Práctica 4 BMI

Memoria

12/05/2020

Adrián Rubio Pintado

Jorge Muñoz Aguado

A continuación, se explicarán las tareas realizadas.

Tarea 1 - Estructuras de datos y recomendación simple.

La estructura de datos de *RatingImpl* se basa en el uso de *HashMap* dentro de otro *HashMap*. De esta forma, el rendimiento conseguido es el óptimo.

En el caso de las similitudes, estas se precalculan para ofrecer una respuesta inmediata en caso de solicitar los datos. (Orden 1)

Tarea 2 - Filtrado colaborativo kNN.

El algoritmo de KNN se basa en comparar los datos con los K vecinos más próximos.

Tarea 2.1 - KNN basado en usuario.

En primer lugar, para ejecutar el algoritmo de KNN se precalculan las similitudes y estos se ordenan, en una cola de prioridad, ya que los K vecinos deben ser los más próximos. También será necesario conocer qué ítems no ha calificado el usuario.

Una vez conocidos los vecinos y los ítems se procede a recomendar pesando la similitud entre los usuarios el valor que le da el otro usuario a ese *item*. Los valores de score se guardarán en un doble *HashMap*.

Tarea 2.2 - KNN basado en usuario normalizado.

Este programa hará uso del mismo algoritmo que el apartado anterior. Pero al calcular el score final se multiplicará por el valor C. Este valor se obtiene al sumar las similitudes de los K vecinos.

Tarea 3 - Recomendación basada en contenido.

Para este apartado, se ha implementado, además de la clase del recommendador pedida, CosineFeatureSimilarity clase que hereda de FeatureSimilarity, que calcula la similitud basada en coseno entre las features.

También se ha implementado la clase CentroidFeatures, que hereda de FeaturesImpl,para adaptar el uso que se le quiere dar de las Features en este apartado. Aquí se almacenan los vectores(usuario-feature).

Tarea 4 - Ampliación de algoritmos.

Tarea 4.1 - ItemNNRecommender.

Para el desarrollo de este ejercicio se ha desarrollado además de la clase ItemNNRecommender, la clase CosineUserSimilarity, que implementa la similitud entre dos items como un coseno entre ellos. Dicha clase calcula y las similitudes en tiempo de ejecucion para aumentar la eficiencia. Para reducir el tamaño de la tabla solo de guarda una solo entrada por cada par de valores(items) en la implementacion(aprovechando la simetria de esta). Dicha tabla se ha implementado como un HashMap<Integer, HashMap<Integer, Double>>.

Además dicho algoritmo se ha optimizado para los casos en los que no tiene que calcular el coseno, para evitar iteraciones en los bucles innecesarios,aprovechando la simetría de las tablas.

Tarea 4.2 - Similitud de Jaccard.

Se ha implementado JaccardFeatureSimilarity como extension de la clase feature FeatureSimilarity. Simplemente calcula la similitud entre dos items mediante sus fetures. Concretamente la similitud es la intersección / unión de las features de dos items.

Tarea 4.3 - Otras Variantes.

Se ha implementado **PearsonCorrelationUserSimilarity**, una implementacion de la similitud que implementa la Correlacion de Peason. Es bastante similar a la implentacion de UserCosineSimilarity. En ella se calculan previamente las medias de los ratings de cada user y luego se almacena la correlación entre cada usuario en una tabla, que sera la que luego consulte la función sim llamada por el exterior, teniendo en cuenta para los calculos solo los items que cada par de usuarios tienen en común: esto es tanto para el producto escalar como para el cálculo de los módulos de cada usuario.

En StudentTest se hacen test con los datasets dados, para los recomendadores que se basan en usuario, ahora ya no se le pasa la similitud por coseno, si la implementada en este ejercicio.

También se ha implementado UserKNNMeanCenteredRecommender , en el que para a cada rating de cada vecino, se le resta la media de ese mismo vecino. Finalmente, a la recomendación se le suma la media del usuario a recomendar. De esta manera eliminamos “los residuos” de los ratings.

Tarea 5 - Evaluación

NOTA: Las pruebas se han ejecutado en un ordenador con un Intel Core i5-7200 a 2.71GHz con 8GB de memoria RAM, de los cuales 7,44GB usables.

