Alleviating the Sparsity in Collaborative Filtering using Crowdsourcing*

Jongwuk Lee, Myungha Jang†, Dongwon Lee‡
The Pennsylvania State University, PA, USA
{jxl90,dongwon}@psu.edu

Won-Seok Hwang, Jiwon Hong, Sang-Wook Kim
Hanyang University, Seoul, Korea
{hws23,nowiz,wook}@hanyang.ac.kr

ABSTRACT

As a novel method for alleviating the sparsity problem in collaborative filtering (CF), we explore *crowdsourcing-based CF*, namely CrowdCF, which solicits new ratings from the crowd. We study three key questions that need to be addressed to effectively utilize CrowdCF: (1) how to select items to show for crowd workers to elicit extra ratings, (2) how to decide the minimum quantity asked to the crowd, and (3) how to handle the erroneous ratings. We validate the effectiveness of CrowdCF by conducting offline experiments using real-life datasets and online experiments on Amazon Mechanical Turk. The best configuration of CrowdCF improves system-wide MAE by 0.07 and 0.03, and F1-score by 4% and 2% in offline and online experiments, compared to the state-of-the-art CF algorithm.

1. INTRODUCTION

Collaborative Filtering (CF) is one of the prevalent recommender techniques widely used in e-commerce. The key idea behind CF is to utilize the preference patterns of users for predicting unobserved ratings of *items*, *e.g.*, movies, books, and music. Unlike other recommendation algorithms, CF only exploits the *observed matrix* of m users and n items, $O \in \mathbb{R}^{m \times n}$. One challenging issue in CF is to manage the sparsity of O. Because most users have limited experience on items, the number of observed ratings in O is inevitably insufficient, thereby incurring the *data sparsity* problem. The degree of sparsity in well-known benchmark datasets such as MovieLens and Netflix ranges from 95% to 98%.

To alleviate the sparsity problem, a natural idea is to "fill-up" more cells in O. Specifically, [10] imputes unobserved ratings using heuristics, and [8, 9] exploits external information such as user demographics or trust relationships to complement the sparsity. As an alternative, *Active Learning (AL)* asks target users to provide more ratings in O so that CF learns user preferences more precisely [13]. AL is effective for improving the prediction accuracy for target users, but is inapplicable for all users.

Crowdsourcing is a new computing paradigm for harnessing human computation. Amazon Mechanical Turk (AMT), one of popular crowdsourcing platforms, issues a mirco-task, called *human intelligence task* (*HIT*), that is answered by human participants, called *workers*. In this paper, we propose to improve CF *not* by filling up cells in O, but by *appending more ratings elicited from a large number of crowds.* Our *crowdsourcing-based CF*, namely CrowdCF, is complementary to existing solutions [8, 9, 10, 13] and can be used together. Specifically, we focus on validating if CrowdCF is effective in the movie domain.

PROBLEM 1 Given an observed matrix $O \in \mathbb{R}^{m \times n}$ made by existing users and a new matrix $W \in \mathbb{R}^{w \times n}$ made by the crowd, can the CF algorithm predict an unobserved rating o_{ij} of an arbitrary user in O more accurately using augmented matrix $A \in \mathbb{R}^{(m+w) \times n}$ than using O? ■

Due to the idiosyncratic nature of crowdsourcing, a few important challenges arise. We need to consider diverse factors, *i.e.*, accuracy, budget, and latency. With the focus of the accuracy in the movie recommendation domain, we tackle the following challenging issues in CrowdCF:

Elicitation strategy: It is implausible to show entire items to the crowd for ratings. We select a small set of items that are effective for improving CF, and compare various elicitation strategies in offline and online scenarios, respectively.

Minimum number of ratings: If crowd workers add more sparse ratings to W by rating only 1-2 items per worker, it is unlikely to improve the accuracy of CF. Thus, we need to enforce workers to rate more than a minimum number of ratings in order to collect at least the guaranteed number of ratings. However, setting this minimum effectively is tricky. If it is too low, the quantity of ratings is limited. If too high, the quality of ratings can be sacrificed as workers would be tempted to add random ratings to some items.

Spam worker detection: While the availability of millions of crowd workers is an advantage of CrowdCF, the existence of many spam workers is a downside. It is therefore necessary to detect potential spam workers who give random ratings to movies that they never watched. We use fake-image-based spam detection for our HIT design.

†This work was done while working as a research intern at Penn State University (email: mhjang@cs.umass.edu).
‡Corresponding author
*This work (Grants No. C0006278) was supported by Business for Cooperative R&D between Industry, Academy, and Research Institute funded by Korea Small and Medium Business Administration in 2013.

