado. (si el número de aristas y agrupamiento es elevado, es dificil, como hemos comprobado anteriormente, alcanzar varios resultados de forma simultánea)

- Ante un grafo con una alta conectividad: Erdos-Renyi, Holme-Kim

- Ante un grafo con usuarios aislados: Dual Barabasi (contra: tiempo computacional elevado)

- No obstante, cabe destacar que Erdos-Renyi con diferentes probabilidades produce resultados muy diversos, siendo como ya hemos dicho muy versátil capaz de adaptarse a numerosos grafos, ya que Erdos-Renyi con baja probabilidad ha producido en este trabajo resultados similares a Barabasi-Albert, sin embargo, el tiempo computacional de Erdos-Renyi es el más bajo de todos los modelos simulados.

- También, tal y como hemos visto en las simulaciones, puede resultar muy positivo probar con varios algoritmos y averiguar cual es el que mejor se adapta al conjunto de datos en cuestión.

Inconvenientes encontrados

- El principal desafío en la realización del trabajo ha sido la memoria, incluso con un ordenador de alta RAM, no es posible realizar este tipo de cálculos con conjuntos

de datos de gran tamaño, se han probado muchas alternativas y plataformas para intentar resolver este problema.

- De esta manera se ha procurado de múltiples formas analizar conjuntos de datos de gran tamaño y compararlo con grafos generados; Generar dichos grafos no necesita mucha memoria, realizar operaciones y cálculos con ellos si, como encontrar la media del camino más corto, la clique máxima y operaciones que suponen computacionalmente caras.

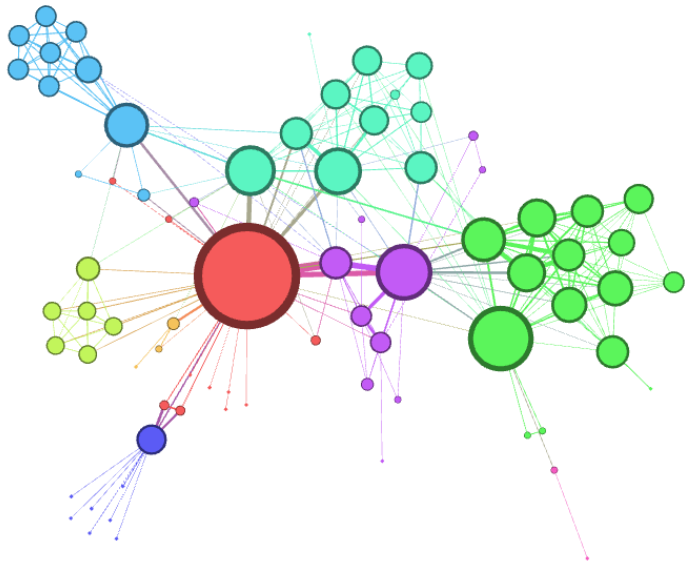
```
%%time
data_dict19 = dict(Erdos =resultados(ER19), Barabasi=resultados(B19), HolmeKim=resultados(HK19))
```

```
Wall time: 7h 43min 47s
```

- Con Google Colaboratory los cálculos de camino son tan pesados que el límite de la suscripción se alcanza rápido (incluyendo camino ni siquiera acaba los cálculos del conjunto de datos) y cuando eso ocurre se restaura el entorno de ejecución y se pierden todas las variables, con lo cual comparar una simulación con otra es prácticamente imposible (en conjuntos de datos de gran tamaño), por ejemplo, calculando los principales valores (incluso eliminando el camino y dejando los imprescindibles) del conjunto de datos Musae, solo podemos calcular este (y debido a que hemos obviado el camino) cuando se intenta simular algún algoritmo más para comparar, al comenzar se interrumpe debido a que se ha alcanzado el máximo de RAM, se restaura el entorno de ejecución y se pierden los valores calculados para Musae.

- Al obtener datos con scraping (para intentar entender la diferencia de coeficiente de agrupamiento y asortatividad en los diferentes conjuntos de datos) el resultado son grafos no conexos con los cuales no se puede analizar cierta información, lo mismo ocurre intentando tomar muestras de conjuntos de datos, se pierde la conectividad

- Gephi es un software pensado más para la representación gráfica y detección de comunidades, que para el análisis y de igual manera no está pensado para muestras de gran tamaño, colapsa.



