2019 IEEE 35th International Conference on Data Engineering (ICDE)

# No, That's Not My Feedback: TV Show Recommendation Using Watchable Interval

Kyung-Jae Cho[1*], Yeon-Chang Lee[1*], Kyungsik Han[2], Jaeho Choi[3], and Sang-Wook Kim[1†]
[1]*Hanyang University*, Seoul, South Korea
{kcho035, lyc0324, wook}@hanyang.ac.kr
[2]*Ajou University*, Suwon, South Korea
kyungsikhan@ajou.ac.kr
[3]*Naver Corporation*, Seongnam, South Korea
choi.jaeho@navercorp.com

***Abstract*—As the number of TV channels increases, it is becoming important to recommend TV shows that users prefer to watch. To this end, we investigate the *inherent characteristics* of implicit feedback given in the TV show domain, and identify the *challenges* for building an effective TV show recommendation. Based on the unique characteristics, we define a user's *watchable interval*, the most important and novel concept in understanding users' true preferences. In order to reflect this new concept into the TV show recommendation, we propose a novel framework based on collaborative filtering. Our framework is composed of (1) preference estimation based on a user's *watchable interval*, (2) preference prediction based on *confidence* exploiting *watchable episodes*, and (3) top-$N$ recommendation considering TV show's *staying and remaining times*. Using a real-world TV show dataset, we demonstrate that our framework effectively solves the challenges and significantly outperforms other existing state-of-the-art methods.**

***Index Terms*—Recommender systems, TV show recommendation, implicit feedback, watchable interval, watchable episode**

## I. Introduction

Collaborative filtering (CF) [1]–[3] is one of the popular recommendation methods that rely on users' past behaviors such as explicit or implicit feedback. *Explicit feedback* refers to the action where users add *direct* preferences (*e.g.*, star rating); its representative domain includes recommendations on movies, products, etc [4], [5]. *Implicit feedback* is an *indirectly* expressed preference, inferred from user behaviors (*e.g.*, browsing history); its representative domain includes recommendations on news, music, etc [6]–[9].

While a recommendation system can use both explicit and implicit feedback, explicit feedback depends on users' willingness to share their thoughts directly, which often times asks users to perform extra work. For this reason, although it is less direct than explicit feedback, implicit feedback has greater availability and can be used better in a practical sense [10]–[12]. Among many recommendation domains, this paper focuses on the domain of TV shows in which various types of implicit feedback are generated and complex interactions exist between users and TV shows.

*The paper has two first authors having contributed equally to this work.
†Corresponding author.

A recommendation system for TV shows has become more important as the number of channels is increasing (*e.g.*, more than 100 channels) [13]. This often makes users difficult or take time to find the TV shows that they prefer. For example, users have to move or stay on some channels for deciding which show to watch continuously. While moving to other channels, they may miss watching some content that they prefer. Therefore, a recommendation system that considers both user preferences and the *currently broadcasting shows* will enhance user experience in watching TV.

Data in a TV show recommendation system have many unique characteristics, as illustrated in Fig. 1. Fig. 1-(a) explains a timetable of TV shows from three channels. Fig. 1-(b) shows various cases of users' watching TV shows (*e.g.*, switching to different episodes of the different TV shows). Fig. 1-(c) shows the composition of a TV show, consisting of a series of its episodes. Based on Fig. 1, we describe the unique characteristics of (1) the TV show and (2) the user.

A TV show can consist of several episodes. Fig. 1-(c) depicts that one TV show, 'Game of Thrones,' consists of several episodes. For each episode of the TV show, (1) its start time and (2) length of time to be broadcast may be different. These two factors determine the *broadcasting interval of each episode*. Fig. 1-(a) depicts that different episodes (*e.g.*, $C1$-1, $C2$-2, and $C3$-2) of three TV shows start at different times and have different lengths of broadcasting time. Therefore, we note that through these characteristics, the time at which a user can watch an episode of a TV show is determined by the broadcasting interval of that episode.

In the case of a user, whenever a user watches TV, she will have (1) different start time of watching and (2) different length of watching time. These two factors determine each user's *TV watching interval*. Fig. 1-(b) shows that users $U1$ and $U2$ have two different TV watching intervals (with dotted boxes). In this case, the episodes of the TV shows that each user (*e.g.*, user $U1$) can watch are only those having the overlapped intervals between *broadcasting intervals* of the episodes and $U1$'s *TV watching interval*. $U1$ selects and then watches only a few episodes that *she prefers* among the episodes of the TV shows that *she can watch*.

For the selected episodes of TV shows, (1) the time at which

2375-026X/19/$31.00 ©2019 IEEE
DOI 10.1109/ICDE.2019.00036

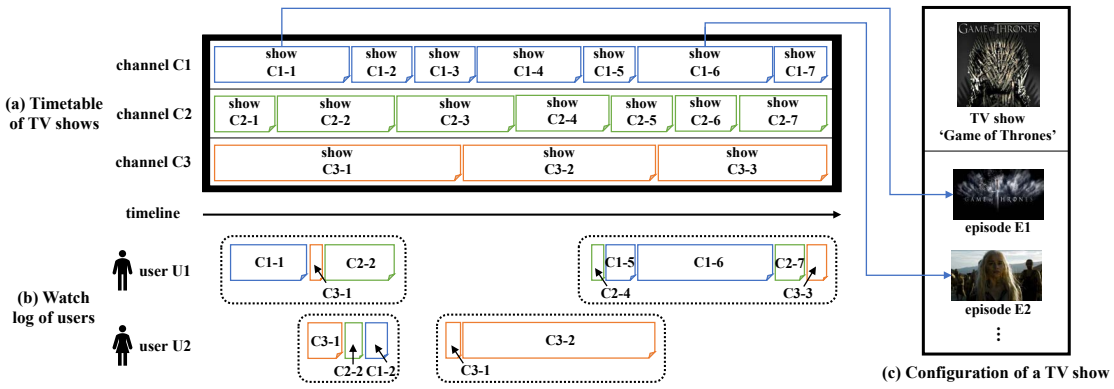



Fig. 1. Data in a TV show recommendation.

$U1$ began watching and (2) the length of time she watched may be different. These two factors determine $U1$'s *watching interval for each episode* of TV shows. Within $U1$'s first TV watching interval, in Fig. 1-(b), we can see that $U1$'s watching intervals for three episodes of TV shows are different. This indicates that *a user's feedback on a TV show is represented by her watching intervals for its episodes.*

Based on these characteristics, prior studies [10], [13]–[15] attempt to infer a user's preference on a TV show. The recent studies [13], [14] have focused on how long she watched the episodes of the TV show during the *entire broadcasting intervals* of the episodes. However, this approach assumes that a user gives feedback even on a part of broadcasting intervals for the episodes that she was not able to watch. Thus, *the feedback that is not expressed by her will be used in the wrong way* to infer her preference on the TV show.

For this reason, in this paper, we claim the importance of considering how long a user watched an episode of a TV show during her "*watchable interval*" (not broadcasting interval) for the episode. A user's *watchable interval for an episode* of a TV show is defined as the overlapped interval between her TV watching interval and the broadcasting interval of the TV show episode. Further, the user's "*watchable episodes*" are defined as the ones that were broadcasted within her whole TV watching intervals (*i.e.*, a set of intervals within which she could watch TV).

We identify three challenges to take our new concept into the TV show recommendation as follows: preference estimation (challenge #1); preference prediction (challenge #2); recommendation (challenge #3). To address all these challenges, we propose a novel TV show recommendation framework, considering a user's watchable interval on an episode.

For challenge #1, we first identify her watchable episodes of the TV show, estimate her preference on each episode with the consideration of a watchable interval on each of those episodes, and aggregate their preferences on all those episodes to estimate a preference of a corresponding TV show. For challenge #2, we employ two CF models, *weighted alternating least square* (wALS) [16] and *neural matrix factorization* (NeuMF) [11], which use not only both a user's and others' estimated preferences but also the notion of *confidence* for the estimated preferences on TV shows. For challenge #3, we *re-adjust* the scores of the user's predicted preferences by taking the *staying and remaining times* (*i.e.*, time factors) of the episodes at recommendation time into account [15].

Through extensive experiments with a real-world TV show dataset, we demonstrate that not only each of our ideas, strategies, and models in our framework is effective but also our final framework integrating all of them is the most effective in terms of recommendation accuracy. In addition, we show our proposed framework consistently and universally outperforms the state-of-the-art TV show recommendation methods for both of the following two test sets: (1) a set of correct answers from only the TV shows each user could watch in the past (*i.e.*, a test set for watchable TV shows, *W-test set* in short); (2) a set of correct answers from only the TV shows each user could not watch in the past (*i.e.*, a test set for non-watchable TV shows, *NW-test set* in short). Specifically, our framework remarkably outperforms PM [10], PNM [14], ShowTime [15], and RecTime [13] up to 250% / 330%, 57% / 247%, 14% / 123%, and 40% / 429%, respectively in terms of *normalized discounted cumulative gain* (nDCG) for *W-test set* / *NW-test set*.

Our contributions are summarized as follows:

- **Watchable interval**: We introduce a novel notion of *watchable intervals for watchable episodes*, which is very important in developing a TV show recommendation framework.
- **Preference estimation**: We estimate a user's preference on a watchable TV show by exploiting *her watchable interval for its watchable episodes*.
- **Preference prediction**: We predict a user's preference on a non-watchable TV show by considering the *confidence* on the estimated preferences on watchable TV shows.
- **TV show recommendation framework**: We develop a novel recommendation framework by integrating our preference estimation, preference prediction, and recommendation methods.
- **Evaluation**: We demonstrate that our proposed framework significantly outperforms the state-of-the-art methods for both *W-test set* and *NW-test set*.