2. RELATED WORK

Collaborative Filtering. As one of the popular techniques, *Collaborative Filtering (CF)* has been widely used in recommender systems. In general, CF can be categorized as two types: *memory-based* and *model-based* approaches [1]: (1) memory-based approaches exploit the neighborhood by computing the similarity between users or items, and (2) model-based approaches design a model to learn users' preference patterns by leveraging observed user ratings as training data. Since CF is usually computed with an observed matrix that is extremely sparse, it inherently suffers from the *data sparsity* problem. To alleviate this problem, the *imputation-based* approach [10] fills more cells by estimating the ratings of unrated items from other rated items. Huang et al. [8] employed *demographic information* as external resources in addition to observed ratings, and the *trust-based* approach [9] makes use of trust relationships between users to complement the sparse observed matrix.

Active Learning to CF. As an alternative approach to alleviating the sparsity problem in CF, *Active Learning (AL)* focuses on identifying the preferences of target users, as reviewed in [13]. Some work developed active learning algorithms on the basis of model-based approaches. Specifically, [3] observed that the usefulness of collected ratings can be different depending on items rated by each user, thereby developing a *multiple-cause vector quantization model*. However, this assumed that users can provide the ratings for any items. To relax this assumption, [7] extended Bayesian active learning by distinguishing the items that the user can rate. Our crowdsourcing-based CF is inspired by AL, but is designed to "augment" the observed matrix by adding more new ratings by the crowd.

Crowdsourcing. Crowdsourcing has recently become popular for integrating human and machine computations, which is well-surveyed in [4]. Crowdsourcing is known to be effective for improving the accuracy of machine-based algorithms, *e.g.*, word processing [2], image search [15], and query processing [6].

3. PROPOSED FRAMEWORK

Given a set of m users $\mathcal{U} = \{u_1, \ldots, u_m\}$ and a set of n items $\mathcal{E} = \{e_1, \ldots, e_n\}$, a set of ratings can be represented by an *observed matrix* $O \in \mathbb{R}^{m \times n}$, where each entry o_{ij} corresponds to the rating of user u_i for item e_j. If o_{ij} is "observed," it is associated with a positive integer value, *e.g.*, $o_{ij} \in \{1, \ldots, 5\}$ in movie datasets. Otherwise, o_{ij} is set by zero, implying an "unobserved" rating. Let y_{ij} denote a binary variable for o_{ij}. If o_{ij} is rated by user u_i, then y_{ij} is one. Otherwise, y_{ij} is zero.

We present a *crowdsourcing-based CF framework*, termed as CrowdCF. Specifically, we address the following challenging issues in CrowdCF: (1) how to select a set of items to show to workers; (2) how to determine the minimum number of ratings that are requested to workers; and (3) how to remove spam workers who incur noisy ratings.

3.1 Designing Elicitation Strategies

The first step of CrowdCF is to display a set of movie images to workers. Since showing all movie items to workers at once is unreasonable, we select a set of items s ($s \leq n$). An *elicitation strategy* S is represented by a function $S(s, O, \mathcal{E}) = \mathcal{E}_{sel}$. That is, S selects a set of s items $\mathcal{E}_{sel} = \{e_1, \ldots, e_s\}$ out of $\mathcal{E}$ using O, *i.e.*, $\mathcal{E}_{sel} \subseteq \mathcal{E}$.

To decide $\mathcal{E}_{sel}$, we consider two criteria: the number of collected ratings and the usefulness of ratings. First, in order to elicit as many ratings as possible from crowd workers, an ideal S should select the items that workers are likely to rate, *e.g.*, popular or highly-rated movies. In this case, workers can elicit sufficient ratings for sparse items. Second, an ideal S needs to select informative items. This prevents that the rated items between workers do not overlap too much, and helps to get more informative ratings. It is thus important to balance the two criteria.

We explore various strategies that take the two criteria into account [5, 11, 12]. The elicitation strategies are independent of item domains and CF algorithms. In this paper, the elicitation strategies are plugged to probabilistic matrix factorization [14], which is known as the state-of-the-art CF algorithm. Given a function for measuring their criteria, each strategy sorts items in decreasing order of function scores, and selects a set of s items with highest scores.

Random (Rand): As a baseline strategy, Rand chooses s items out of $\mathcal{E}$ in a random manner.

Popularity (Pop) [11]: Pop uses the number of user ratings for each item, *i.e.*, the *frequency* of items. The frequency $freq(e_j)$ is computed by the number of observed ratings, *i.e.*, $freq(e_j) = \sum_{i=1}^{m} y_{ij}$.

Highest rating (HRating) [5]: With the same goal as Pop, HRating employs the average of observed ratings of item e_j, *i.e.*, $rating(e_j) = \sum_{i=1}^{m} o_{ij} / \sum_{i=1}^{m} y_{ij}$. This implies that workers are likely to give more ratings for items with highest average ratings.

