A Framework for Training Hybrid Recommender Systems

Simon Bremer[1,2], Alan Schelten[2], Enrico Lohmann[2], Martin Kleinsteuber[1,2]
[1]Technical University of Munich
[2]Mercateo AG
{simon.bremer,kleinsteuber}@tum.de
{alan.schelten,enrico.lohmann}@mercateo.com

ABSTRACT

Recommender Systems (RS) are widely used to provide users with personalized suggestions taken from an extended variety of items. One of the major challenges of RS is the accuracy in cold-start situations where little feedback is available for a user or an item. Exploiting available user and item metadata helps to cope with this problem. We propose a hybrid training framework consisting of two predictors, a collaborative filtering instance and a metadata-based instance relying on content and demographic data. Our framework supports a wide range of algorithms to be used as predictors. The cross-training mechanism we design minimizes the weaknesses of one instance by updating its training with predicted data from the other instance. A sophisticated sampling function selects ratings to be predicted for cross-training

We evaluate our framework conducting multiple experiments on the MovieLens 100K dataset, simulating different scenarios including user and item cold-start. Our framework outperforms state-of-the-art algorithms and is able to provide accurate predictions across all tested scenarios.

ACM Reference format:
Simon Bremer[1,2], Alan Schelten[2], Enrico Lohmann[2], Martin Kleinsteuber[1,2]. 2017. A Framework for Training Hybrid Recommender Systems. In *Proceedings of RecSysKTL Workshop @ ACM RecSys '17, August 27, 2017, Como, Italy,* , 8 pages.
https://doi.org/N/A

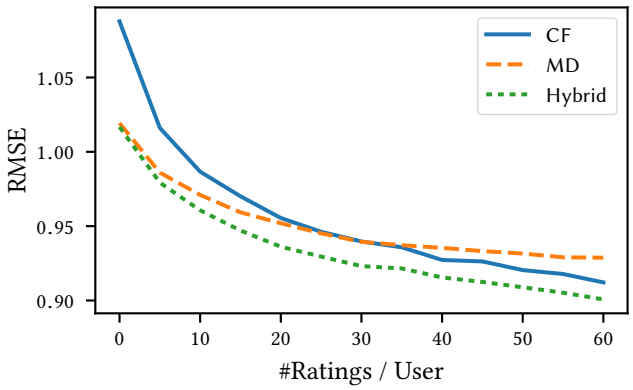

Figure 1: Root-mean-square error (RMSE) of Collaborative Filtering (CF), Metadata-based Filtering (MD) and our proposed Hybrid Framework on the MovieLens 100K dataset simulating user cold-start. In case users have no (0) ratings in the training set (complete user cold-start), MD clearly outperforms CF as predicted. Performance increases with more feedback available from users and at ~30 ratings CF starts to outperform MD on this dataset. Our hybrid framework outperforms both CF and MD in all cases. A detailed description of this experiment can be found in section 4.6.2.

1 INTRODUCTION

Recommender Systems (RS) are nowadays of tremendous importance across a multitude of different areas within the field of digital technology. While the amount of choice given to a user or customer has been growing continuously, users find it increasingly difficult to come up with a satisfactory selection without being overwhelmed by quantity.

Recommender systems help providers to offer users personalized suggestions in order to improve user experience. A number of different approaches are common to generate suggestions. The two most prominent and widely used methods are Collaborative Filtering (CF) and Content-based Filtering (CN) techniques. CF algorithms only take user feedback into consideration. Feedback can be given in an explicit (e.g. movie rating 1-5 stars) or implicit (e.g. user performed search query) way. CN algorithms mainly rely on known item properties to discover items similar to each other fitting the user's preference.

One major difficulty of CF and CN methods is that limited amount of feedback from some users (e.g. new/inactive) usually results in poor recommendations for those users. A user who has not given any feedback has not expressed any preference, therefore it is not possible to compute personalized recommendations. This phenomenon is called the cold-start (CS) or ramp-up problem. We distinguish between user cold-start (little feedback from specific users), item cold-start (little feedback to specific items) and system cold-start (both user and item cold-start). If information about users is available, Demographic Filtering (DM) can be applied and reasonable predictions can be made in case of user cold-start. Instead of processing feedback which is not yet available for a cold-start user, DM exploits user properties available through metadata to generate recommendations based on preferences of users with a similar demographic profile. In turn, CN helps to overcome item cold-start issues by taking item metadata into consideration. Alone however CN fails to deal with a user cold-start. As CN and DM both process metadata, algorithms often combine both methods if metadata is available for users and items. We dub the combination of CN

RecSysKTL Workshop @ ACM RecSys '17, August 27, 2017, Como, Italy

and DM Metadata-based Filtering (MD). While MD algorithms are able to produce reasonable results in case of user or item cold-start and can deal with a system cold-start as well, experiments have shown that CF usually outperforms MD as soon as some amount of feedback has been given (see Figure 1).

In order to archive optimal overall performance it is necessary to combine the results of both CF and MD. The challenge of designing hybrid systems is subject to ongoing research. We present a framework to combine the advantages of both algorithms.

1.1 Our Contribution

We contribute a framework for training a hybrid CF-MD model consisting of one instance of each, CF and MD. The framework utilizes cross-training, an approach using prediction results of one instance to train the other and vice versa. We design a sophisticated sampling method to select specific user/item interactions for cross-training. This enables us to increase the overall performance of both predictors over entries with little as well as a with lot of feedback. Our method is able to cope with cold-start, a major drawback of the popular CF approach. At the same time prediction results for entries with lots of feedback also improve.

