Hybrid collaborative recommendations: Practical considerations and tools to develop a recommender

1 Introduction

The diversity as a concept has been proved by the evolution to be very successful. The researches follow this idea in many modern approaches. The ensemble methods in the machine learning often outperform standalone methods. In the context of the recommender systems, similar idea is used as so-called hybrid recommenders.

As a rule, two standard recommendation techniques became established [Balabanovic *et al.*, 1997]: *collaborative filtering* and *content-based recommendation*. In a collaborative filtering a recommender relies on the user evaluation of the items and based on that it calculates the user and item similarity [Kim *et al.*, 2009]. Such a recommender is able to find users that have a similar taste to a given user or the items that would be similarly rated by the other users [Balabanovic *et al.*, 1997]. Content-based recommenders utilize content representation of the items (i.e., a set of attributes that characterize an item) and predict, whether a user would like items that are similar to those he/she preferred in the past.

It is clear that all these techniques have their advantages and disadvantages, which researchers try to solve in many different ways. Luckily, characteristics of these approaches are often disjointed and thus their reasonable combination can result in better performing approach.

Therefore, we adopt the following definition. *Hybrid recommender systems* are based on the combination of two or more monolithic recommendation techniques, such that the advantages of one recommender are utilized in order to solve the disadvantages of the other recommender [Burke *et al.*, 2011]. Hybrid recommendation is sometimes referred as another recommendation technique (alongside to content-based and collaborative filtering) [Adomavicius *et al.*, 2005].

Hybrid recommenders were introduced to increase an accuracy of existing monolithic techniques and also to reduce or eliminate their major drawbacks (such as the cold-start problem, sparsity, or user outliers). The most common technique is to employ collaborative filtering with the combination of other technique (e.g., content-based or knowledge-based recommender) [Burke, 2002]. However, there are also other combinations of techniques that are suitable depending on the studied problem and sometimes even a domain.

Hybrid recommender systems also attracted many companies and they became employed as a part of their products. In the domain of news, Google developed [Liu *et al.*, 2010] a hybrid recommender that was aimed to combine both user interests and trends in Google News (a combination of content-based recommendation and collaborative filtering). As they reported, proposed hybrid improved Click-Through-Rate by 30% (in comparison to the baseline collaborative approach).

In 2006 a movie streaming company Netflix announced a competition, where the goal was to recommend movies such that the proposed algorithm made an improvement over 10% in comparison to the Netflix baseline recommender. Several solutions were proposed (such as [Koren, 2009]) and many of them were actually the hybrid approaches based on the collaborative filtering recommendation (including the winning ones). Netflix also uses hybrid recommenders nowadays [Gomez-Uribe *et al.*, 2016], for example in the search results, where it combines user movie playbacks, search data and metadata.

The recommender system domain historically connects academia with the business. As a result, plenty of libraries and frameworks have been proposed. These cover various programing languages and as a rule implement state-of-the-art approaches. To give a short overview, we present a comparison of several recommendations libraries and frameworks with emphasis on the hybrid recommenders.

The typical evaluation of a recommenders starts with offline experiments. To obtain a reliable and comparable results, appropriate methodology and dataset is a necessity. In this chapter, we also compare typical datasets used for the evaluation of recommenders with metrics reported by several authors.

The chapter systematically covers the following topics:

- Overview of hybrid recommender techniques with emphasis on pros and cons (Section 1.2.)
- Comparison of recommender system libraries and frameworks (Section 1.3.)
- Practical hints for the evaluation of recommenders (Section 1.4.)
- Comparison of datasets and reported metrics (Section 1.4.4.)

2 Hybrid recommender systems - pros and cons

Hybrid recommender systems were proposed to improve existing monolithic recommendation techniques. By using and combining these techniques, hybrids aim to improve recommendation performance (from several points of view) and thus enhance overall user experience. Correspondingly, there are problems that monolithic techniques suffer from and hybrid recommenders are able to solve.

2.1 *Types of combinations*

When picking a hybrid recommendation technique, we have to select, which combination will be utilized. In other words, which monolithic recommendation techniques will be employed and also how they will be combined.

Following the Burke's taxonomy [Burke, 2002] we distinguish between seven basic types of hybrid combinations: weighted, switching, mixed, feature combination, cascade, meta-level, and feature augmentation. We would like to emphasize that these combinations types define how two or more approaches are combined (no restrictions for specific recommender type).

In *weighted hybrid*, the underlying recommenders calculate a score for an item and these scores are then combined to produce a final (single) score for the recommended item. [Hornung *et al.*, 2010] built a weighted music recommender that combined collaborative recommender (for track similarity) and two content-based recommenders (for tag and time similarity). To enrich the final list of recommendations, they also generated the additional serendipitous music tracks by considering a similarity of the users. Moreover, the famous Netflix prize winner algorithm combined 24 monolithic predictors in order to provide final estimate. The gradient boosted decision trees were used to combine single models covering neighborhood, matrix factorization or regression models [Koren, 2009].

Switching hybrid specifies a condition, which determines which recommendation technique will be selected and used for the recommendation (depending on the situation). A switching hybrid was proposed in [Ghazanfar *et al.*, 2014], where the authors utilized clustering approach to detect the gray-sheep users. These gray-sheep users then received recommendations generated by the separate content-based recommender.

In *mixed hybrid*, each underlying recommender generates a list of recommendations that are combined to produce a final recommendation. In other words, both lists are presented to the user. A TV recommendation using mixed hybrid was proposed in [Barragans-Martínez *et al.*, 2010], where the collaborative filtering was combined with the content-based recommendation. During the merging strategy, they used an average rating of TV shows (calculated by the recommenders).

Feature combination uses multiple types of features that are combined to learn a single recommender model. For instance, the ratings of users combined with the content features of the specific item [Basu *et al.*, 1998]. [Zanker *et al.*, 2009] utilized a single collaborative filtering recommender that combined various features (called *rating domains*), such as the navigation actions, viewed items, items added to the shopping basket, or the user context.

Cascade hybrid is based on the idea of refinements, where the first-level recommender generates recommended items. The role of the second recommender is to adjust the items returned by the first recommender, but here the focus is only on those items that need refinements. Lampropoulos [Lampropoulos *et al.*, 2012] presented a cascade hybrid that employed a two-step solution. Firstly, a content-based recommender was used as a one-class classifier that identified the items suitable for a particular user. Then, a second-level collaborative filtering recommender assigned ratings to the items identified by the content-based recommender.

In *meta-level hybrid*, the first recommender learns a model, which is used as an input to the second recommender. By analyzing the rule-based preferences from historical user interactions, a collaborative filtering model was learned and used as an input to the knowledge-based recommender in [Zanker, 2008].

Similar idea is applied for the *feature augmentation hybrid*, where the *result* (not a model) of the first recommender prediction is used as a feature to the second recommender. [Campos *et al.*, 2010] created a hybrid recommender that used weights produced during the content-based recommendation as an input to the collaborative filtering recommender.

As noted by [Burke, 2002], some combinations (e.g., switching or mixed hybrid) require an initial effort that must be done before we may employ the hybrid strategy. For example, in case of the switching hybrid, we must define the criteria to switch between the recommendation techniques beforehand. The weighted hybrid requires setting the weights that apply for the results of particular hybrids.

For the feature augmentation, the cascade and the meta-level hybrid, a dependency may cause issues, if the second-level recommender relies on the results of the first-level recommender (Table 1.1. summarizes the pros and cons of these combinations).

Another perspective for the hybrid recommender classification was proposed by Aggarwal [Aggarwal, 2016]. He recognizes three high-level types:

- *ensemble design* – analogy to ensemble methods in machine learning. Several algorithms are combined into a single output (switching, weighted, cascade, feature augmentation),
- *monolithic design* – refers to a recommender combining several data sources (feature combination, meta-level),
- *mixed systems* – combines both ensemble and monolithic design.

This taxonomy offers a valuable (from the machine learning perspective) view which addresses the nature of Burke's insight.

Table 1.1. Pros and cons of hybrid combinations (based on [Burke, 2002]).

Type	*Pros*	*Cons*
Weighted	Possible to adjust weights of hybrids. Can be used in datasets with implicit feedback.	Value of the particular recommendation techniques should be uniform across the algorithms. All the techniques apply the weights to each item, which may be redundant.
Switching	System is more sensitive to strengths and weaknesses of the particular recommenders.	Switching criteria must be defined.
Mixed	Suitable where it is possible to make a large number of recommendations simultaneously. Allows to recommend both popular and new items.	Combination technique must be employed. Rules for solving conflicting situations must be also defined.
Feature combination	Combines features from several algorithms which results to improved similarities.	May require feature selection in content-based recommender [Basu *et al.*, 1998].
Feature augmentation	Allows to improve an accuracy of a system without modifying it.	A quality of second recommender may depend on the recommendations of the first (augmenting) recommender.
Cascade	Allows to employ second recommender to only relevant items (results of the first recommender). More efficient than weighted.	Quality of second-level recommender may depend on the recommendations of the first-level recommender.
Meta-level	Learned model is a compressed representation of the user-item preferences.	A quality of the second recommender may depend on the quality of the representation of the first recommender.

2.2 *Hybrids as a solution for recommendation issues*

There are several issues that standard recommendation techniques suffer from. In the worst-case scenario, it results in an inability to recommend any items. Most of these issues are related to how recommender systems work.

We further examine the problems related to the collaborative filtering recommendations (the problems collaborative filtering is either suffering from or is able to help to deal with):

- *cold-start problem* (a problem of a new user/item, or a new context in case of the context-aware recommender systems),
- *over-specialization* (inability to recommend items outside-the-box),
- *sparsity* (of a user-item matrix),
- *extremes* (gray and black sheep),
- *lack of diversity*.

Hybrid recommenders are capable of reducing these problems by hybridization of collaborative filtering with the other recommendation technique.

2.2.1 *Cold-start problem*

One of the most notable problems occurs when a *new user* or a *new item* is introduced to a recommender. This problem is also referred as a *cold-start problem*. Here, the recommender fails to generate appropriate recommendations since it does not have enough knowledge about the user preferences.

When a new user appears, a low number of user-item interactions causes that the recommender is unable to unmask user preferences. This problem is usually present in both content-based and collaborative recommenders. Specifically, for the collaborative recommenders, a cold-start problem occurs also when a new item appears. Since it is not rated by any users, it is not possible to score how appropriate would be to recommend such an item [Schein *et al.*, 2002].

