# Recommending Books to be Exchanged Online in the Absence of Wish Lists

**Maria Soledad Pera**
*People and Information Research Team*
*Department of Computer Science, Boise State University, Boise, Idaho 83725, U.S.A.*
*Email: solepera@boisestate.edu*

**Yiu-Kai Ng**
*Computer Science Department, Brigham Young University, Provo, Utah 84602, U.S.A.*
*Email: ng@compsci.byu.edu*

**An online exchange system is a web service that allows communities to trade items without the burden of manually selecting them, which saves users' time and efforts. Even though online book-exchange systems have been developed, their services can further be improved by reducing the workload imposed on their users. To accomplish this task, we propose a recommendation-based book exchange system, called EasyEx, which identifies potential exchanges for a user solely based on a list of items the user is willing to part with. EasyEx is a novel and unique book-exchange system, since unlike existing online exchange systems, it does not require a user to create and maintain a *wish list*, which is a list of items the user would like to receive as part of the exchange. Instead, EasyEx directly suggests items to users to increase serendipity and as a result expose them to items which may be unfamiliar, but appealing, to them. In identifying books to be exchanged, EasyEx employs known recommendation strategies, i.e., personalized mean and matrix factorization, to predict book ratings, which are treated as the degrees of appeal to a user on recommended books. Furthermore, EasyEx incorporates OptaPlanner, which solves constraint satisfaction problems efficiently, as part of the recommendation-based exchange process to create exchange cycles. Experimental results have verified that EasyEx offers users recommended books that satisfy the users' interests and contributes to the item-exchange mechanism with a new design methodology.**

## Introduction

Internet users have been taking advantage of the free services offered by social websites to exchange items online, which include books, CDs, DVDs, and video games. A significant number of these online item-exchange websites establish a platform for users trading books. Instead of selling used books to bookstores for less than half of their original prices or a fraction of their purchasing costs through garage sales, book owners have found another way to maximize the values of their used books by exchanging them online. The online book-swapping option is attractive, since signing up and accessing these book-exchange websites is free and it just costs the postage to swap books. Existing popular online book-exchange websites include Read-

itswapit,[1] WhatsOnMyBookShelf,[2] PaperBackSwap,[3] and BookMooch,[4] in addition to Socialbib[5] and America's BookShelf.[6].

While Readitswapit, which relies on email notifications to contact its users, provides an interface to allow its users to post books to be swapped with other members and choose books from online libraries that they would like to acquire, a WhatsOnMyBookShelf's user can register a book with a designated point value (called credit) based on the market price of the book and request a book owned by others by redeeming some of their credits. Each WhatsOnMyBookShelf user maintains a wish list[7] of books and is notified through emails when a book included in the wish list is registered to be swapped with another user.

PaperBackSwap users request a book from another user's booklist through email contacts and exchange their books through postal mails. A book credit is earned by the sender and recipient of each transaction. However, in order for a BookMooch user to receive a book from another user, the former must first earn a point by giving away one of his/her books. A BookMooch user can create a wish list on the website so that the user will be automatically notified when a book on the wish list becomes available.

We were inspired by the online swapping task performed by existing book-exchange sites and would like to further enhance their service by simplifying the book-exchange process. We are interested in books, instead of other item-swapping services, such as stock exchanges, since reading fosters people's learning, especially at young ages. The importance of reading has been well-documented in the literature. According to Lifehack,[8] there are a significant number of benefits on regular reading, which include mental stimulation, stress reduction, knowledge acquaintance, vocabulary expansion, memory improvement, analytical thinking skill enhancement, concentration and focus improvement, and free entertainment. Furthermore, an article published by Reader's Digest[9] claims that reading can protect one's brain from Alzheimer's disease, slash stress levels, encourage positive thinking, and fortify friendships.

A number of approaches for item-exchange, including book exchanges, have been presented in the literature (Abbassi and Lakshmanan, 2009; Su, Tung and Zhang, 2012). All of these algorithms, however, require users who participate in the exchange to create and maintain their own wish lists. The wish list of a user must be updated periodically, which takes time and efforts, to reflect the current needs and interests of the user. Consequently, this list may be outdated or incomplete, since the user might not be aware of the existence of other items of interest. With that in mind, we propose a new book-exchange system, called *EasyEx*,[10] which does not require its users to specify their wish lists of books. Instead, EasyEx analyzes books of interest to a user based on a number of commonly-used recommendation algorithms and configures the optimal book exchange cycles involving multiple users using a constraint satisfaction solver.

