MeLU: Meta-Learned User Preference Estimator for Cold-Start Recommendation

Hoyeop Lee, Jinbae Im, Seongwon Jang, Hyunsouk Cho, Sehee Chung
Knowledge AI Lab., NCSOFT Co., South Korea
{hoyeoplee,jinbae,swjang90,dakgalbi,seheechung}@ncsoft.com

ABSTRACT

This paper proposes a recommender system to alleviate the cold-start problem that can estimate user preferences based on only a small number of items. To identify a user's preference in the cold state, existing recommender systems, such as Netflix, initially provide items to a user; we call those items evidence candidates. Recommendations are then made based on the items selected by the user. Previous recommendation studies have two limitations: (1) the users who consumed a few items have poor recommendations and (2) inadequate evidence candidates are used to identify user preferences. We propose a meta-learning-based recommender system called MeLU to overcome these two limitations. From meta-learning, which can rapidly adopt new task with a few examples, MeLU can estimate new user's preferences with a few consumed items. In addition, we provide an evidence candidate selection strategy that determines distinguishing items for customized preference estimation. We validate MeLU with two benchmark datasets, and the proposed model reduces at least 5.92% mean absolute error than two comparative models on the datasets. We also conduct a user study experiment to verify the evidence selection strategy.

KEYWORDS

Recommender systems; Cold-start problem; User preference estimation; Meta-learning

ACM Reference Format:
Hoyeop Lee, Jinbae Im, Seongwon Jang, Hyunsouk Cho, Sehee Chung. 2019. MeLU: Meta-Learned User Preference Estimator for Cold-Start Recommendation . In *The 25th ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD '19), August 4–8, 2019, Anchorage, AK, USA.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3292500.3330859

1 INTRODUCTION

Recommender systems can be generally classified as collaborative filter-based, content-based, or hybrid systems. Collaborative filter-based systems estimate user responses by collecting preference information from numerous users [6, 12, 21]. The predictions are built upon the existing ratings of other users who have similar ratings as the target user. However, such systems cannot handle new users (user cold-start) and new items (item cold-start) because of the lack of user-item interactions. Content-based systems are introduced [15, 17] to solve the cold-start problem. Such systems use user profile information (*e.g.*, gender, nationality, religion, and political stance) and the contents of the items to make recommendations. The systems might have a limitation suggesting the same items to the users who have similar contents regardless of items that user already rated. Hybrid systems, which are based on a collaborative filter and utilize content information, are widely used in various applications [3, 9, 23]. However, these systems are unfit for a recommendation when the user-item interaction data are sparse. Moreover, due to privacy issues, collecting personal information is challenging, which might result in the user cold-start problem.

To avoid the user cold-start problem due to privacy issues, many web-based systems, such as Netflix, recommend items based on only minimal user information. Netflix initially presents popular movies and television programs to new users: we call these videos the *evidence candidates*. Then, the user chooses the videos that he/she likes among the candidates. Afterward, the system recommends some programs based on the videos selected by the user. Recently, to improve performance, the recommendations have been made using deep learning methods [10, 24, 28]; however, the cold-start problem remains for new users who rate only a few items.

Previous recommender systems are limited by two important issues. First, a system should be able to recommend items to new users who have made a few ratings. New users leave the system when they receive poor recommendations initially [14]. However, the existing systems are not designed for users who rate only a few items. The previous system leverages the user profile information to improve the poor performance, but it does not solve the limitation. Consider two unemployed male users of a movie service who are in their twenties. One man watches a few science fiction movies, while the other man views a few horror movies. When gender, age, and occupation are provided as the user information, the recommender system might present a very similar movie lists to both men because a few movies cannot figure out their preferences. Second, the existing systems do not provide reliable evidence candidates to estimate user preferences: they show popular items as evidence candidates. Spending time selecting evidence candidates might not be necessary because recommendations naturally become robust as user-item interaction increases. However, we have to deliberately select proper evidence candidates to improve the initial recommendations for new users.

This study proposes a meta-learning-based recommender system called MeLU to resolve the preceding issues. Meta-learning focuses on improving classification or regression performance by learning with only a small amount of training data. A recommender system has similar characteristics with meta-learning, because it focuses on predicting a user's preferences based on only a small number of

KDD '19, August 4–8, 2019, Anchorage, AK, USA

ACM ISBN 978-1-4503-6201-6/19/08…$15.00
https://doi.org/10.1145/3292500.3330859

consumed items. We consider the Model-Agnostic Meta-Learning (MAML) algorithm [4], which allows estimation of customized preference directly based on an individual user's item-consumption history, even when only a small number of items have been consumed. In contrast to collaborative filter-based systems, which find other users who have similar ratings as the target user, the proposed system considers the items consumed by the target user only. Moreover, we suggest evidence candidate selection strategy for the MAML-based recommender system which can substantially enhance the initial recommendation performance for new users by selecting distinguishing items for customized preference estimation.

Our study provides four main contributions. First, we alleviate the user cold-start problem by adopting the MAML concept in the recommender system. Second, to the best of our knowledge, this study is the first to identify reliable evidence candidates to improve the initial recommendation performance of new users. Third, MeLU can provide personalized model to each user using their unique item-consumption history. Lastly, the proposed system and evidence candidate selection strategy are validated with two benchmark datasets and a user survey.

The remainder of this paper is organized as follows. We briefly describe meta-learning and the relationship between meta-learning and the proposed recommender model in Section 2. We present the proposed system and evidence candidate selection strategy in Section 3. Section 4 evaluates the recommendation performance of the proposed method by means of two comparative methods for benchmark datasets. In addition, through user study, we compare the proposed model-based evidence candidates with popularity-based evidence candidates in Section 5. Finally, we conclude this paper and discuss future research in Section 6.

2 META-LEARNING

Meta-learning, also called learning-to-learn, aims to train a model that can rapidly adapt to a new task which is not used during the training with a few examples [26]. Meta-learning is inspired by the human learning process, which can quickly learn new tasks based on a small number of examples. Meta-learning can be classified into three types: metric-based, memory-based, and optimization-based meta-learning. Previous researches [8, 20, 22, 27] on metric-based and memory-based meta-learning are concentrated on classification problems; the work in [25] took a concept of the metric-based meta-learning algorithm in the recommender system to predict whether a user consumes an item or not. The system could make a recommendation based on item-consumption but cannot provide a personalized model to users. By contrast, we consider taking a concept of optimization-based meta-learning [4, 11] to recommender system, which can serve a personalized recommender model by reflecting item-consumption of each user.

