



OPEN Active learning algorithm for alleviating the user cold start problem of recommender systems

Toon De Pessemier[✉], Bruno Willems & Luc Martens

A key challenge in recommender systems is how to profile new users. A popular solution for this problem is to use active learning strategies. These strategies request ratings for a small set of carefully selected items to reveal the preferences of new users. In this paper, we propose a new decision tree-based algorithm for selecting these items. Treating the recommender system as a black box, the ratings collected from interviewing new users are passed on to the recommender system with the intention of improving its performance. Extensive offline evaluation with two data sets and various recommender algorithms shows that our algorithm does indeed improve the performance of the underlying recommender algorithm if users are able to rate most of the items that are presented to them during the interview. However, online evaluation with 50 real users could not prove that our algorithm does indeed have a positive impact on the performance of the underlying recommender algorithm. This reveals the discrepancy between offline and online evaluations of active learning techniques applied in the context of recommender systems. This is due to the fact that real users are not always able to rate the item selected by the active learning algorithm and therefore cannot provide the requested information, in contrast to many machine learning scenarios where the labeling of all samples is possible. Hence, further research is required to provide more certainty regarding the impact of active learning strategies on recommender algorithms.

Keywords Active learning, Decision Trees, User Cold Start, Recommender Systems

Most recent decades saw the rise of massively popular web services, such as Youtube, Amazon, Netflix and Spotify. As the number of users and items on these platforms started to rapidly increase, so did the need for an intelligent system that would help users find the most relevant items on the platform. A race had begun to design and build the best recommender system (RS) possible in order to serve this massive number of users.

Recommender systems (RSs) are software tools and techniques designed to assist users in the content selection process and therefore try to solve the problem of information overload by offering personalized suggestions to users. A RS relies on feedback provided by the user to determine what the user's preferences are towards certain item categories or individual items. The better a RS is able to understand what a user likes and dislikes, the better it is at matching users with items on the platform.

Filtering techniques used by RSs can be divided into three main categories, i.e. content-based, collaborative and hybrid filtering techniques. Each technique has its own unique approach when it comes to selecting the most appropriate items based on the user's preferences. Content-based filtering (CBF) techniques are domain-dependent since they require knowledge about the items and their attributes. The RS builds a user profile that captures the user's preferences for these item attributes. CBF uses vector space models such as Term Frequency Inverse Document Frequency, probabilistic models such as Naive Bayes Classifiers, Decision Trees and Neural Networks to model the relationship between different items¹⁻⁴. CBF tries to recommend items similar to the ones the user liked in the past. Collaborative filtering (CF) techniques are domain independent prediction techniques used for content that cannot be easily and adequately described by metadata, such as movies and music^{5,6}. CF uses a user-item matrix from which it recommends items to users. This matrix captures the preferences for items by users. Hybrid filtering (HF) combines multiple filtering techniques⁷⁻⁹. This can increase the accuracy and effectiveness of the recommendations because the disadvantages of one technique can be compensated by the advantages of another technique.

A major problem with RSs is that they often fail to generate suitable recommendations for new users. This problem is known as the user cold start problem¹⁰ and it is caused by a lack of information (i.e., ratings) regarding

Ghent University, Belgium, Imec, Belgium, Waves, iGent - Technologiepark 126, Ghent 9052, Belgium. ✉email: toon.depessemier@ugent.be

the preferences of a new user. Especially CF-based RSs struggle with this problem since they solely rely on ratings collected from its users to determine which items would be interesting for the target users¹¹.

A promising solution for the user cold start problem is active learning (AL). AL is often used in the machine learning domain when acquiring large amount of labeled training data is expensive or even impossible. AL specifies how to select data points such that when they are given a label, the machine learning model gains as much useful information as possible with the least amount of labeled training data^{12,13}. In cases where gathering labeled samples is difficult, time consuming, or expensive, AL is an excellent solution for screening valuable samples and reducing sample labeling time, e.g., for face detection for rail transit passengers¹⁴. Applying AL to gather labeled data is also beneficial in cases where decisions have to be made based on an imbalanced number of samples over the different classes, e.g., for the identification of daily activities such as “standing” vs “jumping”¹⁵. Within the RS domain, AL tackles the problem of obtaining high quality information that better represents the user’s preferences and improves the recommendation quality. This is done by carefully selecting a set of items and asking the user to rate these items. With AL, the goal is to acquire a limited number of additional ratings that will help the RS generate better recommendations for its users¹¹. Asking a new user to accurately rate multiple items requires considerable effort from this user. And thus, asking too many questions could frustrate the user and make him/her quit the system prematurely. On the other hand, asking too few questions might not provide the RS with enough useful information regarding the user’s preferences to generate accurate recommendations.

This paper focuses on applying AL to improve the performance of various popular recommender algorithms. More specifically, these recommender algorithms are treated as black boxes and are simply given ratings collected by the AL algorithm. The goal of this research was to design a new AL algorithm that is able to have a positive impact on the performance of several recommender algorithms without making any assumptions about these recommender algorithms apart from the fact that they require ratings to determine suitable recommendations.

The remainder of this paper is organized as follows. In Section “[Related work](#)”, an overview is given of existing AL strategies currently used within the RS domain. Our newly proposed AL algorithm is presented in Section “[Methods](#)”. Section “[Results and discussion](#)” discusses the results obtained through both offline and online evaluation of the proposed AL algorithm, as well as the limitations of the study. Finally, a conclusion is given in Section “[Conclusion](#)”.

Related work

An excellent overview of various AL strategies is presented in¹¹. To summarize, an AL strategy is either non-personalized or personalized. The former means that each new user is asked to rate the same set of items while the latter adjusts its set of items based on the ratings given by the user, i.e. each new user must rate a different set of items. Furthermore, these two categories can be further divided into a single-heuristic and combined-heuristic subcategory. Single-heuristic strategies implement a unique item selection rule while combined-heuristic strategies combine multiple single-heuristic strategies in order to better estimate items’ usefulness for the RS. These heuristics can for instance try to reduce the uncertainty of the RS regarding the user’s preferences by finding the most informative items, or try to increase the chance that the user will be able to rate the selected items by focusing on popular items.