[1] http://www.readitswapit.co.uk

[2] http://whatsonmybookshelf.com/

[3] http://www.paperbackswap.com/index.php

[4] http://www.bookmooch.com/

[5] http://www.socialbib.com

[6] http://www.americasbookshelf.com/

[7] A *wish list* is a list of items that a user wants in return for giving up the items specified in an *item list*.

[8] http://goo.gl/jRzuxq

[9] http://goo.gl/l4J4Rt

[10] An initial design of EasyEx was originally introduced in our RecSys 2015 poster (Pera and Ng, 2015). In the poster, we provide a simple overview of EasyEx and discuss the preliminary analysis conducted to assess our hypothesis, regarding the possibility of proposing exchanges based on recommendation as opposed to wish lists. Design choices, such as explaining the recommendation strategies considered for EasyEx, and detailed empirical analysis, including evaluating exchange performance based on user cohesiveness, were beyond the score of the poster.

EasyEx is novel, since it employs a recommendation-based book-exchange mechanism that suggests books of interest to a user and generates unanticipated exchanges in return. It is unique, since to the best of our knowledge, EasyEx is the only book-exchange system that incorporates a recommendation mechanism in book-exchange cycles. Furthermore, unlike most of the existing item-exchange tools that limit the exchanges to pairs of users, EasyEx can create multiple exchange cycles that involve more than two users at a time. EasyEx is especially useful for users who have the desire to acquire unknown books of interest which otherwise are overlooked by the users. EasyEx is valuable and attractive in book exchanges, since users and books can find each other by pure serendipity. This fortunate discovery offers the users books to read to gain new knowledge or for entertainment purpose. More importantly, our proposed strategy is domain independent, which explains why EasyEx can easily be extended to other domains of interest, other than books. While EasyEx relies on a well-known optimization tool for book exchanges, one of the main contributions of EasyEx is in configuring the *hard* and *soft constraints* (to be introduced in Our Book-Exchange System section) imposed by the optimizer on the book-exchange domain. Empirical studies conducted using the BookCrossing data-set have verified that books involved in the book-exchange cycles and recommended by EasyEx are appealing to users and the cycles are optimal in terms of involving the maximal number of potential users with books to exchange. This diversifies the types of books involved in the exchange and increases the chance for a user to acquire a desired book.

## Related Work

Recommender systems have been a topic of research interest for more than two decades (Ricci, Rokach, Shapira and Kantor, 2011). Majority of the works in this research area focus on identifying items, such as books, movies, restaurants, and activities, of interest to an individual or a group of users. This identification can be accomplished using popular techniques, including collaborative filtering and content-based analysis, as well as state-of-the-art strategies that rely on matrix factorization (Bauer and Nanopoulos, 2014), context adaptation (Hariri, Mobasher and Burke, 2014), and review analysis (Ling, Lyu and King, 2014). There is no lack of literature on book recommenders (Ricci, Rokach, Shapira and Kantor, 2011). However, recommenders presented in the aforementioned works adopt traditional strategies that directly suggest items but cannot offer recommendation-based exchange transactions, which can be achieved by EasyEx.

In 2016 CLEF, a new task is proposed (Bogers et al., 2016) that is to some extend related to EasyEx. The focus of this task is to examine online forums, where some users asked for book recommendations and others provided them. While the proposed task is connected to EasyEx, in CLEF's task no real book transactions exist, making it possible to offer the same recommendation to multiple requesters. EasyEx, however, is constrained by a limited number of books people want to share, making the task more challenging, since EasyEx needs to satisfy the demand of everybody. In addition, a book request posted on an online forum can be treated as a natural language based *wish list*. In EasyEx, there is no wish list provided by the user, which is automatically inferred. In response to the CLEF's task, Ziak et al. (2016) introduce a system that combines two strategies, classification and linking. The former examines n-grams in forums' posts to determine whether they correspond to book requests, while the latter links books to requests using a lookup table. Htait et al. (2016), on the other hand, present a book recommendation system which automatically generates queries that are submitted to Amazon to locate books to be recommended to users. Their proposed strategy is based on the combination of book tags specified in the requests, in addition to other extracted information, including price, number of pages and publication date of books, as well as information about previous books

