Efficient Top-N Recommendation for Very Large Scale Binary Rated Datasets

Fabio Aiolli
University of Padova, Italy
aiolli@math.unipd.it

ABSTRACT

We present a simple and scalable algorithm for top-N recommendation able to deal with very large datasets and (binary rated) implicit feedback. We focus on memory-based collaborative filtering algorithms similar to the well known neighboor based technique for explicit feedback. The major difference, that makes the algorithm particularly scalable, is that it uses positive feedback only and no explicit computation of the complete (user-by-user or item-by-item) similarity matrix needs to be performed.

The study of the proposed algorithm has been conducted on data from the Million Songs Dataset (MSD) challenge whose task was to suggest a set of songs (out of more than 380k available songs) to more than 100k users given half of the user listening history and complete listening history of other 1 million people.

In particular, we investigate on the entire recommendation pipeline, starting from the definition of suitable similarity and scoring functions and suggestions on how to aggregate multiple ranking strategies to define the overall recommendation. The technique we are proposing extends and improves the one that already won the MSD challenge last year.

Categories and Subject Descriptors

H.4 [**Information Systems Applications**]: Miscellaneous; H.3.3 [**Information Storage and Retrieval**]: Information Search and Retrieval

General Terms

Algorithms

Keywords

Collaborative Filtering, Top-N Recommendation, Implicit Feedback, Million Song Dataset Challenge

1. INTRODUCTION

Recommender systems are common tools to improve customer experience in e-commerce applications. These systems typically use the history of their interaction with the users to improve future recommendations. Different kinds of interactions can be tracked. For example, users can be required to leave an *explicit feedback* (a vote, a rate, or any degree of satisfaction) for the available items. In contrast, the system can autonomously keep track of user behaviors, for instance by keeping the history of user purchases, browsing activity of the users, etc. This second case is usually referred to as *implicit feedback*. Web Retrieval literature is plenty of works dealing with the implicit feedback problem while less effort has been devoted so far to recommendation settings (see [7, 15, 11]).

The Million Song Dataset Challenge [10] was a large scale, music recommendation challenge, where the task is to predict which songs a user will listen to, provided the listening history of a user. The challenge was based on the Million Song Dataset (MSD), a freely-available collection of meta data for one million of contemporary songs (e.g. song titles, artists, year of publication, and much more) [4]. About one hundred and fifty teams participated to the challenge. The subset of data actually used in the challenge was the so called Taste Profile Subset that consists of more than 48 million triplets *(user,song,count)* gathered from user listening histories. Data consists of about 1.2 million users and covers more than 380,000 songs in MSD. The user-item matrix is very sparse as the fraction of non-zero entries (a.k.a. density) is only 0.01%.

The task of the challenge was to recommend the most appropriate songs for a user given half of her listening history and the complete history of another 1 million users. Thus, the challenge focused on the ordering of the songs on the basis of the relevance for a given user (top-N recommendation), and this makes the particular problem different from the more classical problem of predicting rates a user will give to unseen items [6, 13]. For example, popular tasks like Netflix [3] and Movielens fall in this last case. A second important characteristic of the MSD problem is that we do not have explicit or direct feedback about what users like and how much they like it (implicit feedback). In fact, we only have information of the form "user u listened to song i" without any knowledge about whether user *u* actually liked song i or not. A third important aspect of the MSD data is the presence of meta data concerning songs including title, artist, year of publication, etc. An interesting open question then was whether this additional information could help or not. Finally, given the huge size of the datasets involved, time and memory efficiency of the method used turned out to be another very important issue in the challenge.

The most popular technique adopted in recommender systems is Collaborative Filtering (CF) where the matrix of rates users have previously assigned to the items is used to discover other users with similar behaviors as the active user (i.e. the user for which we want

RecSys'13, October 12–16, 2013, Hong Kong, China.

http://dx.doi.org/10.1145/2507157.2507189

to make the prediction). Current CF approaches can be grouped in the two classes of neighborhood and model-based methods.

The main intuition in neighborhood-based methods is that, if other users, similar to the active user, already purchased a certain item, then it is likely that the active user will like that item as well. Similarly, knowing that a set of items are often purchased together (they are similar in some sense), then, if the active user has bought one of them, probably he/she will be interested to the other too. Both views have been effectively adopted in recent literature for explicit feedback settings.

In this paper, we demonstrate that the choice of the right prediction technique or similarity measure is not all that we need to obtain a good recommender. Here, we propose a flexible pipeline of steps each one with its own (small) set of parameters to tune for the specific domain. Moreover, we show that, although the item-based view has turned out more useful to win the MSD competition, the user-based view also brings useful and diverse information that can be aggregated to boost the performance of the recommendation.

In Section 2, collaborative filtering is described and proposed as a first approach to solve the problem of MSD. In particular, we briefly discuss the most popular state-of-the-art techniques: model based and memory based CF methods. In Section 3, we propose the *asymmetric cosine* similarity, a parameterized similarity function that can be adapted to different applicative domains. Furthermore, the notion of locality is taken in account. In Section 4 a new memory based CF approach is presented which is particularly suitable to tasks with implicit feedback. Finally, in Section 5, empirical results of the proposed techniques are presented and discussed.

2. CF AND THE MILLION SONG DATASET

Collaborative Filtering (CF) techniques use a database in the form of a user-item matrix R of preferences. A typical CF setting consists of a set $\mathcal{U}$ of n users, a set $\mathcal{I}$ of m items, and a user-item matrix $R = \{r_{ui}\} \in \mathbb{R}^{n \times m}$ represents how much user u likes item i. In this paper, we assume binary ratings $r_{ui} \in \{0, 1\}$ as this was the setting of the MSD challenge[1]. In the MSD setting, an entry $r_{ui} = 1$ represents the fact that user u have listened to the song i. In the following, we refer to terms item or song interchangeably. The MSD challenge task can be properly described as a *top-N recommendation* task, that is, for any *active user* u, we want to identify a list of N ($N = 500$ in the challenge) items $I_u \subseteq \mathcal{I}$ she/he will like the most. Clearly, this set must be disjoint with the set of items already rated (purchased, or listened to) by the active user.