(Les Miserables en Gephi)

Pluses y mejoras que se pueden realizar en un proyecto de más tamaño

- Estudiar detenidamente la asortatividad de las redes, ya que en general en los artículos se considera que es positiva para modelar y formular algoritmos, pero analizando conjuntos de datos encontramos que la mayoría de ellas son negativas; esto hace preguntarnos si los conjuntos de datos no son fiables o representativos, a pesar de que algunos de ellos pertenecen a fuentes muy fiables.

- Sin duda otro tema merecedor de estudio es el coeficiente de agrupamiento, los valores oscilan en un margen increíblemente amplio y es difícil entender como puede variar tanto de unos conjuntos de datos a otros, cabe destacar que todos los conjuntos de datos utilizados en este trabajo son de fuentes fiables. Otros conjuntos de datos encontrados en la web:

*Coeficiente de agrupamiento de otros conjuntos de datos (Twitter, Github)[11]

```
nx.average_clustering(TW)
```

```
0.05087681843971928
```

```
nx.average_clustering(GU)
```

```
0.0722208382203323
```

- Networkx dispone de muchos resultados en forma de generadores e iteradores, saberlos utilizar de forma apropiada puede solucionar muchos problemas de memoria.

- Conseguir pickles de modelos pre-entrenados

- Librería nxviz, para otro tipo de visualizaciones, hay una actualización en la API y no ha sido posible usarla.

- Análisis con series temporales, ver cómo evoluciona el grafo con el tiempo puede ser un estudio muy interesante.

- Poner peso en las aristas de acuerdo a la importancia que creamos que tenga

- Redes con etiquetas para poder tener más información y mejores sistemas de recomendación.

- Comprobar la teoría de los 6 grados en un conjunto de datos de gran tamaño.

- Scraping de un determinado tópico, limpiar el contenido de las publicaciones con NLP (Natural Language Processing) y analizar cuál ha sido la importancia,

trascendencia de dichas publicaciones, y como en la propagación ciertos usuarios han intervenido (debido al gran número de seguidores que han compartido)

- Utilizar databricks con AWS o Google Cloud (se ha intentado en este proyecto, pero existe algún tipo de error que resolver)

- Análisis de grafos bipartitos (se empezó a analizar uno y puede ser muy útil a la vez que interesante, relación producto-consumidor)

- Egonet.

6. Referencias

[1] Riitta Toivonen, Jukka-Pekka Onnela, Jari Saramäki, Jörkki Hyvönen, Kimmo Kaski "A model for social networks".In: Physica A 371 (2006) 851–860

[2] Paul Erdős and Alfréd Rényi. "On the evolution of random graphs". In: Publ. Math. Inst. Hung. Acad. Sci 5.1 (1960) 17–60

[3] Petter Holme and Beom Jun Kim. "Growing scale-free networks with tunable clustering". In: Physical review E 65.2 (2002) 026107

[4] Albert-László Barabási and Réka Albert. "Emergence of scaling in random networks". In: science 286.5439 (1999) 509–512

[5] https://snap.stanford.edu

[6] https://www.universite-lyon.fr/

[7] https://networkx.org

[8] https://matplotlib.org

[9] https://ericmjl.github.io/

Datasets

[10]https://snap.stanford.edu/data/

[11]https://www.kaggle.com/datasets/

Apéndice

Las funciones más relevantes e imprescindibles usadas

- Módulos necesarios

```
from math import *
import networkx as nx
import numpy as np
import pandas as pd
import matplotlib as mpl
import matplotlib.pyplot as plt
import random
import csv
from statistics import mean as mean
from networkx.algorithms import approximation
from networkx.algorithms import community
from itertools import combinations
```

- Función para calcular las principales características de un grafo

```
def resultados(G):
    return {
        'Nodos': int(len(G.nodes())),
        'Aristas': int(len(G.edges())),
        'Media centralidad de grado': mean(nx.degree_centrality(G).values()),
        'Media centralidad de intermediación': mean(nx.betweenness_centrality(G).values()),
        'Promedio Clustering': nx.average_clustering(G),
        'Promedio camino mas corto': nx.average_shortest_path_length(G),
        'Triángulos': int(len(nx.triangles(G))),
        'Tamaño mayor clique': len(sorted(sorted(nx.find_cliques(G), key=lambda x:len(x))[-1])),
        'Pearson' : nx.degree_pearson_correlation_coefficient(G),
        'Asortatividad': nx.degree_assortativity_coefficient(G)
    }
```