We test our approach on the MovieLens [5] dataset and were able to outperform baseline algorithms as well as previously published results of similar methods.

1.2 Related Work

One of the most popular CF techniques in recent research is the latent feature based matrix factorization (MF). Koren [9] provides an overview covering some extensions to basic factorization. Many additional extensions have been published which improve results or take more data into consideration such as rating timestamps [8] or metadata.

We distinguish between CF and MD methods and use them in a divided fashion in our framework. Other researchers have tried to include both approaches into a single model. Manzato et al. [11] proposed additional latent features for categorical item metadata. Santos et al. [14] outperformed Manzato's model by including biases representing preferences of certain groups of users (e.g. age group) to specific item attributes. Zhang et al. [15] formulated additional combinations of offsets describing a user's taste for genre or a user group's taste for a specific movie, used alongside matrix factorization.

Most factorization models are covered by the model class of Factorization Machines (FM) [13] introduced by Rendle. FM can implement CF while also taking user and item metadata into consideration acting as a one-model hybrid. While implementations of FM like LightFM [10] provide accurate results during normal operation as well as cold start, our CF / MD models can be specifically tuned to perform well in their main area of operation (normal / cold-start).

The technique of merging multiple individual algorithms or their results is an important challenge in RS research. Taking the average or weighted average of results is the simplest ensemble method available. Jahrer et al. [6] presented more sophisticated approaches of merging results from multiple CF algorithms like linear regression, neural networks, decision trees and even stacked multiple merging techniques. Ensemble methods to merge results are just one method of building hybrid systems. Burke [2] has named multiple other possible techniques like cascading results from one model into another to refine the ranking of items.

Another approach of building a hybrid system was presented by Zhang at al. [15]. They built multiple models using different MF extensions or used bagging on the training set to train multiple instances of the same algorithm. We adapt Zhang's method of cross-training (co-training), where predictions of one model are used to train another and vice versa. To pick samples used for cross-training, a confidence is calculated based on the number of ratings per user/item as well as metadata stats like the number of users per gender or age-group. We will describe confidence measures and how we refine cross-training sampling in detail in Section 2.2.

While most contributions addressed until now measure the error of all predictions, other researchers focus on ranking the predicted items and evaluating the ranks. Park et al. [12] used a feature vector matrix multiplication as well as a custom pair-wise loss function to tackle the cold-start problem. Their evaluation only takes ranked recommendations into consideration.

2 HYBRID TRAINING FRAMEWORK

In this section we describe the functionality of our framework and explain cross-training as well as the selection and generation of data for cross-training.

The basic scenario for an RS incorporates a set of users $\mathcal{U}$ and a set of items $\mathcal{I}$. The rating r_{ui} with $u \in \mathcal{U}, i \in \mathcal{I}$ refers to the explicit rating given from a user u to an item i. We call ui an index pair or tuple. All possible tuples are contained in the set $\mathcal{L} = \mathcal{U} \times \mathcal{I}$. The set $\mathcal{K} \subset \mathcal{L}$ consists only of index pairs $ui \in \mathcal{K}$ for which the ground truth of rating r_{ui} is available in the training set. The complement of the training index-set $\mathcal{K}$ denotes all tuples not known: $\overline{\mathcal{K}} = \mathcal{L} \setminus \mathcal{K}$. The training set is defined as: $\mathcal{T} = \{(ui, r_{ui}) | ui \in \mathcal{K}\}$.

Our hybrid framework combines an instance of a CF algorithm as well as an MD algorithm. Both models act as a regression function $\hat{r} : \mathcal{L} \mapsto \mathbb{R}$, giving their prediction $\hat{r}_{ui}$ for an index pair ui.

The framework is universal in the way that both algorithms, CF and MD are interchangeable and almost any regression approach relying on training through labeled data can be inserted. We specify our choice of algorithms and further beneficial properties of possible algorithms in Section 3.

2.1 Outline of the Training Procedure

Step 0: Initial training

Step 1: Predict ratings for index pairs from $\mathcal{S}_{CF}(\overline{\mathcal{K}})$ to construct the teaching set.
Update training of MD using the teaching set and the original training set.

Step 2: Predict ratings for index pairs from $\mathcal{S}_{MD}(\overline{\mathcal{K}})$ to construct the teaching set.
Update training of CF using the teaching set and the original training set.

Step 3: If cross-training stopping criterion is not yet reached, go back to Step 1 and repeat cross-training iteration

2.1.1 *Initial Training.*

To ensure that reasonable results can be predicted to be used in

cross-training, the first step is separate initial training. Both models are individually trained with the training dataset $\mathcal{T}$.

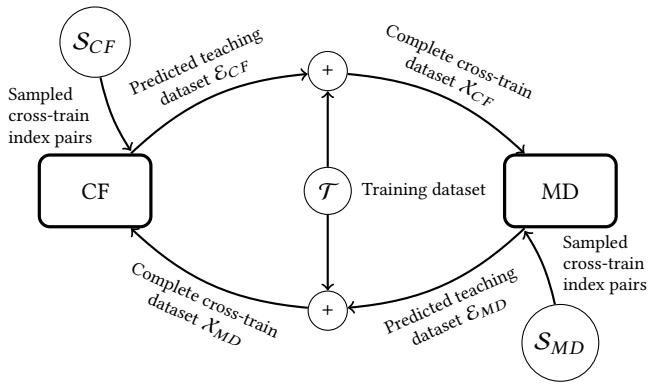

Figure 2: The mechanism of cross-training. The two models CF and MD are trained with a combination of the training data and teaching data predicted by the other model. $\mathcal{S}_{CF}$ and $\mathcal{S}_{MD}$ are sampling functions and supply a set of ui index tuples which are used to predict ratings for the teaching set.

