Memoria P4 BMINF

Jorge Cifuentes y Alejandro Martín

Ejercicios implementados: todos menos el último.

Ejercicio 1

Hemos implementado Features y Ratings usando mapas de mapas, parseando la entrada y rellenándolos. En ratings, el RandomSplit se hace con un random.nextInt para dividir los conjuntos de entrenamiento y test.

También hemos implementado el recomendador Average.

Ejercicio 2

Hemos implementado los recomendadores por vecinos próximos:

UserKNN

En mi pc: 20-40 segundos (4GB ram) Como lo hemos aligerado tanto (ver código):

AbstractRecommender:

Iteramos solo sobre los items que el user no ha valorado aún.

AbstractUserKNNRecommender

Clase básica. Solo calculamos el ranking de vecinos más próximos a cada user una vez, después lo leemos de una tabla hash donde se almacena (este cambio fue el que más tiempo redujo).

CosineUserSimilarity

Para los 3 sumatorios, solo iteramos sobre el set exacto de items (de x e y, de x, y de y), así evitamos iteraciones largas y chequeos de null innecesarios.

Los sumatorios de abajo (raíz de todos los ratings del user al cuadrado) son constantes independientemente del otro user, así que los calculamos una sola vez y los guardamos en una hash

UserKNNRecommender y NormUserKNNRecommender apenas tienen código, está casi todo en AbstractUserKNNRecommender**.**

Ejercicio 3

Hemos implementado el centroide:

CentroidRecommender / CosineFeatureSimilarity

En el Centroide creamos una FeaturesImpl de (Users x Features) y se lo pasamos a la similitud. Con el centroide ya hecho (dos FeaturesImpl) en la similitud, podemos calcular el rating.

Ejercicio 4

ItemNNRecommender / CosineItemSimilarity

Es simple: para cada ítem vecino (k = todos los vecinos) calcula su el rating que le da el usuario (acceso a hash) por la similitud entre el ítem base y el vecino (de nuevo acceso a hash, ya que precalculamos todas las similitudes en CosineItemSimilarity, de nuevo teniendo en cuenta la simetría).

ItemNNRecommender / JaccardFeatureSimilarity

Solo varía que en la similitud, hacemos Jaccard: tamaño de la intersección de sus características entre tamaño de la unión de estas.

En el ItemNN cabe comentar que, para reducir el tiempo, hemos guardado la suma de ratings de cada item, para no tener que recalculalro cada vez (mapa de item – double). También hemos aprovechado que el vecindario de items es simétrico (sim (i, j) = sim (j, i)), y al iterar calculamos la similitud del actual con los siguientes, ya que la similitud con los anteriores ya está hecha en la anterior iteración.

UserKnnRecommender / PearsonSimilarity

Hemos añadido un **StudentTest** que hace una recomendación kNN basada en usuario, pero que para la similitud aplica la correlación de Pearson.

Ejercicio 5

Hemos añadido la salida en .txt se adjunta en la entrega, y el tiempo de ejecución corresponde a ejecutar la métrica, no a la primera ejecución normal. También hay que decir que, lógicamente, las métricas que salen de un RandomSplit variarán ligeramente su valor.

En cuanto **al conjunto pequeño**, el que mejor precisión tiene es el UserBasedkNN (0.11). También es el que tiene mejor recall (0.2). El segundo en ambos aspectos es el ItemBasedkNN con coseno. El mejor RMSE (el más pequeño, al ser una suma de errores de estimación) es el del NormalizedUserBasedkNN. El de la versión sin normalizar es más alto, ya que predice rankings, pero no ratings (no están en rango 0-5). El peor es claramente la recomendación por mayoría, dado que no es nada personalizado a cada usuario, por lo que el error entre rating real y estimado es muy grande.

En el **conjunto grande** se repiten estas situaciones, con diferentes números pero bastante parecidos.