2.1 Model-based Collaborative Filtering

Model-based CF techniques construct a model of the information contained in the matrix R. There are many proposed techniques of this type, including Bayesian models, Clustering models, Latent Factor models, and Classification/Regression models, just to name a few.

In more recent literature about CF, Matrix Factorization (MF) techniques [9] have become a very popular and effective choice to implement the CF idea. In this kind of models one tries to learn an embedding of both users and items into a smaller dimensional space. More formally, one needs to find two matrices $X \in \mathbb{R}^{k \times n}$ and $Y \in \mathbb{R}^{k \times m}$ such that $R \approx X^\top Y$, in such a way to minimize a loss over training data. A common choice for this loss is the root mean square error (RMSE).

Despite MF is recognized to be a state-of-the-art technique in CF, we note that it has some drawbacks that made it impractical and unsuccessful for the MSD task. First of all, training the corresponding model is computationally honerous and this fact is crucial when the size of the matrix R is very large as in the MSD case. Second, since MF is tipically modelled as a regression problem, then it seems unsuitable for implicit feedback tasks and some modifications to the original technique are needed [7]. In the MSD, for example, we have binary values of relevance and the value 0 cannot properly be considered as unrelevant since the no-action on an item can be due to many other reasons beyond not liking it (the user can be unaware of the existence of the song, for example). Third, MF techniques typically solve the associated minimization problem by using gradient descent algorithms which do not guarantee the convergence to a global minimum and, it is well known, the rate of convergence is quite low when near to local minima. Finally, matrix factorization based baselines provided by the organizers at the beginning of the challenge, and final results by other team entries, have confirmed quite poor results of MF based algorithms for this particular task, thus supporting our previous claims.

2.2 Memory-based Collaborative Filtering

In Memory-based Collaborative Filtering (MBCF) the entire user-item matrix is used to generate a prediction. In this case, there is not a real training of the model. Instead, a priori knowledge about typical behaviours of a user is exploited to some extent. Given a new user for which we want to obtain the prediction, the set of items to suggest are computed looking at "similar" users. This strategy is typically referred to as *user-based recommendation*. Alternatively, in the *item-based recommendation* strategy, one computes the most similar items for the items that have been already rated by the active user, and then prefer those items to form the final recommendation. There are many different proposal on how to combine the information provided by similar users/items (see [13] for a good survey). However, most of them are tailored to classical recommendation systems and they are not promptly compliant with the implicit feedback setting. More importantly, computing the neirest neighbors requires the computation of similarities for every pair of users or songs. This is simply infeasible in our domain given the huge size of the datasets involved.

In principle, the MBCF strategy can be used to do prediction for the implicit feedback setting as well and this can be done in the following very general ways.

In the *user-based* type of recommendation, the scoring function, on the basis of which the recommendation is made, is computed by

$$\hat{r}^U_{ui} = \sum_{v \in \mathcal{U}} w_{uv} r_{vi} = \sum_{v \in \mathcal{U}(i)} w_{uv}, \tag{1}$$

that is, the score obtained on an item for a target user is proportional to the similarities between the target user u and other users v that have rated before the item i ($v \in \mathcal{U}(i)$). This score will be higher for items which are often rated by similar users.

[1]Note that in this definition we are neglecting the information given by the *count* attribute of the triplets indicating how many times the song has been listened to by a user. However, the organizers of the challenge have warned us on the fact that this attribute is likely to be unreliable and absolutely not correlated with likings.

On the other hand, within a *item-based* type of recommendation [5, 12], the target item i is associated with a score

$$\hat{r}^I_{ui} = \sum_{j \in \mathcal{I}} w_{ij} r_{uj} = \sum_{j \in \mathcal{I}(u)} w_{ij}, \quad (2)$$

and hence, the score is proportional to the similarities between item i and other items already purchased by the user u ($j \in \mathcal{I}(u)$).

2.3 MF and MBCF unified

We end the section by noticing that, in both user-based and item-based MBCF, the user and item contributions are decomposed, that is, we can write

$$\hat{r}^U_{ui} = \mathbf{x}_u^\top \mathbf{y}_i, \text{ with } \mathbf{x}_u, \mathbf{y}_i \in \mathbb{R}^n,$$

and

$$\hat{r}^I_{ui} = \mathbf{x}_u^\top \mathbf{y}_i, \text{ with } \mathbf{x}_u, \mathbf{y}_i \in \mathbb{R}^m.$$

In other words, similarly to the MF case, we are implicitly defining an embedding for users and items. This embedding is performed onto an m-dimensional space in user based recommendation systems ($k = m$) or performed onto an n-dimensional space in item based recommendation systems ($k = m$).

3. SIMILARITIES FOR CF

The cosine similarity is undoubtely the most popular measure of correlation between two vectors. An important characteristic of this similarity measure is that it is symmetric and this feature can be important for certain applications. However, in our opinion, there are cases where the simmetry of the similarity measure does not match exactly the features of a domain. As we will see in the following, asymmetries are very common in CF applications. Now, we generalize the cosine similarity and suggest a nice probabilistic interpretation for binary valued vectors.