Entropy (Ent) [11]: Ent uses the *entropy* for considering the uncertainty of observed ratings. Entropy is the dispersion of observed ratings, *i.e.*, $ent(e_j) = -\sum_{i=1}^{5} p_{ij} log_2(p_{ij})$, where p_{ij} is the probability of observed ratings that are equal to i for e_j.

Highest rating0 (HRating0): This improves HRating by considering unobserved ratings in O. When computing the average rating of item e_j, this takes into account the ratio of unobserved ratings, *i.e.*, $rating0(e_j) = \sum_{i=1}^{m} o_{ij}/m$.

Entropy0 (Ent0) [12]: To improve Ent, Ent0 exploits a *weighted entropy* function by considering unobserved ratings, *i.e.*, $ent0(e_j) = -\sum_{i=0}^{5} w_i p_{ij} log_2(p_{ij})$. As the ratio of unobserved ratings is much higher than that of observed ratings, w_0 is empirically tuned, and other weights $w_1, \ldots, w_5$ are set by $(1 - w_0)/5$.

Harmonic mean of entropy and logarithm of frequency (HELF) [12]: To combine Ent and Pop, HELF [12] exploits the harmonic mean of two metrics, *i.e.*, $helf(e_j) = 2 \times \frac{log_2(freq(e_j)) \times ent(e_j)}{log_2(freq(e_j)) + ent(e_j)}$. Because the two metrics have different ranges for $log_2(freq(e_j))$ and $ent(e_j)$, it is non-trivial to normalize the metric scores. To address this problem, we optimized HELF by employing the *rankings* of items, *i.e.*, $rank(freq(e_j))$ and $rank(ent(e_j))$, which can combine two metrics independently of score ranges. That is, we leverage the harmonic mean of two rankings for e_j, *i.e.*, $helf_{rank}(e_j) = 2 \times \frac{rank(freq(e_j)) \times rank(ent(e_j))}{rank(freq(e_j)) + rank(ent(e_j))}$.

In addition to these strategies, there are a number of plau-

sible strategies. Some strategies can be further optimized by leveraging domain-specific and algorithm-specific properties. Furthermore, different metrics can be used, *e.g.*, *diversification* and *coverage*. The presented strategies can be used as a guideline for designing better strategies.

3.2 Controlling Minimum Number of Ratings

Once $\mathcal{E}_{sel}$ is chosen by strategy S and presented to workers, each worker will rate only a subset of $\mathcal{E}_{sel}$ for a variety of reasons (*e.g.*, a worker cannot rate unseen movie items). Let $\mathcal{E}_{rated}$ be a set of r rated items, *i.e.*, $\mathcal{E}_{rated} = \{e_1, \dots e_r\}$ such that $\mathcal{E}_{rated} \subseteq \mathcal{E}_{sel}$, and let w be the number of workers. When $\mathcal{E}_{rated}$ is collected from w workers, supplementary ratings can be represented by a *worker matrix* $W \in \mathbb{R}^{w \times n}$. By vertically concatenating two matrices O and W, we can acquire an *augmented matrix* $A \in \mathbb{R}^{(m+w) \times n}$.

Note that if workers give ratings to only a few items (*e.g.*, rating only 2 movies from 100 movies presented, $|\mathcal{E}_{rated}| = 2$ and $|\mathcal{E}_{sel}| = 100$), then concatenating the new ratings is unlikely to improve the accuracy of CF, possibly making the whole matrix even sparser. We thus enforce workers to rate at least the *minimum number of items*, denoted by r_{min}. Otherwise, we reject the ratings from those workers without paying rewards.

The challenging issue is then how to determine r_{min}. Similar to the elicitation strategies, we need to consider two criteria: *worker effort* and the *accuracy* of ratings. When r_{min} is too high, we can accrue many ratings but workers are tempted to give more noisy ratings to meet the requirement. On the other hand, when r_{min} is too low, workers may give more accurate ratings to only the small number of items, yielding low impact to quality of worker matrix W.

Toward this issue, we propose three heuristics as the guideline for setting r_{min}. They can be used for designing a cost model for the trade-off between the two criteria. Let us use the frequency function for user u_i, *i.e.*, $freq(u_i) = \sum_{j=1}^{n} y_{ij}$, to denote the number of items rated by u_i.

Minimum frequency: We calculate the distribution of user ratings from O, and treat the *minimum* number of user ratings in O as the minimum effort that a worker should put in, *i.e.*, $r_{min} = min_{u_i \in \mathcal{U}}(freq(u_i))$.

