IIC 3633 - Recommender Systems
Homework Assignment 1

Paula Navarrete Campos & Astrid San Martín
Department of Computer Science
School of Engineering
Pontifical Catholic University
pcnavarr@uc.cl, aesanmar@uc.cl

This work develops a recommendation system for the beer industry based on Collaborative Filtering. Schafer et al. (2007) [1] and Koren et al. (2009) [3] disentangle key concepts on collaborative filtering (CF), exposing its main uses, algorithms and design decisions regarding rating systems and ratings acquisition. There is some common grounded theory to take into account when designing and developing a recommender system, mainly regarding approach, data nature and purpose of the software.

I. Introduction and Objectives

Collaborative Filtering is the process of filtering or evaluating items using the opinions of other people, taking its roots in the old human behavior of sharing opinions. There are some key concepts in this approach to making recommendations.

- *Rating* consists on associating two things, user and item, usually through some value. This can be visualized through a *ratings matrix*, where each row represents a user and each column represents an item, with the intersection being the value of the rating (the absence of value means that the user has not evaluated the item). The measure of the rating can be a scalar (numerical measure), binary (as decisions type agree/disagee, like/dislike) or unary (for example, if the user saw the item, or if he rated it positively).
- *User* corresponds to the person who provides ratings to the system or who uses it to receive information.
- Collaborative filtering systems produce recommendations or predictions for a user and at least one item, which can be anything that can be “evaluated” by a human.

II. Exploratory Data Analysis

We present in this section a brief analysis of the data. It is important to get acquainted with the different kind of data we are dealing, its shape and distribution have a lot to say when choosing an optimal solution. All the code produced to generate this report can be consulted in the following GitHub Repo url: https://github.com/paulanavarretec/RecSys-Tarea1.

Table VI shows the first and last rows of the available data.

userID	itemID	styleID	rating	brewerID	timestamp
4924	11757	84	4.5	1199	1247372118
4924	5441	2	4.5	1199	1209176445
4924	19960	84	5.0	1199	1223914717
...					
6118	20470	64	4.0	394	1204330634
6118	1324	59	3.5	263	1212967655
7268	1504	13	5.0	568	1157647130

TABLE I
Raw data

userID	8318
itemID	1836
styleID	95
rating	10
brewerID	210
timestamp	43905

TABLE II
Number of unique instances for each feature

As we can see, users and items are represented by an integer value, style and brewer too, rating is represented with a decimal number and they all range in a wide spectrum. But we need a little bit more information to see if we have enough users and/or items to be able to predict something.

Table II depicts how many different users, items, styles, rating, brewerID and timestamps there are in the whole set. It shows we have a large sample, compounded by many users and (a lot less) items, and we have more users than items (as expected). We have ten different decimal ratings (we guess 0 to 5 incremented by 0.5). We can see we have a lot less different styles and brewers.

1) **Density**

- Ratings
  Table III takes a deeper look into the shape of the data, to do that, we use different statistic functions provided by *pandas library*. It is important to notice that the overall mean rating is 3.8 and the lowest 25% of the ratings are below 3.5 and the highest 25% of the ratings are above 4.5.

  The skewness of a sample is a measure for symmetry, and encompasses the lack of it. Any

mean	3.865
std	0.713
min	0.0
25%	3.5
50%	4.0
75%	4.5

TABLE III
RATING DESCRIPTION

symmetric data should have a skewness near zero: negative values indicate data that are skewed left and positive values for the skewness indicate data that are skewed right. By skewed left, we mean that the left tail is long relative to the right tail and the skewness of the rating feature ($skew = -1.01$) refects this.

The kurtosis of a sample is a measure of whether the data are heavy-tailed or light-tailed relative to a normal distribution which has a kurtosis of zero. Data sets with high kurtosis (positive values) tend to have heavy tails or outliers, which is our case fot the ratings ($kurt = 1.6$), and data sets with low kurtosis (negative valueas) tend to have light tails, or lack of outliers.

All characteristics mentioned above can be visualized on figure 1. We can appreciate the ratings are biased to 4.0 and they tend to accumulate around it.

