Combining multiple metadata types in movies recommendation using ensemble algorithms

http://www.producao.usp.br/handle/BDPI/48641

Downloaded from: Biblioteca Digital da Produção Intelectual - BDPI, Universidade de São Paulo
Combining Multiple Metadata Types in Movies Recommendation Using Ensemble Algorithms

Bruno Cabral  
Department of Computer Science  
Federal University of Bahia  
Salvador, Brazil  
bruno.cabral@ufba.br

Renato D. Beltrão  
Mathematics and Computing Institute - University of São Paulo  
São Carlos, SP – Brazil  
rdompleri@usp.br

Marcelo G. Manzato  
Mathematics and Computing Institute - University of São Paulo  
São Carlos, SP – Brazil  
mmanzato@icmc.usp.br

Frederico Araújo Durão  
Department of Computer Science  
Federal University of Bahia  
Salvador, Brazil  
freddurao@dcc.ufba.br

ABSTRACT
In this paper, we analyze the application of ensemble algorithms to improve the ranking recommendation problem with multiple metadata. We propose three generic ensemble strategies that do not require modification of the recommender algorithm. They combine predictions from a recommender trained with distinct metadata into a unified rank of recommended items. The proposed strategies are Most Pleasure, Best of All and Genetic Algorithm Weighting. The evaluation using the HetRec 2011 MovieLens 2k dataset with five different metadata (genres, tags, directors, actors and countries) shows that our proposed ensemble algorithms achieve a considerable 7% improvement in the Mean Average Precision even with state-of-art collaborative filtering algorithms.

Categories and Subject Descriptors
H.3.3 [Information Search and Retrieval]: Information Filtering

General Terms
Design, Algorithms

Keywords
recommendation; ensemble; metadata; movie; collaborative filtering

1. INTRODUCTION
Recommender systems have become increasingly popular and widely adopted by many sites and services. They are important tools in assisting users to filter what is relevant in this complex information world. There are a number of ways to build recommender systems; they are classified as content-based filtering, collaborative filtering or the hybrid approach, which combines both filtering strategies [1, 5].

Content-based filtering recommends multimedia content to the user based on a profile containing information regarding the content, such as genre, keywords, subject, etc. These metadata are weighted according to past ratings, in order to characterize the user’s main interests. However, this approach has problems such as over-specialization [1] and limited performance due to metadata scarcity or quality. An alternative to this problem is the collaborative filtering, which is based on clusters of similar users or items. One disadvantage of collaborative filtering is the computational effort spent to calculate similarity between users and/or items in a vectorial space composed of user ratings in a user-item matrix.

Such limitations have inspired researchers to use matrix factorization techniques, such as Singular Value Decomposition (SVD), in order to extract latent semantic relationships between users and items, transforming the vectorial space into a feature space containing topics of interest [20, 11, 17, 10]. Nevertheless, other challenges have to be dealt with, such as sparsity, overfitting and data distortion caused by imputation methods [10].

Considering the limitations and challenges depicted above, hybrid recommenders play an important role because they group together the benefits of content based and collaborative filtering. It is known that limitations of both approaches, such as the cold start problem, overspecialization and limited content analysis, can be reduced when combining both strategies into a unified model [1]. However, most recent systems which exploit latent factor models do not consider the metadata associated to the content, which could provide significant and meaningful information about the user’s interests. Another issue of current metadata aware recommenders is that usually they support only one type of item attribute at a time. To overcome this issue, Beltrão et al. [3] analyzed the performance of a recommender using multiple types of metadata, by concatenating the dif-
different pieces of information, and although the performance improved, the results were still modest.

Similarly to Beltrão et al. [3], this paper proposes a different approach for handling multiple metadata, using ensemble algorithms. We use three different ensemble strategies to combine different metadata, but with the advantage that it does not require the algorithm to be modified, or to be trained multiple times with the same dataset, and therefore, it can be used in all current Recommender Systems.

This work is structured as follows: in Section 2 we review related works that use ensemble algorithms; in Section 3 we briefly describe the models considered in this evaluation; in Section 4 we detail our proposed Ensemble framework and strategies; Section 5 presents the evaluation and validation of the approach with HetRec dataset with 855598 ratings, and analysis of the performance of the three proposed strategies; and finally, in Section 6 we discuss the final remarks, future work and acknowledgments.