There are several domains, which suffer from the new item cold-start problem more as others. In some domains recommended items are relevant for only a short time period (e.g., news, discounts) and thus the value of the recommended item decreases exponentially over the time.

The cold-start problem is not usually an isolated state of the system, but it is a process (its effect decay over the time, i.e., user activity). It is clear, that there is no specific line (e.g., an amount of user ratings) to be recognized as the "no cold-start". In [Visnovsky *et al.*, 2014], authors analyzed the influence of the amount of user rating to the quality of user similarity search (cluster quality). As we can see (Figure 1.1), the increasing number of the user ratings logarithmically improves the cluster quality. For the MovieLens dataset approx. 50 ratings are required to obtain similar clusters as considering all the user ratings.

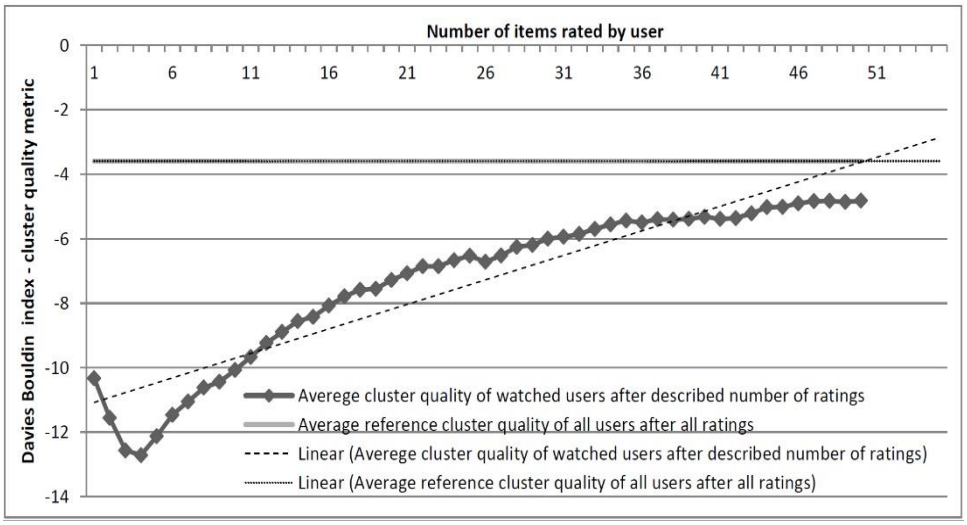

Fig. 1.1. The influence of the amount of user ratings to the cluster quality (similar user search task) in MovieLens dataset [Višňovský *et al.*, 2014].

However, a new item problem does not affect the content-based recommenders, hence the content-based recommender can extract item properties without any user ratings[1]. Therefore, the content-based recommenders can be used to reduce the cold-start problem of collaborative filtering [Ronen *et al.*, 2013]. Moreover, several approaches aim at addressing not only important content-based features, but also important features selection [Cella *et al.*, 2017].

Hybrid recommendation is able to solve the cold-start problem for both a new user and a new item. One example is a work of Schein [Schein *et al.*, 2002], where they fit a model using content and collaborative information. They present a two-way aspect model and Naïve Bayes recommender that uses content features in order to predict the ratings for the non-rated items.

The cold-start problem was further explored in [Braunhofer *et al.*, 2014], where the authors applied hybrid recommender to solve a problem of *a new context*. A cold-start problem of a new context occurs when an existing user is exposed to a new contextual situation. They proposed a switching hybrid recommender that combined a demographic-based context-aware recommender and demographics-based context-aware recommender.

However, as they outlined, evaluating such a hybrid recommender that was in addition extended by the contextual feature was a demanding task since there was a lack of large datasets suitable for this task. There are several datasets that can be used for this task: STS [Elahi *et al.*, 2013], CoMoDa [Odic *et al.*, 2013], and Music [Baltrunas *et al.*, 2011].

2.2.2 *Over-specialization*

One of the shortcomings of the content-based recommenders is that they are not able to recommend the *outside-the-box* items, also referred as a problem of *over-specialization* [Shardanand *et al.*, 1995]. Since the content-based recommender relies on the content descriptions during the user preferences analysis, it is limited to find the similar items to only those that user previously liked.

For example, if a user watches movie from the comedy and adventure genre, content-based recommender learns this information and builds a user model that is used to recommend only movies from these genres. Therefore, it may fail to recommend, for example horror movies even if a user would appreciate some.

On the contrary, there are domains where it is useless to recommend similar items. If someone bought an expensive camera, he/she probably won't buy another (within some reasonable time period).

Here, the hybridization can be beneficial if we combine both collaborative filtering and content-based recommendation. In such a hybrid environment, collaborative recommendation can be helpful in recommending the items outside-the-box. Moreover, the hybridization may eliminate the trade-off between recommendation accuracy and diversity of recommended items [Yoshii *et al.*, 2008].

On the contrary, the specific settings and domain characteristics may bring the over-specialization problem to the collaborative recommenders as well. As the collaborative filtering usually uses the most similar users, if these are highly consistent (and recommender is not designed to bring diversity), only highly specific items will be recommended (similarly to the content-based over-specialization problem).

The over-specialization problem refers to recommending highly tailored items to user past preferences. This often results to the problem of diversity lack. These are, however, two separate concepts. We may lack the diversity of recommended items without over-specialization problem (e.g., user likes adventure and receives sci-fi recommendations).

2.2.3 *Sparsity*

Real-world web applications contain tremendous amount of content and users. This is unfortunately a problem for the collaborative recommender approaches, which often use a user-item matrix (Figure 1.2). In fact, such a matrix is extremely sparse in an average system. This is usually a result of the fact that many users interact with only few items.

[1] This assumes that content (and similarity search) can be processed and computed immediately.

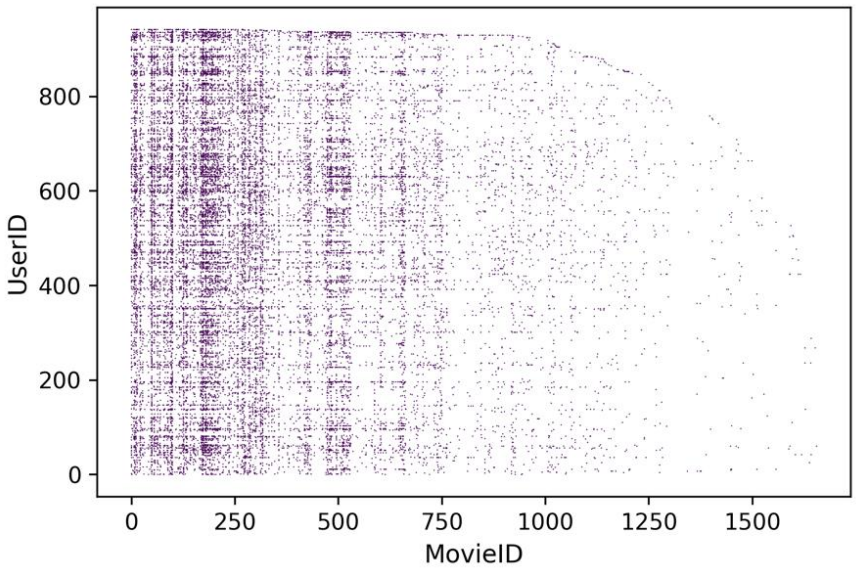

Fig. 1.2. User-item interaction matrix from the MovieLens 100k dataset.

One example is the MovieLens 20M dataset [Harper *et al.*, 2015], which contains 27 000 items (movies) and 138 000 users. An upper bound for the maximum number of ratings is therefore 3,726 * 10^9, however the dataset contains only 2 * 10^7 ratings.

Fortunately, sparsity is yet another issue of the recommender systems that can be reduced by the hybrid recommenders. By utilizing a hybrid model, the missing items from the matrix can be calculated, which solves the problem of sparsity. Several hybridization types are helpful, e.g., the feature combination. By combining several recommender sources (e.g., content and collaborative), we reduce the rating matrix sparsity. An example hybrid recommender was proposed in [Kim *et al.*, 2012], where the authors used a social network and trust scores between users to reduce data sparsity.

2.2.4 *Gray and black sheep*

Another problem of the standard recommendation techniques is the specific users (*extremes*), for which a particular approach can be not sufficient enough. Here we distinguish between two basic extremes: *gray sheep* and *black sheep*.

Gray sheep users do not have consistent opinions and thus do not clearly fall into any of the groups of people sharing the same opinion [Claypool *et al.*, 1999]. This problem occurs namely in the small and medium community of users.

Also, as noted in [Claypool *et al.*, 1999], unlike in the cold-start problem, even by gathering more ratings from such users, a recommender is unable to produce precise predictions. Depending on the dataset and a number of gray sheep users [Ghazanfar *et al.*, 2014], a presence of the gray sheep users may affect the quality of the recommendation for the whole community.

On the other hand, black sheep users [McCrae *et al.*, 2004] have no or few people that they correlate with. Therefore, recommendation approaches relying on the user-to-user correlations are unable to generate any predictions. Su et al. pointed out that although this is clearly a failure of the recommender system, non-electronic recommenders are unable to properly recommend items to black sheep users as well [Su *et al.*, 2009]. Therefore, we may consider such a failure to be acceptable.

Both gray and black sheep users cannot benefit from the collaborative recommendation. This is a consequence of the inability of the recommender to find a relationship between such a user and other users in the community. Similarly, a demographic recommender may have the same issue, since it uses demographic information about the users to categorize them into groups. However, here the solution of the problem is a hybridization where a collaborative or demographic recommender can be combined with a content-based recommendation.

[Ghazanfar *et al.*, 2014] utilized K-means clustering to identify the gray sheep users and proposed a switching hybrid recommender that was able to decrease the recommendation error rate by switching between the collaborative filtering and the content-based recommender.

2.3 *Drawbacks of the hybrid recommenders*

One reason to employ a hybrid recommendation is to *improve the performance* of individual – *monolithic* recommenders, such that the hybrid recommender performs better than any underlying recommender.

However, this requires that the underlying recommenders should be also well-tuned such that they are able to recommend items with satisfying accuracy. If the underlying recommender performs poorly, a hybrid recommender may fail in improving the accuracy and it may end up with the drop, indeed.

We need to choose which recommendation techniques we need to employ and optimize its parameters. Moreover, these recommender techniques need to be properly evaluated. For this step, it is required to have a good knowledge of the underlying recommendation techniques, but also, we need to understand the domain.