Average frequency: Given O, a *sparsity* function for the degree of sparsity is calculated as $sp(O) = 1 - \frac{\sum_{i=1}^{m} freq(u_i)}{mn}$. If the sparsity of A decreases from that of O, it can be viewed as one benefit of CrowdCF. That is, the ideal scenario in CrowdCF meets the following inequality, $sp(A) \leq sp(O) \Leftrightarrow \frac{sp(O)+sp(W)}{2} \leq sp(O) \Leftrightarrow sp(W) \leq sp(O)$. Unfolding the inequality further, we get: $1 - \frac{\sum_{i=1}^{w} freq(u_i)}{wn} \leq 1 - \frac{\sum_{i=1}^{m} freq(u_i)}{mn} \Leftrightarrow \frac{\sum_{i=1}^{w} freq(u_i)}{w} \geq \frac{\sum_{i=1}^{m} freq(u_i)}{m}$. That is, in order to reduce the sparsity, the average number of ratings that we collect from workers should be no smaller than that from O. r_{min} can be set by the *average* number of user ratings in O, *i.e.*, $r_{min} = \frac{\sum_{i=1}^{m} freq(u_i)}{m}$.

Median frequency: We observed that O follows the Zipifan distribution in general. As a balanced way of the two heuristics, r_{min} can be set by the *median* ratings in O, *i.e.*, $r_{min} = median_{u_i \in \mathcal{U}}(freq(u_i))$.

3.3 Filtering Noisy Ratings

Compared to the case of active learning, the quality of ratings obtained from crowd-sourcing can be nosier. *Spam workers* may provide random ratings for movies that they have not watched to get paid. We thus have to identify and exclude those spam workers who incur noisy ratings. Specifically, we consider the following two features:

	MovieRating ($\mathcal{D}_{MR}$)	MovieLens ($\mathcal{D}_{ML1}$)	MovieLens ($\mathcal{D}_{ML2}$)
# of users (m)	500	943	6,040
# of items (n)	1,000	1,682	3,952
# of ratings	43,865	100,000	1,000,209
Avg. # of ratings	87.73	106.04	165.6

Table 1: Statistics on three datasets

Task-independent features: Regardless of tasks, we can avoid ill-qualified workers using AMT features such as geolocation, HIT approval rate, or category of workers. Empirically, we found that the average quality of movie ratings by "master" workers[1] was significantly higher than those of regular workers.

Task-dependent features: Depending on the given tasks, we use three domain-specific features. (1) Fake movie filtering: we inject fake movies to $\mathcal{E}_{sel}$, and we regard workers as spam workers if they rate the fake movies beyond a threshold. (2) The time spent for a HIT: workers who finish given HITs abnormally faster than others can be viewed as potential spam workers. (3) Correlation between ratings by a worker and average ratings from O: if the correlation is abnormally low, it may indicate that the worker randomly gave ratings.

4. OFFLINE EVALUATION

In this section, we conducted offline evaluation to simulate crowdsourcing. We randomly selected users as crowd workers. The main reason why we conducted the offline evaluation is, based on the assumption that crowd workers have similar characteristics with existing users, we can validate the effectiveness of CrowdCF as done in existing works for active learning [5, 11, 12], and test extensive experimental settings for parameter optimization.

4.1 Experimental Setup

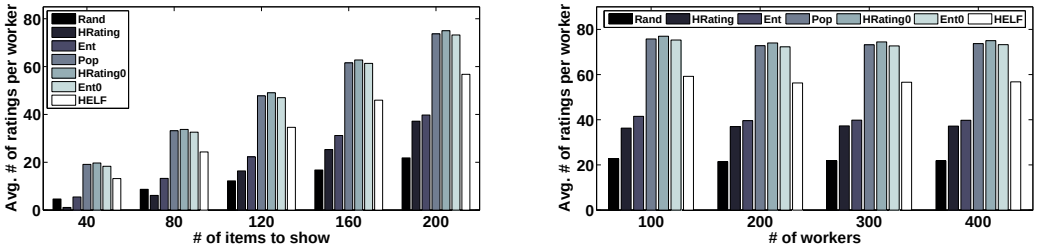
We used three datasets ($\mathcal{D}_{MR}$, $\mathcal{D}_{ML1}$, and $\mathcal{D}_{ML2}$), obtained from real-life movie datasets, *i.e.*, MovieRating[2] and MovieLens[3]. The details of the three datasets are summarized in Table 1. To simulate crowdsourcing in CF, each dataset is divided into three partitions: *crowdsourced* ($\mathcal{D}^W$), *observed* ($\mathcal{D}^O$), and *validated* ($\mathcal{D}^V$), respectively.

Crowdsourced ($\mathcal{D}^W$): To decide crowd workers, a total set of users is divided into disjoint two sets: existing users and simulated workers. The workers are randomly chosen among users with at least 70 ratings (*i.e.*, $freq(u_i) \geq 70$). These ratings in $\mathcal{D}^W$ are represented by new matrix W.

Observed ($\mathcal{D}^O$): Except for the crowdsourced partition, the rest of the dataset is divided into two partitions: observed and validated ratings. The observed ratings are used as an input for CF algorithms, represented by O.

Validated ($\mathcal{D}^V$): The validated ratings are unknown to CF algorithms, and are only used for evaluating the accuracy.

