Case Study: Kaggle MSD challenge competition

May 19, 2015

Problem Statement

Given:

1. The full listening history for 1M users
2. half of the listening history for 110K users (called the visible listening history),

propose a ranked list of 500 songs per user, in decreasing probability of this particular user listening to that song.

The results will be evaluated using the Mean Average Precision (truncated at 500) of the proposed ranking against the second half of the 110K users listening history. This second half of the listening history was not available to the competitors, which needed to send their proposition and receive in return the result of the Mean Average Precision through a web interface hosted at Kaggle.

Overview of possible methods

Content based

Content based methods extract song characteristics (such as Genre, beat, lyrics, ...) to build a model (Clustering, Bayesian Network, ...) and recommend songs similar to those the user has listened in the past.

Context based

Context base methods use social information such as comments, music review, tags, ... to predict similarities between songs. Other information such as characteristic of the user for which we want to make a prediction, such as its age, education, location, ... might also be taken into account to make a prediction.

Collaborative Filtering

Collaborative Filtering techniques[1] are based on the assumption that the listening history of all users provides valuable information that one could exploit to give recommendations to a specific user.

Different flavors of Collaborative filtering exists, which can be classified (at high level) by:

- their use of a model:
  - MEMORY BASED Collaborative Filtering methods use the entire set of user history to make a prediction, typically using a KNN approach to find similar users and propose the songs they listen to.
  - MODEL BASED Collaborative Filtering methods build and train a data mining prediction model (Clustering, Bayesian Network, ...) of the users preferences, that will later be used to provide a prediction for a specific user.
- their focus:
  - USER BASED Collaborative Filtering methods compute the similarities between the user A for which we would like to make a prediction and all other users, then proposing to user A the songs listened by those similar users, ranked in decreasing order of user A predicted preference.
  - ITEM BASED Collaborative Filtering methods, on the other hand, first pre-compute the similiarities between each songs, and use those similarities to provide user A with a ranked list of songs similar to the ones he listened to .

Dimensionality Reduction

Dimensionality Reduction methods like the Principal Component Analysis can be used to reduce the amount of data calculations are based on, then improving response times and often, also, recommendation performances.

Singular Value Decomposition can also be used to provide recommendations on its own.

Hybrid Methods

Different approaches can be used to combine the results of the several recommendation strategies into one, in an attempt to get higher recommendation performances. Those hybrid methods can be classified as follow:

- MONOLITHIC: creates a single recommendation component, using a complex recommending algorithm in its cores:
  - FEATURE COMBINATION: combines the scores of several scoring algorithms into one score used to rank the songs
- PARALLEL: uses two or more simple ranking algorithms in parallel to creates a higher level ranking:
  - WEIGHTING: assigns a weight to each ranking algorithm then, for each final recommendation, “randomly” chooses the next recommendation of one of the rankings according to their weights
  - SWITCHING: uses an oracle to decide which ranking algorithm to use for a specific user, by looking at the characteristics of this particular user (history length, number of nearest neighbors, sensibility to popular songs, ...)

- PIPE-LINED: uses two or more simple ranking algorithms one after each other to create a final ranking:
  - CASCADE: each successive algorithm excludes songs from the list of all possible songs to recommend, until the last recommender provide its ranking on the remaining songs.
  - META-LEVEL: each algorithm builds a model that can be exploited by the next one, until the last algorithm provides a ranking based on the model it received.

Chosen approach

The MSD challenge provides a listening history in the form of triplets <user, song, number of times listened>. Plenty of content information about the songs is also available in the Million Song Dataset, such as music genre, beat information, etc.

For this case study, I chose to evaluate the performances of Collaborative Filtering methods, based upon the user listening history, and more specifically of Memory Based methods (both User Based and Item Based), using a simple weighted sum strategy to score the items for a particular user, following the method used in [2].

User and Item Based Collaborative Filtering

Collaborative Filtering methods recommend songs to a user u by associating to each song i a score s_{ui} proportional, or inversely proportional, to the predicted probability that user u will listen to song i, then ranking them in decreasing or increasing order according to the score type

In the following, I will use the words "song" and "item" interchangeably.

Given:

- $\mathcal{U}$ as the set of n users $u =< i_1, i_2, ..., i_m >$
- $\mathcal{I}$ as the set of m items $i =< u_1, u_2, ..., u_n >$
- $\mathcal{U}(i)$ as the set of users listening to item i
- $\mathcal{I}(u)$ as the set of items listened by user i
- w_{uv} as the similarity between users u and v
- w_{ij} as the similarity between songs i and j
- $f(w)$ as the scoring function

The score associated to item i for user u using a User Based Collaborative Filtering strategy is then:

$$s_{ui} = \sum_{v \in \mathcal{U}(i)} f(w_{uv})$$

meaning s_{ui} is proportional to the similarities between user u and other users v listening to at least one song that user $\underline{u}$ listened to.

And using an Item Base Collaborative Filtering strategy, the score associated to item i for user u is:

$$s_{ui} = \sum_{j \in \mathcal{I}(u)} f(w_{ij})$$

meaning s_{ui} is proportional to the similarities between songs i listened by user u and songs j listened by other users who also listened to one song i

We still need to define 3 components:

- the scoring function $f(w)$
- the similarity measures w_{uv} and w_{ij}
- the feature construction method of the vectors $u =< i_1, i_2, ..., i_m >$ and $i =< u_1, u_2, ..., u_n >$

Scoring function

The scoring function $f(w)$ determines how much a weight for a similar user v or similar item j influence the overall score. By taking:

$$f(w) = w^q$$

we give more importance to the most similar users/items when q is higher. When $q = 1$, the overall score is simply the sum of the similarities. q is a parameter to tune in order to optimize the overall performance of the Collaborative Filtering algorithm.