3.1 Asymmetric Cosine (asymC) Similarity

Consider three sets $\mathcal{A}, \mathcal{B}, \mathcal{C}$ such that $|\mathcal{C}| = q$ and assume two relations exist, namely $\mathcal{R}_\mathcal{A} \subset \mathcal{A} \times \mathcal{C}$, and $\mathcal{R}_\mathcal{B} \subset \mathcal{B} \times \mathcal{C}$. Now, let $a \in \mathcal{A}$ and $b \in \mathcal{B}$, we can define binary q-dimensional vector representations ($\mathbf{a} \in \{0,1\}^q, \mathbf{b} \in \{0,1\}^q$), for a and b respectively, on the basis of the corresponding relation. Specifically, each position in $\mathbf{a}$ and $\mathbf{b}$ corresponds to a specific element $c \in \mathcal{C}$ and it is set to 1 whenever $(a,c) \in \mathcal{R}_\mathcal{A}$, and $(b,c) \in \mathcal{R}_\mathcal{B}$, respectively.

Now, we can formally define the conditional probabilities $P(a|b)$, that is, the probability that given $c \in \mathcal{C}$ such that $(b,c) \in \mathcal{R}_\mathcal{B}$, then $(a,c) \in \mathcal{R}_\mathcal{A}$ also holds. Similarly, we can define $P(b|a)$. These probabilities can be computed with simple vector operations:

$$P(a|b) = \frac{\mathbf{a}^\top \mathbf{b}}{\mathbf{b}^\top \mathbf{b}} \text{ and } P(b|a) = \frac{\mathbf{a}^\top \mathbf{b}}{\mathbf{a}^\top \mathbf{a}}$$

where $\mathbf{a}^\top \mathbf{b}$ computes the number of times $(a,c) \in \mathcal{R}_\mathcal{A}$ and $(b,c) \in \mathcal{R}_\mathcal{B}$ co-occur. Similarly, $\mathbf{a}^\top \mathbf{a} = ||\mathbf{a}||^2$ and $\mathbf{b}^\top \mathbf{b} = ||\mathbf{b}||^2$ computes the number of times $(a,c) \in \mathcal{R}_\mathcal{A}$ occurs, and the number of times $(b,c) \in \mathcal{R}_\mathcal{B}$ occurs, respectively.

Nicely, with the definition above, the cosine similarity function can be seen as a product of the square roots of the two conditional probabilities:

$$S_{1/2}(a,b) := \cos(\mathbf{a}, \mathbf{b}) = \frac{\mathbf{a}^\top \mathbf{b}}{||\mathbf{a}|| \cdot ||\mathbf{b}||} = P(a|b)^{\frac{1}{2}} P(b|a)^{\frac{1}{2}} \quad (3)$$

The idea behind the asymmetric cosine similarity is to give asymmetric weights to the conditional probabilities in the formula above:

$$S_\alpha(a,b) = P(a|b)^\alpha P(b|a)^{1-\alpha} = \frac{\mathbf{a}^\top \mathbf{b}}{||\mathbf{a}||^{2\alpha} ||\mathbf{b}||^{2(1-\alpha)}}$$

where $0 \leq \alpha \leq 1$.

Using the notation of sets, and setting $\mathcal{R}(x) = \{c \in \mathcal{C} | (x,c) \in \mathcal{R}\}$, then we can simply write:

$$S_\alpha(a,b) = \frac{|\mathcal{R}_\mathcal{A}(a) \cap \mathcal{R}_\mathcal{B}(b)|}{|\mathcal{R}_\mathcal{A}(a)|^\alpha |\mathcal{R}_\mathcal{B}(b)|^{1-\alpha}}$$

Finally, note that eq. (3) still holds for real valued (non binary) vector representations. Clearly, in this case, the strict probabilistic interpretation cannot be applied but we can still consider $P(a|b)$ as a function indicating how much information we can get from knowing $(b,c) \in \mathcal{R}_\mathcal{B}$ in order to predict whether $(a,c) \in \mathcal{R}_\mathcal{A}$.

3.2 User-based and Song-based similarity

Cosine and Pearson's are standard measures of correlation in CF applications. To the best of our knowledge not much has been done until now to adapt these similarities to given problems. Our opinion is that it cannot exist a single similarity measure that can fit all possible domains where collaborative filtering is used. To bridge this gap and try to fit different problems, we consider a parametric family of similarities obtained by instanciating the asymmetric cosine both for the user-based and item-based setting.

MSD data have not relevance grades since the ratings are binary values. This is a first simplification that, as we have seen, we can exploit in the definition of the similarity functions. In the case of binary rates the cosine similarity can be computed as in the following. Let $\mathcal{I}(u)$ be the set of items rated by a generic user u, then the cosine similarity between two users u, v is defined by

$$w_{uv} = S_{1/2}(u,v) = \frac{|\mathcal{I}(u) \cap \mathcal{I}(v)|}{|\mathcal{I}(u)|^{\frac{1}{2}} |\mathcal{I}(v)|^{\frac{1}{2}}}$$

and, similarly for items, by setting $\mathcal{U}(i)$ the set of users which have rated item i, we obtain:

$$w_{ij} = S_{1/2}(i,j) = \frac{|\mathcal{U}(i) \cap \mathcal{U}(j)|}{|\mathcal{U}(i)|^{\frac{1}{2}} |\mathcal{U}(j)|^{\frac{1}{2}}}.$$

Note that, especially for the item case, we are more interested in computing how likely it is that an item will be appreciated by a user when we *already* know that the same user likes another item. It is clear that this definition is not symmetric. For example, say we know that a user already listened to the U2's song "Party Girl" which we know is not so popular. Then, probably that user is a fan of U2 and we can easily predict that she/he would also like something far more popular by U2, like "Sunday Bloody Sunday" for instance. Clearly, the opposite is not true.