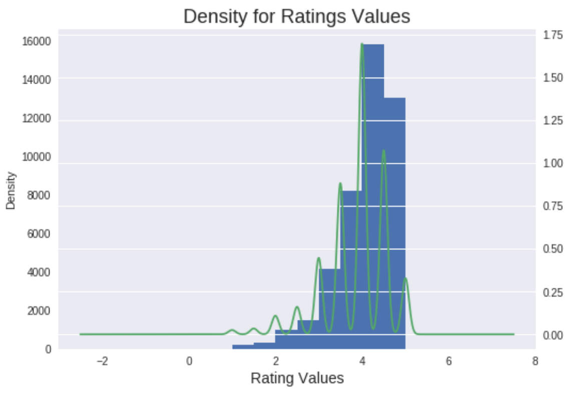

Fig. 1. Density for user rating observations

Users: The shape of the figure 2 is congruent with the tendency to have less users in the beginnig (the so-called early adopters) giving lots of ratings and helping the system to learn, and afterwads, a lot of new users giving a lot of less reviews. The same can be chequed out analogously for the items. We checked skewness ($skew = 0.5$) and kurtosis($kurt = -0.9$) and confirmed bigger right tail and lack of outliers.

Items: It is interesting what the figure 3 shows. We can see a sort of different waves, characterized by a peak of ratings for some items, but a comparable bigger tail of items rated comparatively less. This may suggest that new items were introduced in 4 to 5 waves along time, where we initial introductions brought several more reviews per item introduced than the later, making sense with the previous analysis of users.

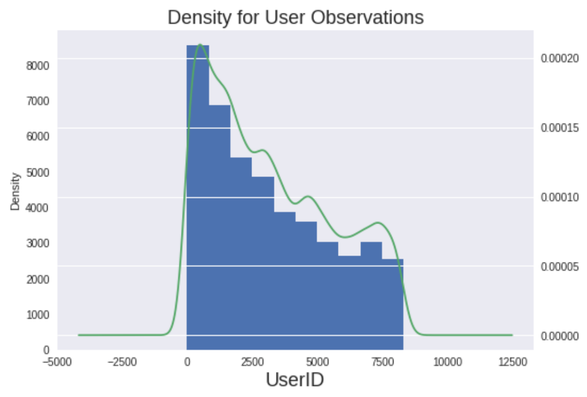

Fig. 2. Density for user rating observations

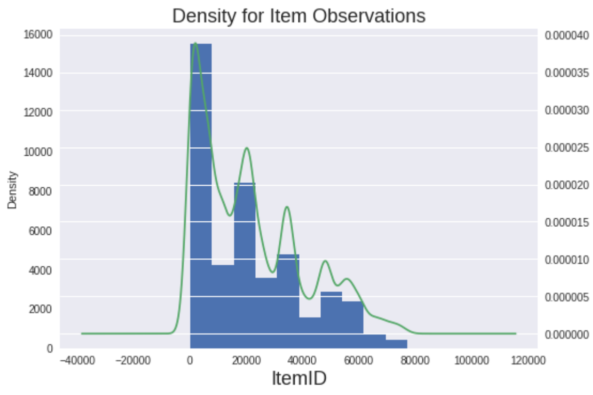

Fig. 3. Density for item rating observations

2) **Distribution**

- **User/Item**: we calculated the number of ratings per user. Table IV shows a brief description of the data. The table is very informative, we can see that at least 25% of the users have given just 1 ranting, and that no more than 25% of the users have given more than 5 ratings. In average, users give 5 ratings. The positive value for the skewness ($skew = 5$) indicates that the right tail is bigger than the left, wich is consistent with our previous knowledge that there should be many users rating few items and much more less rating a lot. The kurtosis ($kurt = 38.7$) high value is also consistent with this, showing the presence of outliers (users at the top of table). This can be clearly seen in the distribution figure below. Figure 4 depicts this, although all this analysis relates userID with number of ratings, this tells us that the users with very few ratings are the ones with higher userID (maybe the ones that joined the

	userID	userRatings
count	8318	8318
mean	4159.91	5.33
std	2401.78	9.9
min	1	1
25%	2080	1
50%	4159	2
75%	6239	5
max	8320	181

TABLE IV

DESCRIPTION OF USER OBSERVATIONS PER USERID

	userID	userRatings
count	1836	1836
mean	40440	24
std	22286	113
min	175	1
25%	21640.	1
50%	41525	2
75%	60511	7
max	77207	2205