Similarity measures

The most common similarity measure used to compare data in a highly dimensional and highly sparse environment is *cosine similarity*, giving the angle between 2 vectors. Its values range from 1.0 for two completely similar vectors, to -1.0 for 2 vectors of opposite direction, passing by 0.0 for two orthogonal vectors that don't share any information at all.

For User Based Collaborative Filtering, the similarity w_{uv} can be defined as the dot product of the vectors $\mathcal{I}(u)$ and $\mathcal{I}(v)$ divided by the lengths of the 2 vectors:

$$w_{uv} = \frac{\mathcal{I}(u) \times \mathcal{I}(v)}{|\mathcal{I}(u)|^{\frac{1}{2}} |\mathcal{I}(v)|^{\frac{1}{2}}}$$

For Item Based Collaborative Filtering, the similarity w_{ij} is defined similarly:

$$w_{ij} = \frac{\mathcal{U}(i) \times \mathcal{U}(j)}{|\mathcal{U}(i)|^{\frac{1}{2}} |\mathcal{U}(j)|^{\frac{1}{2}}}$$

In [2], the winner of the competition proposes to introduce a parameter α in the cosine similarity, as follow:

$$w_{uv} = \frac{\mathcal{I}(u) \times \mathcal{I}(v)}{|\mathcal{I}(u)|^{\alpha} |\mathcal{I}(v)|^{1-\alpha}}$$

$$w_{ij} = \frac{\mathcal{U}(i) \times \mathcal{U}(j)}{|\mathcal{U}(i)|^{\alpha}|\mathcal{U}(j)|^{1-\alpha}}$$

allowing to tune the parameter α from 0.0 to 1.0 to optimize the performance of the overall algorithm.

Feature Construction

There are many ways to build the vectors u and i from the listening history triplets. In this case study, I chose to evaluate the following feature construction methods:

- binary construction method:
  - $u_i = 1$ if user u listened to song i, and 0 otherwise
  - $i_u = 1$ if song i was listened user u, and 0 otherwise
- frequency construction method:
  - $u_i =$ the number of time user u listened to song i
  - $i_u =$ the number of time song i was listened by user u
- normed frequency construction method:
  - $u_i =$ the euclidean norm of the frequency vector
  - $i_u =$ the euclidean norm of the frequency vector
- TF-IDF construction method, making an analogy between the song listening history and a corpus of text documents, we apply the Term Frequency, Inverse Document Frequency [3] to compute the feature vectors:
  - $u_i = tf(u,i) \times idf(i)$, normalized with the euclidean norm where $tf(u,i)$ is the number of times user u listened to song i, and $idf(i) = log(\frac{m}{|\mathcal{U}(i)|})$ with m as the number of songs in the Dataset. This has the impact of decreasing the weight of a song that is listened by many users, in an attempt to minimize the popularity bias that Collaborative Filtering methods usualy suffer from.
  - $i_u = tf(i,u) \times idf(u)$, normalized with the euclidean norm where $tf(i,u)$ is the number users u listening to song i, and $idf(u) = log(\frac{n}{|\mathcal{I}(u)|})$ with n as the number of users in the Dataset, decreasing the weight of a user listening to many different songs.

Hybrid methods

After evaluationg the performances of the User Based and Item Based CF algorithms alone, I will also evaluate the performances of their combinations using 3 different methods.

Ranking Aggregation

Using 2 different recommendation algorithms, we recommend, for each rank in the final aggregated recommendation:

- the next recommendation from the first algorithm with a probability θ,
- or the next recommendation from the second recommendation algorithm with a probability $1 - \theta$.

If the next song recommended by one of those algorithm is already in the final ranking, we of course don't add it a second time.

Feature Combination

This method requires to get, from each algorithm, not only the rank, but also the score associated with each recommendation.

I use a linear combination of the scores received from the 2 different algorithms, in the form:

$$s_{Combined} = \theta s_{Alg1} + (1 - \theta) s_{Alg2}$$

This will of course only work if the 2 algorithms have comparable scoring methods.

Algorithm Switching

This method requires to define an "oracle" helping to choose, for each user, the full recommendation from the first or the second algorithm.

The oracle chosen in this case study is simply the length of the user's listening history, but other could be investiguated, like the number of nearest neighbors for a particular user, or its sensibility to popular songs, etc.

A variant of this method could use the oracle to tune the parameter theta in the Ranking Aggregation or the Feature Combination algorithm.

Performance evaluation

There are many performance measures available to evaluate recommender systems.

The Kaggle MSD Challenge Competition uses the Truncated Mean Average Precision (MAP) to evaluate recommendation submissions. If the Precision at rank k for user u is denoted by:

$$P_{uk} = \frac{1}{k} \sum_{l=1}^{k} R_{ul}$$

$$\text{where: } R_{ul} = \begin{cases} 1 & \text{if song at rank } l \text{ is a relevant song} \\ 0 & \text{otherwise} \end{cases}$$

The Average Precision for user u, truncated at rank K, is then:

$$AP(K)_u = \frac{1}{\tau_u} \sum_{k=1}^{K} P_{uk} R_{uk}$$

$$\text{where: } \tau_u = \min\left(K, R_{uK}\right)$$

It is the average of Precision's at each rank where a relevant document was found.