read by users. Unlike EasyEx, the strategies discussed in (Ziak et al., 2016) and (Htait et al., 2016) are based on users' requests, which can be treated as a wish list, but is differed from EasyEx in this aspect. More importantly, neither of these strategies can handle book exchanges, and are only limited to book recommendations in the traditional sense. To the best of our knowledge, there is no existing work on recommendation-based book exchanges, and for this reason we discuss existing works on item exchange in the remaining of this section.

A number of item-exchange algorithms are introduced in (Abbassi and Lakshmanan, 2009; Abbassi, Lakshmanan and Xie, 2013; Su, Tung and Zhang, 2012). The exchange algorithm proposed by Abbassi and Lakshmanan (2009) examines two lists, the item list and the wish list, created by each user of popular online community sites, such as Readitswapit, peerflix.com, and swap.com. Using these lists, the algorithm applies a synchronous model that maximizes the items that can be exchanged among users. Likewise, Su et al. (2012) consider the resources and needs of users of an online community system and identify candidate exchanges that can be beneficial to each individual user. These exchanges are generated using the Binary Value-based Exchange Model, which considers two users at a time and identifies resources owned by these users to swap, provided that they have similar values as determined by the respective resource rates established on the site. Abbassi et al. (2013), who recognize the long waiting times required by synchronous models in identifying potential exchanges, propose an asynchronous model which depends on virtual credit points given to item givers that can be redeemed by users against items they wish to receive. The model relies on a greedy algorithm and a linear programming rounding heuristic to enhance the exchange process. Unlike these strategies that rely on wish lists, EasyEx infers user preferences and suggests books of interest to a user. Contrasting to the tools in (Abbassi and Lakshmanan, 2009; Abbassi, Lakshmanan and Xie, 2013; Su, Tung and Zhang, 2012), EasyEx does not restrict the exchanges to two users at a time or to short cycles of very few users.

It is well-known that the item-exchange problem is NP-Complete, and researchers have tried to find approximate, effective solutions to the problem (Abraham, Blum and Sandholm, 2007; Anderson, Ashlagi, Gamarnik and Roth, 2015). Abraham et al. (2007) propose a tree-search approach to identify a local minima solution using a linear integer programming formulation of the exchange problem. Anderson et al. (2015), on the other hand, present various techniques based on integer programming, in addition to the well-known travelling salesman problem, to find optimal solutions for kidney exchange among long chains of patients. Similar to the strategies in (Abraham, Blum and Sandholm, 2007; Anderson, Ashlagi, Gamarnik and Roth, 2015), EasyEx aims to identify exchanges that are favorable to all of the users involved in the exchange process. Unlike the exchange strategies proposed in (Abraham, Blum and Sandholm, 2007; Anderson, Ashlagi, Gamarnik and Roth, 2015) which initiate the exchange process on known, solid facts about kidney-related features, EasyEx infers the interests and preference of users to predict the likelihood of books appealing to individual users prior to initiating the exchange process.

## Our Book-Exchange System

In this section, we present the design methodology of EasyEx. An overview of our proposed recommendation-based book-exchange process is shown in Figure 1.

### Candidate Transactions

Consider a given group of users, each of which is associated with a list of books the user is interested in exchanging for new reading material. EasyEx initiates its recommendation-based