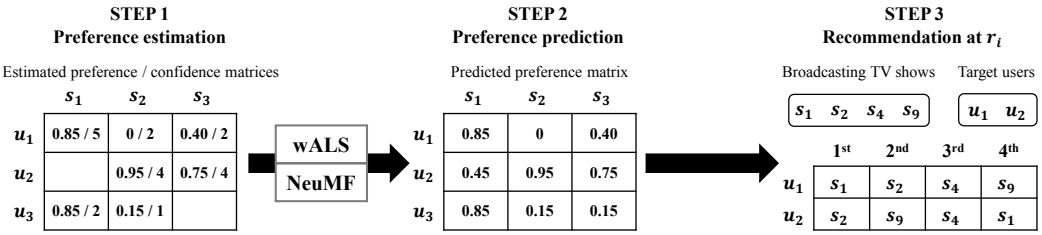



Fig. 2. Overview of our proposed framework.

The rest of this paper is organized as follows: Section II overviews existing studies for the TV show recommendation. Section III presents the overall procedure of the proposed framework, and Section IV evaluates the effectiveness of the proposed framework through extensive experiments. Finally, Section V summarizes and concludes the paper.

## II. Related Work

There has been a long-history of recommending TV shows in the recommender systems area. In early days of research, much research [17], [18] focused on constructing a framework that elicits user's *explicit feedback* rather than exploiting user's implicit feedback on TV shows. Based on the explicit feedback, conventional recommendation approaches such as content-based and CF-based approaches were used in a naive way to recommend a user's preferable TV shows.

However, eliciting explicit feedback to the user needs extra time and cost. For this reason, in recent years, much research [10], [13]–[15] has been carried out to infer the preferences of each TV show by analyzing a user's *implicit feedback*. In this section, we briefly review the state-of-the-art work for TV show recommendations based on a user's implicit feedback.

PM [10] *estimates* a user's preferences on TV shows by simply considering whether she watched the shows or not. PM sets a preference to 1 for all *watched* TV shows (otherwise to 0), and then *predicts* users' final preferences based on wALS, matrix factorization with a confidence. For the watched TV shows, a confidence is set to the number of watched times, and for the non-watched TV shows, it is set to 1. Finally, PM *recommends* to each user the TV shows having the highest predicted preferences for her.

PNM [14] *estimates* a user's preference on a TV show by considering the ratio of her watching time for the show to the broadcasting time of the show. For the non-watched shows or the shows watched for *a little time*, the preference is set to a very small value (*e.g.*, 0). Then, based on PMF (matrix factorization that uses a confidence similar to PM), PNM *predicts* users' final preferences for all TV shows. Finally, PNM *recommends* to each user the TV shows having the highest predicted preferences for her.

ShowTime [15] *estimates* the preference on a TV show by the sum of actually watched hours of its episodes, and then uses each estimated preference as a final preference without additional *prediction*. For *recommendation*, ShowTime *re-adjusts* a user's predicted preference on a TV show by using her *staying time* and the *remaining time* of the TV show at the time of recommendation. Finally, based on the re-adjusted preferences, ShowTime *recommends* to each user the TV shows having the highest predicted preferences for her.[1]

Lastly, RecTime [13] divides the broadcasting interval of a TV show into two factors in order to *estimate* the preference of a user for a TV show. One factor is to utilize the entrance time of the user and the other factor is to use the watching time during the *broadcasting interval* of each TV show. Then, RecTime *predicts* users' final preferences for all TV shows based on tensor factorization, PARAFAC. Finally, RecTime *recommends* to each user the TV shows having the highest predicted preferences for her.

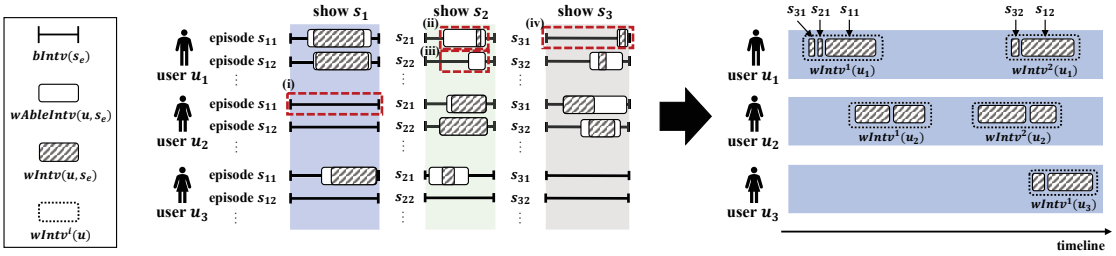
Although the aforementioned existing studies provided important insights on TV show recommendation, they still have some limitations with respect to inferring users' preferences on TV shows. First, PM [10] assumes that a user *equally prefers* all watched TV shows (*i.e.*, naive one-class setting), without considering the length of watching time for each TV show. Second, ShowTime [15] does not take into account the broadcasting interval of each TV show. In this case, the predicted preferences can be biased toward long TV shows. Lastly, PNM [14] and RecTime [13] have focused on how long a user watched the episodes of a TV show during the entire broadcasting intervals of the episodes. However, they assume that the user gives a feedback even on a part of broadcasting intervals for the TV show's episodes that she was not able to watch.

## III. Proposed Framework

To address the limitations of the existing work, we propose a novel TV show recommendation framework. It estimates and predicts the preferences of each user's TV show based on the *watchable interval*, which is a new notion primarily considered in our proposed framework. The framework includes three steps to address the three challenges mentioned in Section I: (1) preference estimation, (2) preference prediction, and (3) recommendation. Fig. 2 illustrates the process of the sequential execution of the framework for each step.

For step 1, we estimate a user $u$'s preference on a TV show $s$ by using *her watchable and watching intervals for its episodes* $s_e$, and the *broadcasting intervals of* $s_e$. Then, based on the estimated preferences, we construct an estimated preference matrix $E = (e_{u,s})_{m \times n}$ of $m$ users and $n$ TV shows. In $E$, when a user $u$ can watch a TV show $s$ more than once, $e_{u,s}$ is

[1] ShowTime can recommend only the TV shows that a user has watched.




(a) Three users' implicit feedback on three TV shows

(b) Time sequence for users' implicit feedback

Fig. 3. Examples of users' watch logs for TV shows.

filled with a value of 0 or more; otherwise, it is *left empty* as in Fig. 2. In addition, we calculate the confidence for each $e_{u,s}$ by exploiting the number of $u$'s watchable episodes in TV show $s$. We then construct a confidence matrix $C = (c_{u,s})_{m \times n}$ of $m$ users and $n$ TV shows based on the computed confidence. The matrices of step 1 in Fig. 2 refer to an example of $E$ and $C$ for three users and three TV shows. In each cell of the matrix, we have two values '$x$ / $y$', where '$x$' means the estimated preference and '$y$' means the confidence.

For step 2, we perform a well known CF technique, *weighted alternating least squares (wALS) or neural matrix factorization (NeuMF)*, based on $E$ and $C$, to predict users' TV show preferences. Then, based on these predicted preferences, we construct a predicted preference matrix $\hat{E} = (\hat{e}_{u,s})_{m \times n}$ of $m$ users and $n$ TV shows. The matrix of step 2 in Fig. 2 shows an example of $\hat{E}$.

For step 3, we re-adjust $\hat{e}_{u,s}$ of target user $u$ for TV show $s$, by considering the *staying and remaining times* of the broadcasting TV shows at *recommendation time* $r_i$. Finally, we recommend top-$N$ TV shows based on the final re-adjusted preferences of TV shows. Step 3 in Fig. 2 shows a recommendation result for $u_1$ and $u_2$ who are currently watching TV at time $r_i$.

### A. Notations

In this subsection, we first present intuitive definitions and examples of the notations using Fig. 3. Fig. 3 visually shows the examples of three users' watch logs for three TV shows: Fig. 3-(a) shows the implicit feedback of three users for the episodes of three TV shows. Note that Fig. 3-(a) depicts users' feedback *regardless of time* order. Fig. 3-(b) shows the *time sequence* of each user's feedback on the TV show episodes shown in Fig. 3-(a). For example, in Fig. 3-(b), user $u_1$ watched the TV shows in the order of $s_{31}$, $s_{21}$, and $s_{11}$ and ended watching TV. Later, $u_1$ watched the TV shows again in the order of $s_{32}$ and $s_{12}$.

We denote a *broadcasting interval of an episode* $s_e$ of a TV show $s$ as $bIntv(s_e)$. $bIntv(s_e)$ is represented by the starting and ending times of $s_e$. In Fig. 3-(a), a '$|-|$' represents a broadcasting interval for an episode. We see that episodes of three TV shows (*i.e.*, $s_1$, $s_2$, and $s_3$) in Fig. 3-(a) have the different lengths of $bIntv(s_e)$. We denote a set of all $s_e$ of a TV show $s$ as $eSet(s)$. Because a user can watch TV several times by her schedule, we denote the *i-th TV watching interval of a user* $u$ as $wIntv^i(u)$. $wIntv^i(u)$ is represented by the starting and ending times for the *i*-th TV watching of $u$. Fig. 3-(b) shows that $u_1$ has two different TV watching intervals (dotted boxes). We denote a set of all watching intervals $wIntv^i(u)$ of $u$ as $wIntvSet(u)$.