The Mean Average Precision is then the average of AP's for all users:

$$MAP(K) = \frac{1}{|\mathcal{U}|} \sum_{u \in \mathcal{U}} AP(K)_u$$

Datasets

To evaluate the performance of the algorithms, we need a training set to generate a recommendation, and a validation set to evaluate it. The MSD Challenge competition provided a training set of 110.000 users and half of their listening history, the second half of the listening history being hold for performance evaluation through an API provided on Kaggle. Since the competition is now closed and the API is not available anymore, I created 3 new Datasets:

- **Dataset_1** is based on the original training set of 110.000 users provided by the competition, each user having a history of 5 to 53 songs. It is further divided into a training set and a validation set by splitting the listening history of each user in 2, giving a training set with a listening history between 2 and 26 songs, and a validation set between 3 and 27 songs.

- **Dataset_2** was build from the full listening history of 1M users. As an attempt to keep a similar distribution as the competition training set, I randomly extracted 110.000 users with a listening history between 10 and 106 songs, giving a training set and validation set both with a listening history per user between 5 and 53 songs.

- **Dataset_3** was also build from the full listening history of 1M users, but released from the constraint on the user history length, in an attempt to see how history length influences the algorithms. However, there are no users listening to less than 10 songs in the full listening history. Dataset_3 contains 50.000 users with a listening history between 10 and 4400 songs.

Results

Collaborative Filtering

Dataset_1

Using the Dataset_1, I evaluated the performances of User Based and Item Based Collaborative Filtering algorithms, using a cosine similarity measure on binary vectors, and varying the parameter α and q to get the best performance for each algorithm.

The synthetic results are shown in Table 1, against a baseline algorithm recommending to each user the most popular songs in deacreasing order of popularity, if the song is not in the training set.

The details of the performances for each pair of parameter (α, q) can be found in annexe to this report.

MAP	Feature Construction	
Algorithm	N/A	binary
Popularity	0.0248	
User Based CF		**0.0789●**
Item Based CF		0.0698

Table 1: Synthetic results for Dataset_1

We can see here that the two algorithms perform better than the baseline populaity algorithm. The User Based method is clearly the winner.

However, those numbers are far from the ones seen in the leaderboard of the competition, which lead me to create the Dataset_2 to get more comparable results.

Dataset_2

Using Dataset_2, I evaluated again the performances of User Based and Item Based Collaborative Filtering algorithms, always with the cosine similarity measure, but trying different feature construction methods.

Again, I compared it against the baseline popularity algorithm. The best performance are summarized in Table 2, and the details of the tuning of α and q are again available in the annexe.

MAP	Feature Construction				
Algorithm	N/A	binary	count	norm	TF-IDF
Popularity	0.3920				
User Based CF		**0.1941●**	0.1462	0.1447	0.1447
Item Based CF		0.1813	0.0830	0.0642	0.0659

Table 2: Synthetic results for Dataset_2

We can see here that:

- the Collaborative Filtering techniques all outperform the baseline popularity algorithm
- the binary feature construction gives better performances than the 3 others in general, which is particularly true for the Item Based CF
- The User Based CF always gives better results than the Item Based CF

Dataset_3

I created Dataset_3 to evaluate the performances of the algorithms on a real life case. The Million Song Dataset full listening history, however, does not provide any listening history smaller than 10 users. A better dataset would start at 1 song listened to really evaluate the impact of the cold start problem.

The results for Dataset_3, shown in Table 3, are slightly lower than those of Dataset_2, either because of the longer listening history, or the smaller number of users (and then of neighbors). However, they still show a consistent beter results for User Based CF over Item Based CF, both outperforming the baseline Popularity algorithm.

MAP	Feature Construction	
Algorithm	N/A	binary
Popularity	0.0380	
User Based CF		**0.1855●**
Item Based CF		0.1671

Table 3: Synthetic results for Dataset_3

Hybrid Methods

After evaluating the performances of single algorithms, I run 3 different hybrid methods on the ones giving the best performance:

- The User Based CF with a binary feature construction and cosine similarity with $\alpha = 0.30$ and $q = 1$
- The Item Based CF with a binary feature construction and cosine similarity with $\alpha = 0.80$ and $q = 1$

Stochastic Aggregation

As shown in Table 4, the best performances for the Stochastic Aggregation were reached with parameter $\theta = 0.4$, meaning that we have, in average, 40% of the recommendations made by the User Based CF, and 60% from the Item Based CF.

θ	MAP
0.1	0.1568
0.2	0.1645
0.3	0.1683
0.4	**0.1692●**
0.5	0.1681
0.6	0.1650
0.7	0.1602
0.8	0.1534
0.9	0.1455

Table 4: MAP results for the Stochastic Aggregation algorithm by varying the parameterθ

This result might seem strange as we saw earlier that the User Based CF performed better than the Item Based CF in general, but might be explained if we look closer at the results.

In Figure 1, the Mean Average Precision of the User Based CF, Item Based CF and Popularity algorithms are plotted by length of the user's listening history. The number of users by listening history is also plotted on a different scale. Here we can see that:

- The performances of the popularity algorithm are clearly below the performances of the Collaborative Filtering algorithms, and are fairly constant for any length of user listening history (only a slight increase).
- The performances of the CF algorithms, on the other hand, both increase as the user listening history gets bigger (weither this is totally due to the algorithm, or partially due to the performance evaluation methodology has not been studied here).
- The vast majority of users have a small listening history. This combined to the low performances of the algorithms in this region account for a general performance below 0.20, while we can nearly reach 0.30 with a bigger listening history.
- Although the User Based CF performs better than the Item Based CF in general, for small listening history the Item Based performs better.

This fourth observation explains why the performances of the Stochastic Aggregation method are maximum while choosing more results from the Item Based CF than from the User Based CF, as most users have a small listening history and the User Based CF performs better for those.