As an extreme alternative to the cosine similarity, we can resort to the conditional probability measure which can be estimated with the following formulas:

$$w_{uv} = S_1(u,v) = P(u|v) = \frac{|\mathcal{I}(u) \cap \mathcal{I}(v)|}{|\mathcal{I}(v)|}$$

and

$$w_{ij} = S_1(i,j) = P(i|j) = \frac{|\mathcal{U}(i) \cap \mathcal{U}(j)|}{|\mathcal{U}(j)|}$$

Previous works (see [8] for example) pointed out that the conditional probability measure of similarity, $P(i|j)$, has the limitation that items which are purchased frequently tend to have higher values not because of their co-occurrence frequency but instead because of their popularity. As we have seen, this might not always be a limitation in a recommendation setting like ours. Experimental results in this paper will confirm that, emphasizing asymmetrically one of the conditionals, will help in general.

In the following of the paper we will use the parametric generalization of the above similarity measures called asymmetric cosine similarity. The parametrization on α permits ad-hoc optimizations of the similarity function for the domain of interest. In our case, this is done by validating on available data.

Summarizing, we have:

$$w_{uv} = S_\alpha(u,v) = \frac{|\mathcal{I}(u) \cap \mathcal{I}(v)|}{|\mathcal{I}(u)|^\alpha |\mathcal{I}(v)|^{1-\alpha}} \quad (4)$$

$$w_{ij} = S_\alpha(i,j) = \frac{|\mathcal{U}(i) \cap \mathcal{U}(j)|}{|\mathcal{U}(i)|^\alpha |\mathcal{U}(j)|^{1-\alpha}} \quad (5)$$

where $\alpha \in [0,1]$ is a parameter to tune.

Note that, this similarity function generalizes the one given in [5] where a similar parameterization is also proposed for the item-based case with strictly empirical motivations. Here, we give a formal justification of the rescaling factor they used by defining the similarity as an asymmetric product of conditional probabilities.

3.3 Locality of the Scoring Function

In Section 2.2 we have seen how the final recommendation is computed by a scoring function that combines the scores obtained using individual users or items. So, it is important to determine how much each individual scoring component influences the overall scoring. MBCF algorithms typically approach this problem by restricting the computation to neirest neighbors. Alternatively, we propose to use a monothonic not decreasing function $f(w)$ on the weights to emphasize/deemphasize similarity contributions in such a way to adjust the *locality* of the scoring function, that is how many, and how much of, nearest users/items really matter in the computation. As we will see, a correct setting of this function turned out to be very useful with the challenge data.

In particular, we use the exponential family of functions, that is $f(w) = w^q$ where $q \in \mathbb{N}$. The effect of this exponentiation in both the eq. (1) and eq. (2) is the following: when q is high, smaller weights drop to zero while higher ones are (relatively) emphasized; at the other extreme, when $q = 0$, the aggregation is performed by simply adding up the ratings. We can note that, in the user-based type of scoring function, this corresponds to take the popularity of an item as its score, while, in the case of item-based type of scoring function, this would turn out in a constant for all items (the number of ratings made by the active user).

4. RECOMMENDATION

In this section, we describe a novel recommendation technique based on the asymmetric cosine combination between user and item embeddings. Furthermore, we describe additional techniques, like calibration and aggregation, to improve the final recommendation.

4.1 Asymmetric Cosine based CF

In Section 2.3 it is shown how the scoring function of a memory-based model can be formulated as $\hat{r}_{ui} = \mathbf{x}_u^\top \mathbf{y}_i$, where $\mathbf{x}_u$ and $\mathbf{y}_i$ are appropriate representations of users and items, respectively. To increase the flexibility of the prediction further we can resort again to the asymmetric cosine similarity defined in Section 3.1 and replace the scoring function with the following:

$$\hat{r}_{ui} = S_\beta(u,i) = \frac{\mathbf{x}_u^\top \mathbf{y}_i}{||\mathbf{x}_u||^{2\beta} ||\mathbf{y}_i||^{2(1-\beta)}} \quad (6)$$

One can easily note that, when $\beta = 1$, the order induced by this scoring function over different songs is the same as the one induced by the standard scoring function because the two scoring functions are basically the same up to a (positive) constant. Since we are focusing on top-N recommendation and we are only interested to the ranking among songs, then we can consider that above as a generalization of the standard memory-based CF scoring function.

Now, we can analyze a little more in depth this new scoring function for the user-based and item-based prediction cases.

User-based prediction. In this case, the embedding is performed in $\mathbb{R}^n$, the space of users, according to:

$$\mathbf{x}_u^{(v)} = w_{uv} \quad \text{and} \quad ||\mathbf{x}_u||^2 = ||\mathbf{w}_u||^2$$

$$\mathbf{y}_i^{(v)} = r_{vi} \quad \text{and} \quad ||\mathbf{y}_i||^2 = |\mathcal{U}(i)|$$

and the user-based scoring function consists of:

$$\hat{r}_{ui} = \frac{\sum_{v \in \mathcal{U}(i)} w_{uv}}{||\mathbf{w}_u||^{2\beta} |\mathcal{U}(i)|^{(1-\beta)}} \propto \frac{\sum_{v \in \mathcal{U}(i)} w_{uv}}{|\mathcal{U}(i)|^{(1-\beta)}}$$

Item-based prediction. In this case, the embedding is performed in $\mathbb{R}^m$, the space of items, according to:

$$\mathbf{x}_u^{(j)} = r_{uj} \quad \text{and} \quad ||\mathbf{x}_u||^2 = |\mathcal{I}(u)|$$

$$\mathbf{y}_i^{(j)} = w_{ij} \quad \text{and} \quad ||\mathbf{y}_i||^2 = ||\mathbf{w}_i||^2$$

and the item-based scoring function consists of:

$$\hat{r}_{ui} = \frac{\sum_{j \in \mathcal{I}(u)} w_{ij}}{|\mathcal{I}(u)|^{\beta} ||\mathbf{w}_i||^{2(1-\beta)}} \propto \frac{\sum_{j \in \mathcal{I}(u)} w_{ij}}{||\mathbf{w}_i||^{2(1-\beta)}}$$

Note that, in the item case, the normalization is defined in terms of the weight norms whose computation requires the computation of the weights w_{ij} for every $j \in \mathcal{I}$ and this is quite inefficient. Then, a further step of preprocessing where this norm is estimated for each item i is needed in this case. In our implementation, this step is implemented by drawing items j from a subsample of items.