A classic example of a non-personalized, single-heuristic AL strategy is a popularity-based algorithm¹⁶ which selects items that have received the highest number of ratings. Presenting new users with popular items increases the chance that they will be able to rate these items and thus, more ratings can be elicited from the user. However, although more ratings can be obtained, the information gained regarding the user’s preferences tend to be much lower since popular items are liked by most users and will probably also be liked by the new user. Furthermore,¹⁶ also introduces a non-personalized, combined-heuristic strategy called *Classique*. This strategy extends a list of randomly selected items for rating elicitation by inserting a popular item that comes from a manually selected list of popular movies. Hence, it attempts to improve the probability that users have experienced and can rate at least one of the presented items.

This paper focuses on personalized, single-heuristic strategies in general and user partitioning strategies in particular. These strategies attempt to partition existing users into groups of users with similar tastes by requesting new users to rate a customized set of items.¹⁷ suggests the use of decision trees (DTs) to create adaptive interviews for new users. The authors of¹⁷ introduce an elegant algorithm for constructing ternary DTs. New users start at the root of the decision tree (DT) and traverse towards the leaf nodes based on the ratings that they give to each of the items selected by the DT. As the DT is ternary, users can rate an item with Like, Dislike, or Unknown.¹⁸ made a successful attempt to incorporate matrix factorization into the tree structures to improve the rating prediction accuracy of the DT. Additionally, the authors of¹⁹ propose to upgrade the 3-way tree structure to a 6-way tree structure. They argue that a 6-way split, i.e., one child node for each each rating from 1 to 5 and one child node for the “Unknown” response, can more accurately detect users’ tastes. We chose to further investigate and develop the AL solution based on DTs since these structures are powerful tools to create adaptive interviews for rating elicitation. They also possess the major benefit of being very transparent and explainable, two characteristics that are highly appreciated within the RS domain. An important similarity between research performed in^{17,18} and¹⁹ is that the DT serves as a prediction model and thus serves the purpose of a RS. Instead, we would like to pass the ratings elicited by the DT to a proper RS and measure the impact that those ratings have on the prediction performance of the underlying recommender algorithm. We intend to investigate the impact AL has on various recommender algorithms when unary ratings are used instead of the classic 5-star ratings. The usage of unary data in our research is motivated by the choice of many online systems to eliminate the traditional 5-star rating system and only rely on implicit data, such as clicks or purchases, for recommendations. Furthermore, we demonstrated that it is possible to combine multiple personalized, single-heuristic strategies into one personalized, combined-heuristic strategy. This can for example be done via a voting mechanism where each strategy votes on the items that it would like to request ratings for from the user.

Methods

This section discusses the main contributions of this paper. First, a detailed description is given of a new AL strategy based on DTs. Next, an overview is presented of the framework that has been used to evaluate the impact AL strategy has on the performance of various recommender algorithms given a certain data set.

Decision tree algorithm

This paper proposes a new AL that is based on a DT algorithm described in¹⁷. It is therefore very important to briefly explain this algorithm. In¹⁷, a ternary DT is constructed and new users have to traverse this tree starting at the root of this tree. Each interior node of the DT is labelled with one item (query). The user is asked to rate this item and based on his response, he traverses down to one of the child nodes. Since the DT of¹⁷ is ternary, the possible responses a user can give are Like, Dislike and Unknown. The latter is appropriate when the user can not rate the item because he is not familiar with it. This rating elicitation process is repeated until the user arrives in one of the leaf nodes of the DT.

Furthermore, each tree node also represents a set of users and is able to predict ratings for items by taking the average of ratings from the users that it represents. To be more precise, let I be the entire set of items, U the entire set of users and R the entire set of ratings between the users in U and items in I . Each tree node t represents a set of users $U_t \subseteq U$. The profile of item i in node t is defined as D_i^t :

$$D_i^t = \{(u, r) \in U \times R \mid u \in U_t\} \quad (1)$$

Each node t is then able to generate a rating prediction \hat{r}_{ti} for the target item i by taking the average of its users' ratings for this item i :

$$\hat{r}_{ti} = \frac{\sum_{(u,r) \in D_i^t} r_{ui} + \lambda_1 \hat{r}_{si}}{|D_i^t| + \lambda_1} \quad (2)$$

Note that equation 2 includes the prediction of the parent node s for the prediction of item i in child node t . This is done to avoid overfitting. Applying regularization with regularization factor λ_1 is especially important when the number of ratings in item profile D_i^t is limited. We used $\lambda_1 = 200$ as further optimizing this hyperparameter was considered irrelevant for the purpose of our research. Since tree nodes can make rating predictions, it is also important to define an appropriate rating prediction error e_{ti} for node t and item i :

$$e_{ti} = \sum_{(u,r) \in D_i^t} (r - \hat{r}_{ui})^2 \quad (3)$$

The total rating prediction error for node t is defined as the sum of its rating prediction errors for each of the items, i.e., $e_t = \sum_{i \in I} e_{ti}$. The construction of the DT is done in a top-down manner by repeatedly splitting a set of users into smaller sets of users with similar taste. For each internal tree node, the best splitting item is the item that partitions the users into three sets of users such that the splitting error is minimized. At tree node t , three candidate child nodes are defined: $tL(i)$, $tD(i)$ and $tU(i)$ representing users who like, dislike and can not rate item i respectively. Assigning item i to node t would result in a splitting error $Err_t(i) = e_{tL} + e_{tD} + e_{tU}$. Note that error e_{tU} is entirely determined by the rating prediction error of the parent node because users in child node $tU(i)$ do not have a rating for item i . The candidate item that will be assigned to node t will be the one with the lowest splitting error $Err_t(i)$. Instead of considering all items as candidate splitting items, an optimization is to only consider a subset of items based on popularity. More specifically, only the items that have the most ratings within this specific user segment (i.e. within this tree node), are considered. Because these items have the most ratings within the segment, the split will lead to a lower error. This idea was suggested in¹⁸ and is referred to as the Most Popular Sampling (MPS) method. By further reducing the number of items based on their amount of ratings, the building speed of the tree model can be increased. However in our experiments this was not needed since all models were created in a time frame of a few minutes.