TABLE I
NOTATIONS

| Notation | Description |
|---|---|
| $u$, $s$, and $s_e$ | User, TV show consisting of multiple episodes, and TV show episode |
| $E = (e_{u,s})_{m \times n}$ | Estimated preference matrix whose entry is $e_{u,s}$ |
| $C = (c_{u,s})_{m \times n}$ | Confidence matrix whose entry is $c_{u,s}$ |
| $\hat{E} = (\hat{e}_{u,s})_{m \times n}$ | Predicted preference matrix whose entry is $\hat{e}_{u,s}$ |
| $ep(u, s_e)$ | $u$'s estimated preference on $s_e$ of $s$ |
| $p_{u,s,r_i}$ | $u$'s predicted preference on $s$ at $r_i$ |
| $wIntv^i(u)$ | TV watching interval that $u$ watched TV at $i^{th}$ time |
| $bIntv(s_e)$ | Broadcasting interval of $s_e$ of $s$ |
| $wAbleIntv(u, s_e)$ | Watchable interval of $s_e$ of $s$ that $u$ could watch |
| $wIntv(u, s_e)$ | Watching interval that $u$ watched $s_e$ of $s$ |
| $wIntvSet(u)$ | A set of $wIntv^i(u)$ |
| $eSet(s)$ | A set of all episodes of $s$ |
| $wAble\_eSet(u, s)$ | A set of watchable episodes of $s$ that $u$ could watch |
| $\lvert \cdot \rvert$ | The length of interval or the number of elements in a set |
| $r_i$ | $i^{th}$ recommendation time point |

We denote a *user $u$'s watchable interval for an episode* $s_e$ of a TV show $s$ as $wAbleIntv(u, s_e)$. $wAbleIntv(u, s_e)$ indicates the overlapped interval between $u$'s $wIntv^i(u)$ and $s_e$'s $bIntv(s_e)$. In Fig. 3-(a), a white box shows a user's watchable interval for an episode of a TV show. We can see that the lengths of $wAbleIntv(u, s_e)$ for different users on the same episode could be different from each other.

We denote a *watchable episode $s_e$ of user $u$* with $|wAbleIntv(u, s_e)| \neq 0$ as $wAbleEpisode$ of $u$. We denote a set of *wAbleEpisode* $s_e$ of a TV show $s$ that $u$ could watch as $wAble\_eSet(u, s)$. In Fig. 3-(a), all episodes with white boxes correspond to the $wAbleEpisodes$ of each user. For example, for $u_1$, both episodes $s_{11}$ and $s_{12}$ are *wAbleEpisodes* of $s_1$, whereas, for $u_2$, both episodes are not $wAbleEpisodes$ of $s_1$. Specifically, (i) in Fig. 3-(a) means that $u_2$ did not watch TV while $s_{11}$ was being broadcast. Thus, $s_{11}$ is not $wAbleEpisode$ for $u_2$.

In addition, we denote a *user $u$'s watching interval for an episode* $s_e$ of a TV show $s$ within $wAbleIntv(u, s_e)$ as $wIntv(u, s_e)$. $wIntv(u, s_e)$ indicates that $u$ has watched $s_e$ of $s$. Note that $wIntv(u, s_e)$ can exist only when $u$ can watch $s_e$ (*i.e.*, $wAbleIntv(u, s_e)$ exists). In Fig. 3-(a), a box filled with diagonal lines shows a user's watching interval for an episode of a TV show. We can see that the lengths of $wIntv(u, s_e)$

for different users on the same episode are different from each other. As in (iii), an episode with no diagonal lined box in a white box (*i.e.*, $wAbleIntv(u, s_e)$ exists, but $wIntv(u, s_e)$ does not exist) means that $u_1$ was able to watch $s_{22}$ but did not actually watch it. This happens because she watched an episode of other TV show. We denote the length of any interval defined above and the size of any set as $|\cdot|$. Table I summarizes a list of notations defined in this paper.

### B. Preference Estimation (PE)

As mentioned in Section I, because $bIntv(s_e)$ is different for each TV show episode $s_e$ and $wIntv^i(u)$ is also different for each user $u$, the time interval at which a user $u$ can watch an episode $s_e$ of a TV show $s$ is determined by $s_e$'s $bIntv(s_e)$ and $u$'s $wIntv^i(u)$ for TV watching. In addition, a user $u$'s feedback on a TV show $s$ is represented by $wIntv(u, s_e)$ for all its episodes $s_e$. Therefore, a user's preference on a TV show should be quantified by considering all of these intervals together (challenge #1).

To address challenge #1, we propose the following two ideas:

- **Idea #1: Consider *wAbleIntvs* of *wAbleEpisodes***: Because a user can give feedback to her *wAbleEpisodes* only in the range of the *wAbleIntv*, the preference of the episode is estimated by considering the ratio of her $wIntv$ to her *wAbleIntv* (*i.e.*, rather than *bIntv* as in PNM and RecTime) on the episode.
- **Idea #2: Aggregate feedback on *wAbleEpisodes***: For estimating a user's preference on a TV show, her preference on each *wAbleEpisode* of the TV show should be first estimated. Then, the final preference of the TV show is estimated by aggregating the estimated preferences of all of her *wAbleEpisodes* of the TV show.

We formally define a set of *wAbleEpisodes* for TV show $s$ of user $u$, $wAble\_eSet(u, s)$, as below:

$$wAble\_eSet(u, s) = \{s_e || wIntv^i(u) \cap bIntv(s_e)| > 0, \\ s_e \in eSet(s), wIntv^i(u) \in wIntvSet(u))\}. \quad (1)$$

Then, we first estimate the preference $ep(u, s_e)$ for each episode $s_e$ included in $wAble\_eSet(u, s)$.[2] To consider idea #1 in $ep(u, s_e)$, we formally define the *wAbleIntv* for an episode $s_e$ of $u$ as follows. $\forall wIntv^i(u) \in wIntvSet(u)$:

$$wAbleIntv(u, s_e) = wIntv^i(u) \cap bIntv(s_e). \quad (2)$$

Using the concept of *wAbleIntv*, we distinguish the feedback given to *wAbleEpisodes* by each user as follows:

- Feedback #1: A user skips a *wAbleEpisode* (*non-preferred*) while finding a user's preferred episode.
- Feedback #2: A user selects and watches a *wAbleEpisode* (*preferred*).
- Feedback #3: A user never reaches a *wAbleEpisode* (*probably non-preferred*).

[2]At step 1, we cannot estimate preferences for the episodes ((i) in Fig. 3-(a)) that are being broadcast when a user is not watching TV because the user has not provided any feedback, which is emphasized in the title of our paper. The preference prediction for these episodes is performed in step 2.

We classify these three types of feedback into two preferences: *positive and negative preferences*. Also, we propose two strategies for estimating these two preferences.

**PE-Strategy 1.** We consider *feedback #2* to be a positive preference because the episode selected by the user is the one that she finally stopped there. Here, we consider a positive preference when it meets the following condition: the ratio of $wIntv(u, s_e)$ to $wAbleIntv(u, s_e)$ is above a threshold $\alpha$. As indicated in idea #1, it is important to consider $u$'s *wAbleIntv* on $s_e$, not $bIntv$ of $s_e$. Note that if we estimate a user $u$'s preference $ep(u, s_e)$ of an episode $s_e$ based on $bIntv(s_e)$, we will reflect the time when user $u$ cannot watch episode $s_e$.

**PE-Strategy 2.** We consider *feedback #1* and *feedback #3* to be negative preferences. Because both the skipped and non-reached episodes given to feedback #1 and feedback #3 were not selected for the user in her $wAbleIntv$, we assume that she does not prefer the episodes. To consider these cases, if the ratio of $wIntv(u, s_e)$ to *wAbleIntv*$(u, s_e)$ is less than a threshold $\alpha$, it is determined that $u$ implicitly expresses that $s_e$ is not interesting to her.

Our intuition for feedback #3 behind PE-strategy 2 relates to the philosophy of *zero-injection* [19]–[21], which is our prior work that infers and further exploits users' negative preferences in existing CF techniques. Based on the interestingness of the items to which users gave feedback, zero-injection finds "some" of the items to which a user did not give feedback (*i.e.*, missing feedback) as *uninteresting items*. It then considers the user's feedback to such uninteresting items as negative feedback even though there is no actual feedback from users.

Here, thanks to $wAbleIntv$, we can distinguish two types of missing feedback (*i.e.*, episodes not watched by the user) of a user in the TV show domain as follows: (1) the episodes that the user could watch but did not watch (feedback #3); (2) the episodes that the user did not watch because they could not watch TV. Therefore, encouraged by the success of zero-injection [19], we consider the episodes with feedback #3 in missing feedback (meaning that those episodes were in $wAbleIntv$) to be implicitly expressed by the user as a *negative* preference. Except for feedback #3, the remaining missing feedback is set to an *unknown* preference as with [19].

For example, (ii) in Fig. 3-(a) (*i.e.*, feedback #1) is determined to be uninteresting by a user $u_1$ after the user watched the episode for a little time, and (iii) in Fig. 3-(a) (*i.e.*, feedback #3) means that it was not even watched by the user. Thus, we estimate $u_1$'s preferences for the episodes $s_{21}$ and $s_{22}$ as 0, rather than 'unknown'.