2. RELATED WORK

An ensemble method combines the predictions of different algorithms, or the same algorithm with different parameters to obtain a final prediction. Ensemble algorithms have been successfully used, for instance, in the Netflix Prize contest consisting of the majority of the top performing solutions. [23, 18].

Most of the related works in the literature point out that ensemble learning has been used in recommender system as a way of combining the prediction of multiple algorithms (heterogeneous ensemble) to create a stronger rank [9], in a technique known as blending. They have been also used with a single collaborative filtering algorithm (single-model or homogeneous ensemble), with methods as Bagging and Boosting [2]. However, those solutions do not consider the multiple metadata present in the items, and are often not practical to implement in a production scenario because of the computational cost and complexity. In the case of heterogeneous ensemble, it needs to train all models in parallel and treat the ensemble as one big model, but unfortunately training 100+ models in parallel and tuning all parameters simultaneously is computationally not feasible [23]. In contrast, the homogeneous ensemble demands the same model to be trained multiple times, and some methods such as Boosting requires that the underlying algorithm be modified to handle the weighted samples. Beltrão et al. [3] tried a different approach and combined multiple metadata by concatenating them, with a modest performance increase.

In comparison to the above approaches, our method uses three different ensemble strategies to combine distinct metadata, but with the advantage that it does not require the algorithm to be modified, or to be trained multiple times with the same dataset, and therefore, it can be used in all of the current Recommender Systems. This is because our method uses the user prediction (which is the least possible information in any Recommender System). Our approach involves two voting strategies and a weighted strategy where the parameters are optimized using a Genetic Algorithm approach.

3. CONSIDERED MODELS

In this section we describe in more details the models used to study and compare the different types of metadata considered in this paper. In the next three subsections, we present a set of metadata aware algorithms which use the Bayesian Personalized Ranking (BPR) framework [6] to personalize a ranking of items using only implicit feedback. These techniques will be considered in our evaluation in the context of movies recommendation.

3.1 Notation

Following the same notation in [10, 12], we use special indexing letters to distinguish users, items and attributes: a user is indicated as \( u \), an item is referred as \( i,j,k \) and an item’s attribute as \( g \). The notation \( r_{ui} \) is used to refer to explicit or implicit feedback from a user \( u \) to an item \( i \). In the first case, it is an integer provided by the user indicating how much he liked the content; in the second, it is just a boolean indicating whether the user consumed or visited the content or not. The prediction of the system about the preference of user \( u \) to item \( i \) is represented by \( \hat{r}_{ui} \), which is a floating point value calculated by the recommender algorithm. The set of pairs \( (u,i) \) for which \( r_{ui} \) is known is represented by the set \( K = \{(u,i)|r_{ui} \text{ is known}\} \).

Additional sets used in this paper are: \( N(u) \) to indicate the set of items for which user \( u \) provided an implicit feedback, and \( N(u) \) to indicate the set of items that is unknown to user \( u \).

3.2 BPR-Linear

The BPR-Linear [6] is an algorithm based on the Bayesian Personalized Ranking (BPR) framework, which uses item attributes in a linear mapping for score estimation. The prediction rule is defined as:

\[
\hat{r}_{ui} = \phi_f(\vec{a}_i) = \sum_{g=1}^{n} w_{ug} a_{ig} ,
\]

where \( \phi_f : \mathbb{R}^n \rightarrow \mathbb{R} \) is a function that maps the item attributes to the general preferences \( \hat{r}_{ui} \) and \( \vec{a}_i \) is a boolean vector of size \( n \) where each element \( a_{ig} \) represents the occurrence or not of an attribute, and \( w_{ug} \) is a weight matrix learned using LearnBPR, which is variation of the stochastic gradient descent technique [7]. This way, we first compute the relative importance between two items:

\[
\hat{s}_{uij} = \hat{r}_{ui} - \hat{r}_{uj} = \sum_{g=1}^{n} w_{ug} a_{ig} - \sum_{g=1}^{n} w_{ug} a_{jg} \\
= \sum_{g=1}^{n} w_{ug} (a_{ig} - a_{jg}) .
\]

Finally, the partial derivative with respect to \( w_{ug} \) is taken:

\[
\frac{\partial}{\partial w_{ug}} \hat{s}_{uij} = (a_{ig} - a_{jg}) ,
\]

which is applied to the LearnBPR Algorithm considering that \( \Theta = (w) \) for all set of users and descriptions.