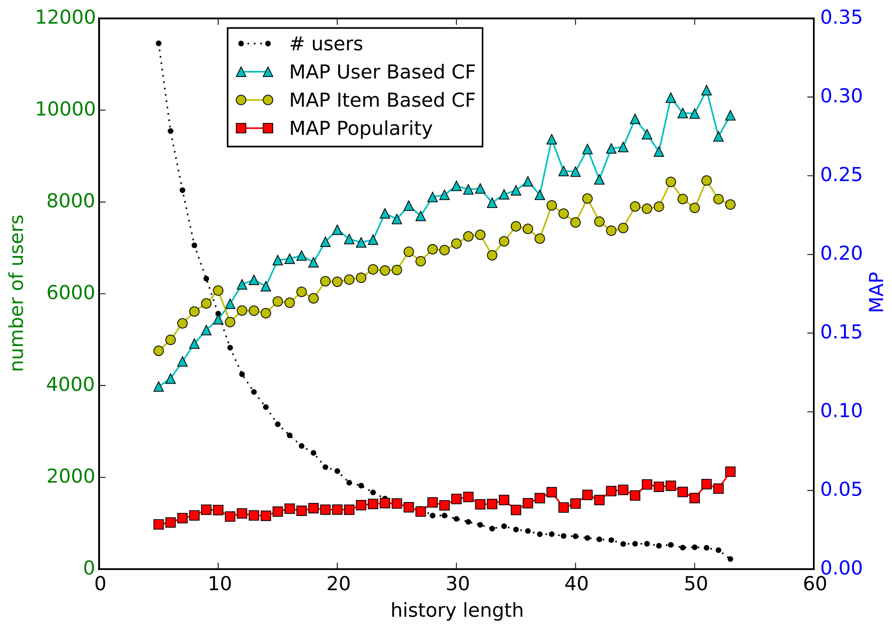

Figure 1: User Based & Item Based CF vs Popularity

Figure 2 shows the performances of the Stochastic Aggregation method against the User Based and Item Based CF. We can see here that the Stochastic Aggregation method performs slightly better than the User Based CF for a small listening history.

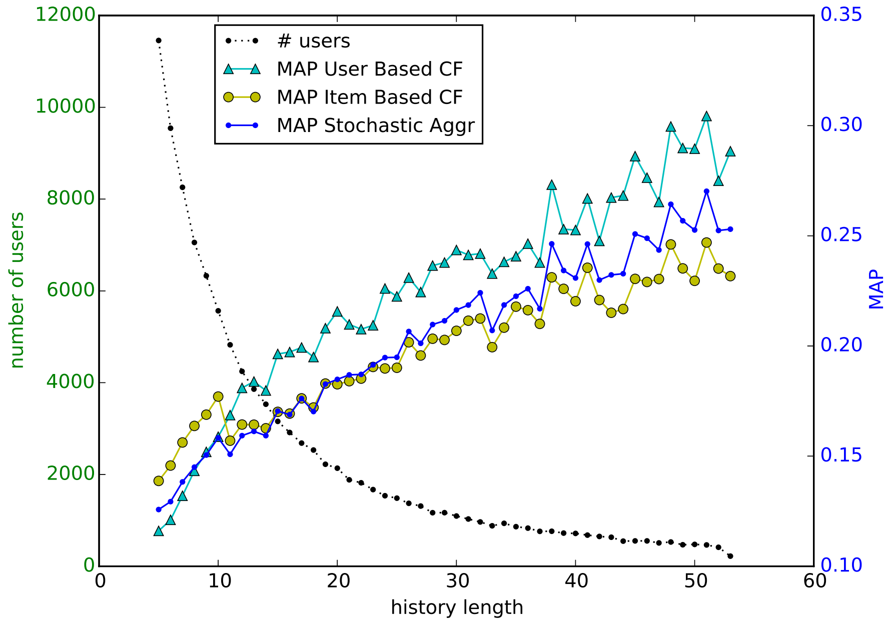

Figure 2: User Based & Item Based CF vs Stochastic Aggregation

The reason of the poor performances of the Stochastic Aggregation method for our dataset is most probably due to the fact that interleaving the ranking of 2 different recommenders pushes the relevant documents lower in the final ranking, as suggests a quick look in the detailed results.

Those lower performances are in contradiction with the restuls in [2], maybe because of the fact that the Item Based CF algorithm in this case study has not been callibrated, which might be the reason the Item Based CF starts to give lower performances than the User Based CF with increasing listening history.

Feature Combination

Varying θ for the Feature Combination method again performs better while giving more importance to the Item Based CF algorithm.

As shown in Table 5, the best performances are reached with $\theta = 0.3$, meaning the score given by the User Based CF accounts for only 30% of the overall score, while the Item Based CF score accounts for 70%.

The Feature Combination performs better than the Stochastic Aggregation. The reason, as we can see in Figure 3, is that it gives more emphasis on the Item Based CF algorithm, which gives better results for a large number of userts.

θ	MAP
0.0	0.1782
0.1	0.1836
0.2	0.1864
0.3	**0.1872●**
0.4	0.1861
0.5	0.1831
0.6	0.1782
0.7	0.1715
0.8	0.1675
0.9	0.1552
1.0	0.1545

Table 5: MAP results for the Stochastic Aggregation algorithm by varying the parameterθ

Algorithm Switching

The last Hybrid Method tested is the Algorithm Switching.

Given the particular drop of performances of the User Based CF for users listening to more than 10 songs, using the user listening history as oracle to decide which ranking to choose from User Based and Item Based CF was an obvious choice.

As shown in Figure 4, The Switching algorithm acheives the best performances of $MAP = 0.2011$, simply by always choosing the best of the 2 algorithms with a very simple rule.

This is however very particular to this dataset and the behavior of the User Based algorithm, and we should keep in mind not to overfit our model. For less obvious situations, it might be interesting to investiguate other oracles.