Based on the two strategies, we estimate $ep(u, s_e)$ as follows:

$$ep(u, s_e) = \begin{cases} 0 & if \frac{|wIntv(u,s_e)|}{|wAbleIntv(u,s_e)|} < \alpha, \\ \frac{|wIntv(u,s_e)|}{|wAbleIntv(u,s_e)|} & otherwise. \end{cases} \quad (3)$$

However, we notice that, if $wAbleIntv(u, s_e)$ is very small, it can be regarded as $u$'s positive preference on $s_e$ even when $u$ watched only a very small part in the corresponding $wAbleIntv(u, s_e)$. For example, as (iv) in Fig. 3-(a), $u_1$'s preference for the episode $s_{31}$ can be estimated (unreasonably)

to be a positive preference even if $u_1$ starts watching at about the end of $s_{31}$. To avoid such mis-understanding, we propose a strategy, PE-Strategy 3, that does not consider these episodes as *wAbleEpisodes*.

**PE-Strategy 3.** Using PE-Strategy 3, we re-define $wAble\_eSet(u,s)$ for TV show $s$ of user $u$ as follows:

$$wAble\_eSet(u,s) = \{s_e | s_e \in wAble\_eSet(u,s), \frac{|wAbleIntv(u,s_e)|}{|bIntv(s_e)|} \geq \beta\}. \quad (4)$$

In other words, $wAble\_eSet(u,s)$ is a set of *wAbleEpisodes* whose ratio of $|wAbleIntv(u,s_e)|$ among $|bIntv(s_e)|$ is greater than a threshold $\beta$.

Finally, for idea #2, we aggregate the preferences $ep(u,s_e)$ for *wAbleEpisodes* $s_e$ included in $wAble\_eSet(u,s)$ to predict user $u$'s preference for TV show $s$, $e_{u,s}$, as follows:

$$e_{u,s} = \frac{1}{|wAble\_eSet(u,s)|} \sum_{s_e \in wAble\_eSet(u,s)} ep(u,s_e). \quad (5)$$

### C. Preference Prediction (PP)

In a TV show domain, the TV shows that have already been watched and those that have not been watched are both considered as candidates for recommendation. We have already estimated the preference of a *watchable* TV show that a user was able to watch. Furthermore, the non-estimated preferences on *non-watchable* TV shows should also be predicted based on the previously estimated preferences on other TV shows (challenge #2).

To address challenge #2, we propose the following two ideas:

- **Idea #3: Consider all preferences on TV shows together by exploiting CF techniques**: We need to predict a user's preference on non-watchable TV shows by considering the user's and other users' preferences on all TV shows.
- **Idea #4: Exploit the confidence of estimated preferences on TV shows**: CF techniques should be performed with *higher weights* in the prediction error for the preferences estimated with *higher confidence*.

We first clarify the importance of idea #4. A user has *different numbers of wAbleEpisodes* for TV shows. A preference of a user on a TV show becomes more reliable as it is estimated from more feedback by the user. To use this intuition, we set the confidence on the estimated preference to be proportional to the amount of feedback (*i.e.*, the number of *wAbleEpisodes*). Towards idea #4, we formally define a notion of *confidence* $c_{u,s}$ of estimated preference $e_{u,s}$ for TV show $s$ of user $u$ is calculated as follows:

$$c_{u,s} = |wAble\_eSet(u,s)|, \quad (6)$$

indicating the number of $u$'s *wAbleEpisodes* (*i.e.*, rather than $u$'s watched episodes as in PM and PNM) of TV show $s$.

After the confidence for each estimated preference is calculated, we should predict preferences by considering all preferences (*i.e.*, idea #3) on TV shows based on the confidence. To do this, we can apply any CF models that utilize the confidence. In this paper, we employ wALS [16] and NeuMF [11]. We show how to predict preferences based on wALS and NeuMF.

wALS and NeuMF optimize the following loss function:

$$\mathcal{L}_{sqr} = \sum_{u,s} c_{u,s}(e_{u,s} - \hat{e}_{u,s})^2. \quad (7)$$

We approximate *E* by performing wALS or NeuMF based on *E* and *C*. The models decompose *E* into two low-rank matrices *X* and *Y* while optimizing a loss function $\mathcal{L}_{sqr}$. The matrices *X* and *Y* represent the features of users and TV shows as latent factors, respectively. Finally, each model predicts preference $\hat{e}_{u,s}$ by their own formulation.

**PP-Model 1 (wALS).** wALS simply computes preference $\hat{e}_{u,s}$ of user $u$ on TV show $s$ by an inner product of $X_u$ and $Y_s^T$. In order to factorize *E*, wALS first assigns random values to elements in *Y*, and updates elements in *X* as in Eq. (8) by optimizing the loss function. $\forall 1 \leq u \leq m$:

$$X_{u(\cdot)} = E_{u(\cdot)}\widetilde{C}_{u(\cdot)}Y\{Y^T\widetilde{C}_{u(\cdot)}Y + \lambda(\sum_s c_{u,s})I\}^{-1}, \quad (8)$$

where $\widetilde{C}_{u(\cdot)}$ is a diagonal matrix with the elements of $C_{u(\cdot)}$ on the diagonal, and *I* is an identity matrix. After that, wALS updates elements in *Y* while fixing *X* as in Eq. (9). $\forall 1 \leq s \leq n$:

$$Y_{s(\cdot)} = E^T_{(\cdot)s}\widetilde{C}_{(\cdot)s}X\{X^T\widetilde{C}_{(\cdot)s}X + \lambda(\sum_u c_{u,s})I\}^{-1}. \quad (9)$$

We optimize the loss function by repeating Eq. (8) and Eq. (9) until *X* and *Y* converge to a *local optimum*.

**PP-Model 2 (NeuMF).** NeuMF computes preference $\hat{e}_{u,s}$ of user $u$ on TV show $s$ as follows:

$$\begin{aligned} \phi^{GMF} &= x_u^G \odot y_s^G, \\ \phi^{MLP} &= a_L(W_L^T(a_{L-1}(...a_2(W_2^T \begin{bmatrix} x_u^M \\ y_s^M \end{bmatrix} + b_2)...)) + b_L, \\ \hat{e}_{u,s} &= \sigma(h^T \begin{bmatrix} \phi^{GMF} \\ \phi^{MLP} \end{bmatrix}), \end{aligned} \quad (10)$$

where $\odot$ denotes the element-wise vector product; $x_u^G$ and $x_u^M$ denote the user embedding for *generalized matrix factorization* (GMF) and *multi-layer perceptron* (MLP) parts, respectively; similarly, $y_s^G$ and $y_s^M$ do for TV show embeddings; $W_i$, $b_i$, $a_i$, and $h$ denote the weight matrix, bias vector, activation function for the $i$-th layer's perceptron, and edge weights of the output layer, respectively. Refer to [11] for details on how to solve the optimization problem $\mathcal{L}_{sqr}$.

### D. Recommendation

In a TV show domain, a set of TV shows being broadcast vary depending on the time of recommendation. Therefore, it is necessary to consider not only the predicted preferences of a target user for the TV shows being broadcasted at the *recommendation time* but also the *time factors* of the TV shows (challenge #3).

To address challenge #3, we exploit the following idea:

- **Idea #5: Consider staying and remaining times**[3]

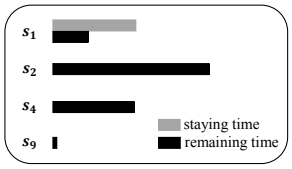



Fig. 4. Time factors of four TV shows considered in recommendation at $r_i$.

Fig. 4 shows an example for the *staying* and *remaining* times of four currently broadcasting TV shows ($s_1$, $s_2$, $s_4$, and $s_9$) for a target user $u_1$ at recommendation time $r_i$. If a user $u$ (*e.g.*, $u_1$ in Fig. 4) has been watching an episode $s_e$ of a TV show $s$ (*e.g.*, $s_1$ in Fig. 4) for a long time at recommendation time, it is reasonable to judge that episode $s_e$ is preferred by her even if her predicted preference on show $s$ appears not that high. In addition, if the remaining time of other episode $s_e$ (*e.g.*, $s_9$ in Fig. 4) is small at recommendation time $r_i$, it is reasonable to judge user $u$ (*e.g.*, $u_1$ in Fig. 4) may not want to watch $s_e$ even if the predicted preference on $s$ appears high. Therefore, to consider idea #5, we re-adjust the score of a predicted preference by considering staying and remaining times of *wAbleEpisodes* at *recommendation time point* $r_i$. We calculate user $u$'s re-adjusted (predicted) preference to TV show $s$ at time $r_i$, $p_{u,s,r_i}$, as follows:

$$p_{u,s,r_i} = \hat{e}_{u,s} \times prob(watch|st(u, r_i, s_e), rt(r_i, s_e)), \qquad (11)$$

Here, $prob(watch|st(u, r_i, s_e), rt(r_i, s_e))$ denotes a probability of user $u$'s $wAbleEpisode$ $s_e$ given with her staying time $st(u, r_i, s_e)$ and remaining time $rt(r_i, s_e)$. Note that $st(u, r_i, s_e) = 0$ for all other broadcasting episodes except for the currently watching episode. The probability increases as her staying time on $s_e$ increases and decreases as the remaining time of $s_e$ decreases. Refer to [15] for more details on how to calculate the $prob(watch|st(u, r_i, s_e), rt(r_i, s_e))$.

Finally, our framework selects and recommends top-$N$ TV shows to each user according to the re-adjusted (predicted) preferences finally computed.

## IV. Evaluation

In this section, we evaluate our framework via extensive experiments. We designed the experiments, aiming at answering the following key research questions:

- RQ1. Does the proposed PE *accurately estimate* the users' preferences on *watchable* TV shows by feedback?
- RQ2. Does the proposed PP *accurately predict* the users' preferences on *non-watchable* TV shows?
- RQ3. How much *accurate* is our framework, compared with other state-of-the-art methods?
- RQ4. How is the accuracy of our framework with *different values of two parameters*, $\alpha$ and $\beta$?
- RQ5. How much *scalable* is the training of our framework?