TABLE V

DESCRIPTION OF RATING OBSERVATIONS PER ITEMID

system afterwards the early adopters),

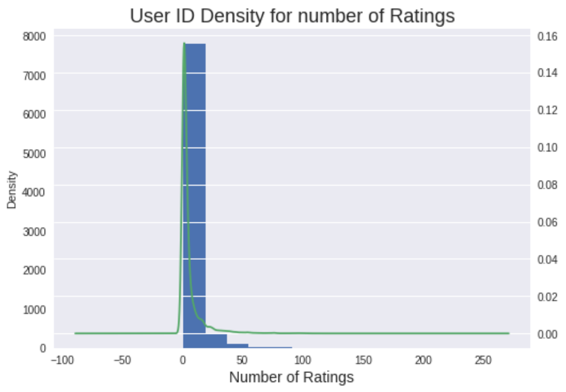

Fig. 4. Density for item rating observations

We worked the data a little in order to explore how the distribution of number of ratings per user behaves. We got a positive value for the skewness ($skew = 0.96$) indicates that the right tail is bigger than the left, which is consistent with our previous knowledge that there should be few users rating a lot of items and substantially more rating just a few. The kurtosis positive value ($kurt = 1.67$) is also consistent with this, showing the presence of outliers (users at the top of table that are far away from the mean). This can be clearly seen in the distribution figure 5.

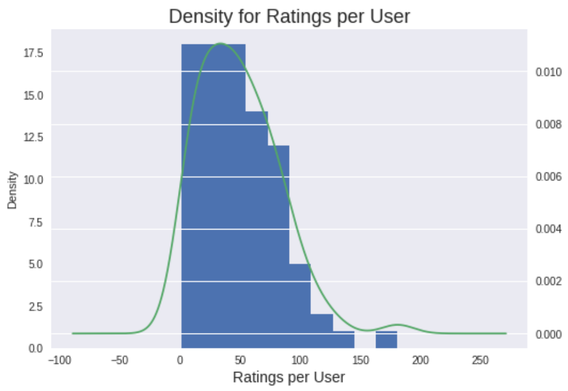

Fig. 5. Distribution for user number of ratings

- **Item/User**: Making the same analysis as before for the rated items distribution, we worked the data to get the number of ratings per item. Table V resumes the relevant information. We can see that at least 25% of the items have been rated just once and that no more than 25% of the items have been rated more than 7 times. Items get approximately 24 ratings In average. The positive value for the skewness ($skew = 10.32$) indicates that the right tail is bigger than the left, which is consistent with our previous knowledge that there should be many items with few user ratings and much more less with a lot of rating. The kurtosis extreme high value ($kurt = 136.46$) is also consistent with this, showing the presence of outliers (items at the top of table, we infer the most popular items). This can be clearly seen in the distribution figure 6.

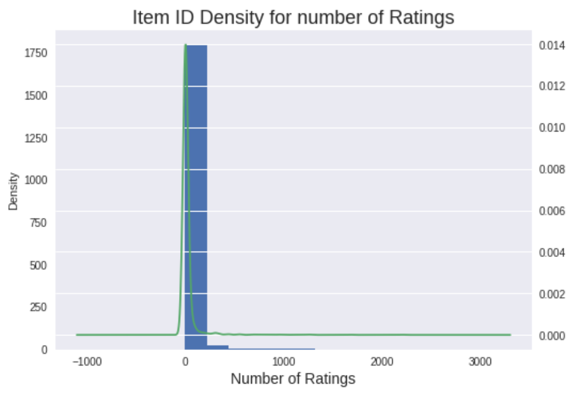

Fig. 6. Distribution for user itemRatings

Although, same as before, the figure relates itemID with number of ratings, this tells us that the items with a lot of ratings are the ones with lower itemID (maybe the items that joined the system in the first stages and got much more time of exposure to user than the later ones), we worked the data to explore how the distribution of number of ratings per item behaves. We found that at least 25% of the items have less than 40 ratings and at most 25% of the items hace more tan 230 ratings, on average items have aproximatelly 200 ratings. The skewness ($skew = 3.02$) indicates that the right tail is bigger than the left, wich is consistent with what we have discussed and the kurtosis ($kurt = 11.14$) value reveals the presence of the

Most Popular items outliers (items at the end of table that are far away from the mean). This can be clearly seen in the distribution figure 7.