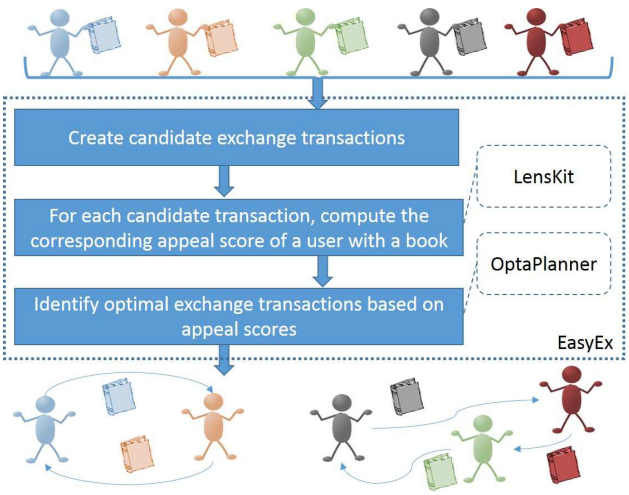


Figure 1: An overview of the recommendation-based book-exchange process of EasyEx

exchange process by identifying potential exchange transactions among the users. Each transaction consists of a user and candidate book to be received by the user.

EasyEx generates candidate exchange transactions by examining all the possible combinations between users and the corresponding books other users are willing to exchange. To avoid suggesting books to a user $U$ that $U$ already has, EasyEx excludes transactions that involve $U$ and a candidate book already in $U$'s item list.

Each candidate transaction is assigned an appeal score, which reflects the predicted degree of interest of a user in receiving a particular book as part of the exchange process. EasyEx analyzes the degree of appeal of each user with each candidate book during its optimization step to generate the optimal set of exchanges. The design goal of EasyEx is to maximize the degrees of appeal of users on books they receive in return.

## Appeal Scores

In this section, we discuss two recommendation strategies, Personalized Mean and Matrix Factorization, which are combined using Multiple Regression to compute the exchange appeal score of a book for a given user.

### Personalized Mean

Personalized mean (or PersMean in short) algorithm is a user-item-based recommendation outlined in LensKit.[11] As shown in Equation 1, PersMean computes $\hat{r}_{u,i}$, the predicted rating of item $i$ for user $u$, using the user and item average rating offsets from the global rating.

$$\hat{r}_{u,i} = \mu + d_i + d_u \tag{1}$$

where $\mu$ is the global mean rating, i.e., the overall average rating, and $d_i$ and $d_u$ are defined by using average offsets as

$$d_u = \frac{1}{|I_u|} \sum_{i \in I_u} (r_{u,i} - \mu) \tag{2}$$

[11] http://lenskit.org/documentation/evaluator/walk through/

$$d_i = \frac{1}{|U_i|} \sum_{u \in U_i} (r_{u,i} - d_u - \mu) \tag{3}$$

where $I_u$ is the set of items rated by $u$, $U_i$ is the set of users who have rated $i$, $r_{u,i}$ is the rating $u$ provided for $i$, and $\mu$ is as defined in Equation 1.

EasyEx adopts PersMean as one of its recommendation strategies, since PersMean is a simple, yet effective, baseline prediction algorithm which is easy to use and understand.

**Matrix Factorization**

Even though PersMean is effective, it primarily depends on the *average* rating score estimated for a given user or an item, i.e., book in our case, which by itself is a limitation, since individual rating prediction is based on the balance between users' and items' average ratings. For this reason, we have also considered a matrix-factorization-based strategy (Ricci, Rokach, Shapira and Kantor, 2011) for predicting the degree of interest of a user on a book. Matrix Factorization strategies are used to learn the latent characteristics of users and items so that it is possible to predict unknown ratings using these latent characteristics (Jamali and Ester, 2010). In other words, these strategies capture the relationships between users, items, and ratings by describing users and items based on a number of factors, i.e., characteristics, inferred from rating patterns. These factors are then exploited to make recommendations. While there are a number of models for identifying latent semantic factors (Blei, Ng and Jordan, 2003), in information retrieval, models based on Singular Value Decomposition (SVD) are well-established (Ricci, Rokach, Shapira and Kantor, 2011). EasyEx relies on the FunkSVD implementation of the matrix factorization recommendation strategy provided by the LensKit Toolkit.[12]

EasyEx's prediction strategy based on matrix factorization represents each item $i$ (i.e., a book) and user $u$ as $n$-dimensional vectors of the form $q_i$, $p_u \in R^n$, such that the vector components of $q_i$ capture the degree to which each factor applies to the corresponding item $i$, and the vector components of $p_u$ measure the degree of interest of $u$ on items (Ricci, Rokach, Shapira and Kantor, 2011). The rating predicted for $u$ on $i$, i.e., $\hat{r}_{u,i}$, is computed using Equation 4.