Toy Datset

	RMSE	P@10	Recall@10	Tiempo ejecución(ms)
RandomRecommender	2.769572106879894	0.04	0.3	1
MajorityRecommender	2.0816659994661326	0.04	0.3	0
AverageRecommender	2.0	0.02	0.2	0
UserKNNRecommender(cosine)	2.1682015457649984	0.04	0.3	1

NormUserKNNRecommender(cosine)	2.2663805671070065	0.02	0.2	1
ItemNNRecommender(cosine)	1.8134282014947813	0.04	0.3	6
CentroidRecommender	2.5584610206210856	0.04	0.3	5
ItemNNRecommender(Jaccard)	1.3161849203880367	0.02	0.2	2
UserKNNRecommender(Pearson)	1.653325635754177	0.02	0.2	1
UserKNNMeanCenteredRecommender(cosine)	1.935684668098964	0.08	0.3333333333333333	2

MovieLens Dataset

	RMSE	P@10	Recall@10	Tiempo ejecución
RandomRecommender	2.7613376369347957	0.003278688524590165	9.606044367078514E-4	866ms
MajorityRecommender	517.7134050311381	0.15360655737704492	0.07722389141231985	1s 79ms
AverageRecommender	1.1756433972419038	9.836065573770492E-4	7.520089913012578E-5	1s 53ms
UserKNNRecommender	22.058948705029394	0.2563934426229504	0.15602604992980434	42s 215ms
NormUserKNNRecommender	0.8927932998005895	0.012786885245901632	0.011047407871294475	41s 962ms
ItemNNRecommender	280.1496642313801	0.22524590163934394	0.15016969934763346	4min 37s 129ms

CentroidRecommender	0.0	0.0	0.0	1s 442ms
ItemNNRecommender(Jaccard)	6.011111265532899	0.020655737704918003	0.007107356403410827	1min 31s 243ms
UserKNNRecommender(Pearson)	4.562481633237943	0.2199999999999997	0.13412589615761222	22s 114ms
UserKNNMeanCenteredRecommender(cosine)	3.264083676880708	0.19852459016393403	0.11345680711361363	44s 955ms

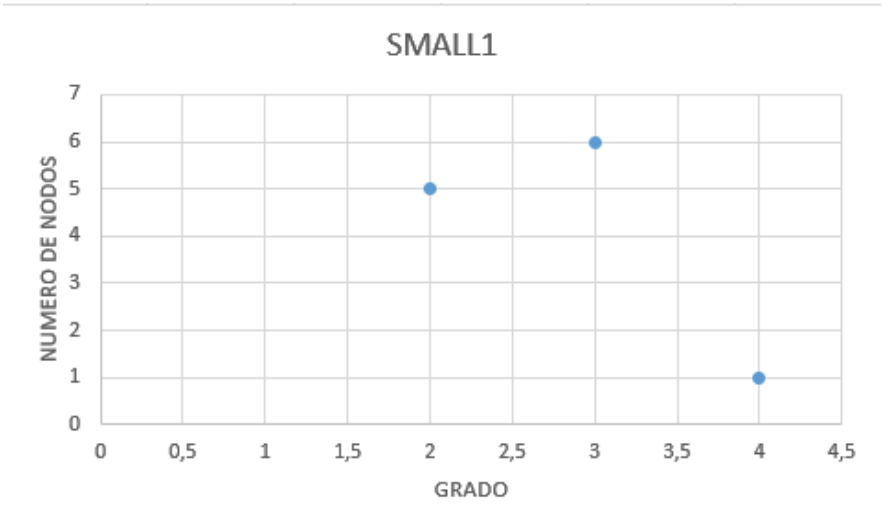
Tarea 6 - Preliminares.

DISTRIBUCIONES DE LOS GRADOS

En los siguientes apartados **u1** denotará el promedio del grado promedio de los amigos de los nodos y **u2** el grado promedio sobre las relaciones de amistad(arcos).