3.3 BPR-Mapping

The BPR-Mapping was also proposed by Gantner et al. [6]; the key difference is that it uses the linear mapping depicted in Subsection 3.2 to enhance the item factors which will be later used in an extended matrix factorization prediction rule. Such an extension of matrix factorization is optimized for Bayesian Personalized Ranking (BPR-MF) [19]
that can deal with the cold-start problem, yielding accurate and fast attribute-aware item recommendation. Gantner et al. [6] address the case where new users and items are added by first computing the latent feature vectors from attributes like the user’s age or movie’s genres, and then using those estimated latent feature vectors to compute the score from the underlying matrix factorization (MF) model.

The model considers the matrix factorization prediction rule:

$$\hat{r}_{ui} = b_{ui} + p_u^T q_i = b_{ui} + \sum_{f=1}^{k} p_{uf} q_{if},$$

where each user $u$ is associated with a user-factors vector $p_u \in \mathbb{R}^k$, and each item $i$ with an item-factors vector $q_i \in \mathbb{R}^k$. The baseline $b_{ui}$ is defined as $b_{ui} = \mu + b_u + b_i$, and indicates the distinct estimates of users and items in comparison to the overall rating average $\mu$.

From this model, the item factors are mapped according to their attributes as:

$$\hat{r}_{ui} = b_{ui} + \sum_{f=1}^{k} p_{uf} \phi_f(\tilde{a}_i),$$

where $\phi_f(\tilde{a}_i)$ has the same definition as in Equation 1.

### 3.4 MABPR

One disadvantage of the previous BPR algorithms is that they are not able to infer any conclusion when the items $i$ and $j$ are known (or both are unknown). In other words, if an item has been viewed by the user, it is possible to conclude that this content is preferred over all other unknown items, as it aroused a particular interest to him than the others. On the other hand, when both items are known (or both are unknown), it is not possible to infer which one is preferred over the other because the system only has the positive/negative feedback from the user. Consequently, those pairs which belong to the same class (positive or negative) will not be able to be ranked accordingly, as the model will be learned only by using the specific case where one item is known and the other is not.

To overcome this limitation, Manzato et al. [13] proposed an extension to the BPR technique which also considers metadata from items in order to infer the relative importance of two items.

It starts by redefining the set $D_K$ which contains the data used during training to $D_K := \{(u, i, j)|i \in N(u) \& j \in N(u) \lor i \in N(u) \& j \in N(u) \& |G(i)| > 0 \& |G(j)| > 0\}$ to consider the metadata available in the specified case, while also considering items without descriptions.

Figure 1 shows how the proposed extension affects the relationship between items $i$ and $j$ with respect to the preferences of user $u$. Because items $i_2$, $i_4$ and $i_6$ are known, the system has to analyze their metadata to infer which one is preferred over the other. This is the role of function $\delta(i, j)$, which is defined as:

$$\delta(i, j) = \begin{cases} + & \text{if } \varphi(u, i) > \varphi(u, j), \\ - & \text{if } \varphi(u, i) < \varphi(u, j), \\ ? & \text{otherwise}, \end{cases}$$

where $\varphi(u, .)$ is defined as:

$$\varphi(u,. ) = \frac{1}{|G(\cdot)|} \sum_{g \in G(\cdot)} w_{ug},$$

and $w_{ug}$ is a weight indicating how much $u$ likes a description $g \in G(\cdot)$.

This approach enhances the BPR algorithm with further insight about the user’s preferences by considering his personal opinions about particular descriptions of items. Such metadata can be of any type: genres of movies/music, keywords, list of actors, authors, etc.

The mechanism used to infer such opinions $w_{ug}$ by analyzing only the training data is accomplished by adopting the same linear attribute-to-feature mapping described in Subsection 3.2.