[1] AMT labels the most qualified workers as "master".
[2] `http://www.cs.usyd.edu.au/~irena/movie_data.zip`
[3] `http://www.grouplens.org/node/12`



(a) Varying $|\mathcal{E}_{sel}|$ (b) Varying w

Figure 1: Change of the number of collected ratings for various strategies ($\mathcal{D}_{ML1}$)

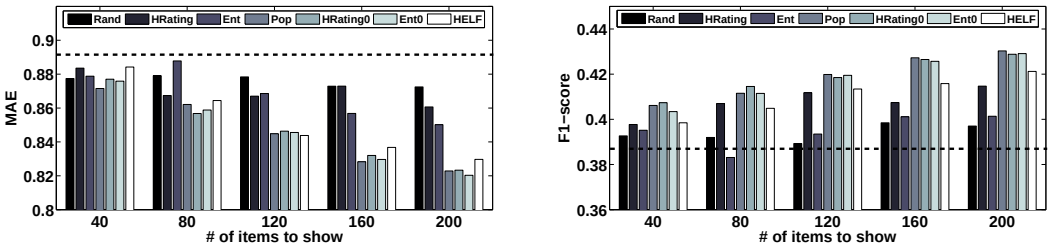


Figure 2: Change of MAE and F1-score with varying $|\mathcal{E}_{sel}|$ for various strategies ($\mathcal{D}_{ML1}$)

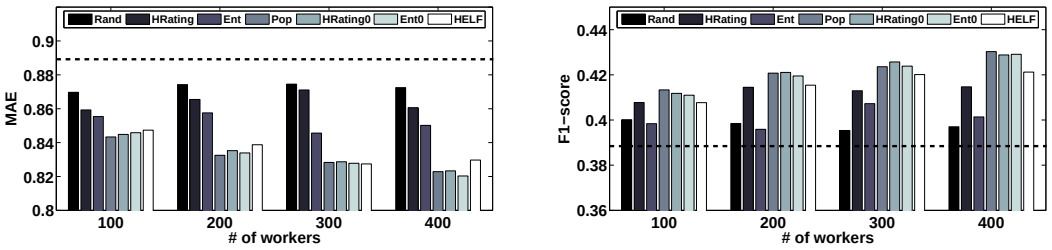


Figure 3: Change of MAE and F1-score with varying w for various strategies ($\mathcal{D}_{ML1}$)

For instance, for $\mathcal{D}_{ML1}$ with 943 users and 100,000 ratings, a range of 100–400 users (with a range of 19,000–73,000 ratings) are randomly chosen as crowd workers. Depending on the strategies, the numbers of ratings in $\mathcal{D}^W$ are ranged from 440–30,000 (Figure 1). 543 users with about 21,600 ratings are used for $\mathcal{D}^O$, and about 5,400 ratings are used for $\mathcal{D}^V$ with five-fold cross validation. Similar break-downs with three partitions are applied for other datasets.

We employed two types of evaluation measures: (1) the number of collected ratings by workers and (2) the accuracy such as *mean absolute error (MAE)*, *F1-score*, and *normalized discounted cumulative gain (NDCG)*. Given two strategies $S1$ and $S2$, if $S1$ yields a higher number of collected ratings than $S2$, $S1$ can be better than $S2$ in CrowdCF. Let V be a set of ratings in $\mathcal{D}^V$, and P be a set of predicted ratings for V. Given P and V, MAE is defined as: $\text{MAE} = \frac{1}{|V|}\sum_{e_{ij}\in V, \widehat{e}_{ij}\in P}|e_{ij} - \widehat{e}_{ij}|$, where $|V|$ is the number of ratings in V and $\widehat{e}_{ij}$ is the predicted rating of $e_{ij} \in V$. Let V_{high} and P_{high} be $\{e_{ij} \in V | e_{ij} = 4 \text{ or } 5\}$ and $\{\widehat{e}_{ij} \in P | \widehat{e}_{ij} = 4 \text{ or } 5\}$, respectively. Note that P_{high} is computed by rounding off predicted ratings. F1-score is computed by harmonic mean for *precision* and *recall* with P_{high} and V_{high}. We also used NDCG to consider more weights for higher ratings, and found that the results for NDCG show similar tendency to F1-score. (Due to the space limitation, we omit the results for NDCG, but provide the details of NDCG results in this web page[4].)

