Utilizing Metadata to Select a Recommendation Algorithm for a User or an Item

Alexander Nechaev[1][0000−0002−0464−9961], Nataly Zhukova[2][0000−0001−5877−4461], and Vasily Meltsov[1][0000−0001−5479−9979]

[1] Vyatka State University, Kirov, Russia
[2] ITMO University, St. Petersburg, Russia
dapqa@yandex.ru, nazhukova@mail.ru, meltsov69@mail.ru

Abstract. In general, recommender systems solve the problem of information overload by helping users of services to find items in which they are interested. There are plenty of algorithms and models that can be used to build such a subset of items, and their performance may vary not only for different datasets but for separate parts of a single one. This issue leads to the "algorithm selection problem". Many state-of-art solutions to this problem are based on meta-learning. It associates the task features with the performance of the base-level algorithms and models. This paper presents the method of recommendation algorithm selection for particular users (or items) that uses binary representations of both explicit metadata of them and computable statistical meta-features. There are two different techniques within the proposed method, which are based on classification or clustering of such binary data, respectively. The meta-learning process is almost automated. The findings of the experiments prove that the usage of the method for recommendation algorithm selection is reasonable and effective. In most cases, a recommender system that uses the metamodel shows lower rating prediction errors compared to any other one utilizing a single model or algorithm for all the users (or items), while in a small number of tests their performance is just the same. The detailed analysis of the evaluation results allows for affirming that the described metamodels can be used in real-world systems to improve the experience of particular users.

Keywords: Recommender System · Algorithm Selection Problem · Meta-Learning · Collaborative Filtering.

1 Introduction

Numerous modern companies provide their clients with a great variety of services, products, and content. Such diversity leads to the problem of information overload that has an adverse effect on both user experience and service growth. Recommender systems (RSs) are used to solve this problem effectively. They

automatically create suggestions of items that most likely lie in a user's area of interest.

Modern recommendation algorithms are actively used in real-world systems [1]. Even a moderate improvement in the personalization of the collections of items is beneficial for both services' clients and its owners. Moreover, the existence of instruments predicting a user's interests makes it possible to use recommender systems in such non-standard areas as medicine and law. Nonetheless, the accuracy of these algorithms is relatively low. The need for functioning in the poorly formalizable domain of people's interests leads to the vast complexity of the development of effective algorithms as well as to the inability to create universal ones. Usually, RS designers implement and evaluate different models and several versions of their compositions. This approach needs a significant number of computational resources, but the result can be suboptimal [2].

The crucial problem that arises is that the performance of a model may vary for different users and items within a single dataset. For example, a matrix factorization model can predict the interests of a particular service user precisely, while the k-Nearest-Neighbors algorithm is much more applicable for making recommendations for another individual. A state-of-art solution to this problem is the use of meta-learning.

Meta-learning lies in the creation of metamodels that associate task features with algorithms' performance. Such metamodels can be separated into the global-level (of a dataset), middle-level (of particular users or items), and micro-level (of particular rows) ones [3]. Micro-level metamodels have recently been proposed and not been studied enough, since they need massive and dense datasets to be effective. There are plenty of researches within the first two levels, including the meta-recommender system methods. Within this field, the contribution of Cuhna et al. [4,5] and their meta-research [2] is particularly notable. The authors' studies provide an in-depth review of the existing meta-learning approaches to the problem of algorithm selection in the domain of recommender systems.

In most cases, a metamodel utilizes only the data about the base models' performance. The real-world systems are generally aware of users' and items' features, even if they are not used in the recommendations generation process. The core idea behind this research is that a metamodel can take a cue from both such metadata and standard meta-features. Moreover, when a metamodel training dataset is represented in a binary format, it allows for applying the same algorithms in different business cases.

This paper proposes the middle-level automated meta-learning method that uses the recommendation algorithms' evaluation results as well as the binary representation of both users' (or items') metadata and their computable meta-features.