### 3.5 MostPopularByAttributes

This is a simple algorithm similar to the “Same artist - greatest hits” baseline presented on McFee et al. [15]. It recommends a ranked item list ordered by popularity, considering attributes that the user had seen previously, followed by the remaining items also ordered by popularity. For instance, if the user had listened to only Rock music, it will recommend first the most popular Rock songs, followed by other genres.

### 4. PROPOSED ENSEMBLE ALGORITHMS

The algorithms presented in Section 3 support only one metadata per item. This is a point of improvement, as it is common for an item to have multiple metadata. In a previous work, we studied this problem of using multiple metadata by concatenating the different types of attributes as a single metadata x item list [7]; however, the performance improvement was moderate. In this paper, the proposed ensemble framework consists of training the recommender system for each different item metadata and combining them with one of the three ensemble strategies presented next.

The strategies elicited here were inspired by group decision-making strategies that combine several users’ preferences to aggregate item-ranking lists. According to Senot et al [21] there are three categories of strategies, namely majority-based, which strength the “most popular” choice among the group, e.g. Borda Count and Plurality Voting strategies;
Consensus-based strategies, which average somehow all the available choices, e.g. Additive Utilitarian and Average without Misery; and borderline strategies, also known as role-based strategies, which only consider a subset of choices based on user roles or any other relevant criterion, e.g. Dictatorship, Least Misery and Most Pleasure strategies.

Before introducing the ensemble algorithms, we need to recall that our recommenders produce a ranking of items. For generating the recommendations, this Ranking-Oriented Recommender receives as an input a dataset of ratings as a tuple $(u, i, r)$, and outputs a matrix $M_{u, i}$, where $U$ is the set of all users and $I$ is the set of all items known by the recommender system. Each row of the matrix $M$ is composed of a vector of tuples $(i, \hat{r}_{ui})$, ordered by the item score prediction $\hat{r}_{ui}$ for the user $u$. The ensemble algorithms proposed in this paper can be formally defined as a function $f : M^K \rightarrow M$, where the input is a vector of k-predictions and the output is a matrix of the combined predictions.

In Subsection 4.1 we present the Most Pleasure, the simplest ensemble strategy, that combines predictions based on score. In Subsection 4.2, we describe the Best of All strategy, that determines a preferred metadata for a user and uses it to create the ensemble, and finally, in Subsection 4.3 the Weighting strategy is presented; it uses multiple metadata and weighs them with a Genetic Algorithm optimizing the Mean Average Precision (MAP).

4.1 Most Pleasure Strategy

The Most Pleasure strategy is a classic aggregation method, often used for combining individual ratings for group rating [14]. It takes the maximum of individual ratings for a specific item and creates a unified rank. Figure 2 illustrates the Most Pleasure strategy, in which the output comprehends a ranked list of movies with highest ratings from two distinct input sets.

Algorithm 1 shows that it only needs the generated prediction vector as an input. This vector is composed of the predictions from the recommender algorithm trained with one of the item metadata. For each user, a new prediction is created, selecting the highest score of an item among all the individually-trained algorithms.

The idea behind this strategy is that differently trained algorithms have a distinct knowledge about the user’s preferences, and the predicted score can be considered an indicator of the algorithm’s confidence. So the created ensemble is a list of items whose the distinct algorithms have more confidence to recommend.

**Algorithm 1:** Most Pleasure algorithm.

### Input:
- Vector of predictions, $P$

### Output:
- Predictions ensemble $M$

```python
for u = 1,...,#Users do
    for i = 1,...,#Items do
        Select highest $\hat{r}_{ui}$ for the item $i$ among the K-predictions for the user $u$
        $M_{ui} \leftarrow (i, \hat{r}_{ui})$ //Store the highest score
    end
    Sort $M_u$ by $\hat{r}_{ui}$
end
```

![Figure 2: Most Pleasure Strategy.](image)

4.2 Best of All Strategy

The Most Pleasure strategy gives the same weight for different types of metadata. However, it is natural to assume that different types of metadata can affect users differently. In contrast, the Best of All strategy considers the recommendation algorithm that provides the best results for a specific user, and uses this algorithm to provide future predictions as illustrated in Figure 3.