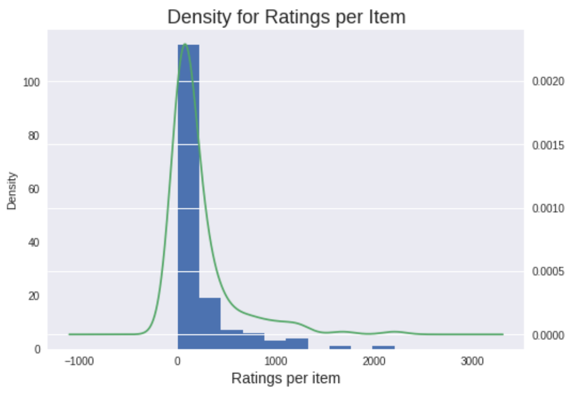

Fig. 7. Distribution for item number of ratings

III. Experimental Design

Describir de forma breve la metodología de los experimentos: Qué se hizo con los datos y por qué (particiones, cross-validation, etc.), qué métodos se usaron y por qué. calcular también tiempo de ejecución, procesamiento y memoria requeridos.

1) **UserKnn:** This algorithm interprets similar users as neighbors, if the user n is similar to the user u, then n is a neighbor of u. It generates a prediction for an item i by analyzing the ratings for i from the users in the neighborhood of u. Where:

$$pred(u,i) = \bar{r} + \frac{\sum\limits_{n \subset N(u)} uSim(u,n) * (r_{ni} - \bar{r}_n)}{\sum\limits_{n \subset N(u)} uSim(u,n)} \tag{1}$$

Donde $N(u)$ son los vecinos de u y corresponde a la predicción para item i que evalúa los ratings para ese i de los vecinos de u. Esta fomulación corrige las variaciones en la escala de rating y pondera cada uno por las similaridades de cada usuario con u (el término inferior corresponde a la normalización del resultado). Una forma de medir la similaridad expresada en la formulación anterior es a través de la correlación de Pearson, donde:

$$uSim(u,n) = \frac{\sum_{i \subset CR(u,n)} (r_{ui} - \bar{r}_u)(r_{ni} - \bar{r}_n)}{\sqrt{\sum\limits_{i \subset CR_{(u,n)}} (r_{ui} - \bar{r}_u)^2} \sqrt{\sum\limits_{i \subset CR_{(u,n)}} (r_{ni} - \bar{r}_n)^2}} \tag{2}$$

Where $CR_{(u,n)}$ are the coevaluated itemes between u and n and represents the similarity between the user u with user n and $\in [-1.0, 1.0]$, where 1.0 corresponds to a perfect similarity and -1.0 to its complement, in this context, we can choose only positive correlations to improve the prediction. This formulation analyzes similarities in ratings for items evaluated together with i (corrating).

Although $UserKnn$ captures how recommendations are arranged and can detect complex patterns, as the data is sparce, it does not achieve general consensus regarding an item and pairs of users with few corratings are prone to throw biased correlations that can dominate the user's neighborhood in question.

The algorithm can be implemented including all the users of the set as neighbors of each user, although by limiting it to the closest k neighbors to each user improves its accuracy and efficiency. The challenge is to choose a well suited k for the dataset. Even so, its implementation is expensive since it requires comparing each user with the complete set, so the time and memory for processing do not scale well as users and ratings increase.

2) **ItemKnn:**, estimaer mejor k

Item-based Nearest Neighbor (IBNN): Corresponden a transponer el problema anterior, mientras que los algoritmos user-based generan predicciones basadas en similaridades entre usuarios, los item-based lo hacen basándose en similaridades entre items, es decir, la predicción para un item se basa en ratings para items similares. Así, una predicción para un usuario u de un item i puede ser representada como la composición de sumas ponderadas de ratings del mismo usuario para los items mas similares de i (análogamente a la derivación (4)).

$$pred(u,i) = \bar{r} + \frac{\sum\limits_{n \subset RI(u)} iSim(i,j) * r_{ui}}{\sum\limits_{n \subset RI(u)} iSim(i,j)} \tag{3}$$

Donde $RI(u)$ corresponde a los itemes evaluados por u.