Once the chosen candidate item has been assigned and the split has been made, it is removed from the set of candidate items that is passed on to the three subtrees. After partitioning the remaining users among their corresponding subtrees, splitting is continued recursively with each of the created child nodes unless the split would result in one or more of the following stop criteria:

1. The depth of the tree would exceed a desired threshold.
2. The number of remaining users would fall below a desired threshold.
3. The splitting error would increase compared to the splitting error of its parent node.

Now, our DT algorithm differs from the DT algorithm described above in two major aspects. The first difference is the number of child nodes that are created after each split. In¹⁷, three child nodes are created representing the users who like, dislike the splitting item, and those who can not rate the splitting item. Since this excludes RSs that rely on unary user ratings to make their predictions, we opted to try out a two-way split. This means that our algorithm creates two child nodes after each split representing users who like the splitting item and the users who can not rate the splitting item. For datasets that do have negative feedback (e.g. dislikes), our AL algorithm can still be used by ignoring the negative feedback. Although some information (i.e. the negative feedback) is not

used by choosing for two nodes, we don't see this as a serious limitation. After all, in many online systems the histogram of ratings is skewed towards the positive ones. In other words, the majority of the ratings are positive and by ignore the negative evaluations, only a small fraction of information is omitted. For the same reason, YouTube, for example, abandoned their star-rating system²⁰.

The second difference is the criteria that define whether or not to continue splitting the DT. We deemed it crucial that sufficient information (i.e., likes) is collected from new users traversing the DT. To this end, the stop criteria were altered in the following way:

1. The depth of the tree would exceed a desired threshold.
2. The number of likes collected would exceed a desired threshold.

In other words, collecting sufficient likes from new users is deemed more important from the perspective of the RS since it will be given the ratings collected from the users after they are done traversing the DT constructed by our algorithm. Nonetheless, should the number of remaining users fall below a desired threshold or the splitting error increase after the split, then we propose to nonetheless continue splitting after reintroducing all users.

Data sets

The MovieLens 1M²¹ and Publiq data sets are used to construct the DT algorithm and train, test and evaluate the performance of the recommender algorithms. The former is a freely available and well-known data set that contains ratings and metadata of users and movies. The latter is a proprietary data set that contains ratings and metadata of Flemish cultural events. These two data sets are quite different to each other, as indicated in Table 1, and it is therefore important to give an overview of their characteristics.

The large difference in average number of ratings per user between the data sets is caused by MovieLens' guarantee that each user in their data set has at least 20 ratings. This might be perceived as a bit artificial since a RS will usually also have to deal with new users who have very few ratings simply because they have not yet had enough time to rate 20 items or more. Consequently, the rating density is much higher for the MovieLens 1M data set compared to the Publiq data set. Another difference is the ratings scale. Whereas MovieLens uses the traditional 5-star rating system, the Publiq data set only contains likes. To ensure a fair comparison, the 1-5 star ratings from the MovieLens 1M data set were transformed to unary ratings (value 1). Ratings of 4 stars or more are considered a positive and converted into a value of 1, while the other ratings are ignored.

Recommender algorithms

As the goal of this paper is to measure the (positive) impact that our AL algorithm has on the performance of RSs, the decision was made to rely on RecPack²², an easy-to-use, flexible and extensible toolkit for top-N recommendation with implicit feedback data. This toolkit ensures the correct implementation and evaluation of various recommender algorithms. RecPack's source code was slightly extended to provide the recommender algorithms with the ratings collected from the new users after they have traversed the DT. Several recommender algorithms were selected based on their popularity:

1. Item K Nearest Neighbours model (Item KNN)²³
2. Embarrassingly Shallow Autoencoders (EASE)²⁴
3. Sparse Linear Methods (SLIM)²⁵
4. Weighted Matrix Factorization algorithm (WMF)²⁶
5. Transparent & Explainable Aspect Space Embedding Recommender (TEASER)^{27,28}

TEASER is a new hybrid recommender algorithm as it relies on HF techniques to predict the ratings of its users. In other words, TEASER uses both item ratings as well as item attributes (metadata) to make its rating predictions whereas the other four recommender algorithms solely rely on item ratings as their only source of information regarding the preferences of its users. By including item metadata and making the user profiles interpretable, TEASER has the additional advantage of transparency because it enables intuitive explanations towards the user. Although this transparency trade-off induces a slightly lower accuracy for some data sets, TEASER is still able to provide decent recommendations in all circumstances²⁸. Therefore it was chosen as the main recommender solution in the online evaluation as well (Section [Online evaluation](#)).

	MovieLens 1M	Publiq
rating scale	1-5	1 (unary)
rating density (%)	4.4684	0.0098
# ratings	1,000,208	615,777
# min ratings/user	20	1
# avg ratings/user	166	1.36
# max ratings/user	2314	488
# users	6,040	452,395
# items	3,706	13,826

Table 1. Overview of data set characteristics.

Experimental evaluation

The aim is to benchmark the performance of our DT algorithm with an offline and online evaluation. A similar approach is used for both evaluation methods, i.e., the impact of our DT algorithm on various recommender algorithms is compared with the impact of a baseline AL on these same recommender algorithms. This baseline AL algorithm presents new users with a certain number of most popular items and is therefore referred to as the Most Popular (MP) algorithm. This baseline has the advantage that popular items have a high probability that the user is familiar with these items and is therefore able to rate these. As a result, more preferences can be elicited from the user, which is a crucial aspect of AL, especially for online evaluations. This popularity-based baseline is a common evaluation approach and is also adopted by related work in the domain of active learning / meta-learning solutions for the cold-start problem of recommender systems²⁹. To ensure a fair comparison, both AL algorithms (our DT and the baseline based on popularity) are given an identical lower- and upper bound on the number of queries they can choose.