2.1.2 *Cross-Training.*

The mechanism of cross-training shown in Figure 2 is designed to improve each model by training it with ratings predicted by the other model. We now describe the process of the first cross-training step (upper half of Figure 2).

First, we draw a number of index pairs from $\overline{\mathcal{K}}$ which are not included in the training set. This is done by the sampling function $\mathcal{S}_{CF}(\overline{\mathcal{K}}) \subset \overline{\mathcal{K}}$ which generates a subset of the unknown index tuples which are to be used for cross-training. As the choice of which and how many tuples are selected from $\overline{\mathcal{K}}$ is vital for the functioning of our whole framework, we provide a detailed description in Section 2.2.

To build the teaching data, a rating $\hat{r}_{ui}$ is predicted for each tuple of the cross-training index set generated by $\mathcal{S}_{CF}$. All predicted ratings and their corresponding index pairs are called the teaching dataset: $\mathcal{E}_{CF} = \left\{(ui, \hat{r}_{ui}^{CF}) | ui \in \mathcal{S}_{CF}(\overline{\mathcal{K}})\right\}$. Teaching data and original training data are concatenated and the resulting cross-training data $\mathcal{X}_{CF} = \mathcal{E}_{CF} \cup \mathcal{T}$ is then used to update the training of MD.

The second cross-training step, using data predicted from MD to train CF (lower half of Figure 2), works in the same manner. Both steps are alternated until a predefined stopping criterion is reached.

We found that a fixed number of cross-training epochs works well. See Section 4.6.3 for an analysis of performance depending on number of epochs.

2.2 Cross-Training Sampling

The choice of how many and especially which user/item tuples are selected for cross-training is an essential part of our framework. We will now explain how we design our sampling functions.

The underlying idea of cross-training is to use the advantages of one algorithm to decrease the impact of weaknesses of the other. Our two regression models do not provide a confidence for a predicted rating. Still, for each index tuple, we can take an educated guess whether CF or MD yield a better prediction. In the case of CF, results are poor in case of cold-start, therefore we can assume that for users and items with few ratings MD will outperform CF. In this case, providing additional training data for cold-start users and items will increase the performance of CF. Vice versa CF is able to predict high quality ratings for users and items with a lot of feedback. Cross-training additional ratings to MD will also increase its overall performance.

Selecting ratings for which one algorithm most likely outperforms the other and which will provide the greatest performance gain is the key part of our framework.

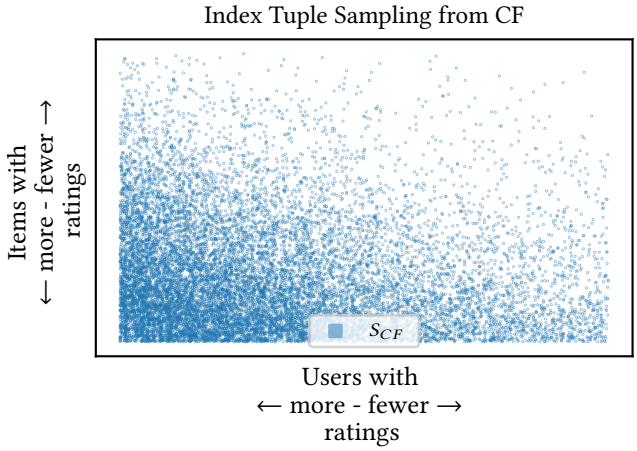

Figure 3: This plot shows the selection of index tuples for cross-training according to users and items. Users (rows) and items (columns) are sorted by ratings. The blue markers, representing index pairs sampled by $\mathcal{S}_{CF}$, concentrate on users/items with many ratings.

2.2.1 *Sampling Index Tuples for Predictions of CF (Figure 3).*

To draw index pairs from $\mathcal{S}_{CF}$ we adopt the method proposed by Zhang at al. [15]. For the selection of index pairs, Zhang introduced a confidence measure $C(\hat{r}_{ui})$ to estimate how accurate a predicted rating will be. In its simplest form it depends on the product of the number of ratings d_u in the training set given by user u and number of ratings d_i, received by item i with a normalization term $\mathcal{N}$:

$$C(\hat{r}_{ui}) = \frac{d_u \times d_i}{\mathcal{N}} \tag{1}$$

Predicting the rating $\hat{r}_{ui}$ for a user u who has rated often and an item i which has been rated often will result in a high confidence using this measure. This reflects the experience that CF generates better predictions as soon as more feedback is available. Zhang builds a probabilistic distribution based on the confidence measure:

$$\mathcal{P}(u, i) = \frac{C(\hat{r}_{ui})}{\sum_{(u', i') \in \overline{\mathcal{K}}} C(\hat{r}_{u'i'})} \tag{2}$$

We use this distribution in $\mathcal{S}_{CF}$ and sample index pairs from $\overline{\mathcal{K}}$ without replacement meaning no duplicates are in the set of

sampled index tuples. Regarding the number of selected samples, we determine the number of cross-training samples depending on the size of $\mathcal{K}$ and the factor δ:

$$|\mathcal{S}_{CF}(\overline{\mathcal{K}})| = \lfloor \delta |\mathcal{K}| \rfloor \tag{3}$$

Through cross-validation we found an optimal value $\delta = 0.15$ for the MovieLens 100K dataset.