Here we should take into consideration the basic *domain characteristics* [Burke *et al.*, 2011]: heterogeneity (of items in the domain), degree of risk (for a user accepting a recommendation), degree of churn (whether a recommender face a continual stream of new items), preferences (stable or unstable), interaction style (implicit or explicit), and scrutability (whether an explanation of recommendation is required by the recommender). Analysis of the domain allows us to choose an appropriate recommendation technique and consider the conditions within which it would run. For instance, in the news domain, where the degree of churn is relatively high, we need to consider the scalability of a hybrid approach.

Explanation of recommendations is a still an open research problem in monolithic recommenders [Herlocker *et al.*, 2000]. In case of hybrid techniques, the problem grows even further, hence we need to properly present an information about the source of the recommendation. By using for instance, a weighed hybrid, it could become cumbersome to determine which recommender contributes the most to the result and even more how this should be presented to a user.

Recently, there have been attempts to solve the issues with the explanation of a hybrid recommendation. For example, Bostandjiev [Bostandjiev *et al.*, 2012] used visual interactive interface that was intended to explain recommendation process and elicit additional user preferences.

This is related to another issue with the hybridization. Not only a particular recommendation technique may need some training phase, but also a hybrid recommender need to be trained in order to handle such particular recommenders. In other words, a hybrid recommender itself adds another parameter that need to be tuned [Campos *et al.*, 2010]. A cross-validation may be employed in order to set parameters (weights), such that the combination of recommendation techniques would fit the problem the best (e.g., which recommenders should be picked for the switching hybrid).

This is a case especially in a weighed hybrid, where the weights need to be estimated. Here, some heuristics may be applied, or these weights can be set with the machine learning. Moreover, these weights can be also personalized, which requires not even more time to train recommender, but also more training instances.

Finally, a hybrid recommender usually requires an additional computation complexity (as more methods are used), which results in worse performance than the monolithic approaches [Cremonesi *et al.*, 2011].

3 Practical implementation considerations

The concept of combining several recommenders to overcome notorious shortcomings is widely accepted. Most of studies in the recommender systems field pointing improved results when used hybrid recommenders. Thanks to this “agreement”, there are plenty of libraries and frameworks implementing (or supporting) hybrid recommender approaches. In this section, we will briefly analyze the most important features of these (Table 1.2).

3.1 *Mrec[2] recommender system library*

Mrec is a Python recommender and evaluation library developed at Mendeley [Mendeley, 2017]. As a part of it, there are several algorithms implemented, which can be used either standalone or as a part of the recommender. The library provides an implementation for:

- SLIM item similarity,
- Weighted matrix factorization WRMF,
- Weighted approximately ranked pairwise ranking loss (WARP),
- Hybrid model which optimizes WARP based on user-item matrix and content features,
- various evaluation metrics (such as Precision, Recall, or Mean Reciprocal Rank).

For a fast development, a command-line interface is available. In addition, the library supports parallelization using IPython. The input for the hybrid recommender consists of the user-item matrix and the content features. A core approach for the library is the WARP algorithm, which reached promising results on the well-established image dataset ImageNet[3] – in the mean of the speed, memory usage, and the performance as well [Weston *et al.*, 2010].

3.2 *Matchbox[4] recommender*

Azure machine learning is getting more and more attention in the last years. The Matchbox recommender, which is available as a part of this machine learning platform, is a large-scale recommender system. It includes both collaborative and content-based approach. These are combined based on the Bayesian probabilistic model.

The main idea is to use the content-based approach first (when a user is relatively new to the system and has only few ratings). Next, the smooth transition to the collaborative filtering is performed as more and more ratings for the user are available.

Two types of content-based features are supported – item and user content features (characteristics). The framework also supports three types of feedback [Stern *et al.*, 2009]: (a) explicit user ratings of items, (b) binary preferences (likes and dislikes), (c) ordinal ratings on a user-specific scale. One of the major shortcomings is the lack of an online training (model has to be retrained periodically).

Model optimal parameters search is offered through the Tune Hyperparameter Module and Cross Validation Module. Also, several metrics to evaluate the performance are available (e.g., MAE, RSME, Precision, AUC).

As the experiments showed [Stern *et al.*, 2009], the content-based features are especially important in the cold-start phase. Together as a hybrid approach, the Matchbox reflects the state-of-the-art performance.

3.3 *Surprise[5] library*

SciPy provides a collection of packages for scientific computation. The Surprise library is a Scikit (SciPy toolkit) library for building and analyzing recommenders [Hug, 2017]. Although it is intended for an easy implementation of custom recommenders, it also provides a range of popular algorithms. The core functionality covers:

- dataset handling (MovieLens and Jester included),
- prediction algorithms – neighborhood methods (kNN), matrix factorization (SVD, SVD++, PMF, NMF), and similarity measures (cosine, Pearson, MSD),
- evaluation support (cross-validation), parameter optimization.

The library itself does not implement any of the hybrid approaches. The ecosystem allows to create custom recommenders, though. In this way, we are able to create a variety of recommenders on the level of a rating prediction or rank reordering.

---

[2] https://mendeley.github.io/mrec

[3] http://www.image-net.org

[4] https://msdn.microsoft.com/en-us/library/azure/dn905987.aspx

[5] http://surpriselib.com

The performance of the algorithms is evaluated based on the RMSE, MAE, or FCP metrics. One of the important characteristics is the documentation, which provides relevant information and a plethora of examples.

Table 1.2. Comparison of libraries and frameworks supporting hybrid recommendation.

Name	*Language*	*Licence*	*Type of combination*	*Evaluation*	*Note*
Mrec	Python	BSD	Weighted, Cascade	yes	-
Matchbox	AzureML	Microsoft online services	Switching	yes	-
Surprise	Python	BSD-3 Clause	–	yes	Custom hybrid implementation is required
LightFM	Python	Apache v2	Feature combination	yes	-
Librec	Java	GNU GPL	Weighted	yes	-
LensKit	Java	LGPL v2.1	Weighted	yes	-
MyMedia Lite	.NET	GNU GPL v3	Weighted	yes	-
Easyrec	Java	GNU GPL	–	no	Custom hybrid implementation is required
Prediction IO	Scala	Apache Licence v2.0	Multiple	yes	-
FluRS	Python	MIT	–	yes	Custom hybrid implementation is required
Seldon	Python	Apache Licence v2.0	Cascade	yes	-
Recommenderlab	R	GNU GPL v2	Weighted	yes	-
Prea	Java	Free BSD	–	yes	Custom hybrid implementation is required
Duine	Java	LGPL v3	Switching	yes	-

3.4 *LightFM[6] library*

Yet another Python implementation. The name is derived from “factorization machines” and combines the content and collaborative ideas [Kula, 2015]. The users and items are represented as the latent vectors, which are defined by the linear combinations of embeddings of the content features (users and items).

Implemented model reflects the data available for the training. If there are no content features provided, it acts as a pure collaborative filtering approach. When the content features are available, these are considered in the optimization process (also useful for the cold-start problem reduction). In total, four loss functions are implemented:

- Logistic,
- Bayesian probabilistic rating,
- Weighted approximate-rank pairwise,
- k-OS Weighted approximate-rank pairwise (k^{th} positive example as a bias).

[6] https://github.com/lyst/lightfm

A model performance evaluation is supported by the implementation of the standard metrics: Precision, Recall, AUC, and Reciprocal rank. Moreover, LightFM allows to easily obtain the MovieLens 100k dataset[7] and use it for the fast experiments.

The LightFM is also available as a Docker container. The documentation provides several examples over various scenarios.

3.5 *Librec*[8]

Librec is a Java library, which includes plenty (over 70) of algorithms implementations. The library consists of several modules, which cover the whole process of recommendation (Figure 1.3.).

The library implements a weighted hybrid recommender, which uses a linear combination of HeatS and ProbS algorithms (derived from the heat and probability spreading) [Zhou *et al.*, 2010]. In total, six types of recommenders are included, while each of them consists of several algorithm implementations:

- Abstract Recommender – provides a set of basic algorithms (e.g., most popular, collaborative, association rules, global average, hybrid),
- Probabilistic Graphical Recommender (e.g., clustering, LDA, PLSA, BUCM),
- Matrix Factorization Recommender (e.g., SVD, BPR, WRMF, RBM),
- Factorization Machines Recommender (e.g., FMALS, FMSGS),
- Social Recommender (e.g., TrustMF, TrustSVD, SOREG, RSTE),
- Tensor Recommender (e.g., BPTF, PITF).

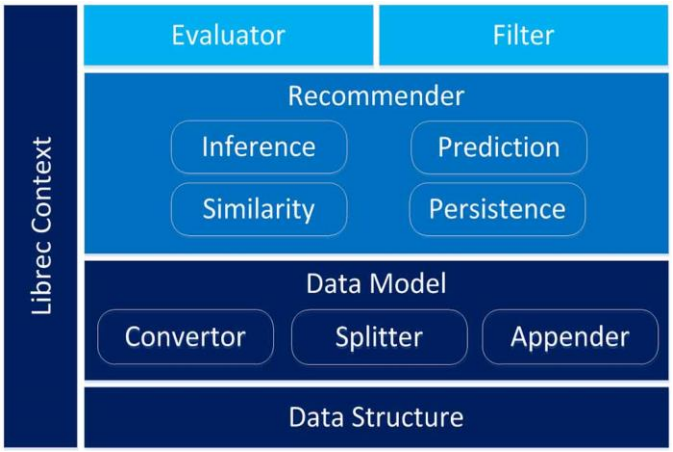

Fig. 1.3. Librec modules overview. The final algorithm is a combination of these components. A set of interfaces allows a flexible implementation of any new algorithms[8].

Several metrics for the performance evaluation are also included, e.g., AUC, nDCG, Precision, Recall, MAE, MPE, and RMSE. Also, a FilmTrust dataset was extracted and included. The documentation provides details for the library usage, with references to the active blogs and discussion forums.

3.6 *LensKit*[9]

Lenskit is an open-source toolkit for building and researching recommender systems, created at University of Minnesota by the GroupLens research group [Ekstrand *et al.*, 2011]. The toolkit was used in over 40 research papers and is also a part of the MovieLens project.

LensKit consists of the several modules focused on the similarity calculation, recommendation, and evaluation of the performance. Four basic algorithms are implemented:

- item-based collaborative filtering,

[7] http://grouplens.org/datasets/movielens/100k/

[8] https://www.librec.net

[9] http://lenskit.org

- user-based collaborative filtering,
- matrix factorization (FunkSVD),
- slope-one rating prediction.

The linear weighted hybrid recommender mechanism is also provided, which allows to combine two recommender lists.