Results summary

Table 6 summarizes all the results of this case study. We can clearly see here the consistency of the results accross 2 difference datasets.

MAP	Dataset	
Algorithm	Dataset_2	Dataset_3
Popularity	0.0392	0.0380
Stochastic Aggregation	*0.1692*	*0.1604*
Item Based CF	0.1813	0.1671
Feature Combination	*0.1861*	*0.1797*
User Based CF	0.1941	0.1855
Switching	***0.2011●***	***0.1880●***

Table 6: Results summary

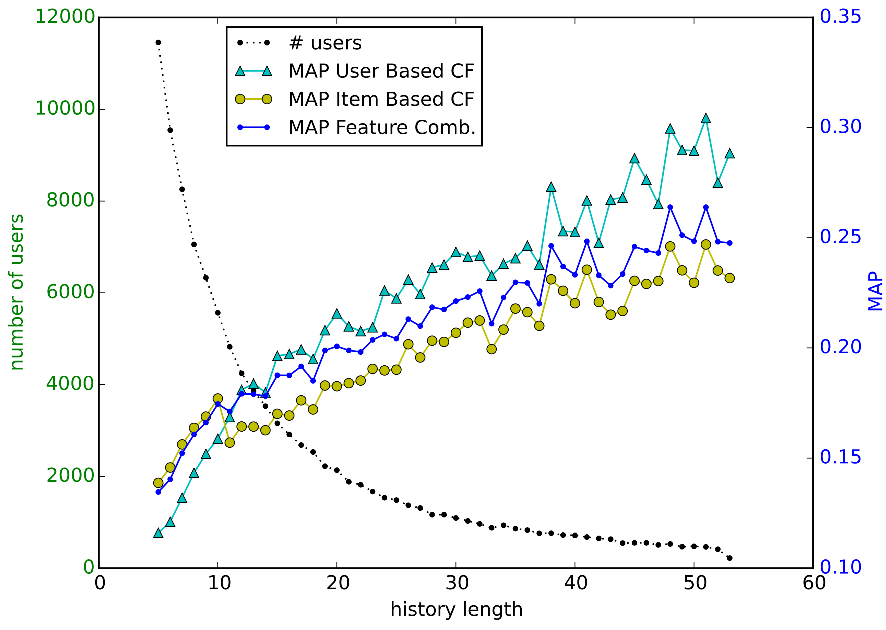

Figure 3: User Based & Item Based CF vs Feature Combination

Conclusion & Future Work

In this case study, I evaluated the performances of different Collaborative Filtering algorithms, mainly following the work of [2].

The results shown here are not fully consistent with those presented in the paper, which I attribute mainly to the fact that I didn't use a callibrated Item Based CF algorithm.

I also tried different feature construction methods, without showing any improvement, and 2 different aggregation methods that proved better than the Ranking Aggregation method (although in the case of the Algorithm Swithing method, this might be due, again, to the use of a non callibrated Item Based CF).

If required to go further, it would be interesting to study in more details the reasons of the poor performances in some situations, knowing that the poor performances are partly due to zero correct recommendation for a large amount of users, especially for those having a short listening history. Understanding the relations between those users and users listening to the same items might help understand if the problem lies in the similarity measure, or if it would definitely be impossible for any algorithm (or human), to give a valuable recommendation.

Although the different feature construction methods evaluated here didn't deliver good performances, analysing the data in more detail could provide insight on how to use the information provided by the number of time a user listened to a particular song, in order to chose more apropriate feature construction method, similarity measures, or scoring function to improve the performances

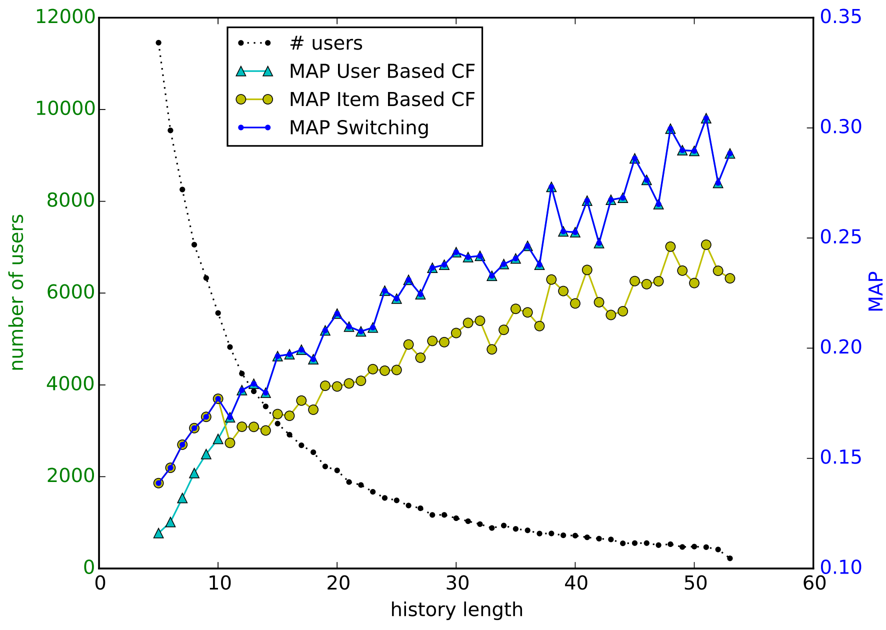

Figure 4: User Based & Item Based CF vs Switching

of Collaborative Filtering algorithms.

Also, studying other hybrid methods such as the Cascade method, maybe in combination with a Content Based model, might help improve performances for user with small listening histories.

Another path to explore would be to try the Personality Diagnosis algorithm, which seems to be a promissing Collaborative Filtering variant.