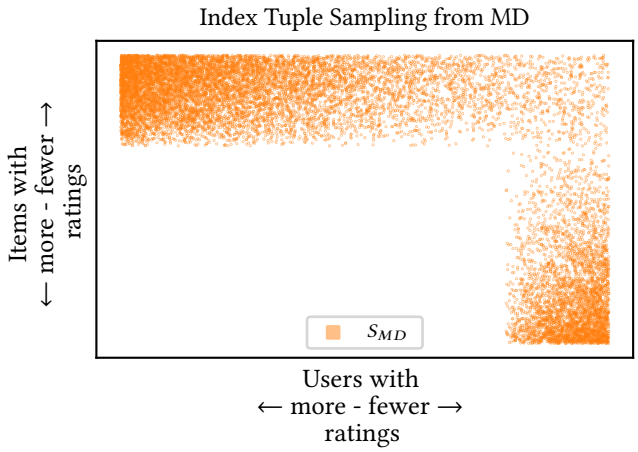

Figure 4: This plot shows the selection of index tuples for cross-training like in Figure 3. Orange markers show tuples from $\mathcal{S}_{MD}$, located mainly in the area of cold-start users but frequently rated items and vice versa. We design the function which samples index pairs to be predicted by MD to take tuples from areas where either the user has few ratings or the item has few ratings, not both. Predictions by MD for tuples where both users and items have very few ratings (system cold-start) are less accurate and therefore not as useful cross-training.

2.2.2 Sampling Index Tuples for Predictions of MD (Figure 4). As described before, CF performs poorly for users and items with little feedback. We will design $\mathcal{S}_{MD}$ in a way that cross-training index pairs are selected specifically among these cold-start users and items. This technique enables us to significantly improve the performance of CF through cross-training.

Experiments have shown that MD produces better results than CF for users with fewer than a certain threshold of ratings (see Figure 1 and Section 4.6.2). We select index tuples for cross-training among these users with fewer than t_{user} ratings. Sampling is done individually for each user to ensure a minimum number of ratings in the complete cross-train dataset for all users. For each user u we select $\lfloor \epsilon_{user} \cdot max(0, t_{user} - d_u) \rfloor$ tuples, where the factor $\epsilon_{user} \in \mathbb{R}_+$ controls the amount of tuples selected for this user. d_u stands for the number of ratings by user u in the training dataset. The corresponding items i sampled for the cross-training tuples are selected randomly but giving a higher probability to items with a high amount of feedback. This is done in a linear fashion according to the number of ratings of an item i with a probability $P(i) = \frac{d_i}{\mathcal{N}}$ where $\mathcal{N}$ normalizes the term to a sum of 1 and d_i refers to the number or ratings given to item i in the training set.

In the same manner we select additional tuples for each item for which the a number of training ratings is below threshold t_{item}. We then go on to select $\lfloor \epsilon_{item} \cdot max(0, t_{item} - d_i) \rfloor$ tuples for each item, again using a factor $\epsilon_{item} \in \mathbb{R}_+$. The distribution to pick corresponding users u for the tuples is $P(u) = \frac{d_u}{\mathcal{N}}$ again with normalization term $\mathcal{N}$.

Tuples for cold-start users are selected favoring items with many ratings contained in the training set over others through probability $P(i)$. This is done deliberately since the prediction of MD is more reliable if at least the item was rated, in contrast to the case in which neither user nor item were rated at all. The same logic applies to cold-start items where tuples with active users are preferred.

Again the tuples are selected without replacement, avoiding redundant index pairs in the teaching set. The total amount of cross-training samples from MD is therefore:

$$\begin{aligned} |\mathcal{S}_{MD}(\overline{\mathcal{K}})| = &\sum_{u \in \mathcal{U}} \lfloor \epsilon_{user} max(0, t_{user} - d_u) \rfloor \\ &+ \sum_{i \in \mathcal{I}} \lfloor \epsilon_{item} max(0, t_{item} - d_i) \rfloor \end{aligned} \tag{4}$$

latex The hyperparameters t_{user}, t_{item}, ϵ_{user} and ϵ_{item} of our framework are determined through cross-validation.

3 SPECIFIC MODELS

This section contains a detailed description of the algorithms we choose to use for the individual models in our experiments. Some constraints apply to algorithms in order to be used as part of our framework. Both models must act as regression functions, predicting a value referring to the explicit feedback based on the index tuple (u, i). Additional metadata about user and item is used by MD. Furthermore algorithms which offer the functionality of iterative training to update their parameters do not require complete retraining during each cross-training epoch. Applying iterative updates during cross-training instead of retraining the model greatly speeds up the process. We constructed both algorithms using stochastic gradient descent as optimizer and employ iterative updates.

3.1 Collaborative Filtering

CF is an often used method in RS and is therefore subject to continuous research. We use SVD++ (Koren [9]), which is a matrix factorization (MF) algorithm that has been proven to perform well.

The MF approach for CF was popularized by Koren and Webb (under pseudonym of Simon Funk) [1] during the Netflix challenge. Both users and items are represented by latent feature vectors $q_i, p_u \in \mathbb{R}^k$ of predefined dimensionality k. Values between 5 and 100 are frequently used for k in research. A higher dimensionality of latent features is able to represent a higher complexity of underlying patterns of user preference while also increasing the risk of overfitting.