Algorithm 2 requires as an input the i) recommendation algorithm, ii) a training dataset, iii) a probe dataset, and iv) the vector of item’s metadata. Differently from the Most Pleasure strategy, this one requires a probe run to determine which is the best performing algorithm. Therefore, the dataset is divided in training and probe. The algorithm is primarily trained using each of item metadata individually. Then, for each user, a probe is made to determine the metadata with the highest performance. This performance is indicated by the Mean Average Precision (MAP) metric [8], often used for ranked recommendations. Finally, the algorithms are retrained using all data (including the probe set), and the final ensemble is the result of the combination of predictions using, for each user, the prediction from the algorithm with the highest performance in the probe test.

The idea behind this algorithm is that a single metadata can greatly influence the user’s preferences, and this should be used for future predictions. For instance, if a User A enjoys films from a particular genre such as “horror”, and other User B enjoys films of some specific theme such as “bloody films”, the ensemble will contain predictions from the recommendation algorithm trained with both: the genre metadata for User A, i.e. “horror”, and a keyword metadata for user B, i.e. “bloody”.

4.3 Weighting Strategy

One drawback of the Best of All strategy is that it considers that only one metadata influences the user preference.
Input: T - Training dataset of rating \(<U,I,R>\)
Input: P - Probe dataset of rating \(<U,I,R>\)
Input: A - Vector of Metadata
Input: PredAlg - the Base prediction algorithm
Output: Predictions ensemble M

for \(m = 1,...,#Metadata\) do
| \(K_m \leftarrow\) PredAlg Trained with T dataset and \(A_u\) |
end

for \(u = 1,...,#Users\) do
| Evaluate all \(K\) algorithms against the \(P\) dataset and select the one with highest MAP for the user \(u\) as \(\text{highest}_u\) |
end

for \(m = 1,...,#Metadata\) do
| \(K_m \leftarrow\) PredAlg Trained with T+P dataset and \(A_u\) |
end

for \(u = 1,...,#Users\) do
| \(r_u \leftarrow K_{\text{highest}_u} u\) |
| \(M_u \leftarrow r_u\) |
end

Algorithm 2: Best of All algorithm.

However, it is natural to assume that the interests of a user may be influenced by more than one metadata, and with different levels. The Weighting strategy considers all available metadata assigning different weights for each prediction as illustrated in Figure 4.

Similarly to the previous strategy, the Algorithm 3 requires as an input the i) recommendation algorithm, ii) a training and probe dataset, and iii) the vector of item metadata. After training the algorithm using each of item metadata individually, a probe run is also needed; however, the objective is to determine the optimal weights for each user. This is an optimization problem and was solved using a Genetic Algorithm (GA). GA is particularly appealing for this type of problem due to its ability to handle multi-objective problems. In addition, the parallelism of GA allows the search space to be covered with less likelihood of returning local extremes [16].

The probe part consists of running the GA to find out the optimal weights. We implemented our algorithm using the GA Framework proposed by Newcombe [16], where the weights are the chromosomes, and the fitness function is the MAP score against the probe dataset. Other GA characteristics includes the use of 5% of Elitism, Double Point crossing-over, and Binary Mutations. Finally, the algorithms are retrained using all data (including the probe set), and the final ensemble uses, as the item score, the sum of individual predictions multiplied by the weights found in the probe phase and divided by the total number of metadata.

Algorithm 3: Weighting algorithm.

The idea behind it is that the different types of metadata influence differently the user preference. Still in the context of movies, let us consider two users: User A, that enjoys films from a determinate set of genres, but do not care about the production country and User B, that does not care about film genre or country of production. For the User A, the ensemble should give a higher weight for the film genre, and a lower weight for the production country. In contrast, to the User B, the ensemble should equally distribute the weights between those metadata.

5. EVALUATION

In the evaluation presented in this paper, we compared the combination of five different types of metadata: actors, directors, genres, tags and countries using the recommendation algorithms previously described in Section 3 and the ensemble algorithms described in Section 4. All algorithms were implemented using the MyMediaLite library [7], which provides the needed infrastructure such as matrix factorization algorithms and error measure methods. To measure the accuracy of recommendations, we used the Mean Average Precision (MAP).