2 Method

The formal statement of the research problem is as follows. There is a dataset consisting of (u, i, r_{ui}) rows that show what rating r_{ui} has the user u given to the item i. Also, there are users' and items' metadata stored as binary vectors. Several recommendations models (base models) are trained on the part of a dataset, and evaluated on another part, which has given the predictions $(u, i, \widehat{r}_{ui})$. The task is to create a metamodel that selects the best base model for a particular user (or item), i.e. makes a personalized selection.

It should be stated that below is the solution that utilizes users' metadata. The solution that uses items' metadata is the same.

At this stage, two different solutions to the problem are proposed. On the one hand, one can take the evaluation results as the starting point and make a dataset showing the correspondence between metadata of the user and the best model for them. In this case, the task is to solve a classification problem. On the other hand, one can start from the nature of users, group them using particular features, and select the best model for each built group. In this case, the task is to solve a clustering problem [6].

While a metamodel is being trained, its input is built from two parts. The first one is a set of explicit users' metadata, taken from a dataset, e.g. age, sex, or profession. The second one contains meta-features computed on the basis of users' rating information, i.e., some standard values used in meta-learning.

Fig. 1 shows the proposed recommender system structure.

Both parts are represented as binary vectors. It is assumed that users' metadata are already binary, while the computable features are converted after a calculation.

Both proposed meta-learning techniques suggest the users are grouped. The best base model must be chosen for each users' group U from the whole set of models M. The selection is performed by the equation

$$m_{best} = \underset{m \in M}{\operatorname{argmax}}\left(\sum_{u \in U} f(m, u)\right), \tag{1}$$

where $f(m, u)$ is the utility function for the model m and the user u that grows together with the model's performance. The definition of such a function depends on the error metric in use. Since RMSE is highly applicable to the task of rating prediction, the utility function is defined as follows.

Let $\mathrm{r} \in \mathbb{R}^N$ be the vector of actual ratings r_j, $\widehat{\mathrm{r}}_m \in \mathbb{R}^N$ be the vectors of predictions $\widehat{r}_{mj}$ which model m has given for the row j. The vector $\mathrm{e}^{min} \in \mathbb{R}^N$ contains values

$$e_j^{min} = \min_{m \in M}((\widehat{r}_{mj} - r_j)^2) \tag{2}$$

equal to minimal squares of error for each row. Thus, the utility function for the model m and the user u is computed by the formula

$$f(m, u) = \sum_{u,j \in 1..N} (e_j^{min} - (\widehat{r}_{mj} - r_j)^2), \tag{3}$$

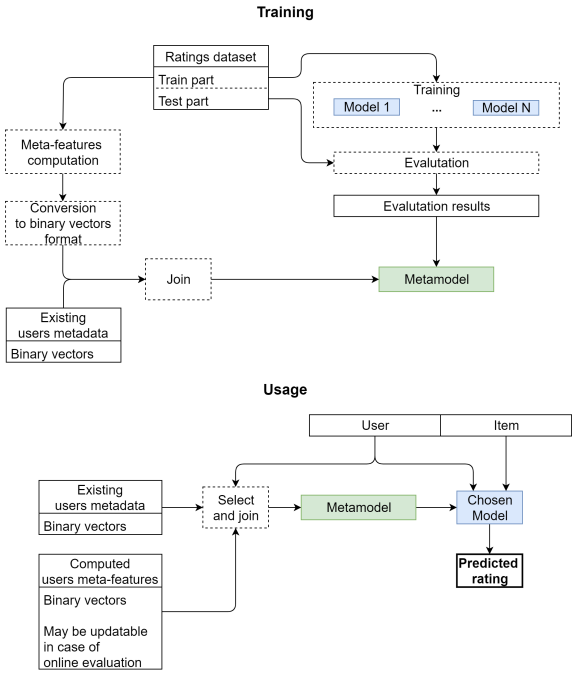

Fig. 1. The proposed recommender system structure

i.e. as the cumulative difference between the minimal squared errors and the models' squared errors.

When one chooses an algorithm for a users' group, they should consider the fact that several base-level models can show roughly equal performance, but one of them is the best on a whole dataset, i.e., fits the task and its domain more than others. Another model may show a higher level of utility not only because of its accordance with this users' group but due to the existence of outlying values in a training dataset, causing errors in modeling of interests. It may lead to the situation when a wrong base-level model that has learned several random records, but not the actual patterns of users' behavior, is selected.