** Pearson mide la asortatividad, se han calculado ambos para ver si existía una diferencia apreciable (no hay diferencia apreciable)

- Función para devolver dataframe con diferentes probabilidades y modelo Holme-Kim

```
def data_frame(G, n, m, nombre):

    HKlist = [resultados(nx.powerlaw_cluster_graph(n, m, j, seed=20160)) for j in np.arange(0.1, 1, 0.1)]

    HKlist.append(resultados(G))

    df = pd.DataFrame(HKlist)

    df.index = ['HK p(0.1)', 'HK p(0.2)', 'HK p(0.3)', 'HK p(0.4)','HK p(0.5)', 'HK p(0.6)','HK p(0.7)', 'HK p(0.8)','\
                'HK p(0.9)', nombre]

    return df
```

- Función para devolver dataframe con diferentes probabilidades del modelo Barabasi-Albert

```
def data_frame2(G, n, m1, m2, nombre):

    BAlist = [resultados(nx.dual_barabasi_albert_graph(n, m1, m2, j, seed=20160)) for j in np.arange(0.1, 1, 0.1)]

    BAlist.append(resultados(G))

    df = pd.DataFrame(BAlist)

    df.index = ['BA p(0.1)', 'BA p(0.2)', 'BA p(0.3)', 'BA p(0.4)','BA p(0.5)', 'BA p(0.6)','BA p(0.7)', 'BA p(0.8)',\
                'BA p(0.9)', nombre]

    return df
```

- Función para pintar data original contrastada con Holme-Kim

```
def contraste(G, n, m, titulo = '', inicial = ''):

    figure, axis = plt.subplots(2, 2)

    figure.suptitle(f'{titulo} / Holme-Kim')

    axis[0, 0].scatter(x = bet_deg(G)[0], y = bet_deg(G)[1])
    axis[0, 0].scatter(x = bet_deg(nx.powerlaw_cluster_graph(n, m, 0.3, seed=20160))[0],\
                       y = bet_deg(nx.powerlaw_cluster_graph(n, m, 0.3, seed=20160))[1])
    axis[0, 0].set_title('P(0.3)')
    axis[0, 0].legend(f'{inicial}H')
    axis[0, 0].set(ylabel='Cen grado')

    axis[0, 1].scatter(x = bet_deg(G)[0], y = bet_deg(G)[1])
    axis[0, 1].scatter(x = bet_deg(nx.powerlaw_cluster_graph(n, m, 0.5, seed=20160))[0],\
                       y = bet_deg(nx.powerlaw_cluster_graph(n, m, 0.5, seed=20160))[1])
    axis[0, 1].set_title('P(0.5)')
    axis[0, 1].legend(f'{inicial}H')


    axis[1, 0].scatter(x = bet_deg(G)[0], y = bet_deg(G)[1])
    axis[1, 0].scatter(x = bet_deg(nx.powerlaw_cluster_graph(n, m, 0.7, seed=20160))[0],\
                       y = bet_deg(nx.powerlaw_cluster_graph(n, m, 0.7, seed=20160))[1])
    axis[1, 0].set_title('P(0.7)')
    axis[1, 0].legend(f'{inicial}H')
    axis[1, 0].set(xlabel='Cen intermediación', ylabel='Cen grado')


    axis[1, 1].scatter(x = bet_deg(G)[0], y = bet_deg(G)[1])
    axis[1, 1].scatter(x = bet_deg(nx.powerlaw_cluster_graph(n, m, 0.9, seed=20160))[0],\
                       y = bet_deg(nx.powerlaw_cluster_graph(n, m, 0.9, seed=20160))[1])
    axis[1, 1].set_title('P(0.9)')
    axis[1, 1].legend(f'{inicial}H')
    axis[1, 1].set(xlabel='Cen intermediación')
```

```
    for axis in axis.flat:
        axis.label_outer()

    plt.show()
```