To evaluate the performance of build algorithms, two groups of metrics are supported: prediction accuracy metrics (e.g., RMSE, MAE, Coverage) and top-n (or ranking) metrics (e.g., MAP, MRR, Precision, Recall, and nDCG).

Since the toolkit is supported by the one of major recommenders research group, the community is highly active.

3.7 *MyMediaLite*[10]

The library was created and currently is maintained by the research group at University of Hildesheim [Gantner *et al.*, 2011]. Thanks to its academic background, it has been utilized in over 20 research papers. There are several algorithms implemented in the library, while in addition, own approaches are supported as well. Two basic scenarios are feasible – the rating prediction and the item prediction:

- Item recommenders (e.g., Random, Most popular, Incremental),
- Rating prediction (e.g., SlopeOne, BPSO, Latent-feature log linear, Matrix factorization with factor-wise learning).

For the hyperparameter optimization, a grid search and the Nealder-Mead algorithm is used. As a part of the library, a weighted hybrid recommendation is also provided.

Evaluation module includes the cross-validation and the online evaluation for the several standard metrics (e.g., MAE, RMSE, AUC, nDCG, Precision). MyMediaLite also supports the real-time incremental updates for the selected recommenders.

3.8 *Easyrec*[11]

Easyrec service is made available for the public usage by using the instance provided by the Smart Agent Technologies of the Research Studios Austria. However, the source code is accessible and allows to run an own instance. Easyrec also supports the third-party plugins to integrate with the popular web-based applications (e.g., Drupal, Mediawiki) via the RESTful Web services.

Several non-personalized and personalized algorithms are already implemented within the service:

- Bought together,
- Popular,
- SlopeOne,
- Association rule miner.

The service is designed such that there is no need to implement any recommenders, which partially limits its possibilities, though. Also, there is no evaluation support provided within the service.

3.9 *PredictionIO*[12]

PredictionIO is currently an incubating project of Apache covering the predictive engines for various machine learning tasks. The platform consists of three parts (Figure 1.4.):

- core machine learning stack (intended for building, evaluating and deploying algorithms),
- event server (unifying the events from multiple platforms),
- a template gallery (a storage of algorithm implementations).

---

[10] http://www.mymedialite.net

[11] http://easyrec.org

[12] http://predictionio.incubator.apache.org

The recommenders template gallery contains a number of implementations aiming at the specific tasks (e.g., in the domain of e-shops):

- Collaborative filtering – user and item based,
- Content based – products similarity,
- Association rules, Frequent pattern,
- Complimentary purchases,
- Personalized ranking,
- Hybrid recommendations.

Among the recommender templates, also a classification, regression, clustering, and NLP tasks are supported. The hybrid idea is supported in each of these tasks, while various combining (ensemble) mechanism can be utilized.

An evaluation of the performance is provided by the Tuning and Evaluation module, which supports the optimal parameters search and standard evaluation metrics (e.g., Precision, Recall, Accuracy).

PredicionIO is a highly scalable platform as it bases on Apache Hadoop, HBase, Spark and ElasticSearch (also available as Docker container). The project benefits from the extensive documentation with a plenty of examples and highly active community of the developers.

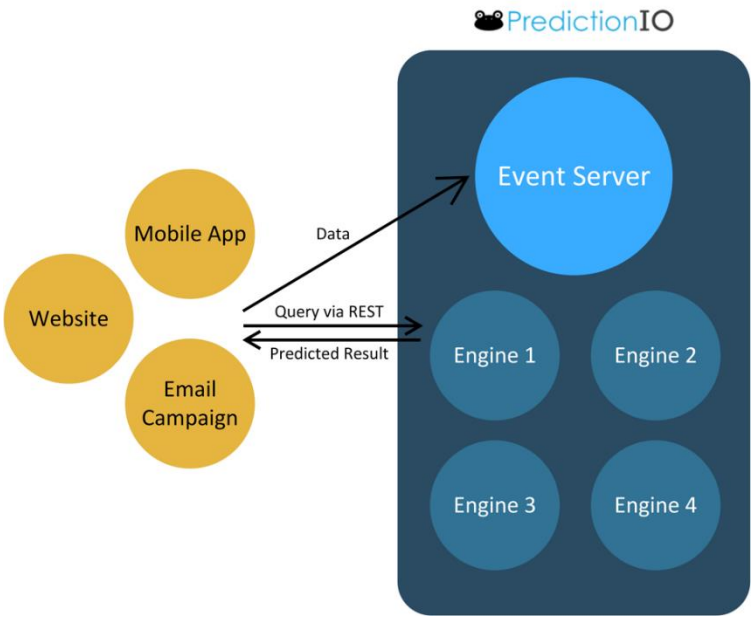

Fig. 1.4. PredictionIO core components[12].

3.10 *FluRS*[13]

Build in Python, FluRS is a small open-source project for an online item recommendation for Python. Its main idea is to provide the "fluent", i.e., incremental recommendation algorithms. Several algorithms are implemented, such as:

- Incremental collaborative filtering (based on the kNN),
- Incremental Matrix factorization and Matrix factorization with BPR optimization,
- Incremental Factorization machines.

A native support for the hybrid recommenders is not provided, on the contrary several metrics for the performance evaluation are available (e.g., MAP, MRR, Precision, Recall). As the project is relatively small and new, the documentation is still evolving.

[13] https://github.com/takuti/flurs

3.11 *Seldon*[14]

Seldon is a platform supporting machine learning tasks intended for the deploy in the production. It runs within a Kubernetes Clusters and supports several model-building tools. Two basic endpoints are available:

- Prediction (several Python pipelines support),
- Recommendation (similar users, latent factor models, association rules, content based, collaborative, hybrid).

Seldon is capable of the weighted and cascading combination of several algorithms and also allows to implement an own hybridization approach. The process of generating recommendations is divided into the offline and the real-time part (Figure 1.5.).

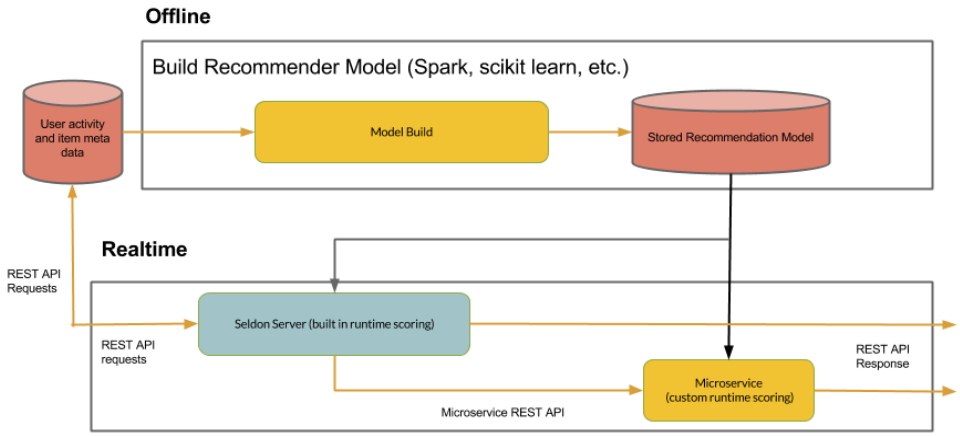

Fig. 1.5. Seldon recommendation components[14].

The platform supports extensive monitoring and analytics via the third-party applications. Also, a paid support for commercial projects is available.

Seldon community is highly active, which results in a rich documentation with many examples.

3.12 *Recommenderlab*[15]

Framework Recommenderlab aims at providing general research infrastructure rather than to create recommender applications. Authors focus on optimizing the process of experiments covering efficient data handling, easy incorporation of algorithms and evaluation [Hahsler, 2017]. Several algorithms are available within the framework:

- Collaborative filtering – user and item based,
- Association rules,
- Most popular, Random,
- Hybrid recommenders – weighting scheme.

The framework supports standard evaluation metrics (e.g., MAE, RMSE, ROC, Precision, Recall). Similarly, the cross-validation, bootstrap sampling serves for the model performance comparison. As a standard for the R libraries, Recommenderlab is well documented and a variety of examples is included in the documentation.

3.13 *Prea*[16]

Toolkit Prea is focused on the collaborative filtering approaches [Lee *et al.*, 2012]. Natively, there is no hybridization technique available, but the support for the own recommenders is provided. Comparing to the

[14] https://www.seldon.io

[15] http://lyle.smu.edu/IDA/recommenderlab

[16] http://prea.gatech.edu

Mahout or MyMedia, Prea also implements several Matrix factorization approaches. Moreover, many additional algorithms are implemented:

- Random, Constant, Average,
- User and Item based, Slope-One,
- Matrix factorization – SVD, NMF, PMF.

For the evaluation of the performance, the toolkit has built-in some basic metrics (e.g., RMSE, MAE, nDCG). Surprisingly, Precision and Recall are not supported. For the data split, a cross-validation is supported. In [Lee *et al.*, 2012], the authors compared the implementations of the algorithm to the MyMedia framework resulting in the very similar values for MAE and RMSE metrics respectively.

3.14 *Duine*[17]

Duine is rather a smaller framework mainly focused on the predictive tasks for the recommendation. Its idea is based on a concept of plugins (e.g., profile models, feedback processors). From the hybrid recommenders perspective, the switching mechanism is utilized. Interesting feature of Duine is an Explanation API, which helps with creating user-friendly explanations for the end-users. Several prediction techniques are implemented:

- Average,
- User-based collaborative filtering,
- Content-based,
- Case-based reasoning.

The framework provides a variety of practical examples; however, the last update comes from 2009. Despite this, it can be used as a solid starting point for the own implementations.

3.15 *Summarization*

Whether attempting to create a new recommender, or using an existing one, recommendation libraries can be useful during the whole development process. However, it is sometimes cumbersome to choose a proper one.

Firstly, it is important to know, which task we aim to do – a rating prediction or an item recommendation (i.e., a Top-N item recommendation). In fact, nearly every library that is capable of the rating prediction can be used for the item recommendation task as well (items can be sorted by their predicted ratings and returned in a Top-N list). However, some libraries are adapted to the item recommendation task and also utilize various techniques in order to improve the order of the recommended items (e.g., learning to rank) with respect to the ranking metrics (such as precision, or recall). One notable example is a LightFM library that uses various loss functions that are meant to optimize the ranking of the items (e.g., Weighted approximate-rank pairwise).