Como los datos con que se compone la predicción están dados por todos los correspondientes al mismo usuario, no es necesario centrar la suma ponderada como en el caso anterior. Análogamente, itemSim() corresponde a la medida de similaridad entre los itemes i y j. La métrica de similaridad mas popular y precisa para calcular esta similaridad es la del coseno subjacente ajustado que se muestra a continuación:

$$iSim(u,n) = \frac{\sum\limits_{u \subset RB_{i,j}} (r_{ui} - \bar{r}_u)(r_{uj} - \bar{r}_u)}{\sqrt{\sum\limits_{u \subset RB_{i,j}} (r_{ui} - \bar{r}_u)^2} \sqrt{\sum\limits_{u \subset RB_{i,j}} (r_{uj} - \bar{r}_u)^2}} \tag{4}$$

Donde el set $RB_{i,j}$ corresponde al set de usuarios que han dado rating tanto a i como a j. Aunque hay evidencia de que IBNN es más preciso que UBNN, de todas formas el tamaño del modelo crece cuadráticamente

Algorithm	parameters	MAE	RMSE	MAP@10	nDCG@10
UserKnn	k=30	0.5014	0.6572	0.0014	0.0010
ItemKnn	k=30	0.4919	0.6586	0.0058	0.0073
SlopeOne	-	0.4841	0.6471	0.0002	0.0002
SVD	f=1000-it=100	0.4536	0.6020	0.0178	0.0107
...					
ALS	-	-	-	-	-

TABLE VI
RESULTS

con el número de items. Hay diferentes técnicas para mejorar el uso de memoria, como limitar el procesamiento hasta k corratings o retener solo las n mejores correlaciones para cada item (esto puede provocar que los itemes correlacionados con los ratings del usuario no contengan el item objetivo)

3) **SlopeOne:**,
4) **SVD:**.
5) **ALS o ALScg:** este modelo lo puede usar para la tarea de ranking. Si tiene problemas con pyreclab, puede usar la biblioteca implicit1.

IV. Results

El informe debe tener una tabla donde las filas presentan los métodos y las columnas las metricas RMSE, MAE, MAP@10 y nDCG@10. Presente en esta tabla cada método sólo una vez. Si el método tiene parámetros, presente la versión del método con los parámetros que le dieron mejores resultados.

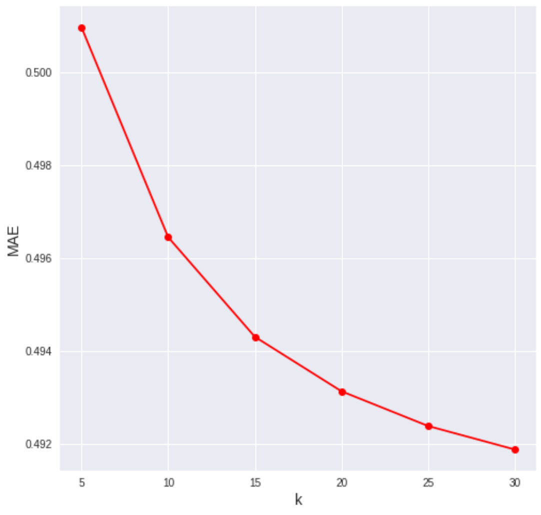

Fig. 8. MAE plot for ItemKnn

V. Results

Describir los resultados obtenidos. Además deben comparar y analizar los diferentes métodos respecto a, al menos,