Consequently, there should be a strong reason to prefer a particular algorithm for a users' group over the overall best one. In this research, the stated problem is solved as follows. After the model m_{U_best} that has the highest utility for a group is chosen, it is calculated how much higher its utility is compared to the one of the overall best model $m_{overall\_best}$. If this value is lower than the constant γ, the model $m_{overall\_best}$ is used for a users' group.

Since a utility function may take strictly non-negative values as well as strictly non-positive ones, the final selection of a model m_U for a users' group U

is performed by equations:

$$utility_U = \sum_{u \in U} f(m_{U\_best}, u), \tag{4}$$

$$utility_{overall} = \sum_{u \in U} f(m_{overall\_best}, u), \tag{5}$$

$$k_{imp} = \max\left(\frac{utility_U}{utility_{overall}}, \frac{utility_{overall}}{utility_U}\right), \tag{6}$$

$$m_U = \begin{cases} m_{U\_best} & \text{if } k_{imp} >= \gamma \\ m_{overall\_best} & \text{if } k_{imp} < \gamma \end{cases}. \tag{7}$$

The meta-learning techniques used within the method are described below.

2.1 Computation of Meta-Features

Meta-features, which are commonly used in metamodel training, can be separated into three groups [7].

1. *Statistical and/or information-theoretical meta-features* describe a dataset or its parts using particular metrics from corresponding sciences, which are applied to data analysis. Such metrics provide certain knowledge about users' behavior and can be used for distinguishing and grouping them. Examples of these features are the record count, the expected value for separate columns, their entropy, skewness, and kurtosis.
2. *Base-level model features.* This group includes hyperparameters and the features of trained base-level models. The examples of the latter include the leaf count in a decision tree or weights of input neurons in the feed-forward network. These metrics can be used to discover implicit dependencies between a model and separate parts of a dataset.
3. The last kind of metrics that are commonly used in meta-features computation is *"landmarkers"*, i.e., fast and rough estimations of different models' performance. To calculate them, one can use simplified models or small parts of a whole dataset.

In terms of the current research, the meta-features from the first group are of particular interest. Base-level model features are not studied, since it is assumed that all the models are of different kinds, and not the ones of a single kind that differ only by hyperparameters. The usage of "landmarkers" is also not studied, as the stated task implies the hybridization of several models that are trained already.

Several statistical features have been chosen; they describe rating dataset parts that correspond to separate users. For each of these features, the algorithm of conversion to 0-or-1 columns is described. The meta-features count has been restricted to three to avoid the increase in the width of the metamodel training dataset. The meta-features are shown in Table 1.

Table 1. Meta-features used in the research

Meta-Feature	Conversion to the Binary Format
Relative count of user's ratings. Equals to count of user's rating divided by the average rating count per user in a whole dataset.	Three columns for ranges $[0; 0.75)$, $[0.75; 1.25)$, $[1.25; +\infty)$.
Standard deviation of user's ratings.	Two columns for ranges: $[0;$ 0.1 of maximal possible rating), [0.1 of maximal possible rating; $+\infty)$.
User's ratings skewness.	Two columns for positive and negative values respectively.

2.2 User Classification

Since the input consists of binary vectors, a classifier can distinguish any two users having non-equal metadata vectors. Thus, the set U is defined as the group of users having equal metadata vectors. The best model is selected for each set U using formulas (1) – (7). These actions build the dataset for a classifier, which associates the metadata vector with the base model as a class label. Obviously, if the dataset includes all the possible varieties of binary metadata vectors, the classifier is not needed. However, most likely, the dataset does not cover all of them.

It is considered that the model count is no less than three, so it is better to train an ensemble of binary classifiers using the "One-vs-Rest" strategy. Since the metadata consists of binary vectors, the ensemble can be effectively founded on any decision-tree based classifiers. In this research, the random forest classifier is used.

2.3 User Clustering

The clustering-based technique brings a specific problem. A clusterer can consider some users "noise" and not attach them to any group. In this case, the meta-learning process is organized as follows.