The optimization-based meta-learning algorithm considers the model f and the distribution over task $p(\mathcal{T})$. The algorithm attempts to find desirable parameter θ of model f, as shown in Figure 1. The optimization-based meta-learning algorithm performs local and global updates. From the random initial parameter θ (the starting point of the black arrow), the algorithm samples several tasks from the distribution over task $\mathcal{T}_i \sim p(\mathcal{T})$. The algorithm *locally* updates

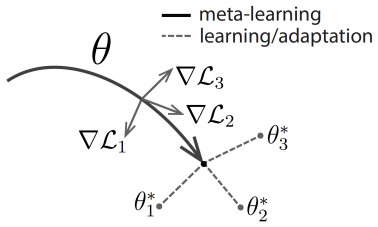

Figure 1: Diagram of the optimization-based meta-learning algorithm [4].

parameter θ to θ_i by gradient $\nabla_\theta \mathcal{L}_i(f_\theta)$ for each tasks $i = 1, \cdots, T$, where T is the number of sampled tasks and $\mathcal{L}_i(f_\theta)$ denotes the training loss on task i with parameter θ. The local updates are represented as gray arrows in the figure. After local updates, for all sampled tasks, the algorithm *globally* updates parameter θ based on $\mathcal{L}_i'(f_{\theta_i})$, which is the test loss on task i with parameter θ_i, so that the globally updated parameter fits into the various tasks.

The optimization-based meta-learning algorithm uses two sets, namely, the support set and query set, for each task. The support set and query set are used for calculating the training loss and test loss on each task, respectively. During the local update, the algorithm adjusts the parameter of the model on each support set (learning process). During the global update, the algorithm trains the parameter to minimize the losses with the adapted parameters on the query sets (learning-to-learn process). When the learning-to-learn process reaches a termination condition for previous tasks, the algorithm only accepts a support set for a new task. Using the support set, the model can adapt to the new task. Note that, the algorithm allows parameters for each task not to be stored; instead, the parameters are calculated via support sets.

We regard each task as estimating a user's preferences in the recommender system. From this inspiration, we propose a MAML-based recommender system that can rapidly estimate a new user's preferences based on only a few user-item interactions. The MAML-based recommender system accounts for the fact that different users have different optimal parameters. Therefore, our MAML-based recommender system provides each user personalized recommendations based on their unique item-consumption history. Consider the two male users from Section 1. The item-consumption history of the first user includes science fiction movies, and that of the second user includes horror movies. Since our recommender system considers movies watched by individual users, the system will recommend science fiction movies or horror movies to each user. Moreover, our model can provide items with large gradient as reliable evidence candidates. An item has a large gradient for item-consumption history if a rating of that item by a new user provides the system with substantial information about the preferences of the user.

3 METHOD

In this section, we describe the details of Meta-Learned User preference estimator (MeLU). We first introduce a user preference estimatior that consists of decision-making layers and output layer with user and item embeddings. Next, we design a personalized user preference estimation model based on MAML that can rapidly

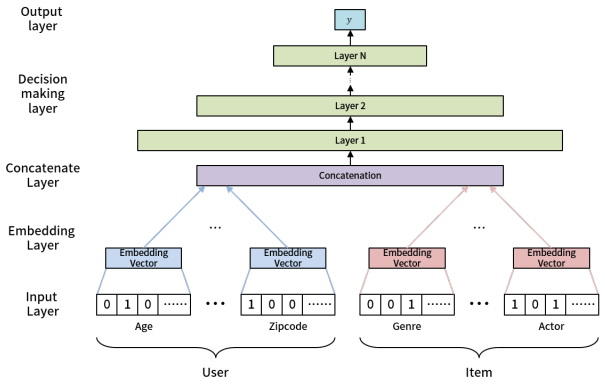

Figure 2: User preference estimatior.

adapt to new tasks (new users). Finally, we suggest an evidence candidate selection strategy based on the personalized user preference estimation model.

3.1 User Preference Estimator

The proposed user preference estimation model for the recommender system is shown in Figure 2. The model takes user content (*e.g.*, age and occupation) and item content (*e.g.*, genre and publication year) as the input. First, the proposed model performs embedding processes based on the input and concatenates the embedded vectors. Second, from the embedding, we model the decision-making process by means of a multilayered neural network, which is widely used in recent recommendation research [3, 6].

We employ the embedding process to extract useful features to estimate user preferences from the contents, because the previous research showed that the embedding process improves the performance in the recommender system [1, 30]. Our model embeds user and item contents in a similar way as in previous work [3]. For categorical content, we generate each content embedding and use the concatenated embeddings as shown in Figure 2. When the number of user contents is P, we define the embedding vector U_i for user i as follows.

$$U_i = [e_{i1}c_{i1}; \cdots ; e_{iP}c_{iP}]^\intercal \tag{1}$$

where c_{ip} is a d_p-dimensional one-hot vector for categorical content $p \in \{1, \cdots, P\}$ of user i and e_{ip} represents the d_e-by-d_p embedding matrix for the corresponding categorical content of the user. d_e and d_p are the embedding dimension and the number of categories for content p, respectively. The items are embedded in the same way. The embedding vector I_j for item j can be described as

$$I_j = \left[e_{j1}c_{j1}; \cdots ; e_{jQ}c_{jQ}\right]^\intercal, \tag{2}$$

where c_{jq} is a d_q-dimensional one-hot vector for categorical content $q \in \{1, \cdots, Q\}$ of item j and e_{jq} represents the d_e-by-d_q embedding matrix for the corresponding categorical content of the item. Q, d_e and d_q are the number of item contents, embedding dimension, and the number of categories for content q, respectively. If there exist continuous contents, the input nodes of the contents are connected directly to the concatenate layer without passing through the embedding layer.