All tests were executed with the HetRec 2011 MovieLens 2k dataset [4], an extension of MovieLens10M dataset, which contains personal ratings and tags about movies. In the dataset, MovieLens movies are linked to the Internet Movie Database (IMDb) and RottenTomatoes (RT) movie review systems. Each movie has its IMDb and RT identifiers, English and Spanish titles, picture URLs, genres, directors, actors (ordered by “popularity”), countries, filming locations, and RT audience and experts’ ratings and scores. The dataset was composed of 2113 users with 855598 ratings on 10197 movies, including the relation between 20 movie genres, 4060 directors, 95321 actors, 72 countries and 13222 tags.

\(^1\)Internet Movie Database, http://www.imdb.com
\(^2\)Rotten Tomatoes, movie critic reviews, http://www.rottentomatoes.com
The three matrix factorization algorithms from Section 3 were evaluated using a fixed latent factor of 10, and as a preliminary run, they achieved the highest MAP score for the majority of cases. The Genetic Algorithm (GA) uses a population of size 40 with 90 generations, a crossover probability of 80% and a mutation probability of 8%. Usually a higher number of generations is used for convergence; however, due to the size of our dataset, a moderated number was used.

We split the dataset randomly in an 80:20 proportion and used as training and evaluation respectively. However, due to the need of a probe run in some of the ensemble strategies presented in section 4, 25% of training dataset was split again to the probe run, resulting in a 60:20:20 split as illustrated in Figure 5. It is important to note that during the evaluation the algorithm is trained with the full training dataset. To summarize, the ensemble was created with an algorithm trained with the 60% dataset and evaluated with the 20% probe dataset, later with the ensemble created, the algorithm was trained again, this time with the full 80% training dataset and evaluated with the evaluation dataset.

Finally, we executed for each algorithm, eight different runs, resulting in total 32 runs. The first five are runs where the algorithm is trained with one of the metadata individually, and are used as baseline for performance evaluations of three ensemble strategies. Thus, we compared the best MAP scores in each algorithm and each metadata. The obtained results are listed in the Table 1.

![Figure 5: Dataset Split.](image)

![Figure 6: MAP score results using the MABPR algorithm. The first five bars are the results for the MABPR recommender algorithm using only one type of metadata, whereas the last three bars are the results for the proposed ensemble algorithms.](image)

![Figure 7: MAP score results using the BPR-Mapping algorithm. The first five bars are the results for the BPR-Mapping recommender algorithm using only one type of metadata, whereas the last three bars are the results for the proposed ensemble algorithms.](image)

Our results indicate the following: We were able to significantly improve the baseline results of using a single metadata in our work. The improvement level was between 1.5% and 7.2%. These improvements were significant as increasing the MAP is a difficult problem, and every increment in MAP is difficult to achieve. Surprisingly, the improvement level was similar among simpler and the complex models, with approximately 7% of improvement discarding the Tags metadata outlier in BPR-Linear algorithm as shown in Figure 8. The Weighting strategy generated the best recommendation for three of the four algorithms, and had the MABPR as the best algorithm to use. The values returned by the algorithms MABPR (Figure 6) and BPR-Mapping (Figure 7) are generally much better than those achieved by the other two algorithms. This is due to the fact that they are state-of-art recommender algorithms. They generated very similar results with a maximum MAP of 0.1838 for MABPR and 0.1803 for BPR-Mapping. On the other hand, the BPR-Linear and MostPopular (Figure 9) achieved a lower MAP, of 0.1510 and 0.1124, respectively. They are simpler algorithms and were used to analyze the ensemble behavior in different contexts.

Indeed, none of the evaluated ensemble method was optimal for all given scenarios. Consequently, one should look
Table 1: Algorithms MAP scores