SMALL1

Con escala decimal:



Grado promedio(AVG g(u)): 2.6666666666666665

u1: 2.8472222222222228

u2: 2.8125

mediana: 3.0

Se cumple la primera y segunda ecuación de la paradoja de la amistad:

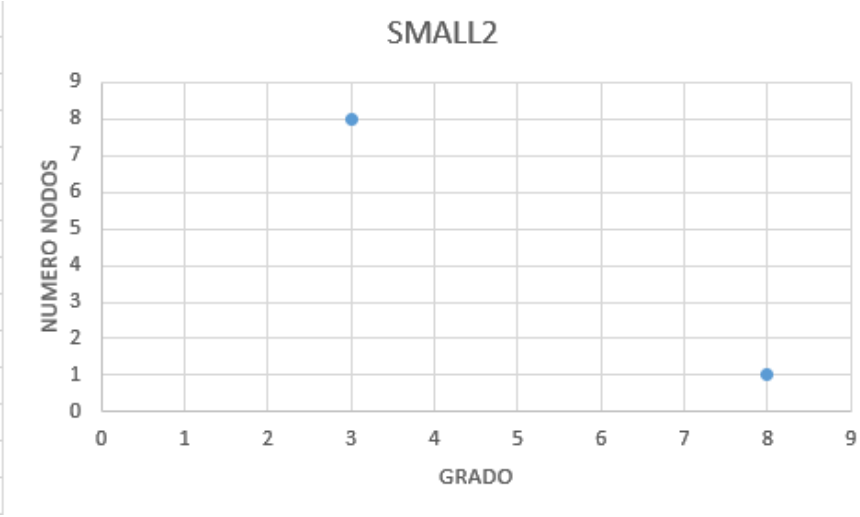
Grado promedio <= u1 **(**2,6 < 2,84)

Grado promedio <= u2: (2,6 < 2,8125)

NO se cumple la tercera ecuación de la paradoja de la amistad:

Mediana <= Grado promedio: 3> 2,6

SMALL2



Grado promedio: 3.5555555555555554

u1: 4.481481481481482

u2: 4.25

mediana: 3.0

Se cumple la primera y segunda ecuación de la paradoja de la amistad:

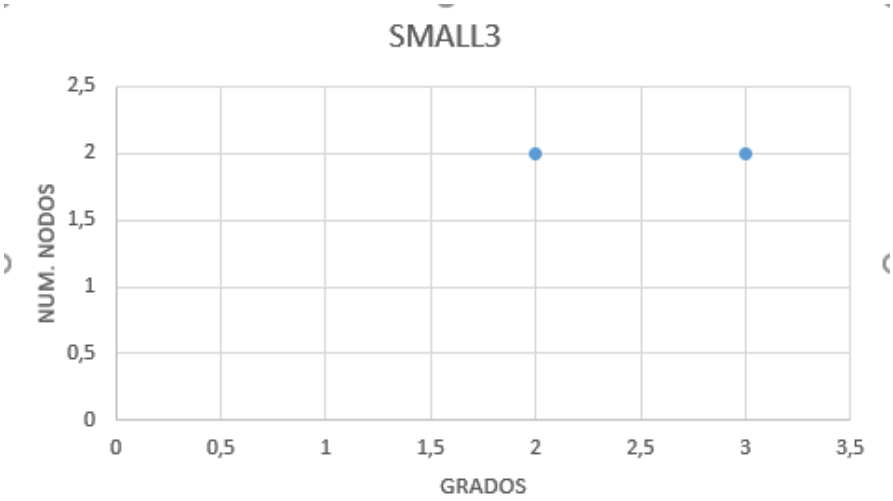
Grado promedio <= u1 **(3.55** < 4.48)

Grado promedio <= u2: (3.55 < 4.25)

NO se cumple la tercera ecuación de la paradoja de la amistad:

Mediana <= Grado promedio: 3> 3.55

SMALL3



En las 3 disponemos de muy pocos datos como para intentar ajustar las gráficas al modelo deseado.

Grado promedio: 2.5

u1: 2.666666666666667

u2: 2.6

mediana: 3.0

Se cumple la primera y segunda ecuación de la paradoja de la amistad:

Grado promedio <= u1 **(**2,5 < 2,66)