[3]This nice idea used in our paper is borrowed from in [15].

TABLE II
Descriptive statistics of the dataset

| | All | Training | Test |
|---|---|---|---|
| **# users** | 2,014 | 2,014 | 1,647 |
| **# TV shows** | 5,211 | 5,211 | 2,014 |
| **# episodes** | 115,796 | 105,428 | 10,368 |
| **# channels** | 67 | 67 | 67 |
| **Avg. # episodes per TV show** | 22.2 | 20.2 | 4.9 |
| **# total log records** | 1,685,820 | 1,541,885 | 143,935 |
| **Avg. # log records per user** | 837.1 | 765.5 | 87.3 |

### A. Experimental Setup

**Dataset:** We used a real-world dataset containing a nine-week *TV watch logs* from users [15]. A single watch log includes a user ID, the channel number that the user watched, and the watching time of the user for the channel. In addition, we used the *Electronic Program Guide* (EPG) dataset [15].[4] The EPG dataset includes channels, TV shows for each channel, episodes for each TV show, and a $bIntv$ for each episode. By combining the watch-log dataset with the EPG dataset, we could understand how long each user has watched each TV show.

For evaluation, we used the most recent one-week of the logs for the test set and the remaining eight weeks for the training set. In [14], users' most preferences on the TV show domain depend on the previous 1-2 month records, which could justify the use of the nine-week dataset in our study.[5]

Table II shows the statistics of the dataset. Here, we present interesting characteristics of the dataset. We first constructed a user-episode matrix that indicates whether each user watched each episode. We then calculated the sparsity of the matrix, which is 99.22%. In traditional recommendation domains (*e.g.*, movie, shopping), the sparsity of the user-item matrix has been calculated based on whether each user used or purchased each item. However, in the TV show domain, it is necessary to consider how long each user watched each episode because TV watching is a continuous action. Therefore, we analyzed *how long a user watched an episode on average* as follows: (sum of all users' $wIntv$ for episodes) / (sum of $bIntv$ of all episodes $\times$ the number of users). Since the value is only 0.17%, its sparsity is 99.83%. These statistics show that our dataset is quite sparse, making it difficult to accurately capture each user's preferences on TV shows. As a result, it is challenging to provide recommendations to users based on this dataset.

Note that finding publicly available datasets in a TV show domain is very challenging. For this reason, most papers, related to TV show recommendation, used only a *single* dataset. For example, [13] and [15], other authors' papers, used the same dataset we used; [10] and [14] used their own single dataset, but neither one is publicly available.

**Competing Methods:** We compare our framework with other state-of-the-art methods to verify its effectiveness. As a baseline method, we employ a non-personalized method: Popular

[4]The EPG dataset was crawled from Daum (www.daum.net), one of the most popular search engines in Korea.

[5]We found that the experimental results on different split ratios (*e.g.*, two/seven weeks for test/training sets) for our dataset are similar to those reported in this paper. This tendency is consistent with the results in [14].

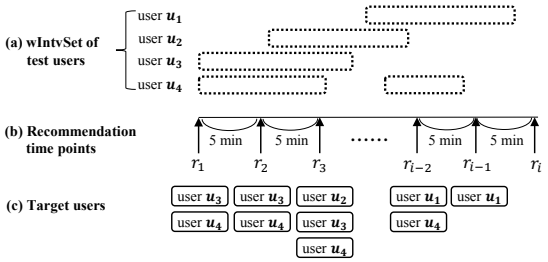



Fig. 5. Evaluation protocols.

which recommends the longest-watched TV shows by users among the currently broadcasting TV shows. In addition, we compare ours with the following state-of-the-art methods: PM [10], PNM [14], ShowTime [15], and RecTime [13]. We used wALS (used in PM), PMF (used in PNM), and PARAFAC (used in RecTime) implemented in the open-source Graphchi [22] and TensorLy [23]. We employed the best performing parameter settings in competing methods, obtained via extensive experiments.

Through extensive experiments, we have found the parameter settings with the best accuracy of our framework. First, the parameter values used in our PE are: $\alpha$ = 0.1; $\beta$ = 0.5. For two variants (*i.e.*, Ours_wALS and Ours_NeuMF) of our PP, the parameter values used in each variant are as follows: Ours_wALS: # iterations, # factors, and a regularization term are 200, 10, and 0.01, respectively; Ours_NeuMF: # iterations, # factors, batch size, and layers are 20, 8, 64, and $[64, 32, 16, 8]$, respectively.

**Evaluation Protocols:** We evaluate the accuracy of the recommendation for the test sets with two cases of TV shows – watchable and non-watchable TV shows. Our test sets represent the sets of TV shows (*i.e.*, correct answers) that each user actually watched *during the test period*. Based on each user's correct answers, we have two test sets as follows:

- *W-test set*: Candidates for recommendation are *all* TV shows broadcasted at each recommendation time. Correct answers are the TV shows that each user was able to watch during the training period and was also actually watched during the test period.
- *NW-test set*: Candidates for recommendation are *all* TV shows broadcasted at each recommendation time. Correct answers are the TV shows that each user was *not* able to watch during the training period but was actually watched during the test period.

Fig. 5 illustrates how our evaluation of TV show recommendation is conducted. Fig. 5-(a) shows the $wIntvSet$ of users in the test set. Fig. 5-(b) shows *recommendation time points* on a global timeline. Recommendations occur in every five minutes. Fig. 5-(c) illustrates the users (*i.e.*, target users for recommendation) who are watching TV shows at each $r_i$. For $r_i$, the target users can be changed (*e.g.*, $u_3$ and $u_4$ at $r_1$). For evaluation, we consider an episode of a TV show that a target user is watching at $r_i$ as a correct answer of recommendation at $r_i$. However, if the target user watched the episode for an interval below a certain threshold $x$ and the ratio of the user's $wAbleIntv$ to the episode's $bIntv$ is less than a certain threshold $y$, we exclude the episode from a set of correct answers (*i.e.*, ground truth) in our evaluation. We set the threshold $x$ as 10% of the $wAbleIntv$ and $y$ as 50% of the $bIntv$.[6] When target users and correct answers are determined, we recommend top-$N$ TV shows to the target users by employing each recommendation method.

TABLE III
ESTIMATION ACCURACY WITH COMBINATIONS OF OUR STRATEGIES

| Metrics | PE(S1) | PE(S1+S2) | PE(S1+S3) | PE(Ω) |
|---|---|---|---|---|
| **Recall@1** | 0.409 | 0.408 | 0.449 | **0.449** |
| **Recall@5** | 0.838 | 0.838 | 0.854 | **0.854** |
| **Precision@1** | 0.409 | 0.408 | 0.449 | **0.449** |
| **Precision@5** | 0.167 | 0.167 | 0.170 | **0.170** |
| **nDCG@1** | 0.409 | 0.408 | 0.449 | **0.449** |
| **nDCG@5** | 0.638 | 0.638 | 0.665 | **0.665** |
| **MRR** | 0.591 | 0.591 | 0.620 | **0.620** |

We measure the accuracy of recommendations by using the four most popularly used metrics: *precision*, *recall*, *nDCG* [24], and *mean reciprocal rank* (*MRR*) [25]. For each metric, we take the average of the values computed over all recommendations as its final accuracy. For all experiments, we conducted *t*-tests with a 95% confidence level to measure accuracy differences. We found that the results in all measures and methods consistently show that the $p$ values are below 0.05, indicating that differences are *statistically significant*. All experiments were conducted in Windows 7 running on Intel Core i7 processor (4.60 GHZ) with 64GB RAM.

### B. Results

**RQ1:** We conduct two experiments – (RQ1-1) the *effectiveness of our PEs using each strategy* and (RQ1-2) the *accuracy comparison* with other methods' PEs. For RQ1-1, we made the following four variants of our PE: PE(S1) using strategy 1 only, PE(S1+S2) using strategies 1 & 2, PE(S1+S3) using strategies 1 & 3, and PE(Ω) using all strategies 1 & 2 & 3. For RQ1, we only measure the accuracy for *W-test set* because PE can be used only for the *watchable TV shows*.

For RQ1-1, Table III shows the accuracies of the four variants of our PE. This result shows that PE(S1+S3) has better accuracy than PE(S1). This result shows that, in estimating each user's preference on each episode, it is effective to improve the accuracy of the PE by considering the ratio of the user's $wIntv$ to the user's $wAbleIntv$ for the episode (*i.e.*, S1) and the ratio of the user's $wAbleIntv$ to the episode's $bIntv$ (*i.e.*, S3) together.

However, there is no difference in accuracies between PE(S1) and PE(S1+S2) and those between PE(S1+S3) and PE(Ω). This means no improvement in accuracy due to S2 in PE. We examine this phenomenon in detail. Suppose that, for some episodes of a TV show $s$, a user $u$ watches the episodes for more than a certain amount of time during $u$'s $wAbleIntv$ for the episodes. In this case, we will determine that $u$ has a positive preference for the episodes by S1 at PE. If

[6]Note that $\alpha$ and $\beta$ indicate parameters for estimating users' preferences, while $x$ and $y$ indicate the parameters for determining correct answers.