Offline evaluation

For the offline evaluation, several experiments were performed with the DT and MP algorithms, together with various data sets (section [Data sets](#)) and various recommender algorithms (section [Recommender algorithms](#)). An experiment consists of multiple steps, each of which is described below.

First, ratings provided by the data set are split into a *train* and *test* set. The *train* set is filled with ratings from training users (70%) who are randomly selected among all users in the system. The remaining users (30%) are considered test users. The training users represent existing users while the test users represent new users entering the system. Second, the ratings from the training users are used to train the AL and the recommender algorithm. In case of our DT algorithm, the ratings are used to construct the DT. Third, the test set is further split into a *test_in* and *test_out* set. The *test_in* set is filled with ratings of test users for specific items selected by the AL algorithm. In case of our DT algorithm, this would mean the ratings given to the splitting items. Figure 1 illustrates how users are partitioned and their ratings are split. Assume users 3, 4 and 9 have been randomly selected as test users. Furthermore, assume items 2, 3 and 4 have been selected by the AL algorithm for rating elicitation purposes. The ratings of the test users for these items are added to the *test_in* set (if available). Their remaining ratings are added to the *test_out* set. The *test_in* set is used by the recommender algorithm to predict the ratings inside the *test_out* set and thus, test the performance of the recommender algorithm. The authors of this paper have opted to measure the performance of the recommender algorithms using the decision support metrics: precision, recall and F1 and the rank metric: normalized discounted cumulative gain (NDCG). For the online experiment, the decision support evaluation was limited to the precision, since it was not feasible to evaluate the relevancy of all items for the test users. Precision indicates how relevant the recommendations are for the target user; recall measures the fraction of relevant items that are actually recommended; and F1 is a combination of the previous two metrics. NDCG indicates how well the recommendations have been ranked by the recommender algorithms.

Online evaluation

For the online evaluation, a user study was performed with the DT and MP algorithm. The target was to collect feedback from 50 real users. As a reward, some randomly chosen participants were given a 10 euro bol.com gift voucher. To facilitate this user study, a web application and Google form were created. This user study (usage of the web application with interview and the questionnaire) and the analysis of the user feedback were carried

	I1	I2	I3	I4	I5	I6	I7	I8	I9	I10
U1		1		1	1					
U2	1									
U3				<u>1</u>			1	1		1
U4		<u>1</u>	<u>1</u>			1				
U5				1	1	1				
U6		1					1			
U7										1
U8			1							
U9		<u>1</u>		<u>1</u>		1				
U10	1				1				1	

Fig. 1. The rating matrix: rows represent ratings from users, columns represent ratings for items.

START INTERVIEW A

START INTERVIEW B

Fig. 2. Users could be asked to complete the two interviews in a random order by clicking one of the two blue buttons.

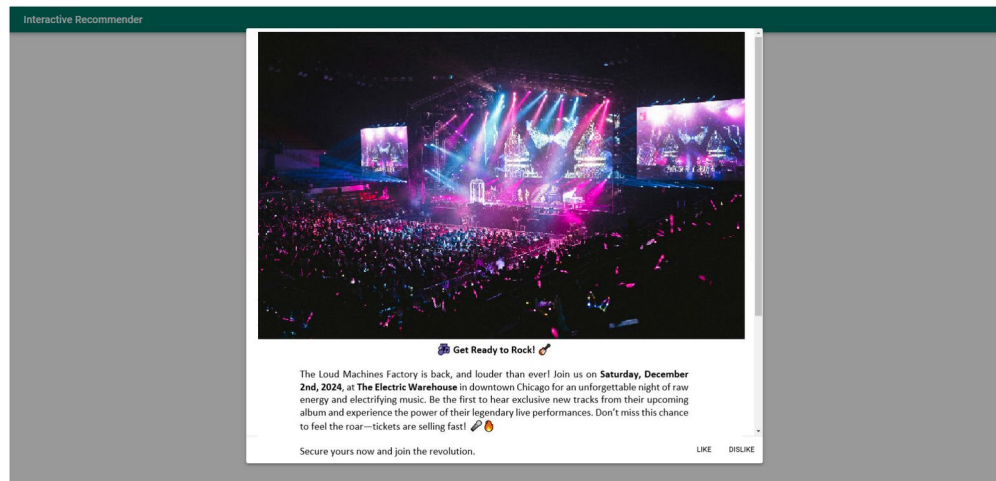


Fig. 3. Screenshot of the interview with a hypothetical event as an example. During an interview, users had to indicate if they (dis)like an event.

out in accordance with the guidelines and regulations regarding data protection and privacy. All experimental protocols were approved by the ethics committee of Ghent University (Belgium). All participants used the web application with recommendations on a voluntary basis and were informed that their click behavior was used to provide personalized event recommendations. It was clearly mentioned that their evaluations of the tool provided in the Google form were used for benchmarking the quality of the algorithms. Before the start of the user study, informed consent was obtained from all participants.

All users in the online test did the experiment with both AL strategies and evaluated the resulting recommendations using the Google form questionnaire, which we refer to as interview A and B. The web application showed how users are asked to complete interviews A and B in a random order, thereby avoiding bias resulting from the order in which the AL strategies are evaluated. Figure 2 shows how users are invited to do the interviews as a blind test, without any indication of the algorithm that is used. During these interviews, users had to rate items selected by the MP (interview A) or DT algorithm (interview B). Figure 3 shows how information is visualized for a hypothetical event. After completing an interview, users received six recommendations generated by the TEASER recommender algorithm, as illustrated in Fig. 4. It is worth mentioning that users were only given the option to like or dislike events. But as explained in Section "Decision tree algorithm", the algorithm only relies on likes and the (second) stop criterium for splitting the DT is also expressed in terms of likes and ignores dislikes. By only using the likes, a more generic AL solution is obtained, which is also usable for unary data set. Indeed, in many online applications, user feedback to feed the recommender is limited to positive-only data coming from purchases or clicks.