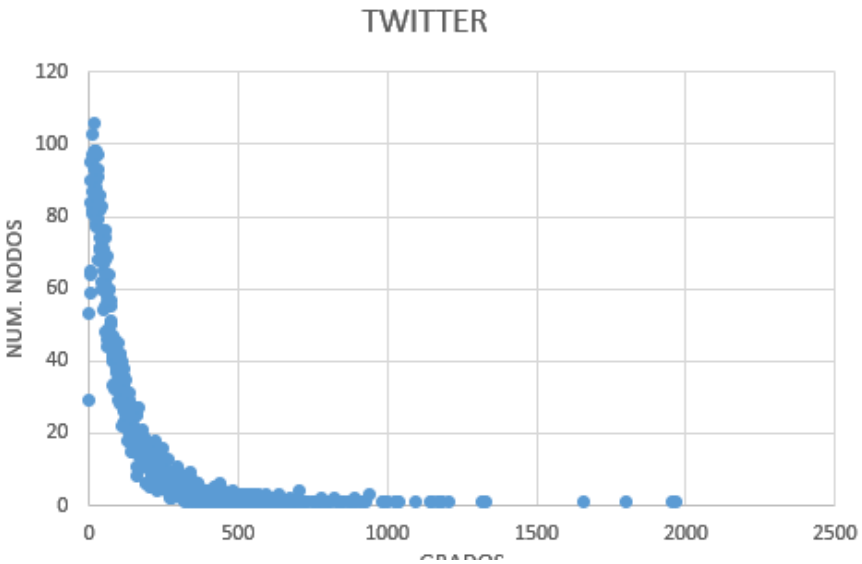
Grado promedio <= u2: (2,5 < 2,6)

NO se cumple la tercera ecuación de la paradoja de la amistad:

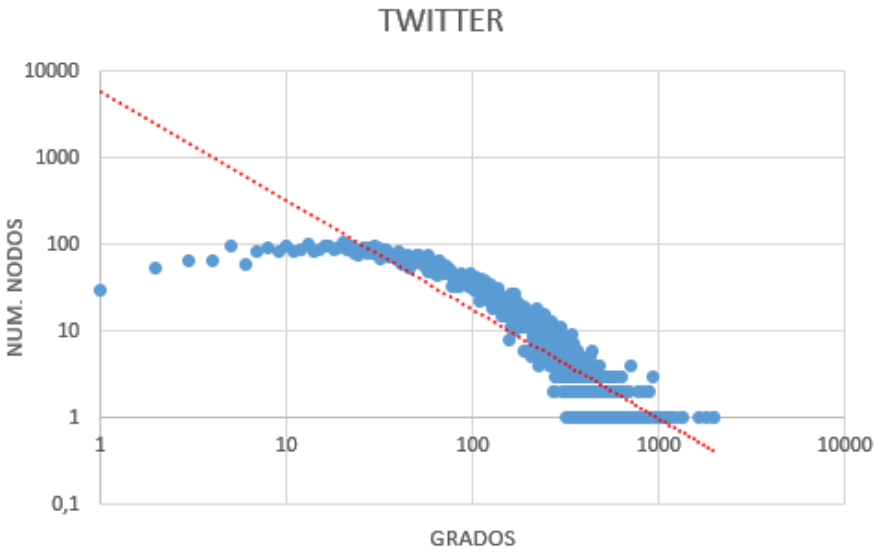
Mediana <= Grado promedio: 3> 2,5

TWITTER

En base decimal, vemos que es candidata a cumplir la Power Law:



Le aplicamos la escala logarítmica(en rojo la línea de tendencia de potencial):



Obsevamos como la Power Law se ajusta, en general, bastante bien.

Grado promedio: 92.21238408615017

u1: 237.106043316598

u2: 230.52376627112082

mediana: 58.0

Se cumple la primera y segunda ecuación de la paradoja de la amistad:

Grado promedio <= u1 **(**92.21 < 237.106)

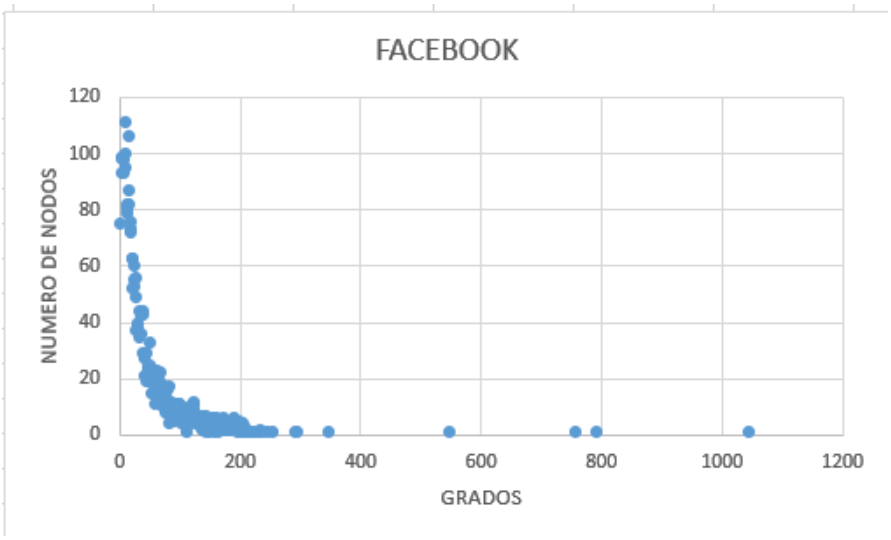
Grado promedio <= u2: (92.21 < 230.52)