En ambos casos el recomendador por la Media (average), aunque tiene poca precisión y recall, no tiene un RMSE muy alto, por lo que se puede intuir que las puntuaciones reales que han tenido que

ser predichas son similares a las reales de los demás usuarios (se mantienen cerca de la media), lo que podría indicar una base homogénea de usuarios.

En ningún conjunto la recomendación por centroide parece especialmente efectiva, tal vez debido a características no completas de los ítems.

	RMSE	P@10	R@10	Tiempo Ejecución
Majority Recommender	590.450927408723	0.06900149031296594	0.11549545350876181	544ms
Average Recommender	1.0302963715887947	4.470938897168406E-4	4.0943108922438936E-4	569ms
User-based kNN	16.576892679651046	0.11162444113263797	0.20584364915145884	23s 476ms
Normalized user-based kNN	0.7762472529350829	0.006855439642324887	0.022219013329701463	22s 482ms
Item-based NN (cosine)	89.12323450784845	0.0906110283159466	0.16141077203464343	2min 57s 553ms
Centroid-based	3.83571008190322	0.015499254843517116	0.02753181265739761	12s 774ms
Item-based NN (Jaccard on ítem features)	4.318787739342637	0.011028315946348718	0.019625454905831966	47s 780ms

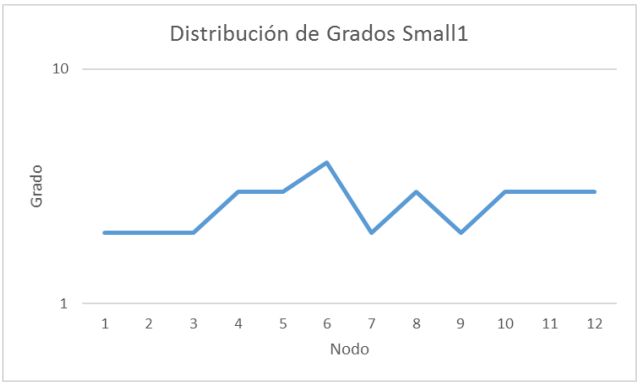
Conjunto pequeño

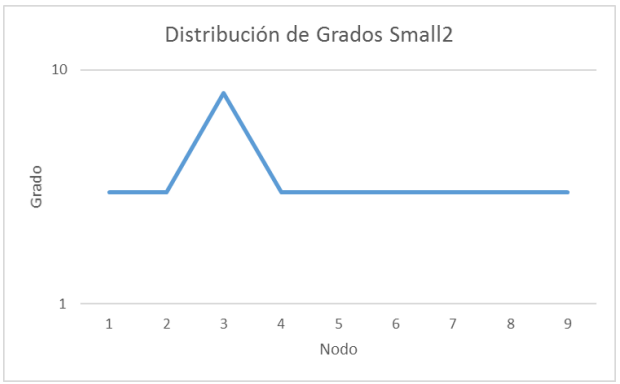
Conjunto grande

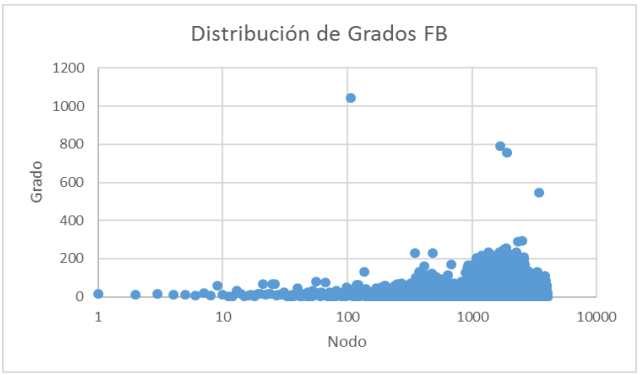
	RMSE	P@10	R@10	Tiempo Ejecución
Majority Recommender	3381.060922745586	0.15475627070515835	0.1306060087686793	1s 865ms
Average Recommender	0.6346974045600309	9.46521533364884E-5	4.1231051642788514E-5	2s 87ms
User-based kNN	34.56044746845366	0.2002366303833417	0.1740237849860835	2min 22s 833ms
Normalized user-based kNN	0.7096894261133271	0.014292475153809811	0.013170644520997995	2min 23s 358ms
Item-based NN (cosine)	360.7664049693128	0.1509701845716987	0.12705654735030028	28min 54s 202ms
Centroid-based	3.641684682936546	0.07988641741599539	0.06878762637838909	52min 48s 532ms
Item-based NN (Jaccard on ítem features)	26.95907774978391	0.10738286796024478	0.10018334500264146	31min 20s 787ms