User preference estimation is conducted by the decision-making and output layers of our model. The embedding dimensions for a user and an item may be different (*i.e.*, $d_e \times p \neq d_e \times q$). Thus, we cannot use a generalized matrix factorization layer [6], which requires equal embedding dimensions. Therefore, we construct the decision-making layer as a N-layer fully-connected neural network. The output layer, which is the subsequent layer of the decision-making layers, estimates user preferences that could be ratings, implicit feedback [7], or dwell times [29]. The layers can be formulated as

$$\begin{aligned} x_0 &= [U_i; I_j], \\ x_1 &= a(\mathbf{W}_1^\intercal x_0 + b_1), \\ &\vdots \\ x_N &= a(\mathbf{W}_N^\intercal x_{N-1} + b_N), \\ \hat{y}_{ij} &= \sigma(\mathbf{W}_o^\intercal x_N + b_o), \end{aligned} \tag{3}$$

where $\mathbf{W}_n$ and b_n are the weight matrix and bias vector for the n-th decision-making layer, and $\mathbf{W}_o$ and b_o are those for the output layer. $\hat{y}_{ij}$ is user i's estimated preference for item j, and a and σ denote the activation functions of the decision-making layer and output layer, respectively. We use rectified linear unit (ReLU) [16] as the activation function a. The activation function σ depends on how user preference is defined. If the goal is to estimate ratings and dwell times, a linear function might be appropriate, whereas the sigmoid function can be used in the case of implicit feedback.

3.2 Meta-Learned User Preference Estimator

Inspired by the concept of optimization-based meta-learning, we put this concept into MeLU to reflect personalized user preferences with only a few user-item interactions. Figure 3 shows how the model employs users' item-consumption history, which is not directly considered in the previous model [3]. Our model considers a user's item-consumption history and preferences, and these inputs act as the support set for the local update. In other words, to reflect a user's tastes, the model updates the parameters in the decision-making layers and the output layer (*i.e.*, $\mathbf{W}$ and b) based on the user's unique item-consumption history. We do not update the embedding vectors for the user and items during the local update to ensure the stability of the learning process, which means that our model assumes users and items do not change, only the users' thoughts change as they interact with the items. Furthermore, unlike MAML, we do not limit the length of the item-consumption history. We extend the idea of the matching network, which is one of the most famous meta-learning algorithms and shows good performance even when the length of the support set (*i.e.*, length of the item-consumption history) is not fixed [2].

Algorithm 1 shows the detailed personalization procedure of the user preference estimation model. First, the parameters in Eqs. 1, 2, and 3 (lines 1-2) are randomly initialized. As the model updates only the parameters in Eq. 3 during the local update, we denote these parameters separately. Second, the model randomly samples the batch of users (line 4). Third, the model locally updates the parameters for user i in the batch by backpropagating the following

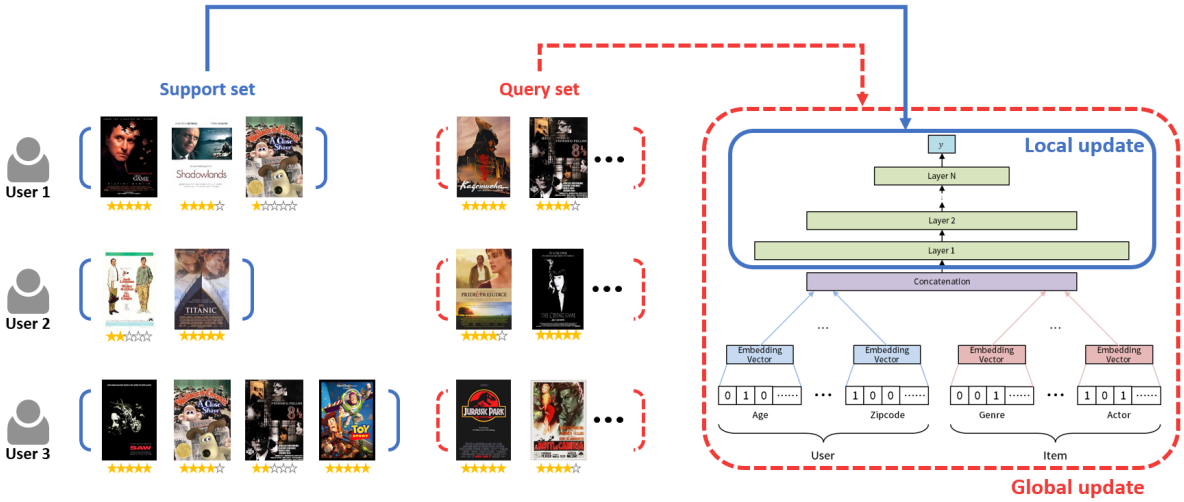

Figure 3: The proposed model, MeLU, receives users' item histories and preferences. Then, the decision-making layers and the output layer (marked with a blue box) are updated based on the support set of each user; that is, the proposed model locally updates the user's decision-making process based on each user's item-consumption pattern. After local updates for the users, all layers (marked with a red-dashed box) are globally updated based on the query sets.

loss function (lines 6-8):

$$\mathcal{L}_i = \frac{1}{|H_i|} \sum_{j \in H_i} (y_{ij} - \hat{y}_{ij})^2$$

where $H_i = \{j|$ item j consumed by user $i\}$ is a set of items consumed by user i, *i.e.*, the support set, and y_{ij} is user i's actual preference for item j. This local update can be considered to be an iteration for personalization, which can be repeated several times. Finally, the algorithm globally updates θ_1 and θ_2 based on loss function $\mathcal{L}'_i$ for corresponding new item-consumption history H'_i (*i.e.*, the query set) with the personalized parameters (line 10). This process aims to find the desirable parameters that achieve good recommendation performance after a few local updates for all users. The algorithm repeats these processes until θ_1 and θ_2 converge. When recommending items to a user, MeLU conducts local updates based on the user's item-consumption history and calculates the preferences for all items that the user has not rated. After that, the model recommends top favored items to the user.