SI se cumple la tercera ecuación de la paradoja de la amistad(ya que la distrib. Es decreciente):

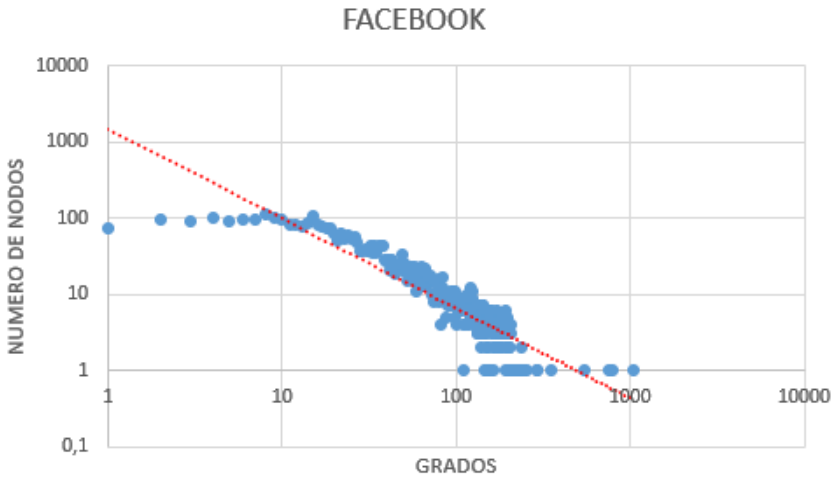
Mediana <= Grado promedio: 58 < 92.21

FACEBOOK

En base decimal, vemos que es candidata a cumplir la Power Law:



Le aplicamos la escala logarítmica (en rojo la línea de tendencia de potencial):



Obsevamos como la Power Law se ajusta, en general, bastante bien.

Grado promedio: 43.69101262688784

u1: 105.55179301541877

u2: 106.56983702427635

mediana: 25.0

Se cumple la primera y segunda ecuación de la paradoja de la amistad:

Grado promedio <= u1 **(**43.69 < 105.55)

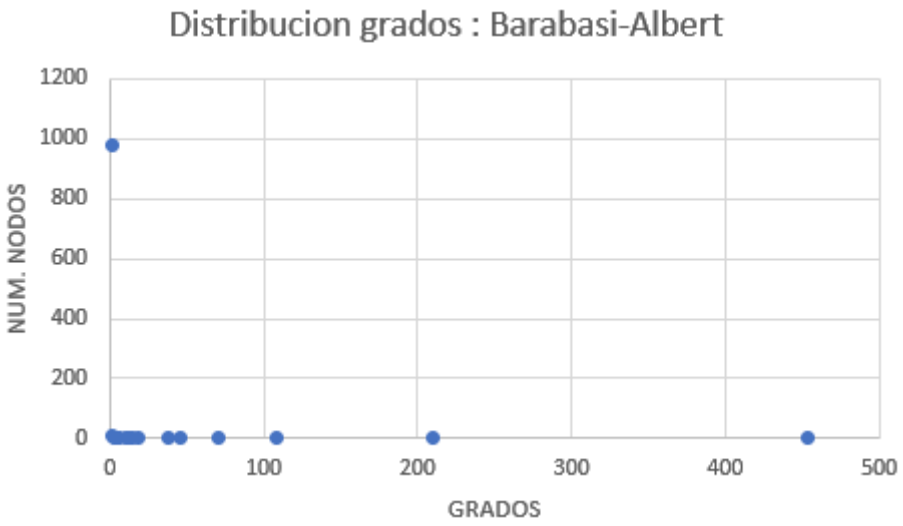
Grado promedio <= u2: (43.69 < 106.5698)