At first, the overall set of users' metadata is clustered. As a result, several clusters are obtained, each of which can be considered as a target group U, along with the separate part of users not belonging to any cluster. The best models for clustered users are chosen using the formulas (1) – (7). For the "noise" group, a metamodel uses the overall best base model, since it is incorrect to consider these users as a neighborhood of any kind.

The task of clustering of binary vectors can be considered as the one of point clustering in the multi-dimensional Euclidean space. It is possible if at least one of the two following conditions is met [8]. The first one is that a dataset must not be very sparse. The second one states that a dataset must not be very wide. It is challenging to provide more precise restrictions, but when both conditions are not met, a clustering model trains on a wide, sparse dataset. In this case, distance metrics applied in Euclidean spaces become almost useless and cannot

be utilized to distinguish points properly. Nonetheless, it is assumed in this research that the count of binary columns is less than 100, while the acceptable density can at least be provided by the computable meta-features.

Thus, the OPTICS model with cosine similarity as a distance metric has been chosen initially to perform clustering. It has been stated heuristically that the farthest two users can differ in no more than two columns and coincide in at least one another column. So, according to the distance metric, the maximal cluster size equals $1 - 1/(\sqrt{2} * \sqrt{2}) = 0.5$.

However, the OPTICS model has a significant disadvantage. Its accuracy is high, but the training process on the datasets used in the experimental study can take an unacceptably long amount of time. This problem can be solved by using the HDBSCAN model [9]. Like OPTICS, it is an improved version of DBCSAN that performs hierarchical clustering without the fixed restriction of maximal cluster radius. It has been discovered empirically that the HDBSCAN model with L2 distance metric shows almost the same level of performance on the used datasets as the OPTICS one with the stated above configuration. It shows highly similar results, but learns much faster. So, this is the HDBSCAN model that is used further.

3 Results and Discussion

The initial pool of algorithms in this research included five collaborative filtering models, implemented in the Surpriselib library [10]: SVD, SVD++, Baseline only, KNN Baseline, Co-Clustering.

Five datasets [11,12,13] were used. They are described in 2. The presented features do not include computable ones. Metadata rows were converted to binary vectors manually in such a way that most of the information was saved, but the width of the dataset did not increase drastically.

Table 2. Summary description about the datasets used in the experimental study

Dataset	No. of ratings	No. of users	No. of user features	No. of items	No. of item features	Rating scale
Movielens 100k	100 000	943	32	1 682	19	1-5
Movielens 1M	1 000 000	6 040	29	3 883	18	1-5
Amazon Phones	82 816	55 975	6	793	20	1-5
Rent the Runway	192 463	105 509	41	5 851	68	1-10
Modcloth	82 723	47 911	17	1 377	8	1-5

Each dataset was used in the following way. It was shuffled randomly and split into three parts A, B, and C with 60, 20%, and 20% of records from the initial dataset, respectively. Each model from the initial pool was trained on part A and evaluated on part B. Based on the evaluation results, the pool of the best base models was built. Next, four metamodels were trained, each of which was classifying/clustering users/items. The γ constant was defined as 1.25. During

the process of classifier training, only 70% of users/items from part B were used. Finally, all the base models and created metamodels were evaluated on the part C. The RMSE was used as the evaluation metric.

The evaluation results are presented in Table 3. Additionally, the table includes the results for a perfect abstract metamodel that chooses the truly best algorithm for each user or item.

Table 3. Evaluation results (RMSE). The best results among the models and the metamodels are bold-faced. MM is for "Metamodel".

Model	Movielens 100k	Movielens 1M	Amazon Phones	Rent the Runway	Modcloth
SVD	0.9433	0.8941	1.5569	1.3941	0.9522
SVD++	**0.9306**	**0.8795**	1.5992	1.4043	0.9547
KNN Baseline	0.9434	0.9133	1.5649	-	**0.9506**
Baseline Only	0.9350	0.9039	**1.5525**	**1.3889**	0.9877
Co-Clustering	0.9713	0.9224	1.6088	1.5531	1.0263
Perfect (by user)	0.8947	0.8558	1.3921	1.2280	0.7877
Perfect (by item)	0.8866	0.8634	1.5003	1.3374	0.9310
MM (user class.)	**0.9296**	0.8798	**1.5371**	1.3752	**0.9144**
MM (item class.)	0.9324	0.8796	1.5463	1.3897	0.9505
MM (user clust.)	0.9317	**0.8795**	1.5375	**1.3746**	**0.9144**
MM (item clust.)	0.9320	**0.8795**	1.5492	1.3894	0.9504