TABLE IV
ESTIMATION ACCURACY OF OUR AND EXISTING METHODS' PES

| Metrics | PE | PM | PNM | ShowTime | RecTime | Popular |
|---|---|---|---|---|---|---|
| **Recall@1** | **0.449** | 0.076 | 0.232 | 0.305 | 0.330 | 0.047 |
| **Recall@5** | **0.854** | 0.368 | 0.723 | 0.740 | 0.755 | 0.363 |
| **Precision@1** | **0.449** | 0.076 | 0.232 | 0.305 | 0.330 | 0.047 |
| **Precision@5** | **0.170** | 0.073 | 0.144 | 0.148 | 0.151 | 0.072 |
| **nDCG@1** | **0.449** | 0.076 | 0.232 | 0.305 | 0.330 | 0.047 |
| **nDCG@5** | **0.665** | 0.219 | 0.486 | 0.530 | 0.550 | 0.204 |
| **MRR** | **0.620** | 0.233 | 0.441 | 0.492 | 0.512 | 0.214 |

$u$ skips or has not watched other remaining episodes of $s$, even though their $wAbleIntv$ exists, then S2 will determine $u$ to have negative preferences (*i.e.*, zero-value) for these episodes. Since, in PE, $u$'s preference for $s$ is computed by aggregating her preferences for all episodes of $s$, we can reflect both positive and negative preferences of $u$ for $s$ due to S2. As a result, we expect that S2 could contribute to improving the accuracy of recommendations in this case.

However, if S1 determines that $u$ has positive preferences for some episodes of $s$, S2 would *rarely* determine that $u$ has negative preferences for other remaining episodes of $s$. This is because a user tends to have a *consistent preference*, either positive or negative, for the episodes of the same TV show. Thus, if S1 has already determined that $u$ has a positive preference for episodes of $s$, then S2 has little effect on recommendation accuracy because an overall preference of $s$ does not much change even after applying S2.

However, if S1 does not determine that $u$ has a positive preference for episodes of $s$, $u$'s preference for $s$ will be *unknown* before S2 is applied. After applying S2, $u$'s preference can be considered to be a negative preference (*i.e.*, zero-value) for some episodes of $s$; thus, the preference for $s$ will change from *unknown to negative* preference. However, since the preference for $s$ is still not positive, $s$ will not be a candidate of top-$N$ recommendation in PE, both before and after applying S2. Therefore, S2 has no effect on the recommendation accuracy improvement in PE as shown in Table III. We note, however, S2 shows a great effect on improving the accuracy of recommendation in PP. The results will be discussed later in detail in RQ2.

For RQ1-2, Table IV shows the accuracy of PE($\Omega$) (we simply call it PE henceforth), existing methods' PEs, and the baseline. We can see that the accuracy of our PE is the greatest. This demonstrates the effectiveness of estimating the preferences based on *wAbleIntv* as well as the limitation of the existing methods based on one-class setting (PM), the sum of watched hours (ShowTime), and $bIntv$ (PNM and RecTime).

Furthermore, we examine in detail the effectiveness of our $wAbleIntv$-based PE by pointing out the limitations of the PEs used in existing methods. To do this, we randomly sample three cases that are successfully matched by our PE as top-1 recommendation but not by the existing methods. We then compute the following four scores for the top-1 TV show (*i.e.*, correct answer) $x$ recommended by our PE: (1) $wIntv$-only score (used in ShowTime) represents the accumulated score of a user's all $wIntv$ for the episodes of $x$; (2) Avg. $bIntv$-based score (used in PNM) represents a user's score on a TV show $x$ by averaging the ratios of $wIntv$ to $bIntv$ for its episodes; (3) Acc. $bIntv$-based score (used in RecTime) represents a user's score on a TV show $x$ by accumulating the ratios of $wIntv$ to $bIntv$ for its episodes; (4) $wAbleIntv$-based score (used in our PE) represents a user's score on a TV show $x$ by considering the ratios of $wIntv$ to $wAbleIntv$ for its episodes.

TABLE V
SCORES FOR TOP-1 TV SHOWS RECOMMENDED BY OUR PE

| Cases | *wIntv*-only score (by ShowTime) | Avg. *bIntv* -based score (by PNM) | Acc. *bIntv* -based score (by RecTime) | *wAbleIntv* -based score (by our PE) |
|---|---|---|---|---|
| **Case 1** | 0.08 (1.00) | 0.26 (0.77) | 0.06 (1.00) | 1.00 |
| **Case 2** | 0.21 (1.00) | 0.19 (0.66) | 0.17 (1.00) | 1.00 |
| **Case 3** | 0.16 (1.00) | 0.21 (0.62) | 0.29 (1.00) | 0.95 |

Table V shows the results.[7] In Case 1, the following values – 0.08, 0.26, 0.06, and 1.00 – outside the parentheses represent $wIntv$-only, avg. $bIntv$-based, acc. $bIntv$-based, and $wAbleIntv$-based scores for top-1 TV show (*i.e.*, correct answer) $x$ recommended by our PE, respectively. The values – 1.00, 0.77, and 1.00 – in the parentheses represent the $wlntv$-only score by ShowTime, the avg. $blntv$-based score by PNM, and the acc. $blntv$-based score by RecTime for the top-1 TV show (not $x$) recommended by ShowTime, PNM, and RecTime, respectively. That is, these three values are the highest $wIntv$-only, avg. $bIntv$-based, and acc. $bIntv$-based scores that TV shows can have in Case 1.

Since the top-1 TV show $x$ recommended by our PE is a correct answer in this experiment, the $wAbleIntv$-based score of $x$ has the highest score of 1. On the other hand, the $wIntv$-only score of $x$ is 0.08, which is significantly lower than the highest $wIntv$-only score of 1 in Case 1. This is because the $wIntv$-only score considers the sum of $wIntv$ for $x$ without considering $u$'s $wAbleIntv$ for $x$ nor the number of times $u$ can watch for $x$. As a result, $u$'s preference could be biased toward the TV show that is frequently or long broadcasted. The avg. and acc. $bIntv$-based scores for $x$ are 0.26 and 0.06, which are significantly lower than the highest avg. and acc. $bIntv$-based scores of 0.77 and 1.00 in Case 1. This is because the $bIntv$-based scores do not exclude the time that $u$ is unable to watch during the $bIntv$ of $x$'s episodes, but regards the time as the $u$' negative feedback. As a result, $u$'s preference of $x$ could be incorrectly predicted as a low preference. Similar tendency is observed in other cases (*i.e.*, Cases 2 and 3 in Table V) as well.

For this reason, ShowTime, PNM, and RecTime miss the correct answers from their recommendation because they did not consider the notion of $wAbleIntv$ in estimating user preferences. In other words, this result shows that our $wAbleIntv$-

[7]Note that $wIntv$-only and acc. $bIntv$-based scores are cumulative values without any normalization, so their scales are quite different from those of the avg. $bIntv$-based and $wAbleIntv$-based scores that have a maximum score of 1 and a minimum score of 0. Therefore, we performed min-max normalization for $wIntv$-only and acc. $bIntv$-based scores in each case to enable relative comparison with the other two scores. For each case, we show $wIntv$-only / acc. $bIntv$-based scores for $x$ and the maximum scores before normalization as follows: Case 1 - 59.60 (752.20) / 1.06 (17.16); Case 2 - 28.40 (133.50) / 0.56 (3.29); Case 3 - 74.30 (460.90) / 1.91 (6.42).

TABLE VI
PREDICTION ACCURACY WITH COMBINATIONS OF OUR STRATEGIES

| Metrics | PP(S1) | PP(S1+S2) | PP(S1+S3) | PP(Ω) |
|---|---|---|---|---|
| **Recall@1** | 0.021 | 0.134 | 0.030 | **0.142** |
| **Recall@5** | 0.157 | **0.418** | 0.181 | 0.417 |
| **Precision@1** | 0.021 | 0.134 | 0.030 | **0.142** |
| **Precision@5** | 0.031 | **0.083** | 0.036 | **0.083** |
| **nDCG@1** | 0.021 | 0.134 | 0.030 | **0.142** |
| **nDCG@5** | 0.087 | 0.278 | 0.104 | **0.282** |
| **MRR** | 0.123 | 0.274 | 0.136 | **0.278** |

TABLE VII
PREDICTION ACCURACY BASED ON OUR AND EXISTING METHODS' PES

| Metrics | PP | PM | PNM | ShowTime | RecTime | Popular |
|---|---|---|---|---|---|---|
| **Recall@1** | **0.142** | 0.042 | 0.040 | 0.018 | 0.020 | 0.005 |
| **Recall@5** | **0.417** | 0.151 | 0.203 | 0.138 | 0.141 | 0.096 |
| **Precision@1** | **0.142** | 0.042 | 0.040 | 0.018 | 0.020 | 0.005 |
| **Precision@5** | **0.083** | 0.030 | 0.040 | 0.027 | 0.028 | 0.019 |
| **nDCG@1** | **0.142** | 0.042 | 0.040 | 0.018 | 0.020 | 0.005 |
| **nDCG@5** | **0.282** | 0.096 | 0.120 | 0.076 | 0.078 | 0.049 |
| **MRR** | **0.278** | 0.132 | 0.149 | 0.113 | 0.115 | 0.094 |

based scores correctly represent user's preferences compared to the $wIntv$-only, avg. $bIntv$-based, and acc. $bIntv$-based scores.