SI se cumple la tercera ecuación de la paradoja de la amistad(distribucion de grado decreciente):

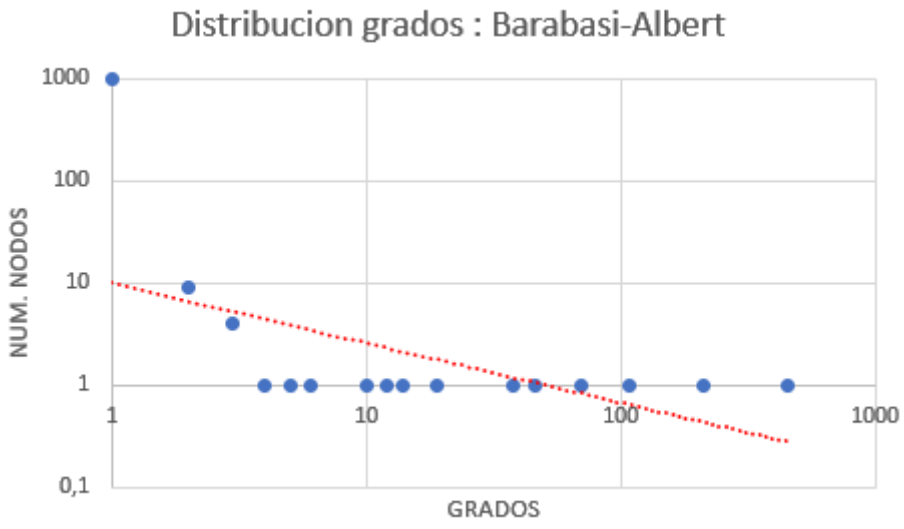
Mediana <= Grado promedio: 25< 43.69

BARABASI-ALBERT

Se ha generado con 2 nodos iniciales semilla,1000iteraciones(es decir 1002 nodos) y 2 enlaces por interacción. En base decimal, vemos que es candidata a cumplir la Power Law:



Le aplicamos la escala logarítmica (en rojo la línea de tendencia de potencial):



Obsevamos como la Power Law se ajusta de manera un tanto pobre.

Grado promedio: 1.99601593625498

u1: 269.36107607663257

u2: 135.8622754491018

mediana: 1.0

Se cumple la primera y segunda ecuación de la paradoja de la amistad:

Grado promedio <= u1 **(**1.996 < 269.36)

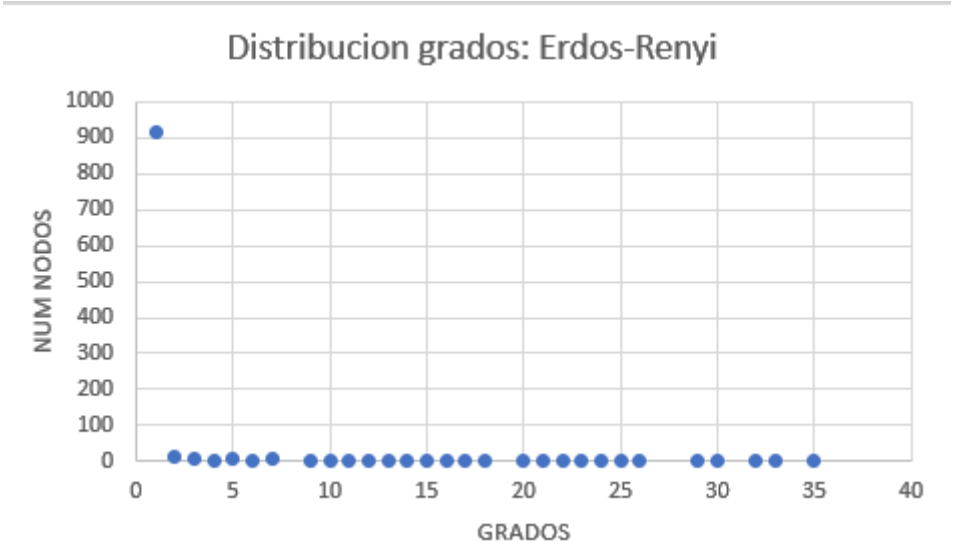
Grado promedio <= u2: (1.996 < 135.86)