It was deemed that users receive enough information about the events such that they can accurately rate them. Figure 5 illustrates how the full description of the recommended event is visualized for an accurate user consideration.

Second, the Google form was used to collect feedback from the users. After completing each interview, the users were asked to enter their ratings for the six recommended items and they also had to answer a questionnaire using a 5-point Likert scale. This scale ranged from 1 = "Strongly disagree" to 5 = "Strongly agree". The questionnaire is displayed in table 2.

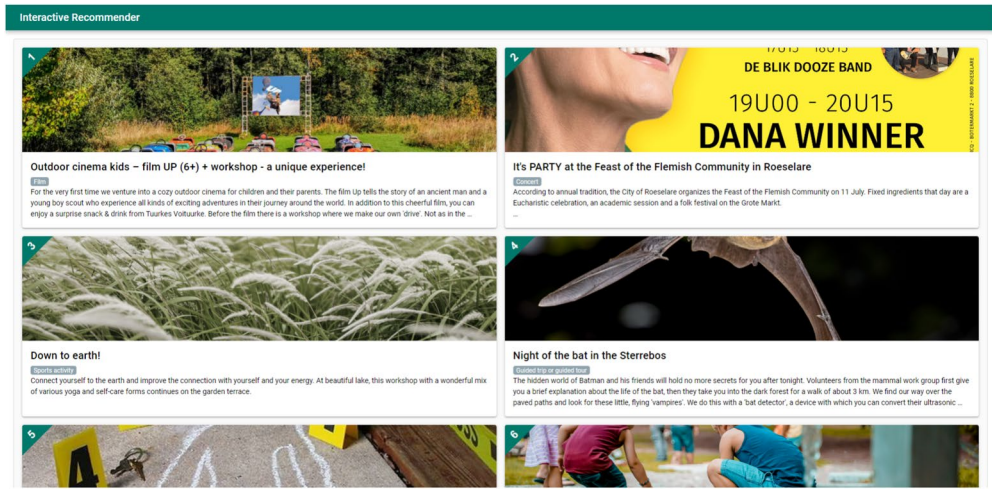


Fig. 4. After completing an interview, users were presented with a list of six recommendations selected by the TEASER recommender algorithm.

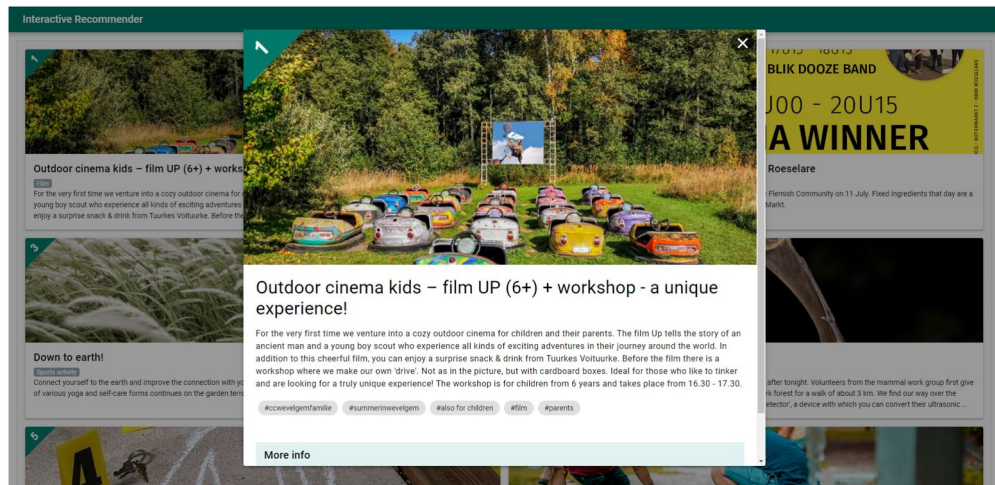


Fig. 5. Users can view the full description of a recommended event by clicking on the event.

#	Question	Answer
1	The events recommended to me are diverse.	1/2/3/4/5
2	The events recommended to me matched my interests.	1/2/3/4/5
3	The recommender system helped me discover new events.	1/2/3/4/5
4	The events recommended to me are ranked appropriately.	1/2/3/4/5
5	The recommender system gave me good suggestions.	1/2/3/4/5

Table 2. Overview of the questionnaire.

Results and discussion

Offline results

Table 3 presents an overview of the results that were collected during the offline evaluation. The first column indicates which combination of AL and recommender algorithm was tested. The first row indicates which data set was used to measure the performance of the recommender algorithm in terms of its capability to generate accurate and well ranked lists of eight recommendations for new users. The hyperparameters of each recommender algorithm were left untouched as it is the intention to adapt the AL to the recommender algorithm and not the other way around. In other words, the goal is to use AL to give a boost to the performance of existing RSs systems who have already been operating for many months or years. However, the hyperparameters

	MovieLens 1M				Publiq			
	Pr@8	Recall	F1	NDCG@8	Pr@8	Recall	F1	NDCG@8
MPxKNN	0.3126	0.0151	0.0288	0.2940	0.0852	0.5009	0.1457	0.3200
MPxEASE	0.3203	0.0155	0.0295	0.3058	0.0980	0.5757	0.1674	0.3460
MPxSLIM	0.3300	0.0159	0.0304	0.3208	0.0912	0.5361	0.1559	0.3306
MPxWMF	0.3064	0.0148	0.0282	0.2860	0.0929	0.5457	0.1587	0.3374
MPxTEASER	0.3396	0.0164	0.0313	0.3348	0.0271	0.1593	0.0463	0.0915
MP average	0.3218	0.0155	0.0296	0.3083	0.0789	0.4635	0.1348	0.3009
DTxKNN	0.5580	0.0270	0.0514	0.5695	0.0663	0.3897	0.1133	0.2673
DTxEASE	0.5075	0.0245	0.0468	0.5252	0.0784	0.4610	0.1341	0.3077
DTxSLIM	0.4666	0.0225	0.0430	0.4812	0.0812	0.4773	0.1388	0.3172
DTxWMF	0.4514	0.0218	0.0416	0.4685	0.0875	0.5143	0.1496	0.3120
DTxTEASER	0.4767	0.0230	0.0439	0.4719	0.0266	0.1562	0.0454	0.0881
DT average	0.4920	0.0238	0.0453	0.5032	0.0680	0.3997	0.1162	0.2585