4.2 Calibration

The focus of the present work is on very large datasets. For this, until now, we have proposed memory and time efficient algorithms that avoid any time and space consuming model training. However, simple statistics from a dataset can still be estimated efficiently and can potentially result useful for recommending.

In this section, we present a simple technique to learn a basic model for songs. The model is trained using positive feedback only hence resulting quite efficient given the sparsity of data. Basically, the

idea is to calibrate the obtained scores for each item by comparing them to the average score on positive user-item pairs.

For each song the mean, and maximum, value of the scoring function obtained for positive pairs is estimated from data. More formally, let $\mathcal{R}$ the relation for which we want to compute the statistics. Then, we define the positive feedback mean and positive feedback maximum as in the following:

$$m_i \approx \mathbb{E}_{(u,i)\in\mathcal{R}}[\hat{r}_{ui}] \quad \text{and} \quad M_i \approx \max_{(u,i)\in\mathcal{R}}[\hat{r}_{ui}]$$

For the MSD data, for example, these statistics are estimated by taking $\hat{r}_{ui}$ values for a sample of training users such that $i \in \mathcal{I}(u)$. So, we need to estimate the average value of the scoring obtained by the same song for user that listened to it.

Now, let be given $0 < \theta < 1$, then the calibrated scoring function is defined as a simple piece-wise linear function:

$$\hat{r}'_{ui} = C(\hat{r}_{ui}) = \begin{cases} \frac{\hat{r}_{ui}}{m_i}\theta & \text{if } \hat{r}_{ui} \leq m_i \\ \theta + \frac{\hat{r}_{ui}-m_i}{M_i-m_i}(1-\theta) & \text{if } m_i < \hat{r}_{ui} \leq M_i \\ 1 & \text{otherwise} \end{cases}$$

In our experiments we used $\theta = 0.5$.

4.3 Aggregation

There are many sources of information available regarding songs. For example, it could be useful to consider the additional metadata which are also available and to construct alternative rankings based on that. In the following, we propose three simple strategies. We assume we have a distribution of probability p_k, such that $\sum_k p_k = 1$ that determines the weight to give to the predictor k to form the aggregated prediction. In our approach the best p_k values are simply determined by validation on training data.

4.3.1 Stochastic Aggregation

Different recommendation strategies are usually individually *precision oriented*, meaning that each strategy is able to correctly recommend a few of the correct songs with high confidence but, other songs which the user likes, cannot be suggested by that particular ranker. Hopefully, if the rankers are diverse, then the rankers can recommend different songs. If this is the case, a possible solution is to predict a final recommendation that contains all the songs for which the single strategies are more confident. The stochastic aggregation strategy used in the challenge can be described in the following way. We assume we are provided with the list of songs, not yet rated by the active user, given in order of confidence, for all the basic strategies. On each step, the recommender randomly choose one of the lists according to a probability distribution p_k over the predictors and recommends the best scored item of the list which has not yet been inserted in the current recommendation.

4.3.2 Linear Aggregation

Another very simple aggregation rule is here presented were the scores given by different predictors are simply averaged. Clearly, in this case, the scores should be comparable. When they are not, we can assume the scores $\hat{r}_{ui}^{(k)}$ are proportional to the probability of observing item i given the user u as estimated by the k-th predictor. In this case, item scores of each predictor can be normalized in such a way that $||\hat{r}_u^{(k)}||_1 = \sum_i \hat{r}_{ui}^{(k)} = 1$.

Data Statistics	min	max	ave	median
users per song	0	110479	125.794	13
songs per user	10	4400	47.45681	27

4.3.3 Borda Aggregation

The last aggregation method we propose is a slight variant of the well known Borda Count method. In this case, each item i gets a score $\sum_k p_k * (|\mathcal{I}| - q_k(i) + 1)$ where $q_k(i) \in [1, |\mathcal{I}|]$ is the position of item i in the list produced by ranker k.

4.4 Optimizations

The computational complexity of the technique proposed mainly depends on the number of weight computations. In the item-based strategy, for each test user, the number of weights to compute is proportional to $\bar{I}|\mathcal{I}|$ where $\bar{I} = \mathbb{E}_u[|\mathcal{I}(u)|]$ where the expectation is taken over visible songs of test users.

In the user-based strategy, for each test user, the number of weights to compute is proportional to $\bar{U}|\mathcal{I}|$ where $\bar{U} = \mathbb{E}_i[|\mathcal{U}(i)|]$ where the expectation is taken over available songs. However, in this second case, we can compute all the $|\mathcal{U}|$ weights for a user u and, the contribution of each weight w_{uv} accumulated on the item scores when the item is in $\mathcal{I}(v)$. This strategy empirically reduced the time required by a factor of 12 on MSD data.

5. EXPERIMENTS AND RESULTS

In the MSD challenge we have: *i)* the full listening history for about 1M users, *ii)* half of the listening history for 110K users (10K validation set, 100K test set), and we have to predict the missing half. We use a "home-made" random validation subset (*HV*) of the original training data of about 900K users of training (*HVtr*, with full listening history). The remaining 100K user's histories has been split in two halves (*HVvi* the visible one, *HVhi* the hidden one).

The experiments presented in this section are based on this *HV* data and compare different similarities and different approaches. The baseline is represented by the simple popularity based method which recommends the most popular songs not yet listened to by the user. Besides the baseline, we report experiments on both the user-based and song-based scoring functions, and an example of the application of ranking aggregation. Given the size of the datasets involved we do not stress on the significance of the presented results. This is confirmed by the fact that the presented results do not differ significantly from the results obtained over the indipendent set of users used as the test set in the challenge.