Algorithm 1 Model-Agnostic Meta-Learning for User Preference Estimator

Require: α, β: step size hyperparameters
1: randomly initialize θ_1 (parameters in Eqs. 1 and 2)
2: randomly initialize θ_2 (parameters in Eq. 3)
3: **while** not converge **do**
4: sample batch of users $B \sim p(\mathcal{B})$
5: **for** user i in B **do**
6: set $\theta_2^i = \theta_2$
7: evaluate $\nabla_{\theta_2^i} \mathcal{L}_i(f_{\theta_1, \theta_2^i})$
8: local update $\theta_2^i \leftarrow \theta_2^i - \alpha \nabla_{\theta_2^i} \mathcal{L}'_i(f_{\theta_1, \theta_2^i})$
9: **end for**
10: global update $\theta_1 \leftarrow \theta_1 - \beta \sum_{i \in B} \nabla_{\theta_1} \mathcal{L}'_i(f_{\theta_1, \theta_2^i})$
$\theta_2 \leftarrow \theta_2 - \beta \sum_{i \in B} \nabla_{\theta_2} \mathcal{L}'_i(f_{\theta_1, \theta_2^i})$
11: **end while**

3.3 Evidence Candidate Selection

We suggest an evidence candidate selection strategy based on the MeLU. The strategy identifies the distinguishing items that can be used to quickly analyze individual preferences of new users in the system. In our model, the larger the average Frobenius norm of whole users' gradient for personalization is (*i.e.*, the gradient of the local update, $\|\nabla_{\theta_2^i} \mathcal{L}_i(f_{\theta_1, \theta_2^i})\|_F$), the better the distinction among users' preferences. When calculating the gradient, we modify $\mathcal{L}_i$ as $\mathcal{L}_i / |\mathcal{L}_i|$, where $|\cdot|$ denotes the absolute value of the input to backpropagate the unit error. Although the items with large gradient are useful for identifying user preferences, a proper assessment might be difficult when the user does not know about the items. Therefore, we also consider the users' awareness of the items. We assume that the more frequent user-item interaction is, the better aware users are of the item. Therefore, for each item, we calculate the gradient-based value and popularity-based value as the average Frobenius norm and the number of interactions for each item using the whole existing user-item pairs, respectively. To scale the units of two values, we normalize the values to range from zero to one and assign the score to each item by multiplying two normalized values. Finally, we define the top k items with the highest scores as evidence candidates. Note that, the scores can vary when entering new user, which directly effects the average Frobenius norm and user-item interactions.

4 EXPERIMENTS

4.1 Experimental Design

We considered two datasets: MovieLens 1M [5] and Bookcrossing [31]. Both datasets provide basic user and item information,

Table 1: Basic statistics and used contents of the MovieLens and Bookcrossing dataset.

	MovieLens	Bookcrossing
Number of users	6,040	278,858
Number of items	3,706	271,379
Number of ratings	1,000,209	1,149,780
Sparsity	95.5316%	99.9985%
User contents	Gender, Age, Occupation, Zip code	Age
Item contents	Publication year, Rate, Genre, Director, Actor	Publication year, Author, Publisher
Range of ratings	$1 \sim 5$	$1 \sim 10$

such as user's age and publication year, and the datasets have explicit feedback information. We divided the items and users into two groups (existing/new) to evaluate the performance under item-cold-start and user-cold-start condition. We prepared four partitions for each dataset: 1) existing items for existing users, 2) existing items for new users, 3) new items for existing users, and 4) new items for new users. The following preprocessing was performed for each dataset:

- **MovieLens 1M**: We collected additional movie contents from IMDb[1]. We divided the movies into movies released before 1997 and after 1998 (approximately 8:2). We regarded movies released before 1997 as existing items and movies released after 1998 as new items, and we randomly selected eighty percent of the users as existing users and the rests as new users.
- **Bookcrossing**: We divided the books into those released before 1997 and those released after 1998 (approximately 5:5). We regarded books released before 1997 as existing items and books released after 1998 as new items, and we randomly selected one-half of the users as existing users and the remainder as new users.

In each partition, we included only users whose item-consumption history length is between thirteen and one hundred. For each user, ten random items from the history were used as the query set (H'_i), and the remaining items were used as the support set (H_i) in Algorithm 1, *i.e.*, the length of the item-consumption history ranges from three to ninety. It was used to determine whether the model can learn consumers' preferences regardless of the length of the item-consumption history. The characteristics of the two datasets are summarized in Table 1.

The proposed model, MeLU, was tested with the following structure. Two layers were used for the decision-making layers with 64 nodes each, and the dimension of all embedding vectors was set to 32. We set the step sizes α and β to 5×10^{-6} and 5×10^{-5} for the MovieLens 1M dataset and to 5×10^{-5} and 5×10^{-4} for the Bookcrossing dataset. The number of local updates was varied from one to five. The batch size and the maximum number of epochs of Algorithm 1 were set to 32 and 30, respectively. All experiments were implemented with Pytorch [19]. The source code of the proposed model is available online[2].

We compared the proposed method with two models: Pairwise Preference Regression (PPR) [18] and Wide & Deep [3]. PPR estimates user preferences via bilinear regression with the one-hot user and item content vectors. Wide & Deep predicts whether the user likes an item, but we modified it into a regression model that estimates preferences. The structure of the Wide & Deep model was the same as that of our user preference estimation model. We did not consider the collaborative-filtering-based approaches for comparison because they cannot estimate preferences for new users and new items.

The performance indicators used in this study were the mean absolute error (MAE) and normalized discounted cumulative gain (nDCG), as shown in Eqs. 4 and 5.

$$MAE = \frac{1}{|U|} \sum_{i \in U} \frac{1}{|H'_i|} \sum_{j \in H'_i} \left| y_{ij} - \hat{y}_{ij} \right|, \tag{4}$$

$$nDCG_k = \frac{1}{|U|} \sum_{i \in U} \frac{DCG^i_k}{IDCG^i_k}, \tag{5}$$

$$DCG^i_k = \sum_{r=1}^{k} \frac{2^{R_{ir}} - 1}{\log_2(1+r)}$$

where R_{ir}, U, and $IDCG^i_k$ are the real rating of user i for the r-th ranked item, a set of users in the test data and the best possible DCG^i_k for user i, respectively.

4.2 Experimental Results

We conducted experiments on a non-cold-start scenario and three cold-start scenarios: (1) recommendation of existing items for new users, (2) recommendation of new items for existing users, and (3) recommendation of new items for new users. The first type of cold-start recommendation scenario can be considered as recommendations for new users of a system. The second type can be considered as making a recommendation on new movies or books. The third is a mixture of the first and second cases and is the most difficult recommendation scenario.