$$\hat{r}_{u,i} = \mu + b_i + b_u + q_i^T p_u \tag{4}$$

where $q_i^T p_u$ is the dot product that captures the overall interest of $u$ in the characteristics that describe $i$ based on the interaction between $u$ and $i$, and $\mu$, $b_i$, and $b_u$ are parameters, i.e., biases, which denote the overall average rating and the observed deviations of $i$ and $u$ from the average, respectively. The biases associated with $i$ and $u$, i.e., $b_i$ and $b_u$, respectively, are estimated by using Equation 5, which aims to minimize the regularized squared error on a set of known ratings, i.e., training instances of the form $(u, i, r_{u,i})$, and $r_{u,i}$ is defined as in Equations 2 and 3.

$$\min_{b*,p*,q*} \sum_{(u,i) \in K} \left( r_{u,i} - \mu - b_i - b_u - q_i^T p_u \right)^2 + \lambda \left( \|q_i\|^2 + \|p_u\|^2 + b_i^2 + b_u^2 \right) \tag{5}$$

where $\mu$, $b_i$, $b_u$, $q_i$, $p_u$, and $q_i^T p_u$ are as defined in Equation 4, $K$ is the set of training instances, and $\lambda$ is a constant that controls the extent of regularization, which is determined by using cross-validation. Note that minimization is performed by gradient descent (Ekstrand, Ludwig, Konstan and Riedl, 2011).

[12] http://lenskit.org/

### Exchange Appeal Scores

To predict the appeal score of a book for a user as part of the exchange process, EasyEx employs multiple linear regression analysis (Wooldridge, 2009), which is a classical statistical technique for building estimation models (Tan, Zhao, and Zhang, 2009). The analysis accounts for the influence of multiple contributing factors, which are derived from the personalized mean and matrix factorization prediction models.

Equation 6 defines a general prediction model based on multiple regression, where $y$ is the dependent variable, which is the predicted exchange appeal score of a book for a user in our case, $\beta_0$ is the intercept parameter, $\beta_1, \ldots, \beta_n$ are the coefficients of regression, $X_i$ $(1 \leq i \leq n)$ is an independent variable (contributing factor), and $n$ is the number of contributing factors (predictors) in the regression analysis (Wooldridge, 2009).

$$y = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \ldots + \beta_n X_n \tag{6}$$

In Equation 6, each unknown parameter, i.e., the intercept and coefficients of the regression, is estimated through a one-time training process using the Ordinary Least Squares method (Wooldridge, 2009). Each training instance refers to exchange transaction ($u$, $b$) involving a book $b$ and a user $u$ and is represented as a vector of the form $<u, b, p_1, p_2, t>$, where $p_i$ is the (value of the) the $i^{th}$ predictor $(1 \leq i \leq 2)$, i.e., predictors based on PersMean and FunkSVD, and $t$ is the target, i.e., the rating assigned to $b$ by $u$. The Ordinary Least Squares method calculates the *residual* of each exchange transaction (i.e., training instance). This is the *difference* between the target rating assigned to $b$ by $u$ and the exchange appeal score predicted using the (values of the) predictors in the vector representation of the corresponding transaction and Equation 6. Unknown parameters are estimated by minimizing the sum of squared differences between residuals of transactions, i.e., training instances.

Using the (development partition of the) benchmark dataset to be introduced in the Experimental Results section, the regression model in Equation 6 is trained to create the prediction model (with its corresponding parameters) for EasyEx, as shown in Equation 7. Using the trained model and a candidate transaction ($u$, $b$), EasyEx computes the exchange appeal score of $u$ with respect to $b$, which is a candidate book to be considered for swapping.

$$EA(u, b) = 0.208 + 0.722 \times PM(u, b) + 0.223 \times MF(u, b) \tag{7}$$

where $PM(u, b)$ and $MF(u, b)$ are the recommendation scores predicted on $b$ for $u$ using the personalized mean and matrix factorization recommendation strategies.