References

[1] *Xiaoyuan Su and Taghi M. Khoshgoftaar.* A survey of collaborative filtering techniques. *Advances in Artificial Intelligence, January 2009.*

[2] *Fabio Aiolli.* A Preliminary Study on a Recommender System for the Million Songs Dataset Challenge

[3] *Juan Ramos.* Using TF-IDF to Determine Word Relevance in Document Queries

ANNEXES

Dataset_1

α	**q=1**	**q=2**	**q=3**	**q=4**	**q=5**
0.00	0.0705	0.0618	0.0570	0.0552	0.0547
0.05	0.0720	0.0630	0.0581	0.0561	0.0553
0.10	0.0725	0.0638	0.0588	0.0562	0.0552
0.15	0.0732	0.0651	0.0591	0.0565	0.0553
0.20	0.0738	0.0658	0.0595	0.0569	0.0556
0.25	0.0748	0.0671	0.0610	0.0577	0.0557
0.30	0.0755	0.0681	0.0618	0.0577	0.0559
0.35	0.0759	0.0689	0.0618	0.0584	0.0560
0.40	0.0769	0.0705	0.0630	0.0592	0.0564
0.45	0.0778	0.0719	0.0653	0.0598	0.0568
0.50	0.0783	0.0726	0.0659	0.0602	0.0570
0.55	0.0786	0.0735	0.0665	0.0606	0.0576
0.60	**0.0789●**	0.0740	0.0673	0.0618	0.0576
0.65	0.0789	0.0742	0.0676	0.0622	0.0576
0.70	0.0784	0.0743	0.0680	0.0625	0.0584
0.75	0.0782	0.0741	0.0682	0.0632	0.0594
0.80	0.0781	0.0736	0.0685	0.0643	0.0608
0.85	0.0780	0.0732	0.0687	0.0654	0.0626
0.90	0.0780	0.0727	0.0689	0.0662	0.0642
0.95	0.0780	0.0721	0.0692	0.0667	0.0649
1.00	0.0730	0.0684	0.0656	0.0643	0.0638

Table 7: Dataset_1, User Based CF, distance=cosine, binary, max neighbours = 50, MAP for different values of α and q

α	**q=1**	**q=2**	**q=3**	**q=4**	**q=5**
0.00	0.0056	0.0052	0.0051	0.0051	0.0051
0.05	0.0088	0.0077	0.0077	0.0078	0.0080
0.10	0.0090	0.0080	0.0082	0.0082	0.0083
0.15	0.0095	0.0086	0.0086	0.0085	0.0085
0.20	0.0103	0.0091	0.0090	0.0090	0.0089
0.25	0.0114	0.0099	0.0097	0.0096	0.0096
0.30	0.0131	0.0110	0.0107	0.0106	0.0106
0.35	0.0158	0.0133	0.0128	0.0127	0.0127
0.40	0.0200	0.0166	0.0159	0.0158	0.0157
0.45	0.0252	0.0208	0.0198	0.0194	0.0192
0.50	0.0308	0.0252	0.0240	0.0235	0.0233
0.55	0.0368	0.0301	0.0286	0.0280	0.0278
0.60	0.0426	0.0352	0.0334	0.0327	0.0324
0.65	0.0484	0.0403	0.0382	0.0375	0.0371
0.70	0.0534	0.0452	0.0429	0.0421	0.0416
0.75	0.0581	0.0497	0.0474	0.0465	0.0461
0.80	0.0622	0.0539	0.0514	0.0504	0.0499
0.85	0.0653	0.0571	0.0546	0.0536	0.0531
0.90	0.0679	0.0599	0.0574	0.0562	0.0556
0.95	**0.0698•**	0.0622	0.0596	0.0584	0.0578
1.00	0.0693	0.0644	0.0630	0.0625	0.0622

Table 8: Dataset_1, Item Based CF, distance=cosine, binary, max neighbours = 50, MAP for different values of α and q

Dataset_2

α	**q=1**	**q=2**	**q=3**	**q=4**	**q=5**
0.00	0.1908	0.1827	0.1703	0.1602	0.1533
0.05	0.1921	0.1839	0.1714	0.1615	0.1539
0.10	0.1926	0.1849	0.1726	0.1625	0.1546
0.15	0.1933	0.1857	0.1732	0.1626	0.1547
0.20	0.1938	0.1866	0.1742	0.1634	0.1551
0.25	0.1940	0.1873	0.1749	0.1639	0.1554
0.30	**0.1941●**	0.1874	0.1754	0.1640	0.1553
0.35	0.1939	0.1874	0.1754	0.1642	0.1551
0.40	0.1937	0.1876	0.1759	0.1645	0.1551
0.45	0.1929	0.1872	0.1758	0.1644	0.1549
0.50	0.1921	0.1868	0.1757	0.1642	0.1546
0.55	0.1916	0.1865	0.1756	0.1640	0.1542
0.60	0.1906	0.1859	0.1754	0.1638	0.1539
0.65	0.1897	0.1852	0.1750	0.1638	0.1537
0.70	0.1889	0.1845	0.1746	0.1636	0.1536
0.75	0.1876	0.1834	0.1736	0.1630	0.1531
0.80	0.1861	0.1817	0.1723	0.1620	0.1524
0.85	0.1849	0.1802	0.1710	0.1609	0.1516
0.90	0.1843	0.1791	0.1696	0.1599	0.1511
0.95	0.1841	0.1777	0.1683	0.1588	0.1505
1.00	0.1776	0.1742	0.1658	0.1571	0.1502

Table 9: Dataset_2, User Based CF, distance=cosine, binary, max neighbours = 50, MAP for different values of α and q