In its simplest form, the inner product is used to predict the explicit rating given by user u to item i: $\hat{r}_{ui} = q_i^T p_u$. Including biases, like global rating average μ and offsets for user b_u and item b_i, has shown to improve the results. Other extensions to the basic MF have already been mentioned in Section 1.2. The SVD++ algorithm proposed by Koren incorporates additional latent vectors

$y_j \in \mathbb{R}^k, j \in \mathcal{I}$ into the factorization. For the set of items $N(u)$ which received implicit feedback by a user u, we compute the sum of these vectors. The complete SVD++ prediction:

$$\hat{r}_{ui} = \mu + b_u + b_i + q_i^T \left(p_u + |N(u)|^{-\frac{1}{2}} \sum_{j \in N(u)} y_j \right) \tag{5}$$

For lots of implicit feedback, the impact of $|N(u)|^{-\frac{1}{2}} \sum_{j \in N(u)} y_j$ increases since more entries are in $N(u)$ and because the normalization uses the square-root. The dataset we use in our experiments does not contain implicit feedback, just explicit ratings. We therefore define that giving a rating can also be considered as implicit feedback. This method is also used by Koren. That way of measuring implicit feedback is, in a way, redundant information, as ratings are already trained. Yet SVD++ has been proven to outperform normal MF [7] not including implicit feedback. We were able to further increase the performance by excluding low ratings (≤ 3 out of 5 on MovieLens 100K) from implicit feedback. This threshold also applies to ratings predicted by MD during cross-training.

All model parameters q_i, y_i and b_i for all $i \in \mathcal{I}$, p_u and b_u for all $u \in \mathcal{U}$ as well as the global average μ are learned by solving the least-squares equation through gradient descent. Regularization is applied to all parameters except μ. We use different regularization values for different parameters:

$$\begin{aligned} \min_{\mu, b_*, q_*, p_*, y_*} & \sum_{(u,i) \in \mathcal{K}} (r_{ui} - \hat{r}_{ui})^2 + \lambda_1 (b_u^2 + b_i^2) \\ & + \lambda_2 \left(||q_i||^2 + ||p_u||^2 + \sum_{j \in N(u)} ||y_j||^2 \right) \end{aligned} \tag{6}$$

A detailed display of gradient descent update rules for parameters can be found in [7].

3.2 Metadata-based Filtering

To cope well with cold-start, relying on known ratings from the training set alone does not suffice, as no personalized predictions can be made for users who have not given any feedback yet. In this case available metadata describing users and items enables an RS to produce reasonable results. We design a custom MD model consisting of a combination of different biases which are motivated by possible causal links.

We assume that all users and items are described by sets of attributes. Examples for such attributes from the MovieLens dataset (Section 4.2) are gender, age and occupation of users. Gender and occupation can be considered categorical fields, those fields have a discrete value in our dataset. To be able to represent continuous fields like age as a set of possible attributes, values are discretized into age groups, each group representing a possible attribute. The set of attributes of a user u is called $D(u)$ (Example: $D(u) = \{is\_female, is\_technician, is\_25-34y/o\}$).

The same feature preprocessing applies to items. Continuous data has to be discretized, categorical fields are simply one-hot encoded into possible attributes. The set of attributes of item i is denoted by $C(i)$. The used MovieLens dataset provides a classification of movies into genres. $C(i)$ will therefore contain one or more genres as attributes (Example: $C(i) = \{action, adventure, romance\}$). In this section we will refer to an item attribute as genre since the dataset provides genres, yet item attributes are not limited to that.

To model the preference of a user accurately and to come up with a reasonable prediction for $\hat{r}_{ui}$ we combine a number of biases and factors to exploit multiple causal links:

3.2.1 User and Item Bias. The global average and user/item biases are the same as for CF and represents a regularized version of average rating of a user and an item:

$$b_{ui} = \mu + b_u + b_i \tag{7}$$

3.2.2 Attribute Bias. For every attribute $d \in D(u)$ and $c \in C(i)$ we apply a bias which states whether a group of users sharing an attribute d rates better or worse than average. The bias works the same for items with attribute c. Normalization is applied according to the number of attributes per user and item.

$$b_{attrib} = \frac{1}{|D(u)|} \sum_{d \in D(u)} b_d + \frac{1}{|C(i)|} \sum_{c \in C(i)} b_c \tag{8}$$

3.2.3 User Attribute to Item Attribute. Now we look at the combination of attributes describing user and item. We introduce a weight h_{cd} corresponding to an offset in the ratings given by users with attribute $d \in D(u)$ for items with attribute $c \in C(i)$ w.r.t. global average μ. An example would be that users with attribute *is_male* tend to rate movies with attribute *action* a bit higher than the global rating average μ. The additional factors g_c for genres and f_d for user attributes can adjust the impact on h_{cd} depending on user attribute and genre:

$$\hat{r}_{ui}^{UA \to IA} = \frac{1}{|C(i)|} \frac{1}{|D(u)|} \sum_{c \in C(i)} g_c \sum_{d \in D(u)} f_d h_{dc} \tag{9}$$

3.2.4 User Attribute to Item. The next part of the prediction models how a homogeneous audience responds to an individual item. "Toy Story" for example might be especially liked among users of age group 0-15. The weight k_{id} corresponds to the rating offset of users with attribute d to the specific item i:

$$\hat{r}_{ui}^{UA \to I} = \frac{1}{|D(u)|} \sum_{d \in D(u)} k_{di} \tag{10}$$

This term works well in the case of user cold-start because no ratings from the current users are required to learn k_{id}, only demographically similar users.

3.2.5 User to Item Attribute. The mapping from user to genre works respectively. Weight l_{cu} models how a specific user u rates items with an attribute c:

$$\hat{r}_{ui}^{U \to IA} = \frac{1}{|C(i)|} \sum_{c \in C(i)} l_{uc} \tag{11}$$

This part of the prediction copes well in the case of item cold-start as it does not matter whether the specific item received feedback.