## Exchange Optimization

Having created candidate transactions for book exchange (as discussed in the Candidate Transactions section) and computed the degrees of exchange appeal for the corresponding transactions (as described in the Appeal Scores section), EasyEx proceeds to identify the optimal set of exchange transactions. This is done by determining the set of transactions that maximizes the number of users that are interested in the books they receive as part of the recommendation-based exchange process. Accomplishing this task requires finding an effective approximate solution. Unfortunately, it has been well-documented in the literature that planning-related problems, such as item-exchange, are NP-Complete (Su, Tung and Zhang, 2012) and thus there is no computational efficient algorithm in solving the problem (Anderson, Ashlagi, Gamarnik and Roth, 2015). To identify an optimal set of swapping transactions that satisfy user expectations

in a timely manner, EasyEx relies on OptaPlanner[13] (Opt, 2015). OptaPlanner is an open source, lightweight, constraint satisfaction solver. It combines sophisticated optimization heuristics and metaheuristics with very efficient score calculations, i.e., intermediate solution assessments, to optimize planning problems, from employee shift rostering (Simic, Simic, Mulitinovic and Djordjevic, 2014) to vehicle routing (DeSmet, 2015). To the best of our knowledge, OptaPlanner has never been applied for optimization problems related to online book swapping. (Further details on OptaPlanner can be found in (Opt, 2015).)

OptaPlanner users are given a number of heuristics and algorithms to choose from, in addition to defining domain-dependent constraints. Due to the page limitation, we only discuss below the OptaPlanner algorithms, heuristics, and constraints employed by EasyEx to generate an optimal solution that maximizes the overall user satisfaction with the proposed swappings, regardless of the number of users participating in the exchange.

The first step in generating the optimal exchange solution is to define the domain of the problem and the corresponding problem constraints, which guide the optimization process. The domain of our book-exchange problem involves (i) a user (identified by a unique user ID), who is a member of an online book-swapping site and must have at least a book to exchange, (ii) a book (identified by a unique book ID), which is available for exchange among users, and (iii) a book exchange planning entity that keeps track of which books have been assigned to which users (with the corresponding degrees of appeal) in an intermediate solution. There are also two constraints to be defined: (i) assigning a book for exchange to each user and (ii) maximizing the average exchange appeal scores of users with books involved in swapping. The former is a *hard* constraint, which is a rule that must not be broken, whereas the latter is a *soft* constraint, which can be broken if it cannot be avoided (Opt, 2015).

With the problem domain and the corresponding constraints well-defined, OptaPlanner proceeds with its initialization step based on the First-fit heuristic. This construction heuristic initiates the process of generating an optimal solution by cycling through all the planning entities (users) and assigning a user a book recommended for exchange. In other words, every user (in default order) is assigned a book, among those available for swapping, that maximizes his degree of appeal. The next step in OptaPlanner's optimization process involves iteratively generating intermediate solutions. In accomplishing this task, OptaPlanner adopts Tabu Search (Glover, 1989), which identifies near-optimal solutions for combinatorial optimization problems. To enhance local search, Tabu Search uses flexible structures memory which facilitates the task of more thoroughly exploiting search information than by rigid memory or memoryless systems (Glover, 1990). Unlike the popular Hill Climbing strategy (Jiang, Ren and Zhao, 2013), Tabu Search avoids "getting stuck" in local optimality by maintaining a "tabu" list of objects (books in our case) that have recently been used (i.e., assigned to a user for exchange) and thus are off-limits to be used for now. Tabu Search depends on a neighborhood search process to seek from one intermediate solution to another one iteratively. After each intermediate solution is generated, OptaPlanner computes a solution score that quantifies the effectiveness of the solution. This score is calculated by averaging the degree of appeal of each user with the book received as part of the exchange based on the current solution under evaluation. Given that the *higher* the solution score is, the *better* the generated intermediate solution is, OptaPlanner explores intermediate solutions until a stopping, i.e., termination, criteria has been reached. In our case, the defined stopping criteria is time-based, i.e., the process of searching for an optimal solution stops when the solution scores computed for the iteratively-generated intermediate solutions do not improve after 5 minutes.[14] After the termination criteria is reached, the set of user-book

[13] http://www.optaplanner.org/

[14] We have empirically determined the stopping criteria to find the effective exchange solutions for sets of users of

pairs that constitute the optimal set of exchanges is generated.