α	**q=1**	**q=2**
0.00	0.1359	0.1126
0.05	0.1375	0.1140
0.10	0.1389	0.1153
0.15	0.1402	0.1167
0.20	0.1415	0.1183
0.25	0.1429	0.1202
0.30	0.1444	0.1227
0.35	0.1454	0.1245
0.40	0.1461	0.1262
0.45	**0.1462●**	0.1266
0.50	0.1447	0.1246
0.55	0.1438	0.1216
0.60	0.1425	0.1171
0.65	0.1398	0.1110
0.70	0.1367	0.1041
0.75	0.1325	0.0956
0.80	0.1282	0.0880
0.85	0.1235	0.0816
0.90	0.1186	0.0764
0.95	0.1130	0.0718
1.00	0.1074	0.0677

Table 10: Dataset_2, User Based CF, distance=cosine, count, max neighbours = 50, MAP for different values of α and q

[

α	**q=1**	**q=2**
0.00	0.1447	0.1246
0.05	0.1447	0.1246
0.10	0.1447	0.1246
0.15	0.1447	0.1246
0.20	0.1447	0.1246
0.25	0.1447	0.1246
0.30	0.1447	0.1246
0.35	0.1447	0.1246
0.40	0.1447	0.1246
0.45	0.1447	0.1246
0.50	**0.1447**●	0.1246
0.55	0.1447	0.1246
0.60	0.1447	0.1246
0.65	0.1447	0.1246
0.70	0.1447	0.1246
0.75	0.1447	0.1246
0.80	0.1447	0.1246
0.85	0.1447	0.1246
0.90	0.1447	0.1246
0.95	0.1447	0.1246
1.00	0.1447	0.1246

Table 11: Dataset_2, User Based CF, distance=cosine, norm, max neighbours = 50, MAP for different values of α and q

α	**q=1**	**q=2**
0.00	0.1395	0.1188
0.05	0.1404	0.1194
0.10	0.1410	0.1197
0.15	0.1417	0.1199
0.20	0.1424	0.1200
0.25	0.1428	0.1201
0.30	0.1433	0.1201
0.35	0.1438	0.1202
0.40	0.1441	0.1199
0.45	0.1444	0.1197
0.50	0.1444	0.1193
0.55	0.1445	0.1190
0.60	0.1447	0.1187
0.65	**0.1447•**	0.1183
0.70	0.1446	0.1178
0.75	0.1446	0.1173
0.80	0.1444	0.1167
0.85	0.1442	0.1162
0.90	0.1440	0.1157
0.95	0.1438	0.1151
1.00	0.1435	0.1144

Table 12: Dataset_2, User Based CF, distance=cosine, tfidf, max neighbours = 50, MAP for different values of α and q

α	**q=1**	**q=2**	**q=3**	**q=4**	**q=5**
0.00	0.0123	0.0097	0.0094	0.0094	0.0094
0.05	0.0179	0.0140	0.0143	0.0153	0.0163
0.10	0.0197	0.0164	0.0173	0.0185	0.0195
0.15	0.0237	0.0197	0.0200	0.0203	0.0206
0.20	0.0306	0.0250	0.0238	0.0235	0.0235
0.25	0.0467	0.0360	0.0323	0.0307	0.0301
0.30	0.0691	0.0509	0.0438	0.0410	0.0395
0.35	0.0899	0.0661	0.0562	0.0519	0.0497
0.40	0.1102	0.0816	0.0695	0.0641	0.0614
0.45	0.1276	0.0959	0.0822	0.0759	0.0727
0.50	0.1424	0.1090	0.0942	0.0874	0.0838
0.55	0.1542	0.1204	0.1049	0.0977	0.0939
0.60	0.1636	0.1300	0.1142	0.1068	0.1030
0.65	0.1710	0.1382	0.1221	0.1147	0.1108
0.70	0.1763	0.1445	0.1286	0.1211	0.1173
0.75	0.1797	0.1495	0.1340	0.1265	0.1225
0.80	**0.1813●**	0.1532	0.1379	0.1305	0.1267
0.85	0.1808	0.1558	0.1409	0.1337	0.1299
0.90	0.1781	0.1573	0.1431	0.1361	0.1323
0.95	0.1736	0.1576	0.1442	0.1375	0.1339
1.00	0.1624	0.1547	0.1445	0.1393	0.1366

Table 13: Dataset_2, Item Based CF, distance=cosine, binary, max neighbours = 50, MAP for different values of α and q

α	**q=1**	**q=2**	**q=3**	**q=4**	**q=5**
0.00	0.0038	0.0031	0.0030	0.0030	0.0029
0.05	0.0044	0.0035	0.0033	0.0033	0.0033
0.10	0.0053	0.0041	0.0039	0.0038	0.0038
0.15	0.0065	0.0048	0.0045	0.0044	0.0044
0.20	0.0085	0.0060	0.0056	0.0054	0.0053
0.25	0.0115	0.0078	0.0072	0.0069	0.0068
0.30	0.0162	0.0105	0.0095	0.0091	0.0089
0.35	0.0233	0.0146	0.0130	0.0124	0.0121
0.40	0.0336	0.0205	0.0179	0.0170	0.0165
0.45	0.0476	0.0289	0.0248	0.0234	0.0227
0.50	0.0642	0.0409	0.0354	0.0333	0.0323
0.55	0.0750	0.0492	0.0431	0.0409	0.0398
0.60	0.0798	0.0522	0.0458	0.0436	0.0425
0.65	0.0823	0.0536	0.0472	0.0450	0.0439
0.70	**0.0830●**	0.0541	0.0480	0.0459	0.0448
0.75	0.0823	0.0541	0.0484	0.0462	0.0453
0.80	0.0808	0.0538	0.0482	0.0463	0.0454
0.85	0.0784	0.0531	0.0480	0.0462	0.0453
0.90	0.0757	0.0523	0.0475	0.0459	0.0451
0.95	0.0727	0.0514	0.0471	0.0456	0.0449
1.00	0.0695	0.0504	0.0466	0.0453	0.0447