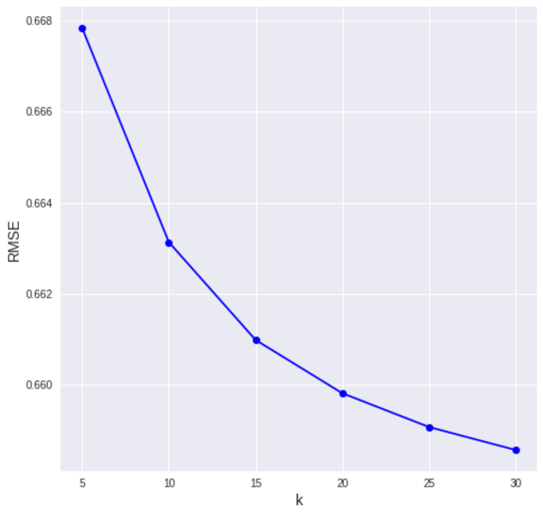

Fig. 9. RMSE plot for ItemKnn

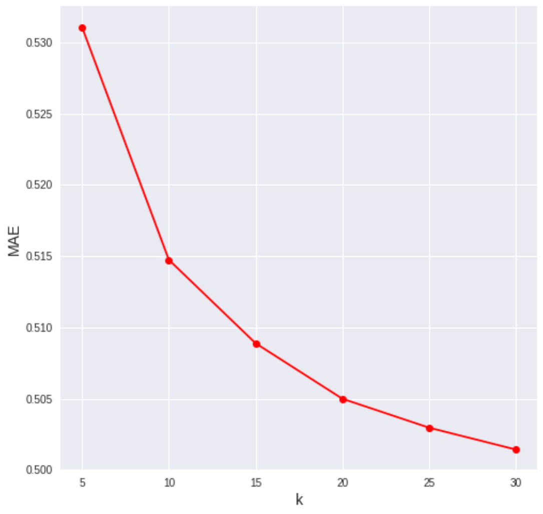

Fig. 10. MAE plot for UserKnn

su implementación (dificultades y otros), tiempo de ejecución, procesamiento y memoria requeridos, y métricas de rendimiento (RMSE y nDCG).

VI. Sensitivity Analysis (Discussion?)

Explicar el efecto de los diferentes parámetros que escogieron en los resultados obtenidos.

1) Item Knn: pretty straightforward. Optimal at 30 neighbors
2) User Knn:

We had to set an improvement threshold of 0.001 to

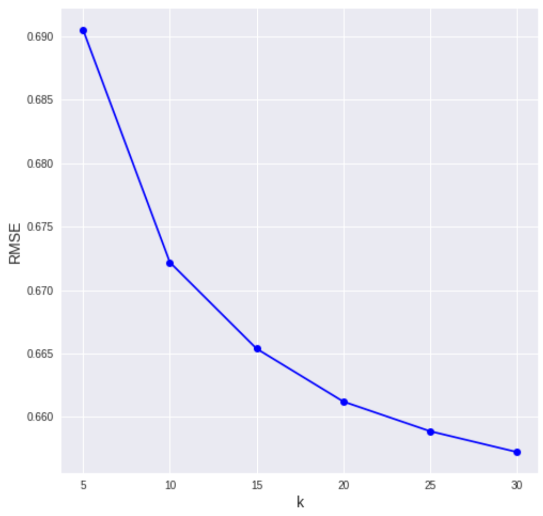

Fig. 11. RMSE plot for UserKnn

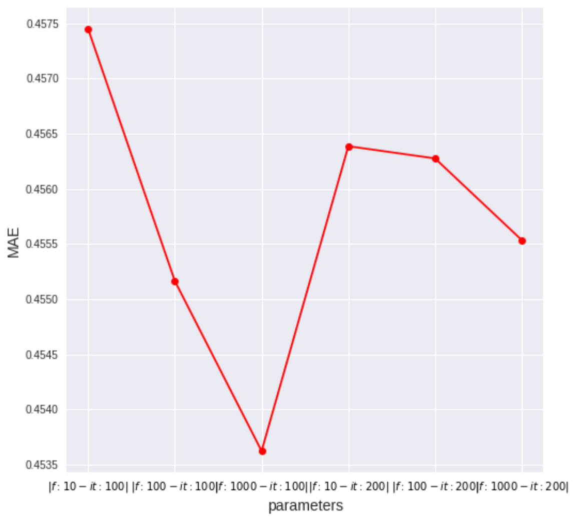

Fig. 12. MAE plot for SVD

stop the descent.