- Misma función aplicada al modelo de Barabasi

```
def contraste2(G, n, m1, m2, titulo = '', inicial = ''):

    figure, axis = plt.subplots(2, 2)

    figure.suptitle(f'{titulo} / Barabasi-Albert')

    axis[0, 0].scatter(x = bet_deg(G)[0], y = bet_deg(G)[1])
    axis[0, 0].scatter(x = bet_deg(nx.dual_barabasi_albert_graph(n, m1, m2, j, seed=20160))[0],\
                       y = bet_deg(nx.dual_barabasi_albert_graph(n, m1, m2, j, seed=20160))[1])
    axis[0, 0].set_title('P(0.3)')
    axis[0, 0].legend(f'{inicial}B')
    axis[0, 0].set(ylabel='Cen grado')

    axis[0, 1].scatter(x = bet_deg(G)[0], y = bet_deg(G)[1])
    axis[0, 1].scatter(x = bet_deg(nx.dual_barabasi_albert_graph(n, m1, m2, j, seed=20160))[0],\
                       y = bet_deg(nx.dual_barabasi_albert_graph(n, m1, m2, j, seed=20160))[1])
    axis[0, 1].set_title('P(0.5)')
    axis[0, 1].legend(f'{inicial}B')


    axis[1, 0].scatter(x = bet_deg(G)[0], y = bet_deg(G)[1])
    axis[1, 0].scatter(x = bet_deg(nx.dual_barabasi_albert_graph(n, m1, m2, j, seed=20160))[0],\
                       y = bet_deg(nx.dual_barabasi_albert_graph(n, m1, m2, j, seed=20160))[1])
    axis[1, 0].set_title('P(0.7)')
    axis[1, 0].legend(f'{inicial}B')
    axis[1, 0].set(xlabel='Cen intermediación', ylabel='Cen grado')


    axis[1, 1].scatter(x = bet_deg(G)[0], y = bet_deg(G)[1])
    axis[1, 1].scatter(x = bet_deg(nx.dual_barabasi_albert_graph(n, m1, m2, j, seed=20160))[0],\
                       y = bet_deg(nx.dual_barabasi_albert_graph(n, m1, m2, j, seed=20160))[1])
    axis[1, 1].set_title('P(0.9)')
    axis[1, 1].legend(f'{inicial}B')
    axis[1, 1].set(xlabel='Cen intermediación')
```

```
   for axis in axis.flat:
       axis.label_outer()

   plt.show()
```

- Función para centralidad, necesaria para llamar a contraste

```
def bet_deg(G):
    bet_cen = nx.betweenness_centrality(G)
    deg_cen = nx.degree_centrality(G)

    return [list(bet_cen.values()), list(deg_cen.values())]
```

- Función para pintar grafo (parámetros escogidos de internet[9], función realizada por mi)

```
def pintar_grafo(G):
    nodes = G.nodes()
    degree = G.degree()
    colors = [degree[n] for n in nodes]
    size = [(degree[n]) for n in nodes]

    #pos = nx.kamada_kawai_layout(G)
    pos = nx.spring_layout(G, k = 0.2)
    cmap = plt.cm.viridis_r
    #cmap = plt.cm.Greys

    vmin = min(colors)
    vmax = max(colors)

    fig = plt.figure(figsize = (7,4), dpi=100)

    nx.draw(G,pos,alpha = 0.8, nodelist = nodes, node_color = 'y', node_size = size,\
            width = 0.2, cmap = cmap, edge_color ='white')
    fig.set_facecolor('#0B243B')

    #plt.legend()
    plt.show()
```

- Función para comprobar si hay self-loops

```
def sloops_list(G):

    return [i for i, j in G.edges() if i == j]
```

- Función para obtener el grado de los vértices del grafo

```
def grado(G):

    return [len(list(T.neighbors(n))) for n in T.nodes()]
```

- Función para obtener el(los) usuario(s) con mayor centralidad de grado

```
def usuarios_importantes(G):

    d_c = nx.degree_centrality(G)
    return [i for i, j in d_c.items() if j == max(list(d_c.values()))]
```

- Función para obtener el(los) usuario(s) con mayor centralidad de intermediación (puentes)

```
def usuarios_importantes2(G):

    b_c = nx.betweenness_centrality(G)
    return [i for i, j in b_c.items() if j == max(list(b_c.values()))]
```

- Ver si un nodo pertenece a un triángulo abierto

  Esta función conseguida en internet [10]

```
def nodo_en_tabierto(G, n):

    en_tabierto = False

    for n1, n2 in combinations(G.neighbors(n), 2):

        if not G.has_edge(n1, n2):

            en_tabierto = True

            break

    return en_tabierto
```

- Función para obtener el subgrafo compuesto por los usuarios importantes y sus vecinos

```
def sub_broadcasters(G, broadcasters):

    return G.subgraph([[nbr for nbr in G.neighbors(n)] for n in broadcasters])
```

Por cuestiones de simplicidad y presentación, solo se han escrito las funciones mas utilizadas e imprescindibles