Table 14: Dataset_2, Item Based CF, distance=cosine, count, max neighbours = 50, MAP for different values of α and q

α	**q=1**	**q=2**	**q=3**	**q=4**	**q=5**
0.00	0.0642	0.0410	0.0354	0.0333	0.0323
0.05	0.0642	0.0410	0.0354	0.0333	0.0323
0.10	0.0642	0.0410	0.0354	0.0333	0.0323
0.15	0.0642	0.0410	0.0354	0.0333	0.0323
0.20	0.0642	0.0410	0.0354	0.0333	0.0323
0.25	0.0642	0.0409	0.0354	0.0333	0.0323
0.30	0.0642	0.0409	0.0354	0.0333	0.0323
0.35	0.0642	0.0409	0.0354	0.0333	0.0323
0.40	0.0642	0.0409	0.0354	0.0333	0.0323
0.45	0.0642	0.0409	0.0354	0.0333	0.0323
0.50	**0.0642●**	0.0410	0.0354	0.0333	0.0323
0.55	0.0642	0.0409	0.0354	0.0333	0.0323
0.60	0.0642	0.0409	0.0354	0.0333	0.0323
0.65	0.0642	0.0409	0.0354	0.0333	0.0323
0.70	0.0642	0.0409	0.0354	0.0333	0.0323
0.75	0.0642	0.0409	0.0354	0.0333	0.0323
0.80	0.0642	0.0409	0.0354	0.0333	0.0323
0.85	0.0642	0.0409	0.0354	0.0333	0.0323
0.90	0.0642	0.0409	0.0354	0.0333	0.0323
0.95	0.0642	0.0409	0.0354	0.0333	0.0323
1.00	0.0642	0.0409	0.0354	0.0333	0.0323

Table 15: Dataset_2, Item Based CF, distance=cosine, norm, max neighbours = 50, MAP for different values of α and q

α	**q=1**	**q=2**	**q=3**	**q=4**	**q=5**
0.00	0.0638	0.0408	0.0354	0.0334	0.0324
0.05	0.0640	0.0408	0.0355	0.0335	0.0325
0.10	0.0640	0.0409	0.0355	0.0335	0.0325
0.15	0.0643	0.0410	0.0356	0.0336	0.0326
0.20	0.0644	0.0410	0.0357	0.0337	0.0327
0.25	0.0645	0.0411	0.0357	0.0337	0.0328
0.30	0.0645	0.0412	0.0358	0.0338	0.0328
0.35	0.0647	0.0413	0.0359	0.0339	0.0329
0.40	0.0648	0.0414	0.0359	0.0340	0.0330
0.45	0.0648	0.0414	0.0360	0.0340	0.0330
0.50	0.0649	0.0415	0.0360	0.0340	0.0330
0.55	0.0651	0.0416	0.0361	0.0341	0.0331
0.60	0.0651	0.0416	0.0361	0.0341	0.0331
0.65	0.0652	0.0417	0.0362	0.0342	0.0332
0.70	0.0654	0.0418	0.0362	0.0343	0.0333
0.75	0.0654	0.0418	0.0363	0.0343	0.0333
0.80	0.0656	0.0419	0.0364	0.0344	0.0334
0.85	0.0657	0.0420	0.0364	0.0344	0.0335
0.90	0.0657	0.0420	0.0365	0.0345	0.0335
0.95	0.0658	0.0420	0.0366	0.0346	0.0335
1.00	**0.0659●**	0.0421	0.0366	0.0346	0.0336

Table 16: Dataset_2, Item Based CF, distance=cosine, tfidf, max neighbours = 50, MAP for different values of α and q

Dataset_3

α	**q=1**	**q=2**
0.00	0.1819	0.1706
0.05	0.1828	0.1718
0.10	0.1839	0.1731
0.15	0.1846	0.1739
0.20	0.1852	0.1748
0.25	**0.1855●**	0.1754
0.30	0.1855	0.1760
0.35	0.1854	0.1762
0.40	0.1851	0.1765
0.45	0.1847	0.1768
0.50	0.1841	0.1767
0.55	0.1829	0.1762
0.60	0.1816	0.1757
0.65	0.1796	0.1745
0.70	0.1773	0.1730
0.75	0.1738	0.1705
0.80	0.1702	0.1676
0.85	0.1662	0.1642
0.90	0.1627	0.1607
0.95	0.1611	0.1579
1.00	0.1485	0.1495

Table 17: Dataset_3, User Based CF, distance=cosine, binary, max neighbours = 50, MAP for different values of α and q

α	**q=1**	**q=2**
0.00	0.0091	0.0084
0.05	0.0138	0.0124
0.10	0.0145	0.0144
0.15	0.0165	0.0174
0.20	0.0208	0.0214
0.25	0.0303	0.0286
0.30	0.0474	0.0399
0.35	0.0674	0.0529
0.40	0.0875	0.0670
0.45	0.1042	0.0797
0.50	0.1195	0.0921
0.55	0.1326	0.1034
0.60	0.1437	0.1137
0.65	0.1525	0.1225
0.70	0.1591	0.1300
0.75	0.1635	0.1362
0.80	0.1664	0.1416
0.85	**0.1671●**	0.1456
0.90	0.1643	0.1481
0.95	0.1596	0.1488
1.00	0.1486	0.1448

Table 18: Dataset_3, Item Based CF, distance=cosine, binary, max neighbours = 50, MAP for different values of α and q