In the training and evaluation phase, the dataset (about users, items, and their interactions) plays an important role. One source is to use the existing recommendation datasets (such as MovieLens; we further describe several available datasets later) or to use the custom ones. The main advantage of the existing datasets is that we are able to compare our results with the research community without the need of implementing the state-of-the-art techniques. However, when creating a recommender for the production environment, there is also a need to evaluate the performance of the recommender on custom data. Here, we may utilize the pre-implemented state-of-the-art recommendation methods.

Some of the libraries support data preprocessing, e.g., the normalization. For instance, (e.g., Surprise) offer tools to handle data pre-processing, or machine learning libraries can be utilized (such as Scikit-learn[18] for Python).

Finally, an important criterion is whether we plan to deploy our recommendation technique to a production environment. This can be usually achieved using every library listed above, however, several of them are more suitable for this task and support the whole pipeline of the process of the recommendation: data preprocessing

[17] http://www.duineframework.org

[18] http://scikit-learn.org

and storage, model training and storage, evaluation, and serving recommendation. Also, there are other relevant factors:

- Do we need to scale to many users (i.e., the process of serving recommendation need to be fast)?
- Do we need to keep the recommendation model up-to-date (i.e., method of recommendation should be capable of a frequent model training without any significant impact on the performance)?
- Is it possible to perform an incremental model updates?

Examples of the libraries that are suitable for the production environment are PredictionIO and Seldon.

4 Practical evaluation considerations

There are several best practices that should be followed when evaluating a recommender system. We list some common principles that are applicable to any technique and add those specific for the hybrid recommendation techniques.

4.1 *Defining an experiment*

One of the first important things that need to be considered during the evaluation is the dataset. Here a correct splitting criterion should be chosen. A common approach is to split the whole dataset into the training and the test set. Usually, it is also suggested to interchange these sets and conduct several successive experiments, such that the train and test set is always different (e.g., k-fold cross validation).

Besides the train and tests sets, also a validation set is important. Especially, in a case of hybrid recommenders, where we need to set and test various parameters, weights, and combinations, the experiments conducted on a validation set are a necessity. The validation set need to be different from the train and test sets.

In the domain of recommenders, there are two basic types of experiments: online and offline. Offline experiments are conducted using the pre-collected datasets [Ricci *et al.*, 2015] (for details see Section 1.4.4.). These experiments are usually performed as soon as the first prototype of the recommender is available and are intended to well-tune the parameters and to explore recommender basic characteristics.

The offline experiments should be not interpreted as the exact measure of the recommender performance. They provide rather the worst-case scenario estimate (as the users do not have a chance to see recommendations and to interact with them).

In an online experiment scenario, we are able to measure how the recommender system influences user behavior, while he/she interacts with the presented items [Ricci *et al.*, 2015]. Clearly, the advantage is that the experiment is conducted with the real users performing the real tasks. Usually, users are split into the groups, where each group experience a different recommendation setup (A/B testing). There are also some drawbacks, such as the risk that the users will be faced a non-relevant recommendation and they leave the experiment too early. There, an online study is done after the offline experiments since the recommender is supposed to be well-tuned and several parameters are already set.

4.2 *Evaluation in hybrid recommenders*

Since the hybrid recommender may consist of different underlying recommender techniques, this need to be considered during the *evaluation*. This also depends on the used combination strategy. Therefore, there are several approaches that were adopted by the researches and were used during their evaluation phase.

The most common practice is to compare the performance of the hybrid recommender (as a whole) to the other state-of-the-art techniques. The state-of-the-art techniques can be either baseline monolithic techniques (such as collaborative filtering) or other hybrid recommenders.

Problem with such a simple approach is that the hybrid recommender is a black-box and we do not know how the underlying recommenders perform. Therefore, many authors also examine different setups of hybrids with respect to the choice of underlying recommenders, used features, and parameters.

Specifically, for the cascade hybrid combination [Lampropoulos *et al.*, 2012], it is suitable to focus on the improvement of the second-level recommender with respect to the first-level recommender (i.e., whether a

second-level recommender is viable to *improve* the performance of the first-level recommender). However, as we have discussed before, it also important to have the first-level recommender well-tuned and evaluated, as well.

In the case of hybrids that combine results of two or more monolithic recommenders, it is more appropriate to evaluate these monolithic approaches separately and then as a whole. For instance, different weights should be set in order to tune the weighted recommender. We can go even further and analyze all the underlying recommenders to compare whether their predictions are similar and how close they are to each other [Hornung *et al.*, 2010].

For instance, in feature combination hybrids, various feature configurations can be investigated [Zanker *et al.*, 2009]. Here, not only different inputs should be considered, but also a different weighing of these inputs. Moreover, if we keen to bring a more personalized experience, a relevance scores of the inputs can be measured for the user (of group of users) separately.

An initial motivation to create hybrid recommenders was not only to improve performance, but also to address the issues associated with the monolithic approaches. However, here we need to focus on the data that we analyze and also for the metrics we choose.

To verify a cold-start problem, a performance for the new user, new item, or new context need to be investigated [Braunhofer *et al.*, 2014]. Based on the combination strategy, several comparisons need to be utilized for each scenario. To illustrate the cold-start problem, there should be also suitable data containing users (or items, or contexts) with the low number of ratings. In some scenarios, these cold-start users (or items, or contexts) may be evaluated separately. If the dataset does not contain such data, we may simulate the cold-start problem by removing the selected number of interactions.

Similarly, for the gray and black sheep users, we first need to identify those users and measure how the recommendation quality differs based on the approach. However, it is usually necessary to verify whether a recommender is able to provide relevant items to non-extreme (i.e., other than gray and black sheep) users.

[Miranda *et al.*, 1999] investigate whether gray sheep users would benefit from using hybrid recommendation rather than collaborative or content-based only. Evaluation procedure was conducted using an online experiment, where these users were actually a part of the study. [Ghazanfar *et al.*, 2014] utilized state-of-the-art datasets and performed offline evaluation, where gray sheep users were separated from the whole dataset and the performance of monolithic approaches was compared to the performance of the switching hybrid.

An issue of diversity is measured as a metric itself – we can measure how diverse the resulting recommendations are or, more precisely, how diverse their *properties* are. For instance, [Burke *et al.*, 2014] analyzed user-based diversity (how users differ within group) and tag-based diversity (how different item tags are) and compared them between three different hybrid approaches.

4.3 *Evaluation frameworks*

Usually, a framework which supports prediction tasks for the recommendation also supports some kind of the performance evaluation. There are, however, frameworks designed specifically for the evaluation of recommender algorithms.

4.3.1 *WrapRec*[19]

WrapRec is a configuration based open-source project (under the MIT license) written in C#. Its idea is to support a fast evaluation of the custom algorithms or the algorithms adopted from the other frameworks [Loni and Said, 2014]. WrapRec architecture is split to three parts (Figure 1.6.).

First one is a module which brings a native support to MyMediaLite and LibFM frameworks. Moreover, also the third-party libraries and custom build recommenders can be evaluated by extending WrapRec. The second module is the Split, which defines the way the data are handled from the train and test evaluation perspective. After the data is loaded through Data reader, they are stored in Data container. By extending the

[19] http://babakx.github.io/WrapRec

split class, a split can be fully customized. Finally, the metrics used for the evaluation are defined in the Evaluation context module. Multiple evaluators are supported.

The WrapRec is a part of the research project CrowdRec based on the Delft University of Technology. It comes with an extensive documentation including several examples.

4.3.2 *Rival*[20]

Another open-source toolkit designed especially for the recommender evaluation is Rival, which is written in Java under LGPL v2.1 license. As it is quite new toolkit, its documentation needs to be improved in the future. Rival consists of four basic modules (important from our point of view).

The evaluation module implements several metrics and strategies for the evaluation. Error metrics include MAE and RMSE, while Precision, Recall, nDCG and MAP as ranking metrics are included.

A recommendation module integrates algorithms from the LensKit and Apache Mahout, which provides a complement to their evaluation tools. The split module is, as expected, responsible for the train and test data splitting. Standard Cross validation is supported. Moreover, a random split and temporal split (considering a timestamp of instances) is available.

Last, but not least, the example module provides examples of the toolkit usage on a real-case scenario. The toolkit is available via the Maven repository.

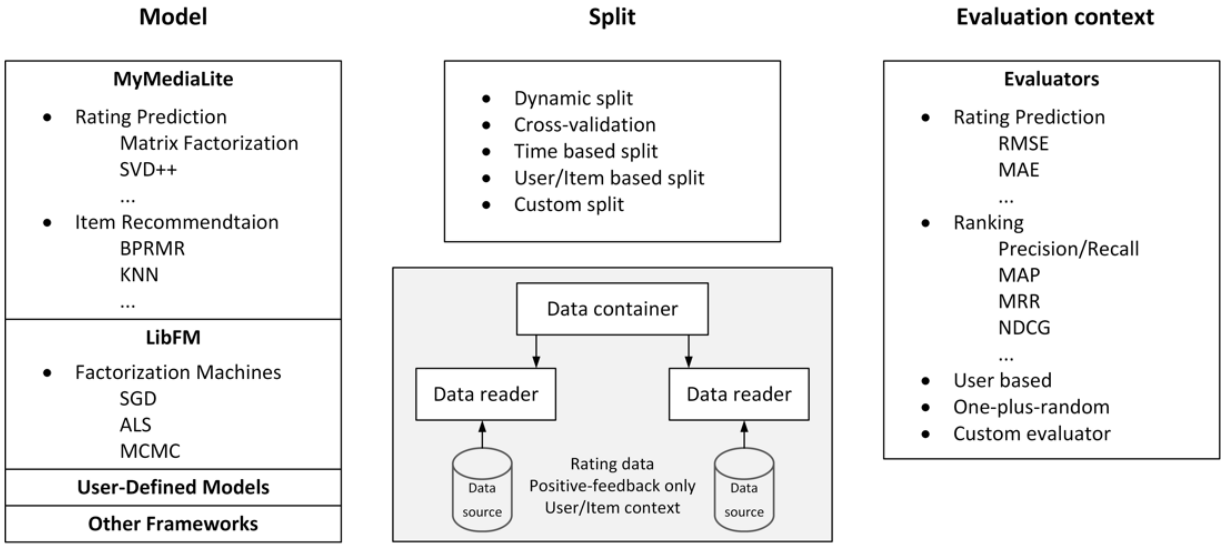

Fig. 1.6. WrapRec evaluation framework architecture[19].

4.4 *Overview of the datasets*

Since the domain of recommendation is largely connected to the industry, there are many real-world datasets available that are being used in research papers. We summarize the available datasets suitable for the collaborative filtering recommendation in the domains such as movies, music, or e-commerce (Table 1.3). We opt for the datasets covered by the research papers. The descriptive characteristics are supplemented by the comparison of the performance in the rating and ranking prediction (Table 1.4).