**RQ2:** We conduct two experiments – (RQ2-1) the *effectiveness of our PPs based on each PE's strategy* and (RQ2-2) the *accuracy comparison* of our PP based on our PE with our PP based on PEs of other methods. Similar to RQ1-1, for RQ2-1, we made the following four variants of our PP: PP(S1) based on PE(S1), PP(S1+S2) based on PE(S1+S2), PP(S1+S3) based on PE(S1+S3), and PP(Ω) based on PE(Ω). For a fair comparison in RQ2-2, we commonly used wALS as a prediction model that analyzes the preferences estimated by each PE, because optimizing wALS is much less complex than NeuMF. For the confidence of the preferences estimated by each PE, we used the values proposed by each method (refer to Section II). Note that, for RQ2, we only measure the accuracy for *NW-test set*, because PP can be used only for the *non-watchable TV shows*.

For RQ2-1, Table VI shows the accuracies for the four variants of our PP. The accuracies increase whenever each strategy is added. Here, we note that, *unlike* PE, PP achieves a significant accuracy improvement by S2. In addition, the accuracy improvement (220% on nDCG@5) between PP(S1) and PP(S1+S2) by S2 is much larger than that (20% on nDCG@5) between PP(S1) and PP(S1+S3) by S3.

Suppose that, for the episodes of a TV show $s$, a user $u$ did not watch any episodes of $s$ for a certain amount of time during $u$'s $wAbleIntv$ for them. In this case, we determine that $u$ does not have a positive preference for $s$ by S1. Therefore, before applying S2, $u$'s preference for $s$ is unknown. Such a TV show $s$ is determined to have new negative preferences (*i.e.*, by adding zero-value) by S2; thus, thanks to S2, (1) the sparse data in a preference matrix becomes denser, and (2) CF can utilize positive and negative preferences of users together. This allows us to more accurately capture latent similarities between users or between items in the process of CF execution.

For RQ2-2, Table VII shows that our PP(Ω) (we simply call it PP hereafter) universally outperforms all existing methods

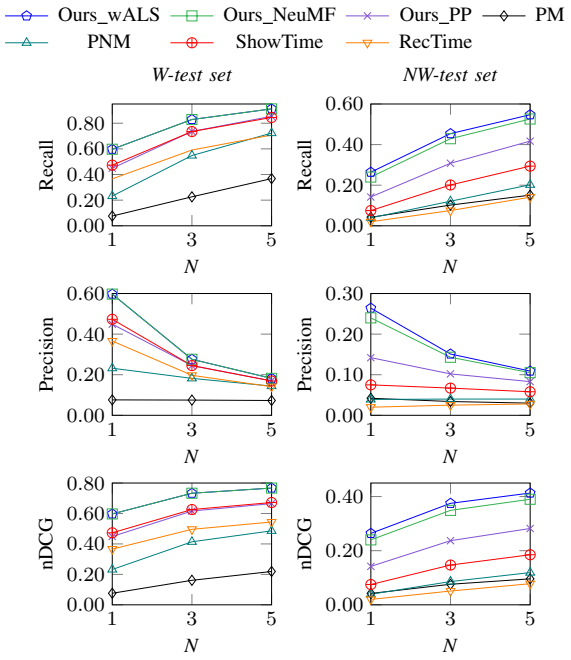



Fig. 6. Accuracy of the state-of-the-art methods and our frameworks.

in terms of accuracy with all metrics. Specifically, our PP dramatically improves the result of nDCG@5 obtained from PM, PNM, ShowTime, RecTime, and Popular by 193%, 135%, 271%, 261%, and 475%, respectively. This result shows that predicting the preferences for non-watchable TV shows using the notion of $wAbleIntv$ is quite effective compared to the competing methods (*i.e.*, not considering the notion of $wAbleIntv$). In other words, this indicates that the quality of the preferences (*i.e.*, input for PP) estimated by our $wAbleIntv$-based PE is good.

**RQ3:** Our recommendation framework that combines the PE, PP, and recommendation methods is tested by three experiments. For RQ3-1, we compare the accuracy of our framework with that of a variant only using PE and PP without using *time factors*, to verify the *effectiveness of the time factors* used in the recommendation step. For RQ3-2, we apply *different CF models*, wALS and NeuMF, to our proposed framework.

For RQ3-3, we conduct *comparative experiments* on all test sets to show whether our framework yields better accuracy than the state-of-the-art methods of PM, PNM, ShowTime, and RecTime. Note that, unlike RQ2-2, PM and PNM uses its own prediction model for *NW-test set*. For ShowTime and RecTime, first ShowTime does not perform prediction on un-watched TV shows. RecTime was experimented only for *genre* recommendation (rather than TV show) in [13] due to the sparsity of the tensor. Although we evaluated RecTime's accuracy in TV show recommendations, the result for *NW-test set* was *in-suitably low* to be included in Fig. 6. Thus, to measure the accuracy of ShowTime and RecTime in *NW-test set*, we apply our PP model after performing ShowTime's and RecTime's PEs.

Fig. 6 shows the accuracies of top-$N$ recommendations

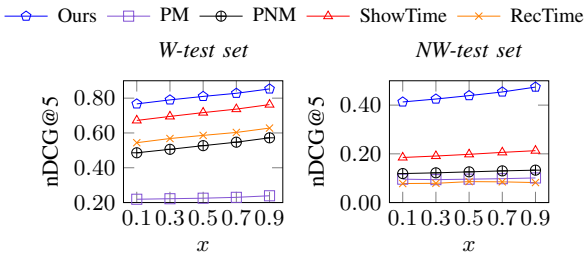



Fig. 7. Accuracy changes with varying $x$ ($y$=0.5).

TABLE VIII
CHANGES IN THE RATIO OF CORRECT ANSWERS WITH INCREASING X

| *x* | 0.1 | 0.3 | 0.5 | 0.7 | 0.9 |
|---|---|---|---|---|---|
| **W-test set** | 79.51% | 73.68% | 66.76% | 58.52% | 45.49% |
| **NW-test set** | 52.12% | 48.99% | 45.46% | 40.78% | 33.86% |

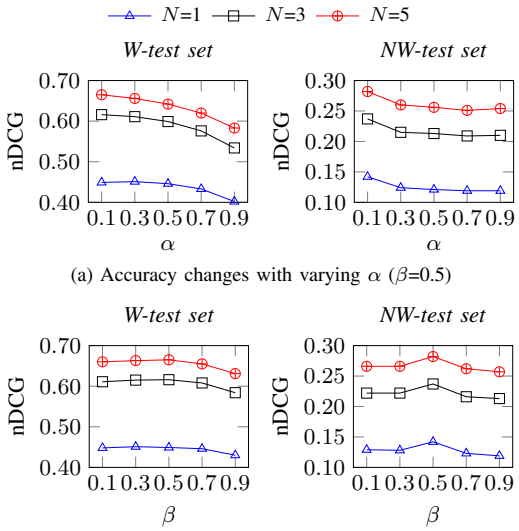
applied to three types of test sets by our PP without using time factors (denotes as Ours_PP[8]), two variants of the proposed framework (denotes as Ours_wALS and Ours_NeuMF) with using time factors, and the state-of-the-art methods. Due to space limitations, we omit the results for MRR, which are consistent with those in Fig. 6. The $x$-axis represents $N$ of top-$N$, and the $y$-axis does the accuracy.

For RQ3-1, the time factors [15] help improve the accuracy significantly. Ours_wALS improves the accuracy of nDCG@5 by 15% (*W-test set*) and 46% (*NW-test set*), compared to Ours_PP. For RQ3-2, Ours_wALS and Ours_NeuMF show comparable accuracy on *W-test set*. On *NW-test set*, however, Ours_wALS shows slightly better accuracy (5.8%) on nDCG@5. For RQ3-3, the proposed framework consistently and universally provides the highest accuracy in all test sets and with all metrics. In particular, for *NW-test set*, our framework improves the accuracy of nDCG@5 *dramatically* up to 330% (PM), 247% (PNM), 123% (ShowTime), and 429% (RecTime).

As noted in evaluation protocols in Section IV-A, we set the threshold $x$ of $wAbleIntv$ to 0.1. For RQ3-4, we analyze the change in the number of correct answers when $x$ changes. In addition, we show the accuracy changes of our framework and the competing methods. Table VIII shows the ratio of correct answers (as $x$ increases) in each of *W-test* and *NW-test sets* among the original correct answers[9] obtained when $x$ is 0. This results indicate that the number of correct answers decreases as $x$ increases. When $x$ is 0.1 in each of *W-test* and *NW-test sets*, it includes 79.51% and 52.12% of the original correct answers, but only 45.49% and 33.86% when $x$ is 0.9.

Fig. 7 shows accuracies of Ours_wALS and four competing methods by varying $x$ (=0.1, 0.3, 0.5, 0.7, 0.9) for *W-test* and *NW-test sets*. Due to space limitations, we omit the results for other metrics and other top-$N$s, which are consistent with those in Fig. 7. The $x$-axis represents the values of $x$ and the $y$-axis does the accuracy. This result shows that our framework consistently and universally outperforms the competing methods regardless of $x$.

[8]This uses wALS as a prediction model as in RQ2-2
[9]281,409 in *W-test set* and 98,590 in *NW-test set*.




(a) Accuracy changes with varying $\alpha$ ($\beta$=0.5)

(b) Accuracy changes with varying $\beta$ ($\alpha$=0.1)

Fig. 8. Accuracy changes with varying $\alpha$ and $\beta$.