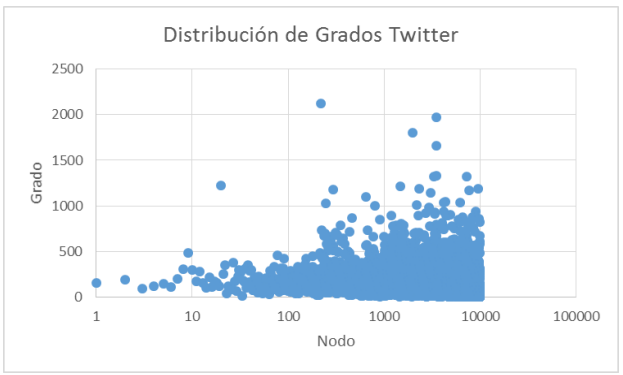
Ejercicio 6

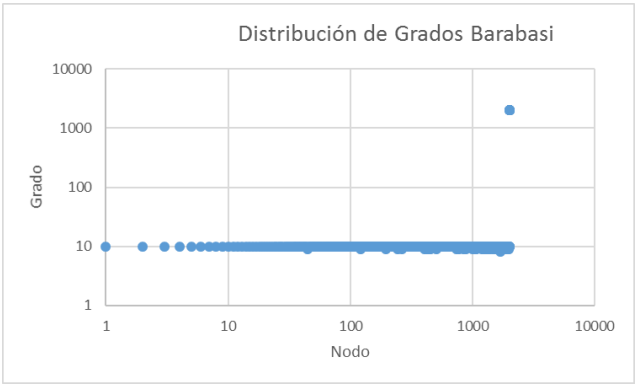
Tras programar los algoritmos para la generación de las redes sociales Barabási-Albert y Erdös-Rényi, se generan las gráficas de las 6 redes sociales.

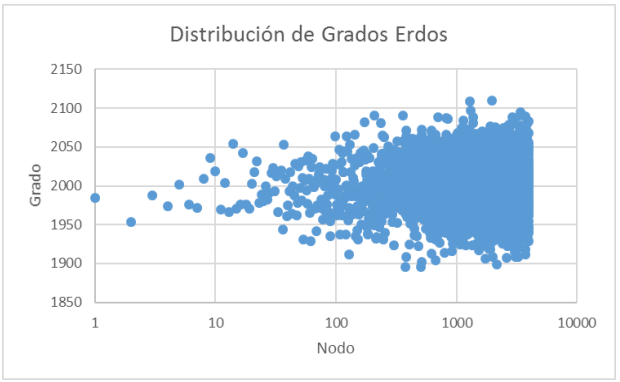












Podemos observar una distribución power law en las redes sociales Fb y Twitter, mientras que las otras redes sociales no tienen dicha distribución.

En cambio, la paradoja de la amistad no se cumple ya que aplicando la fórmula, el número promedio de amigos de los amigos es mayor que el número promedio de amigos por persona.

Ejercicio 7

Los tiempos de ejecución obtenidos al probar las métricas son:

	Facebook	Twitter

	Facebook	Twitter
Coef. Clustering usuario	1s 261ms	14s 700ms
Embeddedness	1min 14s 310ms	45min 23s 731ms
Coef. Clustering global	953ms	26s 354ms
Asortatividad	21ms	186ms