Table 3. Overview of the offline evaluation results after collecting exactly 3 likes, and ask at most 20 queries for each new user. Specific hyperparameter values used by our DT algorithm include a minimum of 200 users to make a split and a maximum of 100 items to consider as splitting items (MPS), and regularization factor $\lambda_1 = 200$. Metrics are calculated based on a list of 8 recommendations.



Fig. 6. The precision and NDCG scores achieved by the TEASER recommender algorithm after applying the DT and MP algorithm. Additionally, 95% confidence intervals were added to each chart.

of the AL, such as the minimum number of users required to make a split or the maximum number of most popular items to consider as candidate splitting items, were optimized through an extensive manual grid search. Hence, the metric scores displayed in Table 3 were calculated using the most optimal hyperparameters for the AL algorithm and default hyperparameters for the recommender algorithm. Additionally, each tested AL algorithm was given the constraints to collect exactly 3 likes from each user user, and not ask more than 20 queries. This range was selected based on the estimated patience a real user would display during the user study when they are asked to read the information of multiple Flemish cultural events and subsequently rate them.

When analyzing the metric scores presented in Table 3, it is clear that our DT seems to have a larger positive impact on the precision (0.4920 vs 0.3218) and ranking capabilities (0.5032 vs 0.3083) of the recommender algorithm when sufficient ratings per user are available (MovieLens). Also for the recall and F1 metric, our DT algorithm boosts the performance compared to the MP algorithm for the MovieLens data set. When users have fewer ratings (Publiq), the MP baseline algorithm seems to gain the upper hand as it has a larger positive impact on the precision (0.0789 vs 0.0680) and ranking capabilities (0.3009 vs 0.2585) of the recommender algorithm. This observation is to be expected as the MP algorithm asks new users to rate a set of most popular items. The more popular an item is, the higher the chance that a new user will be able to rate it¹¹. Remember that the rating density of the MovieLens data set is 4.4684%, which is a lot higher than the 0.0098% rating density of the Publiq data set. Users from the Publiq data set have a higher chance of rating an item presented by the MP algorithm than our DT algorithm. After all, it is still better to obtain an albeit less informative rating than no rating at all when interviewing new users.

Online results

For the user study, the target of 50 participants was reached. The precision and NDCG metric scores were calculated for the six recommendations each user had to rate. As a reminder, the TEASER algorithm and Publiq data set were integrated into a basic web application to serve as a RS. The most optimal hyperparameters were chosen for both AL algorithms via offline evaluation. Both algorithms were tasked to collect ratings for at least 3 items and at most 20 items to ensure that enough information was collected regarding the user's preferences without testing the user's patience too much during the interviews. The results are displayed in Fig. 6. The



Fig. 7. The average answer (5-point Likert scale) given by the participants to each of the 5 questions posed by the questionnaire. Additionally, 95% confidence intervals were added to each chart.

difference between the impact created by both AL algorithms is quite small. However, the results still slightly favour our DT algorithm instead of the baseline MP algorithm. Indeed, a precision of 0.52 (vs 0.49) is achieved by the recommender algorithm after applying our DT algorithm. Additionally, an NDCG score of 0.796 (vs 0.793) is achieved by the TEASER recommender algorithm. However these differences are not significant. The small size of the experiment (50 users, 6 recommendations/user) might be a reason for this.

Furthermore, Fig. 7 displays the average of the answers (5-point Likert scale) given by the participants to the five questions listed in Table 2. Similarly to the metric scores discussed above, the difference between the averages of both AL algorithms is limited. Nonetheless, the answers seem to indicate a slightly more positive impact created by our DT compared to the MP baseline algorithm for most recommendation aspects. This indicates that our DT algorithm slightly improves the diversity (question 1), relevancy (question 2), serendipity (question 3) and ranking (question 4) of the recommendations. It is therefore slightly surprising to see that participants marginally favoured the recommendations after applying the MP algorithm (question 5). None of the differences between answers seem to be significant since the 95% confidence intervals are overlapping.

Discussion, limitations and future work

The obtained offline results show a potential accuracy improvement of the recommendations induced by our DT and in comparison with the MP baseline algorithm *if* there are sufficient ratings per user. In other words, dynamically selecting new items based on the answers given by the user during the rating elicitation process is beneficial for the recommender algorithm's performance *if* users are actually able to rate those items accurately.

However, there are some limitations regarding this study. Firstly, the use of offline data sets doesn't allow to determine if a real user is actually able to provide a rating based on the AL query. Although the offline evaluation showed a clear accuracy improvement, the obtained online results are much more indecisive. This observation can be caused by (wrong) assumption that users can accurately rate an item if they are provided with a description of this item. This inherently reveals the major problem of rating elicitation approaches (AL); most users have only experienced a small number of items and asking them to rate a small number of items increases the chance that they can not accurately rate these items, even if additional information is provided about each item. Our offline evaluation approach did not make this assumption. Indeed, a rating was only elicited from a simulated user if the data set did actually contain a rating for the target item from this user. This is a valuable

reminder that one should be careful when making certain assumptions, especially when this assumption is based on the behaviour of real humans.

Secondly, the tree is constructed based on ratings and users are partitioned into groups until a stopping criteria is reached. After a large number of splits, some of the child nodes may be based on a limited amount of information with the risk of overfitting. A limitation of the current version of the DT is the absence of pruning functionality. An interesting future research track is to investigate if pruning can help to generalize parts of the DT and, in the end improve the accuracy of the recommendations.