A majority of the listed datasets is intended to be used for the evaluation of the explicit feedback, namely to predict the ratings of the items. However, in many scenarios, these ratings can be also used as an implicit feedback, as well and can be utilized to evaluate the Top-N collaborative recommendations. In addition, many selected datasets contain not only the interactions between users and items, but also the characteristics of either the users (demography) or the items (content characteristics) that can be utilized in the hybrid collaborative recommendation.

[20] http://rival.recommenders.net

Table 1.3. Overview of available recommendation datasets.

Dataset	*Released*	*Ratings*	*Users*	*Items*	*Sparsity (%)*
MovieLens 100k	10/2016	100K	1K	9K	98.889
MovieLens 1M	2/2003	1M	6K	4K	95.833
MovieLens 10M	1/2009	10M	72K	10K	98.611
MovieLens 20M	10/2016	20M	138K	27K	99.463
MovieLens 26M	8/2017	26M	270K	45K	99.786
DouBan	2011	16.8M	129K	59K	99.779
Flixter	9/2010	8.2M	1M	49K	99.983
Last.fm 1K	5/2010	19.1M	992	1.5M	98.716
Last.fm 360K	3/2010	17.6M	359K	294K	99.983
Yahoo Music R1	3/2004	11.5M	-	98K	-
Yahoo Music R2	2006	717M	1.8M	136K	99.707
Yahoo Music R3	2006	300K	15K	1K	98.000
Yahoo Music KDD Cup Track 1	2011	262M	1M	625K	99.958
The Million Playlist Dataset	1/2018	1 000 000 playlists with 5-250 of tracks			
Epinions (product ratings)	10/2007	664K	50K	140K	99.990
Epinions (trust ratings)	10/2007	487K	-	-	-
Amazon Product Data	2015	142M	-	-	-
Yelp	2018	5.2M	-	174K	-
Book-Crossing	9/2004	1.1M	279K	271K	99.999
Jester Dataset 1	2003	4.1M	73K	100	43.836
Jester Dataset 2	2012	2.2M	79K	150	81.435

Movies has been one of the most dominant domains in collaborative filtering recommendation for many years. Its popularity is ascribed namely to the Netflix Prize competition, however, also to a great availability of the datasets. The MovieLens datasets[21] published by the GroupLens contain movie ratings and tagging activity of the users on the online portal MovieLens.org. Basic content information is also available, but this can be easily extended through the mapping to the external sources – IMDB[22] and TMDB[23]. Douban is the Chinese social network allowing users to rate, review and recommend movies, music, and books [Ma *et al.*, 2011]. Ma et al. crawled the movie section of the portal and besides the ratings they obtained also 1.7M user-to-user relationships (friends links). Similarly, user-to-user relationships are also available for the Flixster dataset [Jamali and Ester, 2010]. From totally 1M users, the user-movie ratings are available for only 150K of them. Although the number of ratings is considerably lower than in the case of MovieLens or DouBan, Flixster dataset stands out in much greater number of social interactions (26.7M).

Table 1.4. Results of the evaluation of recommendation approaches using the selected datasets.

Dataset	*RMSE*	*MAE*	*Precision*	*Recall*	*F-score*	*AUC*
MovieLens 100k	0.8906[A]		0.3526[A]			
MovieLens 1M	0.8333[B]		-			
MovieLens 10M	0.7764[C]		-			
MovieLens 20M	0.7762[C]		-			
Flixter	1.0954[D]		0.046[E]			
DouBan	0.6988[D]		0.082[E]			
Last.fm 1K	-			0.301[F]		
Yahoo Music T1	21.2634[G]					
Yahoo Music T1	21.879[H]					
Yelp	1.0072[I]	0.7920[I]			0.63[J]	0.85[J]

[21] https://grouplens.org/datasets/movielens/

[22] https://www.imdb.com/

[23] https://themoviedb.org/

Yelp					0.0152[K]	
Epinions		0.9321[L]				
Epinions		0.825[M]				
Amazon (clothing)						0.7961 / 0.7317[N]
Amazon (home)						0.7155 / 0.6396[N]
Amazon					0.033[K]	
Book-Crossing (item-based)			0.0364[O]	0.0732[O]		
Book-Crossing (user-based)			0.0369[O]	0.0576[O]		
Jester Dataset 1	4.1229[P]	3.1606[P]				

A http://www.mymedialite.net
B [Lee *et al.*, 2013]
C [Strub *et al.*, 2016]
D [Ma *et al.*, 2011]
E [Wu *et al.*, 2017]
F [Yang *et al.*, 2012]
G [Zheng *et al.*, 2012]
H [Koenigstein *et al.*, 2011]
I [Hu *et al.*, 2014]
J [Tsai, 2016]
K results are reported at F-Score@5; [Chen *et al.*, 2016]
L detailed results are reported in the paper; [Ma *et al.*, 2008]
M [Pham *et al.*, 2011]
N first result is a warm-start setting, second result is a cold-start setting; [Liu *et al.*, 2017]
O [Ziegler *et al.*, 2005]
P [Takacs *et al.*, 2009]

Yahoo published several recommendation datasets in the domain of *music* with varying amount of information[24]. Yahoo Music R1 contains the rating activity of users over artists (i.e., the items are artists). Yahoo Music R2, a much bigger dataset, contains the rating activity of users on the songs. In addition, there are also metadata available (such as artist, album, genres), though these metadata are represented only using the anonymous identifiers. R2 dataset was used in an evaluation of a data-parallel low-rank matrix factorization [Schelter *et al.*, 2013], where authors applied MapReduce technique to improve the performance of the computations in order to better match production environment requirements. The third dataset – Yahoo Music R3 contains a sample of ratings for the songs that were collected from the users' interactions and from the online survey conducted by Yahoo Research.

As a part of the KDD Cup 2011 competition[25], another music ratings dataset was released that contained 10-years (1999-2009) rating activity on four types of items: tracks, albums, artists, and genres. There are also four different versions of this dataset available varying by the amount of rating activity, where one version is focused on the learning-to-rank problem rather than rating prediction.

Last.fm is a popular online service intended to provide music recommendations based on the music that users listen on their devices. In 2010 there were two datasets released [Celma, 2010]: Last.fm 1K[26] and Last.fm 360K[27]. Last.fm 1K contains full listening history of 992 users represented by the tuple user-artist-track. Last.fm 360K provides history for 359,347 users, however, this dataset contains only information about the playcount of the artists. Both datasets contain users' demography (gender, age, country, sign-up date) and MusicBrainz[28] ID for artists and tracks (if available), which allows to extend the dataset for additional metadata.

For the task of the playlist continuation prediction The Million Playlist Dataset (MPD) was published as a part of the Recommender Systems Challenge 2018[29]. Provided dataset contains 1 million playlists created by the U.S. users in the online music streaming service Spotify[30]. When the dataset was generated, there were several criteria for picking up the representative playlists (e.g., minimum number of artists in a playlist, minimum number of albums in a playlist). In comparison to other datasets available in this domain, one

[24] https://webscope.sandbox.yahoo.com/catalog.php?datatype=r

[25] http://www.kdd.org/kdd2011/kddcup.shtml

[26] http://www.dtic.upf.edu/~ocelma/MusicRecommendationDataset/lastfm-1K.html

[27] http://www.dtic.upf.edu/~ocelma/MusicRecommendationDataset/lastfm-360K.html

[28] https://musicbrainz.org

[29] https://recsys-challenge.spotify.com/

[30] https://www.spotify.com

disadvantage is the missing information about the playlist authors (i.e., user ids). Since at the time of writing this book the challenge was just announced, there were no published papers using this dataset.

Amazon[31] is one of the most famous online retailers with roughly millions of daily users. In 2015 Julian McAuley released[32] the Amazon *product reviews* dataset [McAuley *et al.*, 2015], which contained the reviews and product metadata from the May 1996 – July 2014 period. Reviews consist of numerical ratings, textual reviews, and helpfulness votes. Product metadata include descriptions, categories, price, brand, related products, sales rank, and visual features extracted from the product images (4096-length feature vector extracted using the deep convolutional neural network). Reviews can be retrieved as a whole or as the subsets distinguished by the product categories. In addition, the author provides the raw ratings (a tuple user-item-rating-timestamp) that can be suitable for the training using some of the recommender systems libraries (e.g., MyMediaLite).

Epinions.com was an online service (run by eBay, now discounted) more focused on the opinions of the users on various items (e.g., cars, books, movies, software) [Massa and Avesani, 2007]. Moreover, users were able to "rate" other reviews and express their trust to the reviews of these users (i.e., which reviews they find valuable and which they found offensive, inaccurate, or not valuable at all). Massa et al. crawled Epinions.com website in order to evaluate a trust-aware collaborative filtering recommender. The dataset[33] that they released contained both the ratings of the users on the products, but also the ratings of users on other users' reviews (i.e., the users' trust statements).

Another dataset containing reviews is from Yelp[34], where the businesses are being reviewed. Here, the dataset is published directly by Yelp and is updated periodically, since it is a part of an ongoing research challenge. It contains more than 5 million reviews of 174K businesses supplemented by 200K pictures. Moreover, there are 1.1M tips by 1.3M users available and also aggregated hourly check-ins for each business. Besides the reviews and check-ins, there are several metadata about the businesses (such as address, average rating, whether it is a take-out restaurant, availability of parking, business categories, …) and the users (such as list of friends, votes given by the user, opinions received by other users, …).

In a *book* domain, Book-Crossing dataset[35] was released, which contains implicit and explicit ratings (on a scale 1-10) of people on books from Book-Crossing.com [Ziegler *et al.*, 2005]. Book-Crossing[36] is an online community portal, where the users may exchange books between themselves. There are also demographic data available (users' location and age) and books metadata (title, author, year of publication, publisher, cover images).

Another popular dataset among the research community is from Jester Online Recommender System[37]. Dataset contains users' ratings on *jokes* on a rating scale from -10 to 10 [Goldberg *et al.*, 2001]. What is specific for this dataset is that the number of items is very low (100 and 150), thus resulting in low sparsity of the rating matrix.

4.5 *Summarization*

Generally, an evaluation of any recommender system is a systematic process where a researcher need to take into account several aspects: experiment setup, used data, examined metrics, and desired outcomes. Especially, if we think of hybrid recommender systems, there also other factors, such as whether we need to evaluate any monolithic recommenders, or whether we need to evaluate various combinations of attributes in data.