Table 2 shows the performance of the three methods for the four types of recommendation on the two datasets. The recommendation of existing items for existing users represents the performance on the training dataset. The other three types, which represent cold-start scenarios, indicate the test performance in different cases. We denoted the proposed model with l local updates as MeLU-l. Our model outperforms two comparative methods in three types of cold-start scenarios in the MovieLens and Bookcrossing datasets. PPR showed the best MAE, $nDCG_1$, and $nDCG_3$ in the recommendation of existing items for existing users in the MovieLens dataset, but it showed poor performance in the three cold-start scenarios. From the performance degradation, we could infer that the performance gaps of PPR between the non-cold-start scenario and cold-start scenarios come from the overfitting when sparsity is low. Because the Bookcrossing dataset has large sparsity, the models were not well fitted compared to the MovieLens dataset and it is considered

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

[2] http://github.com/hoyeoplee/MeLU

Table 2: Experimental results on the MovieLens and Bookcrossing datasets. The value of each cell represents the average value, regardless of the length of the user's item-consumption history.

Type	Method	MovieLens			Bookcrossing		
		MAE	$nDCG_1$	$nDCG_3$	*MAE*	$nDCG_1$	$nDCG_3$
Recommendation of existing items for existing users	PPR	**0.1820**	**0.9796**	**0.9831**	3.8092	0.8242	0.8494
	Wide & Deep	0.9047	0.9090	0.9117	1.6206	0.9012	0.9172
	MeLU-1	0.7661	0.8866	0.8904	**0.7799**	**0.9563**	**0.9572**
	MeLU-5	0.7567	0.8870	0.8919	0.7955	0.9546	0.9552
Recommendation of existing items for new users	PPR	1.0748	0.8299	0.8468	3.8430	0.8201	0.8434
	Wide & Deep	1.0694	0.8559	0.8639	2.0457	0.8238	0.8515
	MeLU-1	0.7884	0.8799	0.8810	**1.8701**	**0.8265**	0.8527
	MeLU-5	**0.7854**	**0.8803**	**0.8812**	1.8767	0.8263	**0.8532**
Recommendation of new items for existing users	PPR	1.2441	0.7289	0.7632	3.6821	0.8115	0.8367
	Wide & Deep	1.2655	0.7420	0.7721	2.2648	0.8190	0.8437
	MeLU-1	0.9361	**0.7715**	0.7990	**2.1047**	**0.8202**	**0.8441**
	MeLU-5	**0.9275**	0.7697	**0.8005**	2.1236	0.8190	0.8440
Recommendation of new items for new users	PPR	1.2596	0.7292	0.7634	3.7046	0.8171	0.8381
	Wide & Deep	1.3114	0.7680	0.7874	2.3088	0.8160	0.8405
	MeLU-1	0.9299	**0.7760**	**0.8011**	**2.1475**	**0.8184**	0.8410
	MeLU-5	**0.9235**	0.7752	0.8008	2.1721	**0.8184**	**0.8422**

that overfitting did not occur. Moreover, the proposed method performed well when minimal information about users is available (only age), as for the Bookcrossing dataset. Generally, our model showed good performance in cold-start scenarios, regardless of the amount of user and item information available to the recommender system.

MeLU achieved remarkable improvements on all types of datasets, even in cases with very few local updates. Figure 4 shows the performance of our method for varying the number of iterations for personalization on the two datasets. The *MAE* of the proposed method dramatically decreased after the single iteration for all datasets. After one iteration, slight differences were observed for increasing the number of local updates, in contrast to the results from the existing MAML [4], in which the performance improved as the number of iterations increased. Our model can adapt quickly to users because a single local update is sufficient. The rapid adaptation allows the proposed method to be applied to online recommendation based on user ratings.

Figures 5 and 6 show the *MAE* of our method according to the length of the item-consumption history when the number of local updates is one. As shown in the figures, the recommendation performances of MeLU were superb even when the length of the support set was short. Our model showed robust performance with regard to the length of the item-consumption history. Note that for both datasets, the longer the item-consumption history was, the smaller the number of people. In extreme cases, the number of people was less than five. The performance is believed to be unstable because of insufficient sample size.

We qualitatively analyzed the effect of the local update, which aims to provide personalized recommendations. Figure 7 shows the contour maps of the estimated preferences after one local update for all items of eight MovieLens users. They were divided into four groups, with each group having the same user profiles

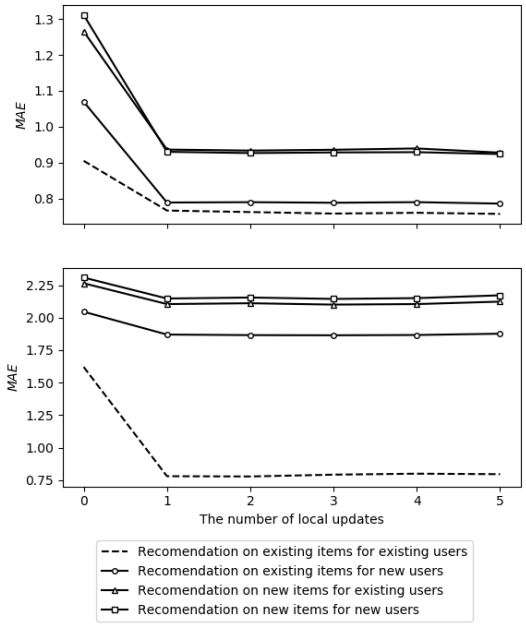

Figure 4: The *MAE* of our method according to the change in the number of local updates on the (top) MovieLens and (bottom) Bookcrossing datasets.

in each row. For visualization, we used t-SNE [13] to reduce the dimension of the item embeddings into two, which was shared by all figures. Differences in the ridge and valley lines were observed in the contour maps within groups as well as between groups. The difference between groups could be caused by differences in user profiles. The difference within groups verified that the proposed

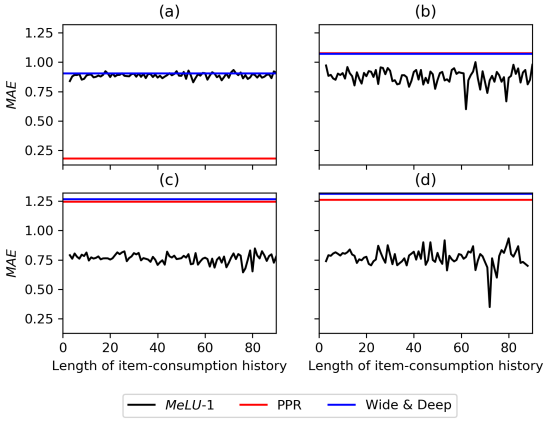

Figure 5: The *MAE* of our method according to the length of item-consumption history on the MovieLens dataset. (a) Recommendation of existing items for existing users. (b) Recommendation of existing items for new users. (c) Recommendation of new items for existing users. (d) Recommendation of new items for new users.