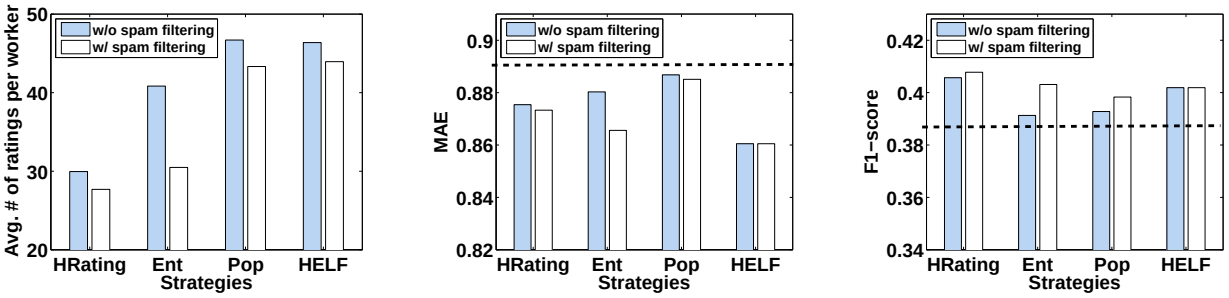
4.2 Experimental Results

We compared various strategies – Rand, HRating, Ent, Pop, HRating0, Ent0, and HELF with the following procedure. (1) Divide the dataset into three partitions – $\mathcal{D}^W$, $\mathcal{D}^O$, and $\mathcal{D}^V$; (2) For each worker, generate a set of items to show $\mathcal{E}_{sel}$, and collect a set of rated items $\mathcal{E}_{rated}$ out of $\mathcal{E}_{sel}$; (3) Run a recommender algorithm, *e.g.*, probabilistic matrix factorization[5] [14]. Note that this algorithm is executed in partitions containing both $\mathcal{D}^W$ and $\mathcal{D}^O$ (*i.e.*, augmented matrix A). We then measured both MAE and F1-score for $\mathcal{D}^V$; and (4) Repeat steps (1)–(3) with five-fold cross validation. For $\mathcal{D}_{ML1}$, we varied the number of items to show $|\mathcal{E}_{sel}|$: 40, 80, 120, 160, and 200, and the number of workers w: 100, 200, 300, and 400. By default, we set $|\mathcal{E}_{sel}| = 200$ and $w = 400$, respectively.

Number of Collected Ratings. Figure 1 reports on the average number of ratings collected per worker for different strategies using $\mathcal{D}_{ML1}$. The effect of parameters was observed as follows: (1) The average number of ratings per worker increased linearly proportional to $|\mathcal{E}_{sel}|$ for all strategies; (2) This stayed constant independently of w; and (3) Pop, HRating0, Ent0, and HELF collected the largest num-

[4] http://pike.psu.edu/download/crowdrec13/

[5] http://www.utstat.toronto.edu/~rsalakhu/BPMF.html



(a) Avg. # of ratings per worker (b) MAE (c) F1-score

Figure 4: Change of avg. # of ratings, MAE, and F1-score for various strategies ($\mathcal{D}_{ML1}$)

ber of ratings in all settings. For instance, when 200 items were shown to each worker, Rand only collected about 20 ratings (10%), and HRating and Ent collected 37∼40 ratings (18∼20%). In contrast, Pop, HRating0, Ent0, and HELF collected 70∼75 ratings (35∼37%). Meanwhile, when $|\mathcal{E}_{sel}|$ was 40 or 80, HRating collected the smallest number of ratings, implying that some items with high ratings may have been "unseen" by workers (*e.g.*, unpopular movies with good reviews by critics). HRating0 and Ent0, considering unobserved ratings in their formulas, collected 35∼40 ratings, thereby improving HRating and Ent. We found that the results of other datasets ($\mathcal{D}_{MR}$ and $\mathcal{D}_{ML2}$) were parallel to those of $\mathcal{D}_{ML1}$. (Please refer to our web page.)

Accuracy. Figure 2 depicts MAE and F1-score for the number of items to show $|\mathcal{E}_{sel}|$. We used a baseline algorithm (dotted line) with probabilistic matrix factorization [14], a state-of-the-art CF algorithm, only using O without crowdsourcing. First and foremost, every strategy in CrowdCF noticeably improved both MAE and F1-score from the baseline. This is because the sparsity in CF becomes low for some items, and the collected ratings in CrowdCF can thus help to predict unobserved ratings. As the answer for our main question in this paper, it empirically validated that *crowdsourcing is indeed effective for improving the accuracy of CF*. When $|\mathcal{E}_{sel}| = 40$, the difference of MAE across strategies is negligible. However, as $|\mathcal{E}_{sel}|$ increases, MAE decreases in Pop, HRating0, Ent0, and HELF. Similarly, increasing F1-score was proportional to $|\mathcal{E}_{sel}|$. In addition, Pop, HRating0, Ent0, and HELF consistently outperformed other strategies. One interesting observation is that Ent was better than HRating for MAE, but worse for F1-score. This is because HRating, focusing on items with high ratings of four or five, is more appropriate for F1-score. These findings were consistently observed in other datasets as well.