The term $\hat{r}_{ui}^{UA \to IA}$ is also used by Santos et al. [14]. Both $\hat{r}_{ui}^{UA \to I}$ and $\hat{r}_{ui}^{U \to IA}$ are also used in a similar fashion by Zhang et al. [15].

The complete expression used for the MD model is:

$$\hat{r}_{ui} = b_{ui} + b_{attrib} + \hat{r}_{ui}^{UA \to IA} + \hat{r}_{ui}^{UA \to I} + \hat{r}_{ui}^{U \to IA} \tag{12}$$

The loss function to be minimized by optimization includes the least squares of errors as well as regularization applying to most parameters:

$$\begin{aligned}
&\min_{\mu, b_*, h_*, f_*, g_*, k_*, l_*} \sum_{(u,i)\in\mathcal{K}} (r_{ui} - \hat{r}_{ui})^2 \\
&+ \lambda_1 \left(b_u^2 + b_i^2 + \frac{1}{|D(u)|} \sum_{d\in D(u)} b_d^2 + \frac{1}{|C(i)|} \sum_{c\in C(i)} b_c^2 \right) \\
&+ \lambda_2 \left(\frac{1}{|C(i)|} \frac{1}{|D(u)|} \sum_{c\in C(i)} \sum_{d\in D(u)} h_{dc}^2 \right. \\
&\qquad \left. + \frac{1}{|D(u)|} \sum_{d\in D(u)} k_{di}^2 + \frac{1}{|C(i)|} \sum_{c\in C(i)} l_{uc}^2 \right)
\end{aligned} \tag{13}$$

3.3 Combining Results

We choose a simple thresholding method to predict ratings by the hybrid system. Hybrid results are taken from CF, except if $\hat{r}_{ui}$ corresponds to a user or an item with very few ratings in which case predictions are taken from MD. We define fixed the cutoff thresholds c_{user} and c_{item} as hyperparameters. Predictions for users and items with fewer than c_{user}/c_{item} ratings are taken from MD.

4 EXPERIMENTS

We test our hybrid framework in a number of experiments to evaluate its performance. The optimal hyperparameters used in our evaluation are determined through cross-validation. Thresholding parameters like t_{user} are selected through observation of experiments like the one conducted in Section 4.6.2.

4.1 Implementation

We implement our model using Python and Keras [3]. Our implementation is published as an open source project and is available on GitHub[3]. To implement our individual models and baselines we use Keras' Functional API[4]. The optimizer we configure Keras to use is Adagrad [4] as it has proven to provide superior results in our application over other optimizers Keras offers. Additional details about the technical implementation are included as in-code documentation.

We use Keras with Theano[5] as backend and were able to significantly speed up our computation by utilizing the GPU. We are able to run a complete 5-fold cross-validation of our hybrid model in under 6 minutes (using Intel® Core™ i7-2760QM / NVIDIA® Quadro® 2000M on Linux).

All ratings (1-5) are linearly transformed to a numerical range of (0.1-0.9) to represent the likelihood for a user to like an item.

It is common practice to take certain precautions to avoid overfitting. We have already described the use of regularization. In addition to this we use early stopping for the initial training. By using a small part of the available data as validation set, not including it in the training set $\mathcal{K}$, and stopping training as soon as a minimum in loss is reached in the validation set. This is accomplished by storing the learned parameters if the validation loss reaches a new minimum, stopping if no new minimum is reached after n epochs (~5) and restoring the saved parameters as they yield the best validation performance.

4.2 Dataset

To test and evaluate our proposed method we use the MovieLens 100K[6] dataset provided by GroupLens [5]. The dataset consists of 100,000 ratings ranging from 1(worst)-5(best). Ratings are given by 943 users to 1682 movies. The resulting sparsity is 93.70%. All users of both sets provided at least 20 ratings, some items have received fewer. Timestamps are included for every rating. We omit those as our research focus does not lie on time dependency. Movies are described by title and a classification into one or multiple of the 19 genres. Metadata of users consists of gender, age and one of 21 occupations. As we work with categorical data, we group the continuous age of the dataset into 7 discretized age groups. Zip codes of users' residences are omitted as well.

The metadata included in the dataset is not very extensive, therefore we cannot expect MD to outperform baselines by a large margin. Gathering more data as well as sophisticated feature engineering can improve results but is often difficult and costly. As the focus of this research lies on our training framework we only use the data provided in the MovieLens dataset.

4.3 Baseline Algorithms

We compare our hybrid algorithm against a number of baseline algorithms:

4.3.1 BiasBaseline. This approach consists only of global, user specific and item specific bias. The prediction term is: $\hat{r}_{ui} = b_{ui}$ with b_{ui} from Section 3.2.1.

4.3.2 SVD. SVD is a common MF approach also incorporating biases. The dot product of an item specific and a user specific latent feature vector is used to predict ratings: $\hat{r}_{ui} = b_{ui} + q_i^T p_u$

4.3.3 Metadata. Our approach processing metadata is described extensively in Section 3.2. We use this model in a standalone mode as a baseline algorithm in order to show that our framework outperforms its components individually.

4.3.4 SVD++. This corresponds to the standalone version of the CF model also used in our hybrid framework. See Section 3.1 for further reference.

4.4 Evaluation Metrics

We measure the performance of our results using two evaluation metrics.