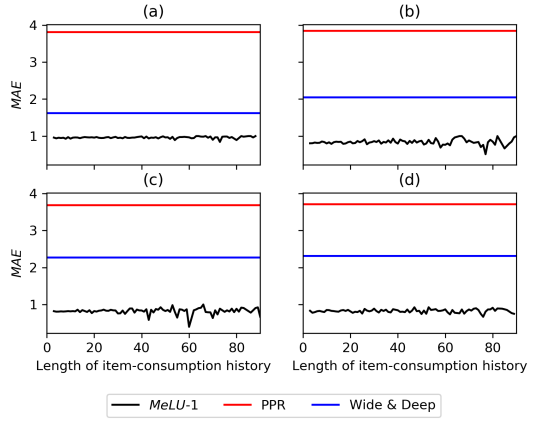

Figure 6: The *MAE* of our method according to the length of item-consumption history on the Bookcrossing dataset. (a) Recommendation of existing items for existing users. (b) Recommendation of existing items for new users. (c) Recommendation of new items for existing users. (d) Recommendation of new items for new users.

algorithm estimates preferences based on movies belonging to the item-consumption history of each user, even when the user profiles are the same. To summarize, MeLU estimated the preferences of users within groups as well as users between groups through the local update.

5 USER STUDY ON EVIDENCE CANDIDATES

5.1 Design of User Study

We conducted user study experiments to compare the proposed and popularity-based evidence candidate selection strategy. In this section, we considered only the MovieLens dataset because more people frequently watch movies than read books. We created the demonstration page for the user study shown in Figure 8. On the first page, as in MovieLens, we collected the user's basic information, such as gender, age, and occupation. On the second page, we presented twenty movies to the user and then obtained item preferences on a five-point Likert scale, which means that we selected twenty evidence candidates. At this time, the movies the user is not familiar with were marked as 'I do not know this movie.' These movies were not included in the user's item-consumption history. We randomly showed one of the popularity-based or our strategy-based movie lists as an A/B test; we listed the two movie lists in Appendix A. When calculating the scores for both evidence candidates, we used existing user-item pairs, and calculated gradients from the model with one local update (*i.e.*, MeLU-1). On the third page, accepting the movies rated by the user among the twenty movies, our model presented new twenty movies that the user might like. We only considered the one-round interaction to verify the effectiveness of extreme user cold-state, but it could be expanded into a multiple-round interactions with the simple expansion on the third page. Additionally, we collected the ratings of the recommended items on a five-point Likert scale, as on the second page. Based on the survey, we evaluated which movie list more accurately discriminated user preferences.

5.2 Results of User Study

We quantitatively validated the evidence candidate selection via a user study. We surveyed eighty-four users in total. Forty-five of the users rated movies extracted by popularity only, and the others rated movies extracted using our strategy. Table 3 shows the summary of the survey results. Naturally, on the second page, our evidence candidates were rarely rated by users compared to the popularity-based evidence candidates. While approximately one-half of the popularity-based evidence candidates were chosen, only one-quarter of our evidence candidates were selected. However, on the third page, the proposed strategy showed better performance on all indicators. The respondents chose more movies, and the average ratings and $nDCG_1$ of the suggested movies were higher. Note that the proposed strategy used smaller evidences than popularity-based strategy for the local update, although the average ratings on the evidences were similar. Therefore, our strategy provides reliable evidence candidates and can quickly identify individual preferences of new users for the items.

We interviewed some of the participants to qualitatively analyze the users' response to the system. The reactions from two groups conflicted. One user assigned to policy A (*i.e.*, popularity-based evidence) said that she could made a lot of choices among the movies on the second page because she knew most of them, but she could not chose any recommended movies on the third page. Another user assigned to the same policy answered that the recommended movies did not seem to grasp his taste. By contrast, a user assigned to policy B (*i.e.*, proposed strategy-based evidences) responded that it was nice to have more preferred movies on the third page than on the second page.