<table>
<thead>
<tr>
<th>Metadata</th>
<th>MABPR</th>
<th>BPR-Mapping</th>
<th>BPR-Linear</th>
<th>MostPopular</th>
</tr>
</thead>
<tbody>
<tr>
<td>Genre</td>
<td>0.1671</td>
<td>0.1662</td>
<td>0.0190</td>
<td>0.0186</td>
</tr>
<tr>
<td>Tags</td>
<td>0.1704</td>
<td>0.1682</td>
<td>0.1486</td>
<td>0.1550</td>
</tr>
<tr>
<td>Directors</td>
<td>0.1687</td>
<td>0.1670</td>
<td>0.0303</td>
<td>0.0504</td>
</tr>
<tr>
<td>Actors</td>
<td>0.1675</td>
<td>0.1646</td>
<td>0.0254</td>
<td>0.0202</td>
</tr>
<tr>
<td>Countries</td>
<td>0.1671</td>
<td>0.1662</td>
<td>0.0250</td>
<td>0.1051</td>
</tr>
<tr>
<td>Most Pleasure</td>
<td>0.1695</td>
<td>0.1670</td>
<td>0.1444</td>
<td>0.1124</td>
</tr>
<tr>
<td>Best of All</td>
<td>0.1761</td>
<td>0.1729</td>
<td>0.1217</td>
<td>0.1081</td>
</tr>
<tr>
<td>Weighting</td>
<td>0.1838</td>
<td>0.1803</td>
<td>0.1510</td>
<td>0.0598</td>
</tr>
<tr>
<td>Improvement</td>
<td>7.2817%</td>
<td>7.1981%</td>
<td>1.5674%</td>
<td>6.8860%</td>
</tr>
</tbody>
</table>

Figure 9: MAP score results using the MostPopular algorithm. The first five bars are the results for the MostPopular recommender algorithm using only one type of metadata, whereas the last three bars are the results for the proposed ensemble algorithms.

for the (base model, ensemble) pair that achieves the best results for the dataset at hand. However, the Weighting ensemble strategy showed as the most effective on three of four scenarios and may be considered as a good candidate to implement in a real world scenario. This is because this strategy uses all metadata to make predictions, and it assigns different weights to the most relevant metadata according to the taste of each individual user. The only scenario in which Weighting did not returned the best results, with the MostPopular algorithm, may be explained by the fact revealed in recent studies from different recommender domains that popular items could highly dominate the recommendation performance[22]. As the most popular movies are often made in the U.S.A, the Countries metadata with the MostPopular algorithm just recommends the general popular movies, a combination what is known to produce an artificially high MAP.

While the Weighting strategy got promising results, the other two strategies should also be considered depending on the scenario. For instance, the MostPleasure strategy is the simplest and straightforward to implement, with a very low overhead as a probe run is not need. Moreover, it got a good performance improvement on the weaker algorithms, and almost did not affect negatively the more complex algorithms. Likewise, the Best of All did produce an even higher improvement, and although it needs a probe run, it do require the GA weight optimization, an expensive step in the process.

Considering only the metadata individually, the Tags is the metadata that returned the best recommendations for three of the four analyzed algorithms, and in BPR-Linear yielded a similar result compared to the ensemble algorithms. This is probably because the Tags contains a more diverse set of information, and, sometimes, may even simulate a combination of metadata. The tags referenced information such keywords, actors, genres, directors, producers. Recommending movies based on a combination of metadata, as seen in Beltrão [3], generates better combinations than with a single metadata.

Finally, we conclude that ensemble algorithms significantly improved the recommender prediction performance, with the Weighting strategy standing out with higher performance on most of the scenarios.

Additionally, the algorithm MABPR obtained the best for the tested data.

6. FINAL REMARKS

In this paper, we have presented a novel approach for combining multiple metadata in recommender systems. Our approach consisted of three different strategies that do not require modification of the recommender algorithm, namely Most Pleasure, Best of All and Genetic Algorithm Weighting. The considered recommender algorithms did not take advantage of multiple item metadata and our ensemble algorithm was able to enable those recommenders to take advantage of this metadata. Most Pleasure, the simplest strategy, consisted of combining the predictions based on score. Best of All determined a single metadata that was more preferred for a user, and finally the Weighting strategy uses multiple metadata and weights them with a Genetic Algorithm that optimizes the MAP.

Empirical evaluation showed a considerable MAP improvement between 1.5% and 7.2% when using the ensemble algorithms, with the Weighting strategy producing the best recommendation for the majority of scenarios. These encouraging results indicate that ensemble algorithms can be used to enhance the recommenders algorithms with multiple metadata.

As future work, we plan to implement more complex ensemble strategies and evaluate the algorithms with a higher number of metadata in order to verify whether multimodal information can generate better recommendations. In order to do so, it will be necessary to find a more extensive
dataset and to evaluate the algorithms runtime performance with this increased work.

7. REFERENCES