Figure 3 reports on MAE and F1-score over varying w. Compared to $|\mathcal{E}_{sel}|$ in Figure 2, MAE and F1-score were less sensitive to w. Specifically, Rand, HRating, and Ent stayed constant for MAE and F1-score. In contrast, Pop, HRating0, Ent0, and HELF improved better as w increases. In addition, we consistently observed that Ent was better than HRating for MAE, but vice versa for F1-score.

We lastly evaluated the minimum number of ratings r_{min} and the effect of spam workers by simulating the ratio δ of noisy ratings. For r_{min}, we found that the best setting of r_{min} is sensitive to the number of items to show. For example, when we show relatively small items, the average frequency or the median of r_{min} leads to lower accuracy, whereas they are good parameters when showing many items. It suggests that the number of items to show should be taken account when setting r_{min}. As for noisy ratings, we found that, as the number of collected ratings increases, the accuracy is improved even if some ratings are noisy. (Please see detailed experimental results in our web page.)

Parameters	Value
Strategies	HRating, Ent, Pop, HELF
# of items to show ($\|\mathcal{E}_{sel}\|$)	204 (real: 180, fake: 24)
Min. # of requested items (r_{min})	20
# of workers per strategy (w)	100
Reward per worker	$0.7

Table 2: Parameter settings used for our HITs

5. ONLINE EVALUATION

In this section, we conduct online experiments on AMT to validate CrowdCF in the real-life scenarios. We explain how to design HIT interface (Section 5.1), evaluate the accuracy of various strategies (Section 5.2), and examine our method used for filtering spam workers (Section 5.3).

5.1 Experimental Setup

We used $\mathcal{D}_{ML1}$ in online experiments. The key difference from the offline evaluation is that we exploit crowdsourced ratings obtained from workers on AMT. We issue a HIT that shows a certain number of movies, and ask workers to rate movies that they have watched. Workers are encouraged to rate as many movies as possible, but required to rate at least the minimum number of movies r_{min}. Each HIT shows a set of items $\mathcal{E}_{sel}$ populated from elicitation strategies. Table 2 shows parameter settings used in the online evaluation. We issued 100 assignments ($w = 100$) for four strategies, collecting ratings from a total of 400 workers, and set r_{min} as 20, evaluating the *minimum frequency* heuristic in Section 3.2. For other two heuristics (*i.e.*, average and median frequency), r_{min} is set by 106 and 80, it is unrealistic to ask workers to rate r_{min} movies.

5.2 Evaluating Elicitation Strategies

We compared various strategies in online settings. It was found in offline experiments that some strategies produced item sets with high correlations. We thus chose four strategies, *i.e.*, HRating, Ent, Pop, and HELF, that had the least correlation with each other.

Number of collected ratings. Figure 4(a) shows the average number of collected ratings per worker over the four strategies. These results compare two scenarios, without

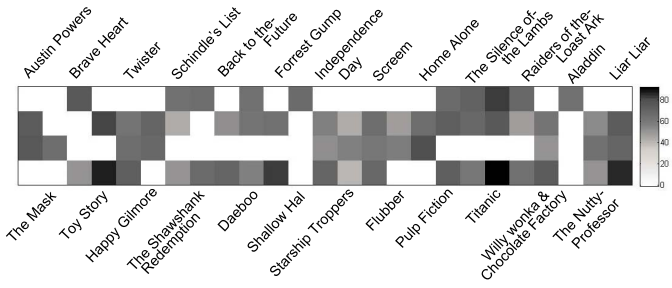


Figure 5: Correlation of rated movies for strategies (Each row from top to bottom corresponds to HRating, HELF, Ent, and Pop, respectively.)

and with spam worker filtering (to be explained in Section 5.3). We found the following observations: (1) Pop collects the largest number of ratings, which is consistent with the offline evaluation; (2) Out of 180 real movies that we showed to workers, we collected 45 ratings (22%) on average, which is much higher than $r_{min} = 20$. This shows that workers participated actively in our online evaluation; and (3) HRating was least effective for collecting ratings, implying that the movies with high ratings are not necessarily the ones that most workers are familiar with.

Figure 5 illustrates the correlation between 25 movies with the most ratings across strategies. The white cell indicates that the movie is not in $\mathcal{E}_{sel}$ for the corresponding strategy. The darker the cell is, the more the number of collected ratings is. We found that some movies are highly correlated although the movie sets to show differ from strategies, *e.g.*, "Pulp Fiction" and "Titanic" are mostly rated by workers regardless of strategies. Therefore, it is an interesting research direction to extend the coverage of selected movies.

Accuracy. Figures 4(b) and 4(c) depict MAE and F1-score for the four strategies. As observed in the offline evaluation, *we verified that crowdsourced ratings improve the accuracy of CF algorithms, compared to the baseline (dotted line) that does not use crowdsourced ratings.* Overall, HELF and HRating were the best strategies for MAE and F1-score, respectively. However, the performance gap between the strategies and the results without crowdsourcing narrows down compared to the offline experiments, as the number of collected ratings in the online evaluation was relatively small. If more ratings are collected, we expect that the accuracy can be improved as observed in offline experiments.