4.4.1 RMSE. The root-mean-square error (RMSE) of all predicted ratings $\hat{r}_{ui}$ of the test set is given as:

$$RMSE = \sqrt{\sum_{ui\in TestSet} \frac{(r_{ui} - \hat{r}_{ui})^2}{|TestSet|}} \tag{14}$$

[3] github.com/sbremer/hybrid_rs
[4] keras.io/getting-started/functional-api-guide/
[5] deeplearning.net/software/theano
[6] grouplens.org/datasets/movielens/100k

4.4.2 Precision@k. Precision, representing a ranking based measurement, is calculated for each user separately. Ratings are predicted for all index tuples of the test dataset of a user. Predicted ratings are then sorted by value and the k items with the highest predicted score are considered actual suggestions. The calculated precision represents the portion of "good" suggestions among those which were predicted top k items. We consider an item recommendation "good" if that item is also among the true top ratings of the user.

Note that the dataset's ground truth contains only integer values as ratings. The true top ratings of a user are determined by choosing the top k ratings. As more than k items can receive a top rating by the user, we also include all tying ratings in the set of true top ratings. This means that the true top ratings for a user may contain more than k items as a user can rate more than k items with a top rating.

4.5 Evaluation Method

All given scores are the mean of results using k-fold cross-validation with $k = 5$. We use 3 different fold methods to evaluate different situations:

4.5.1 Normal k-fold. The standard k-fold function splits the ratings into five parts and cross-validates those. As every user in the dataset has at least 20 ratings and 80% of the ratings are assigned to the training set, it is very likely that every user has at least 10 ratings in the training set. This situation corresponds to a normal test run, without complete cold-start.

4.5.2 User Cold-Start k-fold. This evaluation randomly splits the data by users not by ratings. One fifth of the users and all their given ratings will be used as testing data while the rest is given to the algorithm as training data. Representing complete user cold-start, no ratings of any of the users of the test dataset are available for training.

4.5.3 Item Cold-Start k-fold. To simulate a complete item cold-start, we split all items, taking 80% of items with all corresponding ratings as training set. The test set consists of the last fifth of items and their ratings. The algorithm being evaluated has not seen any ratings of the items in the test set.

4.6 Results

We conduct multiple experiments to evaluate the performance and advantages of our hybrid framework. We include results of multiple baseline algorithms described above, including our CF and MD algorithms in standalone mode. For those baselines, we measure the score of the test dataset after initial training, before applying the proposed cross-training method.

4.6.1 Non and Complete Cold-Start (Table 1). We now show our results for the 3 evaluation folding methods.

The results of competing research are not listed in Table 1 as different evaluation methods were used. In addition other methods do not consider complete user and item cold-start. Zhang et al. [15] evaluated results using a 10-fold cross-validation yielding a best RMSE on MovieLens 100K of 0.8966. We also evaluated our algorithm with 10 folds, outperforming Zhang and achieving an RMSE of 0.8908. Santos et al. present a best RMSE of 0.9123.

	Normal		User CS		Item CS	
Model	RMSE	prec@5	RMSE	prec@5	RMSE	prec@5
BiasBaseline	0.9417	0.7483	1.0212	0.5559	1.0742	0.6035
SVD	0.9246	0.7646	1.0343	0.5538	1.0680	0.6035
Metadata	0.9265	0.7584	1.0173	0.5552	**1.0517**	0.6610
SVD++	0.8996	0.7786	1.0856	0.5317	1.1335	0.6035
Hybrid	**0.8962**	**0.7809**	**1.0169**	**0.5586**	1.0563	**0.6622**

Table 1: Results of the 3 proposed types of 5-fold cross-validation of our hybrid algorithm and baselines on MovieLens 100K.

4.6.2 Different Intensities of User Cold-Start (Figure 1). In addition to our three different cross-validation splits, we tested our approach in case of user cold-start with varying intensity. To do so, we use our user cold-start validation, splitting users into five parts and using one part as the testing dataset. This mode represents a complete cold-start as no ratings of the test set users are in the training set. To simulate cold-start with few, instead of no ratings given by test set users, we randomly take a fixed number of ratings from every user in the test set and include these ratings in the training set. The created situation models users who have given some ratings.

We can observe that for no or very few ratings (fewer than ~30) included in the training set, MD outperforms CF. With more ratings, CF is able to produce more accurate predictions than MD. Our proposed hybrid framework outperforms both its individually trained components, CF and MD, in all tested cases of user cold-start proving that cross-training improves accuracy significantly.

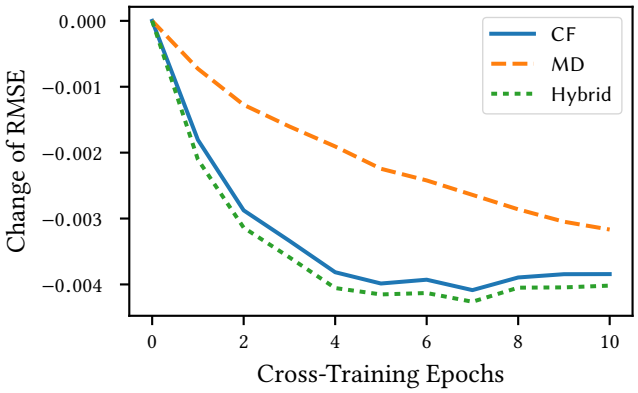

Figure 5: Individual absolute change of root-mean-square error (RMSE) of CF, MD and hybrid framework during cross-training with respect to the performance before cross-training.

4.6.3 Performance of Hybrid Framework by Cross-Training Epochs (Figure 5). We also evaluate how the performance of both framework components, CF and MD, and the combined hybrid results

change when applying cross-training. Figure 5 shows the absolute change of error compared to the results after initial training (epoch zero). This evaluation shows the mean values of a 5-fold cross-validation.