In four out of five cases, the best metamodel had better performance than the best base-level one, since its RMSE on a whole dataset was lower. The greatest benefit from the meta-learning was taken for the Modcloth dataset. In the case of Movielens 1M dataset, the best metamodel had shown exactly the same result as the best base-level one.

The improvement achieved by applying the proposed method was different for different datasets. It is possible to suggest that it depended both on the best base-level model accuracy and on the level of maximal theoretical improvement that could be obtained by per-user (or per-item) algorithm selection. Figure 2 illustrates the relation between these values.

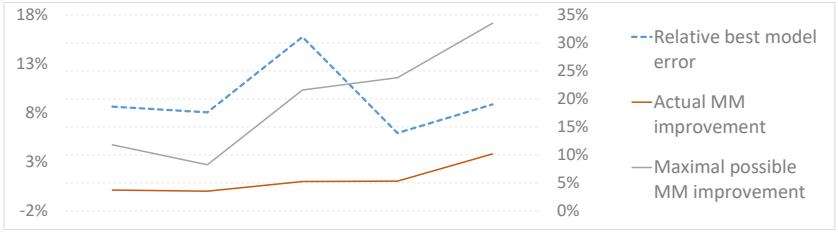

Fig. 2. The relation between the relative best model error and the metamodel improvement. Metamodel improvements are on the primary axis, and relative best model errors are on the secondary axis.

According to the graphs, the actual improvement that the metamodel usage made could be truly dependent on the maximal possible one. At the same time, the possibility of such dependency on the relative best base-level model error was questionable due to the results for the Rent the Runway dataset. It was important that previous experiments that had been performed without statistical meta-features and the usage of the constant γ showed that this dependency could exist, and the results had been generally worse. It allows for affirming that combining explicit metadata together with computable meta-features and performing a restricted selection of the best model for a users' (or items') group is reasonable and increases the accuracy of metamodels.

Both the numerical results and the graphs showed that the actual improvement was far from theoretically possible. Along with that, training accuracy was reasonably good. The graphs in Figure 3 demonstrate how often metamodels have selected the truly best algorithm for a user or an item.

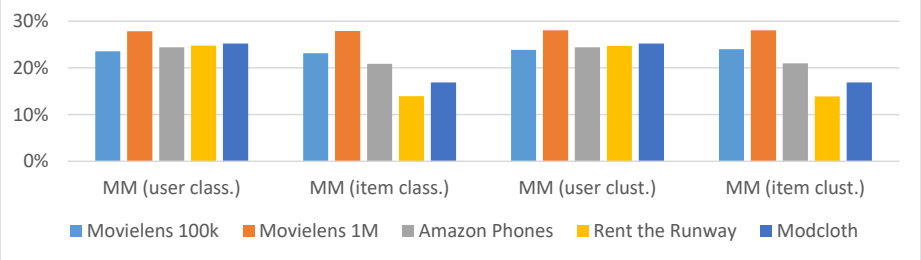

Fig. 3. A perfect algorithm selection rate per metamodels and datasets

According to these measurements, classification and clustering models had a rather similar performance that led to the best selection rate that equaled 24-28%. Worse results showed by item-based models for three out of five datasets could be explained by relatively small item counts in them. It allows for concluding that the bottleneck lies in both dataset size and the complexity of determining the truly best model for users' or items' groups before training.

A pending question within this research is how to choose an optimal value for the constant γ. Figure 4 shows the ratios between the lowest metamodel RMSE and the lowest base-level model RMSE with different γ values. When γ is less or equal to 1, it is equivalent to γ to be ignored (following formulas (4) – (7) from Section 2).