3) SVD:
- Factor analysis
  We had to set an improvement threshold of 0.001 to stop the descent.
  Min MAE factor: 0.455 con 150 factores
  Min RMSE factor: 0.602 con 200 factores
- Iterations analysis
  Plot looks the same as figure 16, the error Plot looks the same, the error diminishment is marginal:

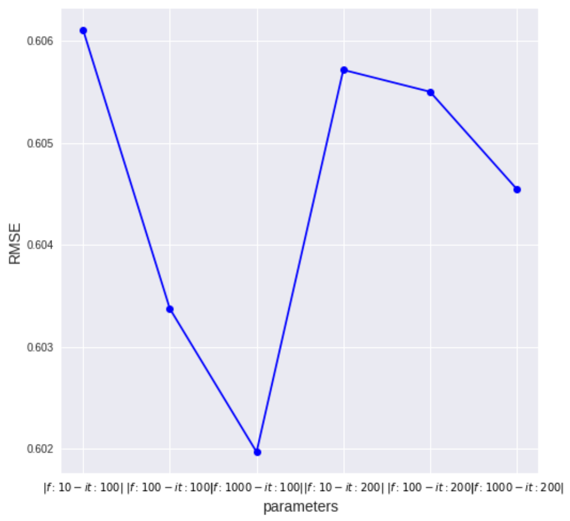

Fig. 13. RMSE plot for SVD

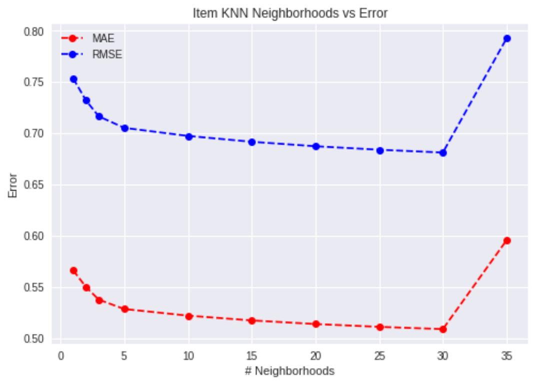

Fig. 14. MAE and RMSE Error against item KNN Neighborhoods

Min MAE iterations: 0.454 con 150 max-iterations
Min RMSE iterations: 0.601 con 200 max-iterations

VII. Conclusion

Conclusiones a las que se llegó luego de realizar la tarea.

References

[1] Schafer, J. B., Frankowski, D., Herlocker, J., Sen, S. (2007). Collaborative filtering recommender systems. In The adaptive web (pp. 291-324). Springer Berlin Heidelberg.
[2] How not to sort by Average Rating, Evan Miller Blog
[3] Koren, Y., Bell, R., Volinsky, C. (2009). Matrix factorization techniques for recommender systems. Computer IEEE Magazine, 42(8), 30–37.
[4] Hu, Y., Koren, Y., Volinsky, C. (2008, December). Collaborative filtering for implicit feedback datasets. In Data Mining, 2008. ICDM'08. Eighth IEEE International Conference on (pp. 263–272). IEEE.
[5] L. Van der Maaten and G. Hinton. Visualizing data using t-sne. Journal of Machine Learning Research, 9(2579-2605):85, 2008.

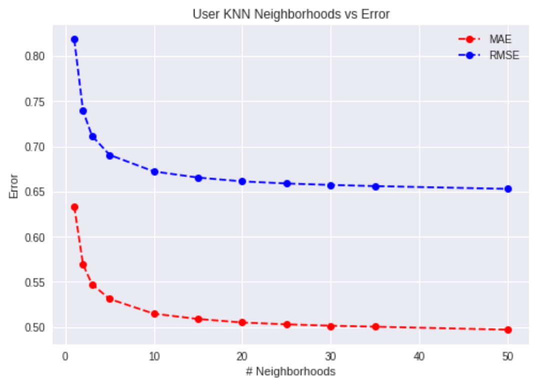

Fig. 15. MAE and RMSE Error against user KNN Neighborhoods

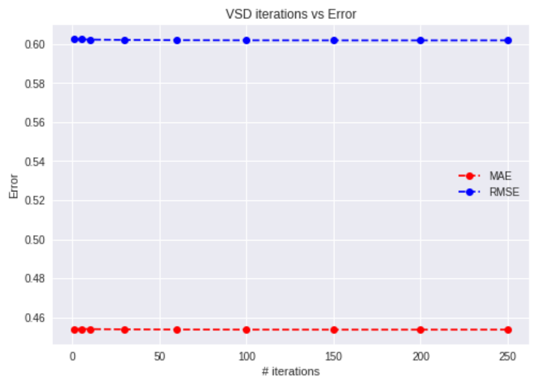

Fig. 16. MAE and RMSE Error against number of factors