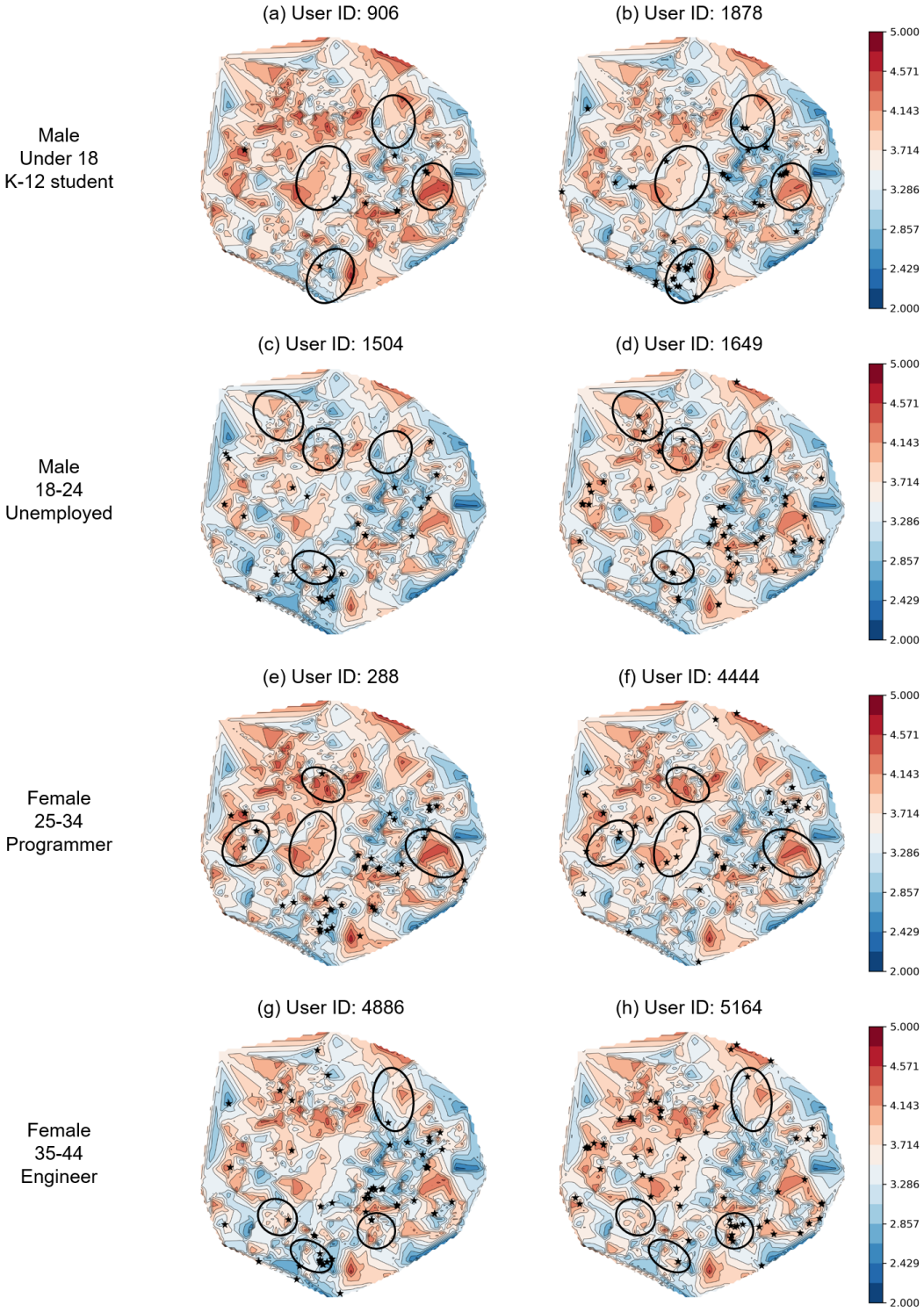

Figure 7: The estimated preference for whole movies by MeLU-1 with respect to users. Users like movies located in the red area and dislike movies in the blue area. The black stars represent the movies rated by the user. In each row, the areas with a difference on the contour map are indicated by black circles.

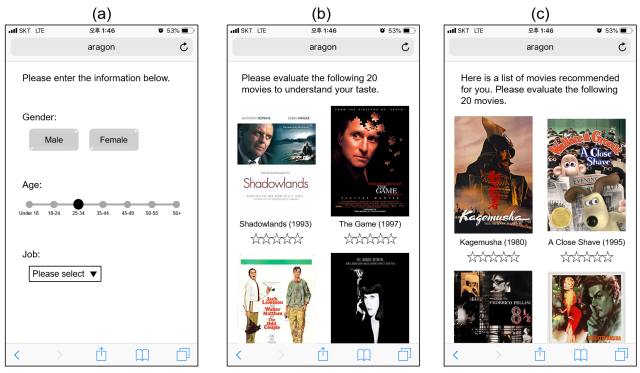

Figure 8: Screenshots of the user survey. (a) First page: survey on user information. (b) Second page: survey on evidence candidates. (c) Third page: survey on recommended items.

Table 3: Survey results on evidence candidate selection. On the left side of the table, we denote the evidence candidates as EC, and the recommended items as RI, which were used to calculate the measures of the corresponding rows.

	Strategy	Popularity-based	Ours
EC	Number of users	45	39
	Avg. number of selection	10.89	5.31
	Avg. ratings	4.02	3.96
RI	Avg. number of selection	3.29	4.28
	Avg. ratings	3.94	4.29
	Avg. $nDCG_1$	0.2756	0.3692

6 CONCLUSION

In this paper, we proposed a MAML-based recommender system that can identify personalized preferences. Our model was able to estimate user preferences based on only a small number of items. We found that the proposed method outperforms two methods on two benchmark datasets. Moreover, we qualitatively found that the local update using item-consumption history effectively identifies user preferences. In addition, based on the local update, we devised a strategy to select evidence candidates. As a result of an A/B test with eighty-four users, the candidates of the proposed strategy were selected less often than those of the popularity-based strategy, but the users were more satisfied with the recommendation results by the proposed strategy-based evidences.

Three promising topics for future research remain. First, the model update cycle must be verified before real-world applications. Although meta-learning is good at learning new things, the performance may not be guaranteed when an entirely new type of user appears in the recommender system. Second, future studies should explore the variants of evidence candidate selection strategy. When calculating the scores for evidence candidate selection, the weighted summation or the addition of other factors may yield better results. Third, the variants of meta-learning based recommender system should be studied. For example, meta-learning based collaborative filter could be designed.