In three out of five cases, the usage of the γ constant was effective. In the case of the Modcloth dataset, there was almost no difference between the evaluated γ values. For the Amazon Phones dataset, the restrictions in the algorithm selection the γ constant brought had affected metamodels' performance negatively. Consequently, the optimal γ value depended on the used dataset and performance of the base-level models. Since the evaluation of several metamodels with different γ values has a high computational cost, it is highly preferable to deter-

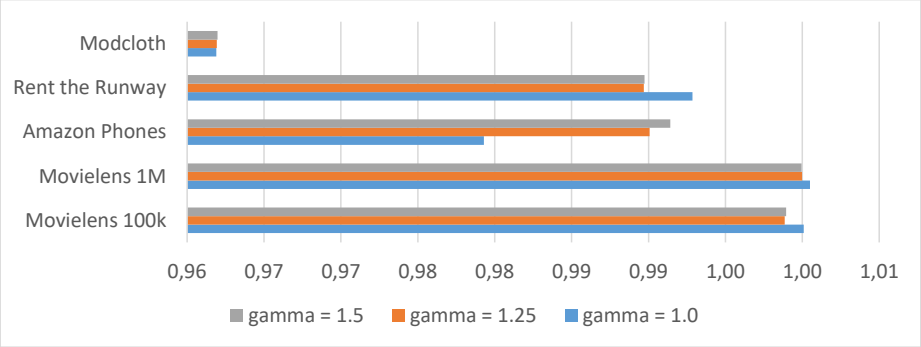

Fig. 4. The lowest metamodel RMSE divided by the lowest base-level model RMSE with different γ values

mine its value before the training. Possibly, this task can be done on the basis of the base-level models' errors. This problem requires additional research.

The impact of the usage of the metamodels on the time needed for recommendation generation has also been considered. Figure 5 shows the average times the base-level and meta- models need to make a single rating prediction on the used datasets. The experiments have been performed using the Intel Core i7-7700HQ CPU and DDR-4 RAM on 2133 MHz frequency. It is significant that the Surpriselib framework evaluates test requests individually, and not in batches. This peculiarity allows the measured time values to be as realistic as possible.

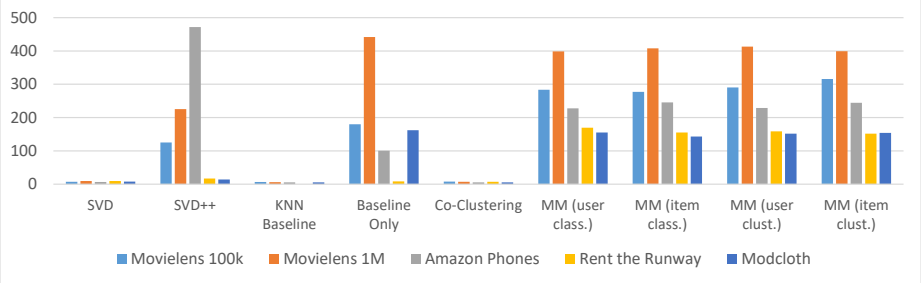

Fig. 5. Average prediction time per record (microseconds)

Depending on a dataset, when a metamodel was used, the average prediction time was ranged between 143 and 413 microseconds. This interval allowed an RS to process thousands of requests per second. The speed of base-level models, in some cases, was comparable to the metamodels' one, but, in some other cases, it was noticeably higher. However, the base-level models have particularly native implementation, while the metamodels are implemented in pure Python without

any optimizations. Besides, it is obvious and seen from the graphs that the speed of a hybrid model depends on the speed of all the models it combines directly. Thus, the prediction time of a recommender system that uses the metamodel is enough for the production but it can be decreased by optimizations and the native implementation.

The results show that all the metamodels are equally performant. To choose one of them in real-world tasks, one should consider the number of users and items in the available data. The classification-based metamodels are preferred in this case, since they can be trained faster than the clustering-based ones. To employ the proposed method in an existing system that includes several recommenders, one can convert users' metadata to the binary vectors format, train the classifier using the existing recommenders' evaluation results, and redirect recommendation requests to the metamodel.

Although the improvement is moderate, the usage of the metamodel that utilizes the peculiarities of a particular user can noticeably increase the quality of their experience. To examine this assumption, one should conduct additional research that aims at the evaluation of ranking measures and uses the appropriate datasets.