We also observed that crowdsourced ratings with spam filtering improved the accuracy compared to the ones without spam filtering in both metrics. The effect of spam workers varies over different strategies. In Ent, the ratio of rated fake movies was 13% including 4 spam workers. Meanwhile, in HELF, the ratio of rated fake movies was only 5%, all of which only checked 3 fake movies. Ent thus shows more drastic performance increase than HELF when spam workers were removed. In addition, as the number of collected ratings increases, crowdsourced ratings are less sensitive to the number of spam workers.

5.3 Filtering Spam Workers

We first applied a task-independent feature (*e.g.*, workers with HIT approval rate $\geq$ 95%) to avoid potential spam workers. We then performed spam filtering using two task-dependent features: *the number of fake movies rated* and *work time to finish the HIT*. When displaying a set of movies, we include some fake movies on purpose, *e.g.*, coming-soon movies to be released in 2014 and fan-made non-existent movies. Using the two features, we labeled a worker w_c as a spam worker, if (1) w_c rated more than 3 fake movies and (2) w_c's work time is too shorter than the average of workers.

To consider how carefully workers rate items, we analyzed the work time that each worker takes to complete the HIT. The assumption is that the more time workers take to rate items, the more careful they are. Our analysis show that obvious spam workers who rated more than five fake movies took about 3 seconds while non-spam workers who rated zero fake movies took about 10 seconds on average. The average work time per item was used as one of the filtering features.

6. CONCLUSION

In this paper, we proposed CrowdCF to enhance the accuracy of CF. In order to best utilize new ratings collected from the crowd, we studied the three important issues raised in CrowdCF: elicitation strategies, the minimum number of ratings, and spammer filtering. We evaluated both offline experiments and online experiments on AMT, and demonstrated that CrowdCF can be the complementary method for alleviating the sparsity problem in CF.

7. REFERENCES

[1] G. Adomavicius and A. Tuzhilin. Toward the next generation of recommender systems: A survey of the state-of-the-art and possible extensions. *IEEE Trans. Knowl. Data Eng.*, 17(6):734–749, 2005.

[2] M. S. Bernstein, G. Little, R. C. Miller, B. Hartmann, M. S. Ackerman, D. R. Karger, D. Crowell, and K. Panovich. Soylent: a word processor with a crowd inside. In *UIST*, pages 313–322, 2010.

[3] C. Boutilier, R. S. Zemel, and B. M. Marlin. Active collaborative filtering. In *UAI*, pages 98–106, 2003.

[4] A. Doan, R. Ramakrishnan, and A. Y. Halevy. Crowdsourcing systems on the world-wide web. *Commun. ACM*, 54(4):86–96, 2011.

[5] M. Elahi, V. Repsys, and F. Ricci. Rating elicitation strategies for collaborative filtering. In *EC-Web*, pages 160–171, 2011.

[6] M. J. Franklin, D. Kossmann, T. Kraska, S. Ramesh, and R. Xin. CrowdDB: answering queries with crowdsourcing. In *SIGMOD*, pages 61–72, 2011.

[7] A. Harpale and Y. Yang. Personalized active learning for collaborative filtering. In *SIGIR*, pages 91–98, 2008.

[8] Z. Huang, H. Chen, and D. D. Zeng. Applying associative retrieval techniques to alleviate the sparsity problem in collaborative filtering. *ACM Trans. Inf. Syst.*, 22(1):116–142, 2004.

[9] M. Jamali and M. Ester. TrustWalker: a random walk model for combining trust-based and item-based recommendation. In *KDD*, pages 397–406, 2009.

[10] H. Ma, I. King, and M. R. Lyu. Effective missing data prediction for collaborative filtering. In *SIGIR*, pages 39–46, 2007.

[11] A. M. Rashid, I. Albert, D. Cosley, S. K. Lam, S. M. McNee, J. A. Konstan, and J. Riedl. Getting to know you: learning new user preferences in recommender systems. In *IUI*, pages 127–134, 2002.

[12] A. M. Rashid, G. Karypis, and J. Riedl. Learning preferences of new users in recommender systems: an information theoretic approach. *SIGKDD Explorations*, 10(2):90–100, 2008.

[13] N. Rubens, D. Kaplan, and M. Sugiyama. Active learning in recommender systems. In *Recommender Systems Handbook*, pages 735–767. Springer, 2011.

[14] R. Salakhutdinov and A. Mnih. Probabilistic matrix factorization. In *NIPS*, 2007.

[15] T. Yan, V. Kumar, and D. Ganesan. CrowdSearch: exploiting crowds for accurate real-time image search on mobile phones. In *MobiSys*, pages 77–90, 2010.