In the following results, when not explicitely indicated, we assume $\beta = 1$ in eq. (6) thus obtaining the standard scoring function for memory-based CF.

5.1 Taste Profile Subset Stats

For completeness we report some statistics about the original challenge training data. In particular, the following table shows the minimum, maximum, and average, number of users per song and songs per user. The median value is also reported.

We can see that the large majority of songs have only few users which have listened to it (less than 13 users for half of the songs) and the large majority of users have listened to few songs (less than 27 for half of the users). These characteristics of the dataset make the top-N recommendation task quite challenging.

5.2 Truncated Mean Average Precision

Conformingly to the challenge, we used the truncated mAP (mean average precision) as the evaluation metric [10]. Let y denote a ranking over items, where $y(p) = i$ means that item i is ranked at position p. The mAP metric emphasizes the top recommendations. For any $k \leq N$, the *precision at k* (π_k) is defined as the proportion of correct recommendations within the top-k of the predicted ranking (assuming the ranking y does not contain the visible songs),

$$\pi_k(u, y) = \frac{1}{k} \sum_{p=1}^{k} r_{uy(p)}$$

For each user the (truncated) average precision is the average precision at each recall point:

$$AP(u, y) = \frac{1}{N_u} \sum_{p=1}^{N} \pi_k(u, y) r_{uy(p)}$$

where N_u is the smaller between N and the number of user u's positively associated songs. Finally, the average of $AP(u, y_u)$'s over all users gives the mean average precision (mAP).

5.3 Results on MSD data

The result obtained on the HV data with the baseline (recommendation by popularity) is presented in Table 1(a). With this strategy, each song i simply gets a score proportional to the number of users $|\mathcal{U}(i)|$ which have listened to it.

Effect of locality (q). In Table 1, we report on experiments that show the effect of the locality parameter q for different strategies: item based and user based, using conditional probability ($\alpha = 0$) and the cosine version ($\alpha = 0.5$). As we can see, beside the case IS with cosine similarity (Table 1c), a correct setting of the parameter q drammatically improves the effectiveness on HV data. We can clearly see that the best performance is reached with the conditional probability on an item based strategy (Table 1b).

Effect of using Asymmetric Cosine (α). In Figure 1, results obtained fixing the parameter q and varying the parameter α for both user and item-based recommendation strategies are given. In the item-based case, the results improve when setting a non-trivial α. In fact, the best result has been obtained for $\alpha = 0.15$.

Effect of Asymmetric Prediction (β). In Table 2 we report the results of user-based AsymC memory-based collaborative filtering algorithm of Section 4.1 with $\alpha = 0.5$. Interestingly, as we can see from the table, the best parameter β always improves (sometimes dramatically) the baseline ($\beta = 1$, results in Table 1(e)).

Effect of Calibration. To test the effect of calibration in item-based recommandation we started with the best parameter setting for uncalibrated recommendation ($\alpha = 0.15, q = 3, mAP@500 = 0.177322$) and calibrated using the technique depicted in Section 4.2. The calibration in fact improve the result, obtaining an interesting $mAP@500 = \mathbf{0.181084}$ (the best result obtained for this dataset so far).
Concerning the effect of calibration on user-based recommendation we started again with the best setting for uncalibrated user-based recommendation ($\alpha = 0.5, q = 4, \beta = 0.7, mAP@500 =$

Baseline	mAP@500
Recommendation by Popularity	0.02262

(a)

IS ($\alpha = 0$)	mAP@500
q=1	0.12224
q=2	0.16581
q=3	**0.17144**
q=4	0.17004
q=5	0.16830

(b)

IS ($\alpha = \frac{1}{2}$)	mAP@500
q=1	**0.16439**
q=2	0.16214
q=3	0.15587
q=4	0.15021
q=5	0.14621

(c)

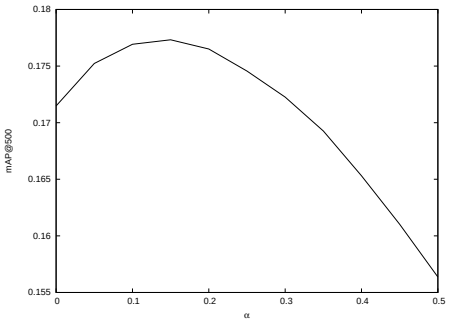
US ($\alpha = 0$)	mAP@500
q=1	0.08030
q=2	0.10747
q=3	0.12479
q=4	0.13298
q=5	**0.13400**
q=6	0.13187
q=7	0.12878

(d)

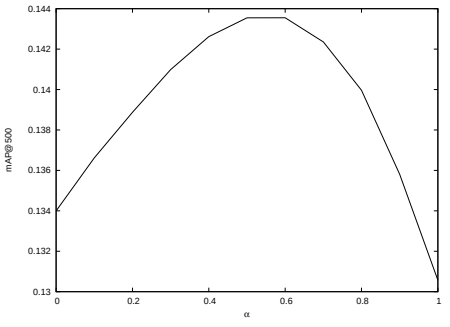
US ($\alpha = \frac{1}{2}$)	mAP@500
q=1	0.07679
q=2	0.10436
q=3	0.12532
q=4	0.13779
q=5	0.14355
q=6	**0.14487**
q=7	0.14352

(e)

Table 1: Results obtained by the baseline, item-based (IS) and user-based (US) CF methods varying the locality parameter (exponent q) of the similarity function.



(a) IS with $0 \leq \alpha \leq 0.5$, $q = 3$, best-mAP@500: $0.177322(\alpha = 0.15)$



(b) US with $0 \leq \alpha \leq 1$, $q = 5$, best-mAP@500: $0.143551(\alpha = 0.6)$

Figure 1: Results obtained by item-based (IS) and user-based (US) CF methods varying the α parameter.