Existing recommendation libraries may simplify this process by providing pre-implemented tools. There are also libraries that are focused specifically on the recommender systems evaluations. In case of data, there are several datasets from a couple of domains publicly available, which allows not only to easily evaluate new

[31] https://www.amazon.com

[32] http://jmcauley.ucsd.edu/data/amazon/

[33] http://www.trustlet.org/downloaded_epinions.html

[34] https://www.yelp.com/dataset

[35] http://www2.informatik.uni-freiburg.de/~cziegler/BX/

[36] https://www.bookcrossing.com

[37] http://eigentaste.berkeley.edu/

recommendation approaches but also to be more transparent while comparing the results (of evaluation) with the research community.

5 Conclusions

Recommender systems have been proved beneficial in many domains. There are several types of recommendation techniques, such as content-based, collaborative, or demographic that are utilized with respect to the domain, user characteristics, item characteristics, and a goal. However, many of these techniques still suffer from various issues that have their roots namely in a lack of data or a specific user behavior.

Hybrid recommender systems were designed to combine the advantages of these techniques in order to solve their major issues and to improve an overall recommendation performance. There are several hybrid combinations that we may choose from and can be used as a template of how to combine two or more recommendation techniques.

Each combination may be suitable for a different scenario and in a different domain. Moreover, we should take into consideration their advantages and disadvantages. It is also important to note that these combinations are not tightened to any particular recommendation techniques. In addition, researchers are not limited to basic combinations, but there are many parameters that can be tuned (e.g., weights, switching criteria). Allowing to tune the parameters brings a possibility to create another level of personalization.

There are several issues related to collaborative filtering that can be reduced by employing a hybrid recommender. Most notable are the cold-start problem, over-specialization, lack of diversity, user extremes, and sparsity. Collaborative filtering may either suffer from these problems or can be used a solution to this problems with the combination of another technique. A major advantage of hybridization is that it enables system to recommend items in some scenarios that collaborative filtering may fail in (such as a new item or a user extreme).

Hybrid recommenders have become a part of many frameworks and libraries in the domain of recommendation. These libraries are also available in various programming languages (such as R, Python, Java, Scala) and support different hybrid combinations. Moreover, many of these libraries are also extendable (e.g., Surprise, Easyrec, FluRS), which allows researchers to easily create their own combinations and yield more valuable research results. They also consist of many other algorithms that are related to the recommendation, such as similarity measures, plenty of matrix factorization approaches, or prediction algorithms.

In most cases, the libraries also provide tools to evaluate performance of the recommendation and measure some basic and also advanced metrics. For this purpose, several state-of-the-art datasets were created and are publicly available as well.

Evaluation of hybrid recommenders inherits many specifics of basic recommendation techniques. Firstly, it is important to pick a suitable dataset and split the data into train and test sets. While designing an experiment, we may opt into online or offline evaluation. Choice of the evaluation metrics should be performed with the respect to the recommendation task.

There are also specifics that are related to the evaluation of the hybrid recommenders. Here we should not only evaluate hybrid recommender as a whole, but also consider evaluating particular underlying recommenders, as well as their parameters. Moreover, if a hybrid recommender was designed to solve the recommendation issues, it is necessary to investigate whether (and how) the hybrid was able to tackle them.

There are also some evaluation frameworks that were designed specifically for this purpose: WrapRec and Rival. Both of them allow to use existing libraries and algorithms. They also offer evaluation measures that are related to the recommendation tasks. Moreover, they are open-source, which allows researchers to do a further development.

Besides all the advantages of the hybrid recommenders, the hybridization may bring up some issues. Hybrids suffer from the problem of the scalability since the training phase may require additional time cost. Another related issue is that we need more data in order to well-tune parameters of hybrid combinations, which is not needed in monolithic approaches.

Yet there are still some unexplored hybrid combinations that were not studied and verified in current literature. Also, many state-of-the-art approaches are only compared to monolithic techniques, with missing

comparison of hybrids between each other. There are also many domains, where the hybrid recommenders were not applied yet.

6 References

Adomavicius, G. and Tuzhilin, A. (2005). Toward the next generation of recommender systems: a survey of the state-of-the-art and possible extensions. IEEE Transactions on Knowledge and Data Engineering, 17(6), 734-749

Aggarwal, C. C. (2016). Recommender Systems: The Textbook. Cham: Springer. ISBN: 978-3-319-29657-9

Balabanovic, M. and Shoham, Y. (1997). Fab: Content-based, collaborative recommendation. Commun. ACM, 40(3), 66-72

Baltrunas, L., Kaminskas, M., Ludwig, B., Moling, O., Ricci, F., Aydin, A., Luke, K.-H., and Schwaiger, R. (2011). InCarMusic: Context-Aware Music Recommendations in a Car, pp. 89-100. Springer Berlin Heidelberg, Berlin, Heidelberg.

Barragans-Martinez, A. B., Costa-Montenegro, E., Burguillo, J. C., Rey-Lopez, M., Mikic-Fonte, F. A., and Peleteiro, A. (2010). A hybrid content-based and item-based collaborative filtering approach to recommend TV programs enhanced with singular value decomposition. Information Sciences, 180(22), pp. 4290-4311.

Basu, C., Hirsh, H., and Cohen, W. (1998). Recommendation as classification: Using social and content-based information in recommendation. In Proceedings of the Fifteenth National/Tenth Conference on Artificial Intelligence/Innovative Applications of Artificial Intelligence, AAAI '98/IAAI '98, pp. 714-720, Menlo Park, CA, USA. American Association for Artificial Intelligence.

Braunhofer, M., Codina, V., and Ricci, F. (2014). Switching hybrid for cold-starting context-aware recommender systems. In Proceedings of the 8th ACM Conference on Recommender Systems, RecSys '14, pp. 349-352, New York, NY, USA. ACM.

Bostandjiev, S., O'Donovan, J., and Hollerer, T. (2012). Tasteweights: A visual interactive hybrid recommender system. In Proceedings of the Sixth ACM Conference on Recommender Systems, RecSys '12, pp. 35-42, New York, NY, USA. ACM

Burke, R. (2002). Hybrid recommender systems: Survey and experiments. User Modeling and User-Adapted Interaction, 12(4), pp. 331-370.

Burke, R. and Ramezani, M. (2011). Matching Recommendation Technologies and Domains, pp. 367-386. Springer US, Boston, MA.

Burke, R., Vahedian, F., and Mobasher, B. (2014). Hybrid Recommendation in Heterogeneous Networks, pages 49-60. Springer International Publishing, Cham.

Celma, O. (2010). Music Recommendation, pp. 43–85. Springer Berlin Heidelberg, Berlin, Heidelberg.

Chen, X., Qin, Z., Zhang, Y., and Xu, T. (2016). Learning to rank features for recommendation over multiple categories. In Proceedings of the 39th International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR '16, pp. 305–314, New York, NY, USA. ACM.

Cremonesi, P., Turrin, R., and Airoldi, F. (2011). Hybrid algorithms for recommending new items. In Proceedings of the 2nd International Workshop on Information Heterogeneity and Fusion in Recommender Systems, HetRec '11, pp. 33-40, New York, NY, USA. ACM.

de Campos, L. M., Fernandez-Luna, J. M., Huete, J. F., and Rueda-Morales, M. A. (2010). Combining content-based and collaborative recommendations: A hybrid approach based on bayesian networks. International Journal of Approximate Reasoning, 51(7): pp. 785-799.

Cella, L., Cereda, S., Quadrana, M., and Cremonesi, P. (2017) Deriving Item Features Relevance from Past User Interactions. Proceedings of the 25th Conference on User Modeling, Adaptation and Personalization, ACM, pp. 275-279.

Claypool, M., Gokhale, A., Miranda, T., Murnikov, P., Netes, D., and Sartin, M. (1999). Combing content-based and collaborative filters in an online newspaper.

Ekstrand, M., D., Ludwig, M., Konstan, J., A., and Riedl, J., T. (2011). Rethinking the recommender research ecosystem: reproducibility, openness, and LensKit. In Proceedings of the fifth ACM conference on Recommender systems (RecSys '11). ACM, New York, NY, USA, pp. 133-140.

Ekstrand, M. and Riedl, J. (2012). When recommenders fail: Predicting recommender failure for algorithm selection and combination. In Proceedings of the Sixth ACM Conference on Recommender Systems, RecSys '12, pp. 233-236, New York, NY, USA. ACM.

Elahi, M., Braunhofer, M., Ricci, F., and Tkalcic, M. (2013). Personality-based active learning for collaborative filtering recommender systems. In Proceeding of the XIIIth International Conference on AI*IA 2013: Advances in Artificial Intelligence - Volume 8249, pp. 360-371, New York, NY, USA. Springer-Verlag New York, Inc

Gantner, Z., Rendle, S., Freudenthaler, Ch. and Schmidt-Thieme, L. (2011). MyMediaLite: a free recommender system library. In Proceedings of the fifth ACM conference on Recommender systems (RecSys '11). ACM, New York, NY, USA, pp. 305-308.

Ghazanfar, M. A. and Prugel-Bennett, A. (2014). Leveraging clustering approaches to solve the gray-sheep users problem in recommender systems. Expert Syst. Appl., 41(7), pp. 3261-3275.

Goldberg, K., Roeder, T., Gupta, D., and Perkins, C. (2001). Eigen-taste: A constant time collaborative filtering algorithm. Information Retrieval, 4(2):133–151.

Gomez-Uribe, C. A. and Hunt, N. (2015). The Netflix Recommender System: Algorithms, Business Value, and Innovation. ACM Trans. Manage. Inf. Syst., 6(4), pp. 13:1-13:19.

Guo, G., Zhang, J., and Yorke-Smith, N. (2016). A novel evidence-based bayesian similarity measure for recommender systems. ACM Trans. Web, 10(2), pp. 8:1–8:30.

Hahsler, M. (2017). recommenderlab: A Framework for Developing and Testing Recommendation Algorithms. Technical report.

Harper, F. M. and Konstan, J. A. (2015). The movielens datasets: History and context. ACM Trans. Interact. Intell. Syst., 5(4), pp. 19:1-19:19.