Thirdly, a limitation of AL in the domain of recommender systems is the bias towards popular items. In traditional ML problems, AL algorithms have typically no limitation regarding the queries for the labels of the samples. For example, for an object recognition task in the domain computer vision, the AL algorithm can select a photo and ask for the label, which specifies where the object is. Either those labels are available, or a human can point out the object in the photo. In any case, an answer (a label) can be given to the AL query. In contrast, in the domain of RSs, the user might not be familiar with the item, and therefore not able to rate the item. This is the case in the offline as well as the online evaluation of AL solutions for RSs. In the offline evaluation, the answer to the AL query is sought in the ratings of the test set. If the specific item is not rated, the AL query cannot be answered, and another item is selected for rating elicitation. In the online evaluation, the user is asked to rate the specific item. Some metadata about the item can help the user, but still the user might feel unable to rate the item without consuming it (e.g. rate a movie without watching it). In this online evaluation, the AL algorithm might end up without additional ratings, because users cannot be asked for ratings over and over again without annoying them.

In the future, we would like to implement and modify other AL algorithms that are not necessarily based on DTs to see if these made observations still apply. It will also be necessary to recruit more than 50 participants for future experiments to provide a higher degree of certainty regarding the obtained results. A large-scale user study will allow a more detailed evaluation of the AL algorithms, also in the long-term. Another interesting future research track is the combination of AL algorithms and meta-learning techniques. The core idea of meta-learning is “learning to learn”, i.e., it leverages prior experience gained from solving tasks to solve new tasks quickly and more efficiently. Technically, meta-learning trains a model over a large number of tasks with limited training samples in each task. It optimizes model parameters according to the adaption performance on these tasks so that the learned model can adapt quickly and generalize well on future tasks. An innovative research track would be to deploy these meta-learning techniques for recommendation tasks and combine these with the extra ratings obtained by AL algorithms.

Conclusion

The goal of helping RSs more quickly learn about the preferences of new users is definitely one worth pursuing. In this paper, we investigated the importance of interviewing new users to quickly obtain crucial information about their preferences. An attempt was made to strike a good balance between gaining enough information by asking sufficient questions and not frustrating the user by asking too many questions. With transparency and balance in mind, it was a natural choice to interview users using a DT. While offline evaluation showed that our DT algorithm created a significant performance gain for the underlying recommender algorithm when using the MovieLens 1M data set, the online evaluation results were much more indecisive. More research is therefore needed to prove that our DT algorithm, and AL in general, does indeed improve the performance of RSs in practice and that interviewing users is the best way to achieve this goal.

Data availability

The data that support the findings of this study are not published online to protect study participant privacy. However, the anonymized data could be made available from the corresponding author, T. De Pessemier, upon reasonable request and for research purposes only. This includes both original data generated in the research (including the logging data of the experiment) and secondary data reuse that supports our results and analyses (including the statistical processing).

Code availability

The authors opted to not disclose the source code because of IP restrictions.

Received: 2 November 2024; Accepted: 30 June 2025

Published online: 08 July 2025

References

- Pazzani, D. Michael J. & Billsus: Content-based recommendation systems. In: Brusilovsky, P., Kobsa, A., Nejdl, W. (eds.) *The Adaptive Web: Methods and Strategies of Web Personalization*, pp. 325–341. Springer, Berlin, Heidelberg (2007). https://doi.org/10.1007/978-3-540-72079-9_10
- Friedman, N., Geiger, D., & Goldszmidt, M.: Bayesian network classifiers. *Machine Learning*, 131–163 (1997) <https://doi.org/10.1023/A:1007465528199>
- Gershman, A., Meisels, A., Luke, K.-H., Rokach, L., Schclar, A., & Sturm, A.: A decision tree based recommender system. *Gesellschaft für Informatik eV* (2010)
- Zhang, Q., Lu, J., & Jin, Y.: Artificial intelligence in recommender systems. *Complex & Intelligent Systems* 7, 439–457 (2021) <https://doi.org/10.1145/2827872>
- Herlocker, J. L., Konstan, J. A., Terveen, L. G. & Riedl, J. T. Evaluating collaborative filtering recommender systems. *ACM Trans. Inf. Syst.* 22(1), 53–53. <https://doi.org/10.1145/963770.963772> (2004).
- Bobadilla, J., Ortega, E., Hernando, A. & Gutiérrez, A. Recommender systems survey. *Knowledge-based systems* 46, 109–132. <https://doi.org/10.1016/j.knsys.2013.03.012> (2013).