US ($\alpha = \frac{1}{2}$)	Best β	mAP@500
q=1	0.3	0.14890
q=2	0.5	0.15801
q=3	0.6	0.16132
q=4	0.7	**0.16229**
q=5	0.8	0.16152
q=6	0.9	0.15975
q=7	0.9	0.15658

(a)

Table 2: User-based AsymC MBCF ($\alpha = 0.5$). Results obtained varying the parameter q and selecting the best parameter β.

(IS, $\alpha = 0.15$, $q = 3$)	(US, $\alpha = 0.3$, $q = 5$)	mAP@500
0.0	1.0	0.14098
0.1	0.9	0.14813
0.2	0.8	0.15559
0.3	0.7	0.16248
0.4	0.6	0.16859
0.5	0.5	0.17362
0.6	0.4	0.17684
0.7	0.3	0.17870
0.8	0.2	**0.17896**
0.9	0.1	0.17813
1.0	0.0	0.17732

(a)

Table 3: Results obtained aggregating the rankings of two different strategies, item-based (IS, $\alpha = 0.15$, $q = 3$) and user-based (US, $\alpha = 0.3$, $q = 5$), by Stochastic Aggregation as discussed in Section 5.3, with different combinations. The $(0.8, 0.2)$ combination corresponds exactly to the winning configuration used in the MSD challenge (see Figure 2).

0.16229) and calibrated using the technique described in Section 4.2 obtaining a $mAP@500 = 0.16487$.

Stochastic Aggregation. Concerning the stochastic aggregation we report results obtained using the exact configuration that allowed us to win the MSD challenge (see Figure 2). In particular, in Table 3, two rankers are combined, and their recommendation aggregated, by using the stochastic algorithm described in Section 4.3.1. In order to maximize the diversity of the two rankers, we aggregated an item-based ranker and a user-based ranker. We can see that the combination improves the performance with respect to the individual rankers on validation data.

Linear Aggregation. In Table 4 we report on results obtained by linearly combining the best performing uncalibrated item-based ranker ($\alpha = 0.15$, $q = 3$, $\beta = 1$) with the best performing uncalibrated user-based ranker ($\alpha = 0.5$, $q = 4$, $\beta = 0.7$) varying the combination parameters. The aggregation is performed according to the algorithm given in Section 4.3.2.

5.4 Comparison with other approaches

We end this section by comparing our with other approaches that have been used in the challenge. Best ranked teams all used variants of memory based CF, besides the 5-th ranked team that used the Absorption algorithm by YouTube [2] which is a graph based

IS($\alpha = 0.15$, $q = 3$)	US($\alpha = 0.5$, $q = 4$,$\beta = 0.7$)	mAP@500
0.5	0.5	0.18014
0.6	0.4	0.18042
0.7	0.3	0.18074
0.8	0.2	**0.18092**
0.9	0.1	0.17853
1.0	0.0	0.17732

(a)

Table 4: Results obtained aggregating the rankings of two different strategies, item-based (IS, $\alpha = 0.15$, $q = 3$) and user-based (US, $\alpha = 0.3$, $q = 5$) AsymC MBCF ($\beta = 0.7$), with different combinations.

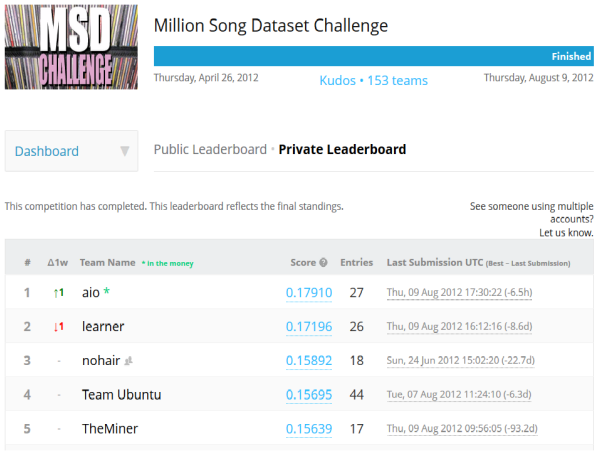


Figure 2: Screenshot of the final MSD challenge leaderboard.

method that performs a random walk on the rating graph to propagate preferences information over the graph. The team placed second used an approach similar to ours to get 17 user-item joint features to use in a learning-to-rank method (RankNet) [14]. On the other side, matrix factorization based techniques showed a very poor performance on this task and people working on that faced serious memory and time efficiency problems. Finally, some teams tried to inject meta data information in the prediction process with scarse results. In our opinion, this can be due to the fact that there is a lot of implicit information contained in the user's history and this is much more than explicit information one can get from metadata. We conclude that meta data information can be more effectively used in a *cold start* setting.

5.5 Results on MovieLens Data

To demonstrate the generality of the approach, we present results on a different dataset, MovieLens1M[2]. This dataset originally consists of more than 1 million of ratings for $3,883$ movies and $6,040$ users. The dataset has been made suitable to implicit feedback by considering the $226,310$ 5-stars ratings as positive feedback and ignoring all the other available ratings, thus obtaining a density of 0.01. Similarly to the MSD dataset, data have been split in a 90% training users and 10% test users. Test user histories has been split again in two halves (visible and hidden). Finally, the task consists on predicting 5-stars movies in the hidden half using 5-stars movies in the visible half and complete history of training users.