SI se cumple la tercera ecuación de la paradoja de la amistad(distrib decreciente):

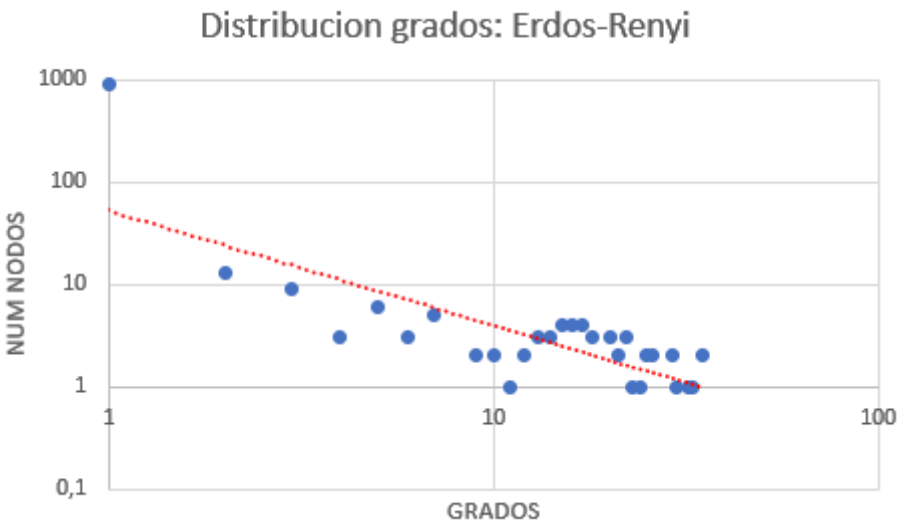
Mediana <= Grado promedio: 1<1.996

ERDOS-RENYI

Se ha generado para 1000 nodos y una probabilidad de formar enlace p = 0.5. En base decimal, vemos que es candidata a cumplir la Power Law:



Le aplicamos la escala logarítmica (en rojo la línea de tendencia de potencial):



No podemos afirmar que se ajuste a la Power Law.

Grado promedio: 1.998003992015968

u1: 18.62710754861409

u2: 10.94005994005994

mediana: 1.0

Se cumple la primera y segunda ecuación de la paradoja de la amistad:

Grado promedio <= u1 **(**1.998 < 18.627)

Grado promedio <= u2: (1.998 < 10.940)

SIse cumple la tercera ecuación de la paradoja de la amistad(distrib decreciente):

Mediana <= Grado promedio: 1 < 1.998

Tarea 7 - Métricas

Las métricas implementadas se han diseñado de acuerdo a las fórmulas expuestas, exceptuando `ClusteringCoefficient`, la cual explicaremos en más detalle a continuación:

El *coeficiente de clustering* se define como triángulos en los que el nodo origen es el mismo que el destino dividido entre todos los posibles triángulos. Para calcular el numerador apenas se requiere de mucho cómputo, pero el segundo problema se convierte en complejo.

Por ello hemos usado otra definición de esta métrica. Se dice que el *coeficiente de clustering* es los nodos vecinos del nodo a conectados entre sí, dividido entre todos los vecinos del nodo a. Este problema ya es más fácil de tratar, y en Java usando los *Set* se puede codificar en apenas unas líneas.

Tarea 7.1 - Rendimiento de las métricas

A continuación, se muestra una tabla con los tiempos obtenidos en las ejecuciones. Es importante destacar que el hardware con el que se cuenta. En concreto las pruebas se han realizado sobre un portátil con un procesador *i5 6200U* con una frecuencia de 2.4 GHz y 8 GB de memoria RAM.

Métrica	Small 1	Small 2	Small 3	Twitter	Facebook
UserClusteringCoefficient	12 ms	3 ms	1 ms	20s 631ms	1s 831ms
Embeddedness	4 ms	1 ms	2 ms	8min 8s 486ms	24s 556ms
ClusteringCoefficient	1 ms	0 ms	0 ms	19s 647ms	1s 517ms
AvgUserMetric	8 ms	1 ms	0 ms	19s 752ms	1s 470ms
Assortativity	1 ms	0 ms	0 ms	264ms	19ms