REFERENCES

[1] Oren Barkan and Noam Koenigstein. 2016. Item2vec: neural item embedding for collaborative filtering. In *IEEE MLSP*. 1–6.
[2] Qi Cai, Yingwei Pan, Ting Yao, Chenggang Yan, and Tao Mei. 2018. Memory Matching Networks for One-Shot Image Recognition. In *CVPR*. 4080–4088.
[3] Heng-Tze Cheng, Levent Koc, Jeremiah Harmsen, Tal Shaked, Tushar Chandra, Hrishi Aradhye, Glen Anderson, Greg Corrado, Wei Chai, Mustafa Ispir, et al. 2016. Wide & deep learning for recommender systems. In *DLRS*. 7–10.
[4] Chelsea Finn, Pieter Abbeel, and Sergey Levine. 2017. Model-Agnostic Meta-Learning for Fast Adaptation of Deep Networks. In *ICML*. 1126–1135.
[5] F Maxwell Harper and Joseph A Konstan. 2016. The movielens datasets: History and context. *ACM Transactions on Interactive Intelligent Systems* 5, 4 (2016), 19.
[6] Xiangnan He, Lizi Liao, Hanwang Zhang, Liqiang Nie, Xia Hu, and Tat-Seng Chua. 2017. Neural collaborative filtering. In *WWW*. 173–182.
[7] Diane Kelly and Jaime Teevan. 2003. Implicit feedback for inferring user preference: a bibliography. In *SIGIR Forum*, Vol. 37. 18–28.
[8] Gregory Koch, Richard Zemel, and Ruslan Salakhutdinov. 2015. Siamese neural networks for one-shot image recognition. In *ICML 2015 Deep Learning Workshop*, Vol. 2.
[9] Pigi Kouki, Shobeir Fakhraei, James Foulds, Magdalini Eirinaki, and Lise Getoor. 2015. Hyper: A flexible and extensible probabilistic framework for hybrid recommender systems. In *RecSys*. 99–106.
[10] Xiaopeng Li and James She. 2017. Collaborative variational autoencoder for recommender systems. In *KDD*. 305–314.
[11] Zhenguo Li, Fengwei Zhou, Fei Chen, and Hang Li. 2017. Meta-SGD: Learning to learn quickly for few shot learning. *arXiv preprint arXiv:1707.09835* (2017).
[12] Greg Linden, Brent Smith, and Jeremy York. 2003. Amazon. com recommendations: Item-to-item collaborative filtering. *IEEE Internet Computing* 1 (2003), 76–80.
[13] Laurens van der Maaten and Geoffrey Hinton. 2008. Visualizing data using t-SNE. *Journal of Machine Learning Research* 9, Nov (2008), 2579–2605.
[14] Sean M McNee, John Riedl, and Joseph A Konstan. 2006. Being accurate is not enough: how accuracy metrics have hurt recommender systems. In *CHI'06 Extended Abstracts*. 1097–1101.
[15] Raymond J Mooney and Loriene Roy. 2000. Content-based book recommending using learning for text categorization. In *DL*. 195–204.
[16] Vinod Nair and Geoffrey E Hinton. 2010. Rectified linear units improve restricted boltzmann machines. In *ICML*. 807–814.
[17] Fedelucio Narducci, Pierpaolo Basile, Cataldo Musto, Pasquale Lops, Annalina Caputo, Marco de Gemmis, Leo Iaquinta, and Giovanni Semeraro. 2016. Concept-based item representations for a cross-lingual content-based recommendation process. *Information Sciences* 374 (2016), 15–31.
[18] Seung-Taek Park and Wei Chu. 2009. Pairwise preference regression for cold-start recommendation. In *RecSys*. 21–28.
[19] Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. 2017. Automatic differentiation in PyTorch. In *NIPS'17 Workshop Autodiff*.
[20] Adam Santoro, Sergey Bartunov, Matthew Botvinick, Daan Wierstra, and Timothy Lillicrap. 2016. Meta-learning with memory-augmented neural networks. In *ICML*. 1842–1850.
[21] Suvash Sedhain, Aditya Krishna Menon, Scott Sanner, and Lexing Xie. 2015. Autorec: Autoencoders meet collaborative filtering. In *WWW*. 111–112.
[22] Jake Snell, Kevin Swersky, and Richard Zemel. 2017. Prototypical networks for few-shot learning. In *NIPS*. 4077–4087.
[23] Florian Strub, Romaric Gaudel, and Jérémie Mary. 2016. Hybrid recommender system based on autoencoders. In *DLRS*. 11–16.
[24] Aaron Van den Oord, Sander Dieleman, and Benjamin Schrauwen. 2013. Deep content-based music recommendation. In *NIPS*. 2643–2651.
[25] Manasi Vartak, Arvind Thiagarajan, Conrado Miranda, Jeshua Bratman, and Hugo Larochelle. 2017. A Meta-Learning Perspective on Cold-Start Recommendations for Items. In *NIPS*. 6904–6914.
[26] Ricardo Vilalta and Youssef Drissi. 2002. A perspective view and survey of meta-learning. *Artificial Intelligence Review* 18, 2 (2002), 77–95.
[27] Oriol Vinyals, Charles Blundell, Tim Lillicrap, Daan Wierstra, et al. 2016. Matching networks for one shot learning. In *NIPS*. 3630–3638.
[28] Jian Wei, Jianhua He, Kai Chen, Yi Zhou, and Zuoyin Tang. 2017. Collaborative filtering and deep learning based recommendation system for cold start items. *Expert Systems with Applications* 69 (2017), 29–39.
[29] Xing Yi, Liangjie Hong, Erheng Zhong, Nanthan Nan Liu, and Suju Rajan. 2014. Beyond clicks: dwell time for personalization. In *RecSys*. 113–120.
[30] Fuzheng Zhang, Nicholas Jing Yuan, Defu Lian, Xing Xie, and Wei-Ying Ma. 2016. Collaborative knowledge base embedding for recommender systems. In *KDD*. 353–362.
[31] Cai-Nicolas Ziegler, Sean M McNee, Joseph A Konstan, and Georg Lausen. 2005. Improving recommendation lists through topic diversification. In *WWW*. 22–32.

APPENDIX

A LISTS OF EVIDENCE CANDIDATES.

Table A.1: Comparison of popularity-based and proposed strategy-based evidence candidates. All movies are sorted by the score of each method. Note that only three movies are overlapped (marked in bold).

Rank	Popularity-based strategy		Our strategy	
	Title	Score	Title	Score
1	Star Wars: Episode IV - A New Hope (1977)	1.00000	Shadowlands (1993)	0.05576
2	Star Wars: Episode V - The Empire Strikes Back (1980)	0.93272	The Game (1997)	0.01032
3	**Star Wars: Episode VI - Return of the Jedi (1983)**	0.89677	The Odd Couple (1968)	0.01021
4	**Terminator 2 (1991)**	0.83963	The Crying Game (1992)	0.00846
5	Jurassic Park (1993)	0.80415	2010 (1984)	0.00375
6	The Silence of the Lambs (1991)	0.78295	**L.A. Confidential (1997)**	0.00282
7	Braveheart (1995)	0.76866	Contact (1997)	0.00224
8	Fargo (1996)	0.76544	Copycat (1995)	0.00216
9	Back to the Future (1985)	0.74332	Jaws (1975)	0.00128
10	Raiders of the Lost Ark (1981)	0.73871	The Terminator (1984)	0.00121
11	Schindler's List (1993)	0.73733	Star Trek: First Contact (1996)	0.00119
12	Men in Black (1997)	0.72581	**Star Wars: Episode VI - Return of the Jedi (1983)**	0.00118
13	**L.A. Confidential (1997)**	0.72442	Beauty and the Beast (1991)	0.00102
14	The Godfather (1972)	0.67742	**Terminator 2 (1991)**	0.00090
15	The Shawshank Redemption (1994)	0.67051	Manhattan (1979)	0.00083
16	The Princess Bride (1987)	0.66728	Twelve Monkeys (1995)	0.00073
17	Groundhog Day (1993)	0.61429	Clerks (1994)	0.00070
18	Forrest Gump (1994)	0.61014	Big (1988)	0.00064
19	Pulp Fiction (1994)	0.59724	Emma (1996)	0.00062
20	E.T. the Extra-Terrestrial (1982)	0.59078	Die Hard (1988)	0.00057