TABLE IX
TRAINING TIME WITH INCREASING NUMBER OF USERS

| **# of users** | 2,000 | 4,000 | 6,000 | 8,000 | 10,000 |
|---|---|---|---|---|---|
| **PE (sec.)** | 21,197 | 45,261 | 66,554 | 93,018 | 118,247 |
| **PP (sec.)** | 1,620 | 3,235 | 4,851 | 6,482 | 8,149 |

**RQ4:** We show the accuracy changes by varying both $\alpha$ and $\beta$, which were obtained from the process to find the best parameter settings. Note that it is difficult to report all the accuracies on every possible pair of two parameters. Here, we only report the changes of Ours_wALS by fixing one parameter as the best value and changing the value (=0.1, 0.3, 0.5, 0.7, 0.9) of the other parameter. Fig. 8-(a) and (b) show the accuracy changes of Ours_wALS with varying $\alpha$ and $\beta$ for *W-test set* and *NW-test set*, respectively. Due to space limitations, we omit the results for other metrics, which are consistent with those in Fig. 8. The $x$-axis represents the values of $\alpha$ and $\beta$, and the $y$-axis does the accuracy of Ours_wALS with different top-$N$s.

Fig. 8-(a) shows that the accuracy of Ours_wALS with $\alpha$ of 0.1, which is the value proposed in this paper, is always highest for all top-$N$, and decreases with increasing $\alpha$ in both test sets. This result shows that it is best to judge an episode to be uninteresting when the ratio of a user's $wIntv$ to its $wAbleIntv$ is less than 10%. Fig. 8-(b) shows different sensitivity of $\beta$ depending on test sets. For *W-test set*, there is almost no change in the accuracy of Ours_wALS when $\beta$ is 0.1-0.5, and decreases as $\beta$ increases from 0.5. On the other hand, for *NW-test set*, the result shows that the accuracy is always the highest for all top-$N$ when $\beta$ is 0.5. This result means that if a user can watch more than 50% of a episode's $bIntv$, we should include it in her *wAbleEpisode*, because it is found to be enough time to represent the preference for the episode.

**RQ5:** We conduct experiments on performance and scalability of our framework. For RQ5-1, we confirmed that the average

time required for our framework to provide recommendations to a user is 0.013s; This result shows that our framework is efficient in real time recommendation. For RQ5-2, we show the changes of training time as the number of users increases.[10] Table IX shows the execution times of the two training steps – PE and PP – of our framework as the number of users varies. This result shows that the execution time for our PE and PP *linearly increases* with the increased number of users; That is, the training of our framework is *scalable* with regard to the number of users.

## V. Conclusions

In this paper, we addressed the problem of recommending TV shows being broadcasted currently to users who are watching TV. We first carefully analyzed the dataset used in the TV show domain and then identified its unique characteristics that are not exposed in other domains. Based on these characteristics, we defined *a user's watchable interval for an episode of a TV show*, which is a novel and important concept in TV show recommendation.

To apply our new concept "watchable interval" to the TV show recommendation, we identified the following three challenges: *preference estimation* (PE), *preference prediction* (PP), and *recommendation*. We proposed a novel framework based on CF with implicit feedback that addresses these challenges. Our framework is composed of a *wAbleIntv-based* PE method, and a *confidence-based* PP method, and a *time factor-considered* recommendation method. Using a real-world TV show dataset, we demonstrated that our framework shows dramatic improvements in recommendation accuracy over the state-of-the-art methods, outperforming PM, PNM, ShowTime, and RecTime up to 250% / 330%, 57% / 247%, 14% / 123%, and 40% / 429%, respectively in terms of nDCG@5 for *W-test set* / *NW-test set*.

Our notion of a watchable interval can be used not only in the TV show domain, but also in other domains such as Amazon's Twitch (www.twitch.tv), Facebook's fb.gg (www.facebook.com/gaming/) that transmit live commentaries of broadcasters *in real time*. Most of such platforms allow users to watch the shows after the show ended (*e.g.*, on-demand). However, given that many users are likely to enjoy watching live shows and communicating with broadcasters in real time, it is still important for users to recommend live shows that they may prefer to watch and interact with. In future work, we plan to expand our framework to consider watch logs for both live and pre-recorded shows.

## Acknowledgment

This work was supported by the NRF grant funded by the MSIT of Korea (No. NRF-2017R1A2B3004581). Also, we thank the Naver Corporation for their support including computing environment and data, which helped us greatly in performing this research successfully.

[10]Experimental results for more than 2,000 users were obtained by adding duplicate logs of the existing users.

## References

[1] B. Sarwar, G. Karypis, J. Konstan, and J. Riedl, "Item-based collaborative filtering recommendation algorithms," in *Proc. of the 10th Int'l Conf. on World Wide Web (WWW)*, pp. 285–295, 2001.

[2] P. Cremonesi, Y. Koren, and R. Turrin, "Performance of recommender algorithms on top-n recommendation tasks," in *Proc. of the 4th ACM Int'l Conf. on Recommender Systems (RecSys)*, pp. 39–46, 2010.

[3] 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. on Knowledge and Data Engineering (TKDE)*, vol. 17, no. 6, pp. 734–749, 2005.

[4] Y. Koren, R. Bell, and C. Volinsky, "Matrix factorization techniques for recommender systems," *Computer*, vol. 42, no. 8, pp. 30–37, 2009.

[5] H.-T. Cheng *et al.*, "Wide & deep learning for recommender systems," in *Proc. of the 1st Workshop on Deep Learning for Recommender Systems (DLRS)*, pp. 7–10, 2016.

[6] O. Celma, "Music recommendation," in *Music Recommendation and Discovery*, pp. 43–85, 2010.

[7] J. Liu, P. Dolan, and E. R. Pedersen, "Personalized news recommendation based on click behavior," in *Proc. of the 15th Int'l Conf. on Intelligent User Interfaces (IUI)*, pp. 31–40, 2010.

[8] W. Pan and L. Chen, "Gbpr: Group preference based bayesian personalized ranking for one-class collaborative filtering." in *Proc. of the 23rd Int'l Joint Conf. on Artificial Intelligence (IJCAI)*, pp. 2691–2697, 2013.

[9] R. Pan *et al.*, "One-class collaborative filtering," in *Proc. of the 8th IEEE Int'l Conf. on Data Mining (ICDM)*, pp. 502–511, 2008.

[10] Y. Hu, Y. Koren, and C. Volinsky, "Collaborative filtering for implicit feedback datasets," in *Proc. of the 8th IEEE Int'l Conf. on Data Mining (ICDM)*, pp. 263–272, 2008.

[11] X. He *et al.*, "Neural collaborative filtering," in *Proc. of the 26th Int'l Conf. on World Wide Web (WWW)*, pp. 173–182, 2017.

[12] X. He *et al.*, "Fast matrix factorization for online recommendation with implicit feedback," in *Proc. of the 39th ACM Int'l Conf. on Research and Development in Information Retrieval (SIGIR)*, pp. 549–558, 2016.

[13] Y. Park, J. Oh, and H. Yu, "Rectime: Real-time recommender system for online broadcasting," *Information Sciences*, vol. 409, pp. 1–16, 2017.

[14] Y. Xin and H. Steck, "Multi-value probabilistic matrix factorization for ip-tv recommendations," in *Proc. of the 15th ACM Int'l Conf. on Recommender Systems (RecSys)*, pp. 221–228, 2011.

[15] J. Oh *et al.*, "When to recommend: A new issue on TV show recommendation," *Information Sciences*, vol. 280, pp. 261–274, 2014.

[16] N. Srebro and T. Jaakkola, "Weighted low-rank approximations," in *Proc. of the 20th Int'l Conf. on Machine Learning (ICML)*, pp. 720–727, 2003.

[17] B. Smyth and P. Cotter, "Personalized electronic program guides for digital TV," *AI Magazine*, vol. 22, no. 2, pp. 89–98, 2001.

[18] P. Cotter and B. Smyth, "Personalisation technologies for the digital TV world," in *Proc. of the 14th European Conf. on Aftifical Intelligence (ECAI)*, pp. 701–705, 2000.

[19] W. S. Hwang, J. Parc, S. W. Kim, J. Lee, and D. Lee, ""Told you I didn't like it": Exploiting uninteresting items for effective collaborative filtering," in *Proc. of the 32nd IEEE Int'l Conf. on Data Engineering (ICDE)*, pp. 349–360, 2016.

[20] Y. Lee, S. W. Kim, S. Park, and X. Xie, "How to impute missing ratings?: Claims, solution, and its application to collaborative filtering," in *Proc. of the 27th Int'l Conf. on World Wide Web (WWW)*, pp. 783–792, 2018.

[21] Y. C. Lee, S. W. Kim, and D. Lee, "gOCCF: Graph-theoretic one-class collaborative filtering based on uninteresting items," in *Proc. of the 32nd Int'l Conf. on Artificial Intelligence (AAAI)*, pp. 3448–3456, 2018.

[22] A. Kyrola, G. Blelloch, and C. Guestrin, "Graphchi: Large-scale graph computation on just a pc," in *Proc. of the 10th USENIX Symp. on Operating Systems Design and Implementation (OSDI)*, pp. 31–46, 2012.

[23] J. Kossaifi, Y. Panagakis, and M. Pantic, "Tensorly: Tensor learning in python," *arXiv preprint arXiv:1610.09555*, 2016.

[24] K. Järvelin and J. Kekäläinen, "IR evaluation methods for retrieving highly relevant documents," in *Proc. of the 23rd ACM Int'l Conf. on Research and Development in Information Retrieval (SIGIR)*, pp. 41–48, 2000.

[25] J. Breese, D. Heckerman, and C. Kadie, "Empirical analysis of predictive algorithms for collaborative filtering," in *Proc. of the 14th Int'l Conf. on Uncertainty in Artificial Intelligence (UAI)*, pp. 43–52, 1998.