[2]http://www.grouplens.org/node/73

Baseline	Best Parameters	mAP@500
User-based kNN	$k = 100$	0.18318

(a)

No	Ranker	Best Parameters	mAP@500
1	IS	$\alpha = 0.6, q = 1$	0.18146
2	IS ($\beta = 0.8$)	$\alpha = 0.55, q = 1$	0.18297
3	US	$\alpha = 0.6, q = 6$	0.18549
4	US ($\beta = 0.8$)	$\alpha = 0.55, q = 5$	**0.18984**

(b)

Aggregated	Best (p_i, p_u)	mAP@500
1 + 3 (Borda)	(0.4,0.6)	0.19008
2 + 4 (Borda)	(0.2,0.8)	**0.19554**
1 + 3 (Stochastic)	(0.5,0.5)	0.19015
2 + 4 (Stochastic)	(0.1,0.9)	0.19052

(c)

Table 5: MovieLens1M results. (a) Best results obtained with the kNN baseline, (b) best results obtained with single rankers (not aggregated), (c) best results obtained with aggregated rankers

Results are presented in Table 5. Specifically, in Table 5(a) the best result obtained with the user-based kNN baseline is reported. In this case, the score of an user-item pair (u, i) is

$$\hat{r}_{ui} = \frac{\sum_{v \in \mathcal{N}_u} w_{uv} r_{vi}}{\sum_{v \in \mathcal{N}_u} w_{uv}}$$

where $\mathcal{N}_u$ is the set of k nearest neighbors of user u according to the similarity w_{uv}. Note that, in this case, differently from the proposed approach, the similarity has to be computed for the entire set of users. This clearly requires far more computational efforts and would be unfeasible for the MSD task. In Table 5(b) the results obtained using symmetric and asymmetric, item and user-based scoring functions are reported. From the table it is clear that user-based rankers are better than item-based ones and asymmetric scoring is beneficial in both the cases. Finally, in Table 5(c) results obtained by aggregation are presented confirming that aggregating item-based and user-based rankers always improves significantly the recommendation performance.

6. CONCLUSION

In this paper we have presented an efficient memory-based CF technique suitable for very large recommendation problems with implicit feedback. The proposed technique extends the one we used to win the MSD challenge [3] (see also [1]). The main contributions of the paper are: a novel scoring function for memory based CF that results particularly effective (and efficient) on implicit rating settings and a new asymmetric similarity measure that can be adapted to the problem at hand that seems to work very well for CF problems. In the near future we want to investigate on the possibility of using metadata information to boost the performance and in a more solid way to aggregate multiple predictions. Finally, it would be interesting to generalize the method to other types of recommendation, including explicit feedback and non binary ratings.

[3]The MSD challenge has been organized by the Computer Audition Lab at UC San Diego and LabROSA at Columbia University, and hosted at www.kaggle.com

7. ACKNOWLEDGMENTS

This work was supported by the Italian Ministry of Education, University, and Research (MIUR) under Project PRIN 2009LNP494_005. Many thanks to Marta Aduasio for helping us with the experiments.

8. REFERENCES

[1] F. Aiolli. A preliminary study on a recommender system for the million songs dataset challenge. In *IIR*, pages 73–83, 2013.

[2] S. Baluja, R. Seth, D. Sivakumar, Y. Jing, J. Yagnik, S. Kumar, D. Ravichandran, and M. Aly. Video suggestion and discovery for youtube: taking random walks through the view graph. In *Proceedings of the 17th international conference on World Wide Web*, WWW '08, pages 895–904, New York, NY, USA, 2008. ACM.

[3] J. Bennett, S. Lanning, and N. Netflix. The netflix prize. In *In KDD Cup and Workshop in conjunction with KDD*, 2007.

[4] T. Bertin-Mahieux, D. P. Ellis, B. Whitman, and P. Lamere. The million song dataset. In *Proceedings of the 12th International Conference on Music Information Retrieval (ISMIR 2011)*, 2011.

[5] M. Deshpande and G. Karypis. Item-based top-*n* recommendation algorithms. *ACM Trans. Inf. Syst.*, 22(1):143–177, 2004.

[6] C. Desrosiers and G. Karypis. A comprehensive survey of neighborhood-based recommendation methods. In *Recommender Systems Handbook*, pages 107–144. 2011.

[7] Y. Hu, Y. Koren, and C. Volinsky. Collaborative filtering for implicit feedback datasets. In *ICDM*, pages 263–272, 2008.

[8] G. Karypis. Evaluation of item-based top-n recommendation algorithms. In *CIKM*, pages 247–254, 2001.

[9] Y. Koren and R. M. Bell. Advances in collaborative filtering. In *Recommender Systems Handbook*, pages 145–186. 2011.

[10] B. McFee, T. Bertin-Mahieux, D. P. Ellis, and G. R. Lanckriet. The million song dataset challenge. In *Proceedings of the 21st international conference companion on World Wide Web*, WWW '12 Companion, pages 909–916, New York, NY, USA, 2012. ACM.

[11] V. C. Ostuni, T. Di Noia, E. Di Sciascio, and R. Mirizzi. Top-n recommendations from implicit feedback leveraging linked open data. In *7th ACM Conference on Recommender Systems (RecSys 2013)*. ACM, ACM Press, 2013.

[12] B. M. Sarwar, G. Karypis, J. A. Konstan, and J. Riedl. Item-based collaborative filtering recommendation algorithms. In *WWW*, pages 285–295, 2001.

[13] X. Su and T. M. Khoshgoftaar. A survey of collaborative filtering techniques. *Advances in Artificial Intelligence*, Jan. 2009.

[14] M. Volkovs and R. Zemel. Collaborative ranking with 17 parameters. In P. Bartlett, F. Pereira, C. Burges, L. Bottou, and K. Weinberger, editors, *Advances in Neural Information Processing Systems 25*, pages 2303–2311. 2012.

[15] J. Wang, A. P. de Vries, and M. J. T. Reinders. A user-item relevance model for log-based collaborative filtering. In M. Lalmas, A. MacFarlane, S. M. RÃijger, A. Tombros, T. Tsikrika, and A. Yavlinsky, editors, *ECIR*, volume 3936 of *Lecture Notes in Computer Science*, pages 37–48. Springer, 2006.