Herlocker, J. L., Konstan, J. A., and Riedl, J. (2000). Explaining collaborative filtering recommendations. In Proceedings of the 2000 ACM Conference on Computer Supported Cooperative Work, CSCW '00, pp. 241-250, New York, NY, USA. ACM

Hornung, T., Ziegler, C.-N., Franz, S., Przyjaciel-Zablocki, M., Schatzle, A., and Lausen, G. (2013). Evaluating hybrid music recommender systems. In Proceedings of the 2013 IEEE/WIC/ACM International Joint Conferences on Web Intelligence (WI) and Intelligent Agent Technologies (IAT) - Volume 01, WI-IAT '13, pp. 57-64, Washington, DC, USA. IEEE Computer Society

Hu, L., Sun, A., and Liu, Y. (2014). Your neighbors affect your ratings: On geographical neighborhood influence to rating prediction. In Proceedings of the 37th International ACM SIGIR Conference on Research & Development in Information Retrieval, SIGIR '14, pp. 345–354, New York, NY, USA. ACM.

Hug, N. (2017). Surprise, a Python library for recommender systems. http://surpriselib.com

Jamali, M. and Ester, M. (2010). A matrix factorization technique with trust propagation for recommendation in social networks. In Proceedings of the Fourth ACM Conference on Recommender Systems, RecSys '10, pp. 135–142, New York, NY, USA. ACM.

Kim, J. K., Jang, M. K., Kim, H. K., and Cho, Y. H. (2009). A hybrid recommendation procedure for new items using preference boundary. In Proceedings of the 11th International Conference on Electronic Commerce, ICEC '09, pp. 289-295, New York, NY, USA. ACM.

Kim, S.-C., Park, C.-S., and Kim, S. K. (2012). A Hybrid Recommendation System Using Trust Scores in a Social Network, pp. 107–112. Springer Netherlands, Dordrecht.

Koenigstein, N., Dror, G., and Koren, Y. (2011). Yahoo! music recommendations: Modeling music ratings with temporal dynamics and item taxonomy. In Proceedings of the Fifth ACM Conference on Recommender Systems, RecSys '11, pp. 165–172, New York, NY, USA. ACM.

Koren, Y. (2009). The bellkor solution to the netflix grand prize.

Kula, M. (2015). Metadata Embeddings for User and Item Cold-start Recommendations. In Proceedings of the Workshop on New Trends in Content-Based Recommender Systems. Ceur-WS, pp. 14-21.

Lampropoulos, A. S., Sotiropoulos, D. N., and Tsihrintzis, G. A. (2012). Evaluation of a cascade hybrid recommendation as a combination of one-class classification and collaborative filtering. In 2012 IEEE 24th International Conference on Tools with Artificial Intelligence, volume 1, pp. 674-681

Lee, J., Kim, S., Lebanon, G., and Singer, Y. (2013). Local low-rank matrix approximation. In Dasgupta, S. and McAllester, D., editors, Proceedings of the 30th International Conference on Machine Learning, volume 28 of Proceedings of Machine Learning Research, pp. 82–90, Atlanta, Georgia, USA.

Lee, J., Sun, M., and Lebanon, G. (2012). PREA: personalized recommendation algorithms toolkit. J. Mach. Learn. Res. 13, 1 (September 2012), pp. 2699-2703.

Liu, J., Dolan, P., and Pedersen, E. R. (2010). Personalized news recommendation based on click behavior. In Proceedings of the 15th International Conference on Intelligent User Interfaces, IUI '10, pp. 31-40, New York, NY, USA. ACM.

Liu, Q., Wu, S., and Wang, L. (2017). Deepstyle: Learning user preferences for visual recommendation. In Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR '17, pp. 841–844, New York, NY, USA. ACM.

Loni, B., and Said, A. (2014). WrapRec: an easy extension of recommender system libraries. In Proceedings of the 8th ACM Conference on Recommender systems (RecSys '14). ACM, New York, NY, USA, pp. 377-378.

Ma, H., Zhou, D., Liu, C., Lyu, M. R., and King, I. (2011). Recommender systems with social regularization. In Proceedings of the Fourth ACM International Conference on Web Search and Data Mining, WSDM '11, pp. 287–296, New York, NY, USA. ACM.

Ma, H., Yang, H., Lyu, M. R., and King, I. (2008). Sorec: Social recommendation using probabilistic matrix factorization. In Proceedings of the 17th ACM Conference on Information and Knowledge Management, CIKM '08, pp. 931–940, New York, NY, USA. ACM.

Massa, P. and Avesani, P. (2007). Trust-aware recommender systems. In Proceedings of the 2007 ACM Conference on Recommender Systems, RecSys'07, pp. 17–24, New York, NY, USA. ACM.

McAuley, J., Targett, C., Shi, Q., and van den Hengel, A. (2015). Image-based recommendations on styles and substitutes. In Proceedings of the 38th International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR '15, pp. 43–52, New York, NY, USA. ACM.

McCrae, J., Piatek, A., and Langley, A. (2004). Collaborative filtering.

Miranda, T., Claypool, M., Gokhale, A., Mir, T., Murnikov, P., Netes, D., & Sartin, M. (1999). Combining content-based and collaborative filters in an online newspaper. In Proceedings of ACM SIGIR Workshop on Recommender Systems.

Mendeley. (2017). Mrec library documentation. http://mendeley.github.io/mrec

Odic, A., Tkalcic, M., Tasic, J. F., and Kosir, A. (2013). Predicting and detecting the relevant contextual information in a movie-recommender system. Interact-ing with Computers, 25(1), pp. 74-90.

Pham, M. C., Cao, Y., Klamma, R., and Jarke, M. (2011). A clustering approach for collaborative filtering recommendation using social network analysis. 17(4):583–604.

Ricci, F., Rokach, L., and Shapira, B. (2015). Recommender Systems Handbook. Springer US, New York, NY, USA, 2nd edition

Ronen, R., Koenigstein, N., Ziklik, E., and Nice, N. (2013). Selecting content-based features for collaborative filtering recommenders. In Proceedings of the 7th ACM Conference on Recommender Systems, RecSys '13, pp. 407-410, New York, NY, USA. ACM.

Said A., and Bellogín, A. (2014). Comparative recommender system evaluation: benchmarking recommendation frameworks. In Proceedings of the 8th ACM Conference on Recommender systems (RecSys '14). ACM, New York, NY, USA, pp. 129-136.

Schein, A. I., Popescul, A., Ungar, L. H., and Pennock, D. M. (2002). Methods and metrics for cold-start recommendations. In Proceedings of the 25th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR '02, pp. 253-260, New York, NY, USA. ACM

Schelter, S., Boden, C., Schenck, M., Alexandrov, A., and Markl, V. (2013). Distributed matrix factorization with mapreduce using a series of broadcast-joins. In Proceedings of the 7th ACM Conference on Recommender Systems, RecSys '13, pp. 281–284, New York, NY, USA. ACM.

Shardanand, U. and Maes, P. (1995). Social information filtering: Algorithms for automating "word of mouth". In Proceedings of the SIGCHI Conference on Human Factors in Computing Systems, CHI'95, pp. 210-217, New York, NY, USA. ACM Press/Addison-Wesley Pub. Co.

Stern, D., H., Herbrich, R. and Graepel, T. (2009). Matchbox: Large Scale Online Bayesian Recommendations. In Proceedings of the 18th international conference on World wide web (WWW '09). ACM, New York, NY, USA, pp. 111-120.

Strub, F., Gaudel, R., and Mary, J. (2016). Hybrid recommender system based on autoencoders. In Proceedings of the 1st Workshop on Deep Learning for Recommender Systems, DLRS 2016, pp. 11–16, New York, NY, USA. ACM.

Su, X. and Khoshgoftaar, T. M. (2009). A survey of collaborative filtering techniques. Advances in Artificial Intelligence, 2009, pp. 4:2–4:2.

Takacs, G., Pilaszy, I., Nemeth, B., and Tikk, D. (2009). Scalable collaborative filtering approaches for large recommender systems. J. Mach. Learn. Res., 10:623–656.

Tsai, C.-H. (2016). A fuzzy-based personalized recommender system for local businesses. In Proceedings of the 27th ACM Conference on Hypertext and Social Media, HT '16, pp. 297–302, New York, NY, USA. ACM.

Višnovský, J., Kaššák, O., Kompan, M. and Bieliková, M. (2014) The Cold Start: Minimal User's Rating Activity Estimation. In 1st Workshop on Recommender Systems for Television and online Video (RecSysTV) in conjunction with 8th ACM Conference on Recommender Systems, Foster City, USA, p. 4.

Weston, J., Bengio, S., and Usunier, N. (2010). Large scale image annotation: learning to rank with joint word-image embeddings. Mach. Learn. 81, 1, pp. 21-35.

Wu, Q., Liu, S., and Miao, C. (2017). Modeling uncertainty driven curiosity for social recommendation. In Proceedings of the International Conference on Web Intelligence, WI '17, pp. 790–798, New York, NY, USA. ACM.

Yang, D., Chen, T., Zhang, W., Lu, Q., and Yu, Y. (2012). Local implicit feedback mining for music recommendation. In Proceedings of the Sixth ACM Conference on Recommender Systems, RecSys '12, pp. 91–98, New York, NY, USA. ACM.

Yoshii, K., Goto, M., Komatani, K., Ogata, T., and Okuno, H. G. (2008). An efficient hybrid music recommender system using an incrementally trainable probabilistic generative model. IEEE Transactions on Audio, Speech, and Language Processing, 16(2), pp. 435-447.

Zanker, M. (2008). A collaborative constraint-based meta-level recommender. In Proceedings of the 2008 ACM Conference on Recommender Systems, RecSys'08, pp. 139-146, New York, NY, USA. ACM.

Zanker, M. and Jessenitschnig, M. (2009). Collaborative feature-combination recommender exploiting explicit and implicit user feedback. In 2009 IEEE Conference on Commerce and Enterprise Computing, pp. 49-56.

Zheng, Z., Chen, T., Liu, N., Yang, Q., and Yu, Y. (2012). Rating prediction with informative ensemble of multi-resolution dynamic models. In Dror, G., Koren, Y., and Weimer, M., editors, Proceedings of KDD Cup 2011, volume 18 of Proceedings of Machine Learning Research, pp. 75–97. PMLR.

Zhou, T., Kuscsik, Z., Liu, J., Medo, M., Wakeling, J., and Zhang, Y. (2010). Solving the apparent diversity-accuracy dilemma of recommender systems. Proceedings of the National Academy of Sciences, 107, 4511--4515.

Ziegler, C.-N., McNee, S. M., Konstan, J. A., and Lausen, G. (2005). Improving recommendation lists through topic diversification. In Proceedings of the 14th International Conference on World Wide Web, WWW '05, pp. 22–32, New York, NY, USA. ACM.