7. Isinkaye, F. O., Folajimi, Y. O. & Ojokoh, B. A. Recommendation systems: Principles, methods and evaluation. *Egyptian Informatics Journal* **16**(3), 261–273. <https://doi.org/10.1016/j.eij.2015.06.005> (2015).
8. Adomavicius, G. & Zhang, J. Impact of data characteristics on recommender systems performance. *ACM Trans. Manag. Inf. Syst.* **3**(1), 1–17. <https://doi.org/10.1145/2151163.2151166> (2012).
9. Davoodi, E., Kianmehr, K. & Afsharchi, M. A semantic social network-based expert recommender system. *Applied Intelligence* **39**, 1–13 (2013).
10. Sethi, R., & Mehrotra, M.: Cold start in recommender systems—a survey from domain perspective. In: *Intelligent Data Communication Technologies and Internet of Things: Proceedings of ICICI 2020* vol. 57, pp. 223–232. Springer, Singapore (2021). https://doi.org/10.1007/978-981-15-9509-7_19
11. Elahi, M., Ricci, F. & Rubens, N. A survey of active learning in collaborative filtering recommender systems. *Computer Science Review* **20**, 29–50. <https://doi.org/10.1016/j.cosrev.2016.05.002> (2016).
12. Settles, B.: Active learning literature survey. In: *Computer Sciences Technical Report 1648*. University of Wisconsin-Madison Department of Computer Sciences, Wisconsin, USA (2009)
13. Wang, M., Feng, T., Shan, Z. & Min, F. Attribute and label distribution driven multi-label active learning. *Applied Intelligence* **52**(10), 11131–11146 (2022).
14. Cao, Z. et al. Face detection for rail transit passengers based on single shot detector and active learning. *Multimed. Tools Appl.* **81**(29), 42433–42456 (2022).
15. Liu, Y. et al. Daily unbalanced action recognition based on active learning. *Multimed. Tools Appl.* **83**(6), 16255–16274 (2024).
16. Rashid, A.M., Albert, I., Cosley, D., Lam, S.K., McNee, S.M., Konstan, J.A., & Riedl, J.: Getting to know you: learning new user preferences in recommender systems. In: *Proceedings of the 7th International Conference on Intelligent User Interfaces*. IUI '02, pp. 127–134. Association for Computing Machinery, New York, NY, USA (2002). <https://doi.org/10.1145/502716.502737>.
17. Golbandi, N., Koren, Y., & Lempel, R.: Adaptive bootstrapping of recommender systems using decision trees. *Proceedings of the Fourth ACM International Conference on Web Search and Data Mining*, 595–604 (2011) <https://doi.org/10.1145/1935826.1935910>
18. Karimi, R., Wistuba, M., Nanopoulos, A., & Schmidt-Thieme, L.: Factorized decision trees for active learning in recommender systems. In: Broy, M., Denert, E. (eds.) *2013 IEEE 25th International Conference on Tools with Artificial Intelligence*, pp. 404–411. IEEE, Herndon, VA, USA (2013). <https://doi.org/10.1145/1935826.1935910>
19. Karimi, R., Nanopoulos, A., & Schmidt-Thieme, L.: Improved questionnaire trees for active learning in recommender systems. In: *LWA, CEUR Workshop Proceedings*, pp. 34–44 (2014)
20. Rajaraman, S.: *Five Stars Dominate Ratings* (2009). <https://blog.youtube/news-and-events/five-stars-dominate-ratings/>
21. Harper, F. M. & Konstan, J. A. The movielens datasets: History and context. *ACM Trans. Interact. Intell. Syst.* **5**(4), 1–19. <https://doi.org/10.1145/2827872> (2015).
22. Michiels, L., Verachtert, R., & Goethals, B.: Recpack: An(other) experimentation toolkit for top-n recommendation using implicit feedback data. In: *Proceedings of the 16th ACM Conference on Recommender Systems*, pp. 648–651. Association for Computing Machinery, New York, NY, USA (2022). <https://doi.org/10.1145/3523227.3551472>
23. Deshpande, M. & Karypis, G. Item-based top-n recommendation algorithms. *ACM Trans. Inf. Syst.* **22**(1), 143–177. <https://doi.org/10.1145/963770.963776> (2004).
24. Steck, H.: Embarrassingly shallow autoencoders for sparse data. In: *The World Wide Web Conference*, New York, NY, USA, pp. 3251–3257 (2019). <https://doi.org/10.1145/3308558.3313710>
25. Ning, X., & Karypis, G.: Slim: Sparse linear methods for top-n recommender systems. In: *2011 IEEE 11th International Conference on Data Mining*, Vancouver, BC, Canada, pp. 497–506 (2011). <https://doi.org/10.1109/ICDM.2011.134>. IEEE
26. Hu, Y., Koren, Y., & Volinsky, C.: Collaborative filtering for implicit feedback datasets. In: *2008 Eighth IEEE International Conference on Data Mining*, Pisa, Italy, pp. 263–272 (2008). <https://doi.org/10.1109/ICDM.2008.22>. IEEE
27. De Pauw, J., Ruymbeek, K., & Goethals, B.: Who do you think i am? interactive user modelling with item metadata. In: *Proceedings of the 16th ACM Conference on Recommender Systems*, Seattle, WA, USA, pp. 640–643 (2022)
28. De Pauw, J., Ruymbeek, K., & Goethals, B.: Modelling users with item metadata for explainable and interactive recommendation (2022). <https://arxiv.org/abs/2207.00350>
29. Lee, H., Im, J., Jang, S., Cho, H., & Chung, S.: Melu: Meta-learned user preference estimator for cold-start recommendation. In: *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pp. 1073–1082 (2019)

Acknowledgements

We would like to thank the Adrem Data lab of the University of Antwerp for their collaboration in this research regarding the design of the webinterface and making the recommender algorithms available.

Author contributions

All authors made substantial contributions to this study. All authors read and approved the final manuscript.

Funding

This research received funding from the Flemish Government under the “Onderzoeksprogramma Artificiële Intelligentie (AI) Vlaanderen” programme.

Declarations

Competing interests

The authors declare no competing interests.

Ethical approval

All human participants involved in the study gave their informed consent.

Consent to participate

When participants agreed to take part in the experiment, they were asked to give their consent for using their click behavior on the dedicated website of this experiment and their answers on the questionnaire for analysis purposes.

Consent for publication

All partners involved in this project gave their consent for publication of this article.

Additional information

Correspondence and requests for materials should be addressed to T.P.

Reprints and permissions information is available at www.nature.com/reprints.

Publisher's note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.

Open Access This article is licensed under a Creative Commons Attribution-NonCommercial-NoDerivatives 4.0 International License, which permits any non-commercial use, sharing, distribution and reproduction in any medium or format, as long as you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons licence, and indicate if you modified the licensed material. You do not have permission under this licence to share adapted material derived from this article or parts of it. The images or other third party material in this article are included in the article's Creative Commons licence, unless indicated otherwise in a credit line to the material. If material is not included in the article's Creative Commons licence and your intended use is not permitted by statutory regulation or exceeds the permitted use, you will need to obtain permission directly from the copyright holder. To view a copy of this licence, visit <http://creativecommons.org/licenses/by-nc-nd/4.0/>.

© The Author(s) 2025