4 Conclusions

The presented work describes a solution to the problem of recommendation algorithm selection for particular users or items within a single service. The paper proposes the middle-level automated meta-learning method that uses the rating prediction algorithms' evaluation results as well as the binary representation of both users' (or items') metadata together with their computable meta-features. Two techniques of creating the metamodel are suggested; they are based on classification and clustering, respectively. Both techniques are applicable for users' features as well as for an items' features.

The metamodels have been evaluated on five open datasets, with the usage of collaborating filtering models on a base level. The experiments prove that the usage of the proposed method is reasonable and effective. The performance in comparison to the base-level models improved up to 3.81% of RMSE value in a best-case scenario and did not decrease in any case. The measures obtained from the evaluation have been analyzed in detail, and the possible directions of future work been described, including the ones aimed at performance enhancement. Current results can be used in real-world systems; they can improve the experience of their users. The additional advantage of the method is that the process of creation of a metamodel is almost automated, and the only manual task is to convert explicit metadata to the binary format.

References

1. Ricci, F., Rokach, L., Shapira, B. eds: Recommender Systems Handbook. Springer US (2015). https://doi.org/10.1007/978-1-4899-7637-6

2. Cunha, T., Soares, C., de Carvalho, A.C.P.L.F.: Metalearning and Recommender Systems: A literature review and empirical study on the algorithm selection problem for Collaborative Filtering. Information Sciences. 423, 128–144 (2018). https://doi.org/10.1016/j.ins.2017.09.050
3. Collins A., Beel J., Tkaczyk D. One-at-a-time: A Meta-Learning Recommender-System for Recommendation-Algorithm Selection on Micro Level. ArXiv abs/1805.12118 (2018).
4. Cunha, T., Soares, C., de Carvalho, A.C.P.L.F.: Selecting Collaborative Filtering Algorithms Using Metalearning. In: Machine Learning and Knowledge Discovery in Databases. pp. 393–409. Springer International Publishing (2016). https://doi.org/10.1007/978-3-319-46227-1_25
5. Cunha, T., Soares, C., de Carvalho, A.C.P.L.F.: CF4CF. In: Proceedings of the 12th ACM Conference on Recommender Systems - RecSys '18. ACM Press (2018). https://doi.org/10.1145/3240323.3240378
6. Meltsov V., Novokshonov P., Repkin D., Nechaev A., Zhukova N.: Development of an intelligent module for monitoring and analysis of client's bank transactions. In: Conference of Open Innovations Association, FRUCT. N 24. 255-262 (2019). https://doi.org/10.23919/fruct.2019.8711931
7. Brazdil, P., Giraud-Carrier, C., Soares, C., Vilalta, R.: Metalearning. Springer Berlin Heidelberg (2009). https://doi.org/10.1007/978-3-540-73263-1
8. Smieja, M., Hajto, K., Tabor, J.: Efficient mixture model for clustering of sparse high dimensional binary data. Data Mining and Knowledge Discovery. 33, 1583–1624 (2019). https://doi.org/10.1007/s10618-019-00635-1
9. Campello, R.J.G.B., Moulavi, D., Sander, J.: Density-Based Clustering Based on Hierarchical Density Estimates. In: Advances in Knowledge Discovery and Data Mining. pp. 160–172. Springer Berlin Heidelberg (2013). https://doi.org/10.1007/978-3-642-37456-2_14
10. Surprise – A Python skicit for recommender systems. http://surpriselib.com/
11. Harper, F.M., Konstan, J.A.: The MovieLens Datasets. ACM Transactions on Interactive Intelligent Systems. 5, 1–19 (2015). https://doi.org/10.1145/2827872
12. Amazon Cell Phones Reviews — Kaggle - https://www.kaggle.com/grikomsn/amazon-cell-phones-reviews
13. Misra, R., Wan, M., McAuley, J.: Decomposing fit semantics for product size recommendation in metric spaces. In: Proceedings of the 12th ACM Conference on Recommender Systems - RecSys '18. ACM Press (2018). https://doi.org/10.1145/3240323.3240398