While initially results of both CF and MD improve significantly, a minimum of CF's error is reached after about 7 epochs. After that we observe an increase of error in the results of CF and our hybrid system. Early stopping of cross-training is implemented to abort at a minimum. We take runtime of the algorithm into consideration when determining the epoch after which to stop, resulting in a trade-off between best improvement through cross-training and lower runtime of the framework.

4.7 Discussion

Our goal was to design a hybrid framework which combines the desirable properties of CF and MD, to be able to provide accurate recommendations over a variety of scenarios including cold-start. We tested our algorithm on the MovieLens 100K dataset and compared the results to a number of baseline algorithms, including our metadata-based approach, in standalone mode.

While the proposed framework is able to yield only slightly more accurate results in case of user cold-start, we clearly outperform all baselines during normal operation, including the frameworks component algorithms. During item cold-start the framework's results are comparable to those of standalone MD, outperforming all other baselines.

In real world application we can expect a mixture of users/items with a broad range of number of ratings per user and item as well as a constant growth of user and item base. While both CF and MD alone show clear weaknesses in some situations, our hybrid approach is superior or at least comparable over all possible cases. We therefore expect a considerable gain in overall performance.

Note that the MovieLens dataset provides little information about users and especially about items. We anticipate that additional and more meaningful metadata will increase the performance of our MD algorithm and therefore the complete framework too.

5 CONCLUSION AND FUTURE WORK

In this work we proposed a hybrid training framework incorporating two RS algorithms, one implementing Collaborative Filtering and one relying on metadata of users and items. Through cross-training we were able to combine the advantages of the individual approaches. Our algorithm is able to predict accurate ratings in situations where users/items have few as well as many ratings. We are able to tackle the cold-start problem while also improving performance in the case of many ratings being available.

Possible future work includes testing our algorithm on different datasets, including e-commerce data from the Mercateo platform. As many components of our framework are interchangeable, we will evaluate different sampling functions as well as other algorithms for CF and MD. In addition we will take steps preparing deployment in online mode, including more evaluation using ranking-based metrics.

6 ACKNOWLEDGEMENTS

This work has been supported by the European Regional Development Fund and the Free State of Saxony via project no. 100272023 *Visual Analytics Interfaces for Big Data Environments - VANDA*.

REFERENCES

[1] B. Webb (aka S. Funk). 2006. Netflix Update: Try This at Home. http://sifter.org/~simon/journal/20061211.html. (2006).

[2] Robin Burke. 2007. Hybrid web recommender systems. *The adaptive web* (2007), 377–408.

[3] François Chollet and others. 2015. Keras. https://github.com/fchollet/keras. (2015).

[4] John Duchi, Elad Hazan, and Yoram Singer. 2011. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of Machine Learning Research* 12, Jul (2011), 2121–2159.

[5] F. Maxwell Harper and Joseph A. Konstan. 2015. The MovieLens Datasets: History and Context. *ACM Trans. Interact. Intell. Syst.* 5, 4, Article 19 (Dec. 2015), 19 pages. https://doi.org/10.1145/2827872

[6] Michael Jahrer, Andreas Töscher, and Robert Legenstein. 2010. Combining predictions for accurate recommender systems. In *Proceedings of the 16th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 693–702.

[7] Yehuda Koren. 2008. Factorization Meets the Neighborhood: A Multifaceted Collaborative Filtering Model. In *Proceedings of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD '08)*. ACM, New York, NY, USA, 426–434. https://doi.org/10.1145/1401890.1401944

[8] Yehuda Koren. 2010. Collaborative filtering with temporal dynamics. *Commun. ACM* 53, 4 (2010), 89–97.

[9] Yehuda Koren, Robert Bell, and Chris Volinsky. 2009. Matrix Factorization Techniques for Recommender Systems. *Computer* 42, 8 (Aug. 2009), 30–37. https://doi.org/10.1109/MC.2009.263

[10] Maciej Kula. 2015. Metadata embeddings for user and item cold-start recommendations. *arXiv preprint arXiv:1507.08439* (2015).

[11] Marcelo Garcia Manzato. 2013. gSVD++: Supporting Implicit Feedback on Recommender Systems with Metadata Awareness. In *Proceedings of the 28th Annual ACM Symposium on Applied Computing (SAC '13)*. ACM, New York, NY, USA, 908–913. https://doi.org/10.1145/2480362.2480536

[12] Seung-Taek Park and Wei Chu. 2009. Pairwise Preference Regression for Cold-start Recommendation. In *Proceedings of the Third ACM Conference on Recommender Systems (RecSys '09)*. ACM, New York, NY, USA, 21–28. https://doi.org/10.1145/1639714.1639720

[13] Steffen Rendle. 2010. Factorization machines. In *Data Mining (ICDM), 2010 IEEE 10th International Conference on*. IEEE, 995–1000.

[14] Edson B. Santos Junior, Marcelo G. Manzato, and Rudinei Goularte. 2013. Hybrid Recommenders: Incorporating Metadata Awareness into Latent Factor Models. In *Proceedings of the 19th Brazilian Symposium on Multimedia and the Web (WebMedia '13)*. ACM, New York, NY, USA, 317–324. https://doi.org/10.1145/2526188.2526197

[15] Mi Zhang, Jie Tang, Xuchen Zhang, and Xiangyang Xue. 2014. Addressing Cold Start in Recommender Systems: A Semi-supervised Co-training Algorithm. In *Proceedings of the 37th International ACM SIGIR Conference on Research & Development in Information Retrieval (SIGIR '14)*. ACM, New York, NY, USA, 73–82. https://doi.org/10.1145/2600428.2609599