As stated in the Introduction section, our proposed book-exchange strategy is domain independent and can easily be extended to other domains of interest, other than books. EasyEx is applicable to domains other than books, since (i) OptaPlanner, which is adopted by EasyEx to generate the optimal exchange solution of items, is easy to configure in defining different item domains, and (ii) the recommendation approach of EasyEx is also applicable to any domain besides books. The variety of these domains, which include movies, songs, (video) games, toys, to name a few, in addition to the book domain, should clearly identify the real applications of EasyEx as an item-exchange system.

## Experimental Results

In this section, we present the results of the empirical studies conducted to assess the performance of EasyEx.[15] In the Datasets section, we introduce the datasets used for training and assessment purposes. In the Recommendation Assessment of EasyEx and Exchange Assessment of EasyEx sections, we demonstrate the correctness of the recommendation and exchange strategies adopted by EasyEx, respectively. In the Overall Assessment of EasyEx section, we verify the overall effectiveness and efficiency of EasyEx.

### Datasets

The dataset employed to evaluate EasyEx and compare the performance of EasyEx with other recommendation approaches is the BookCrossing dataset. The popular BookCrossing dataset was manually created between August and September of 2004 with data extracted from BookCrossing.com. The dataset contains 1.15 million ratings given by 278,858 customers on 271,379 books. The set includes (i) users whose user IDs have been anonymized and represented as integers, (ii) books that have been bookmarked by users and identified by their respective ISBNs, and (iii) book rating information provided by the corresponding users which are expressed on a 1-to-5 scale. The well-known and widely-adopted BookCrossing dataset was selected for assessment purposes, sine it continues to be used as the standard benchmark for book recommendation evaluation (Park, 2013; Subbian et al., 2016; Cremonesi et al., 2014) .

The BookCrossing dataset is comprised of 1,149,780 tuples of the form $<u, b, t>$, where $t$ is the target rating assigned by a given user $u$ to a given book $b$. We removed a number of tuples without target rating from the dataset. To ensure an unbiased evaluation framework, we separated the remaining 433,671 tuples into two disjoint sets, i.e., BkXing-Dev and BkXing-Test, which are used for development and testing purposes, respectively. In separating the dataset, we ensured that all involved users were represented in both splits. For each user in the dataset with 433,671 tuples, we selected one random tuple and included it in BkXing-Test, while the rest of the tuples associated with the same user remained in BkXing-Dev. This process yields 401,248 tuples in BkXing-Dev and 32,423 tuples in BkXing-Test.

### Recommendation Assessment of EasyEx

The quality of exchanges created by EasyEx is dictated by the effectiveness of its rating predictions. To assess the performance of EasyEx in generating high-quality book recommendations,

various sizes with minimal processing time.

[15]Information pertaining to EasyEx's configuration, along with data samples, can be found in https://github.com/BoiseState/Piret-EasyEx.

| *PersMean* | *MF-FunkSVD* | *Regression* **(EasyEx)** |
|---|---|---|
| 0.7871 $\pm$ 0.0069376 | 0.7874 $\pm$ 0.0069376 | 0.7572 $\pm$0.0065875 |

Table 1: Average RMSEs and corresponding standard deviations computed for a number of recommendation strategies using BkXing-Dev

as well as the performance of other recommendation strategies, we depend upon the Root Mean Square Error (RMSE) (Ricci, Rokach, Shapira and Kantor, 2011) (see Equation 8). This performance evaluation metric is used to measure the overall difference among the rating predictions generated by a recommendation strategy with respect to the real (i.e., target) ratings provided by users.

$$RMSE = \sqrt{\frac{\sum_{i=1}^{n}(f(x_i) - y_i)^2}{n}} \tag{8}$$

where $n$ is the total number of items with ratings to be evaluated, $f(x_i)$ is the rating predicted by a system on item $x_i$ ($1 \leq i \leq n$), which is a book in our case, and $y_i$ is an expert-assigned rating to $x_i$.

We chose RMSE for evaluation purpose, since it is a standard measure for evaluating the performance of predictive recommenders (Ricci, Rokach, Shapira and Kantor, 2011) of which EasyEx is one of them. While its top-N counterparts generate lists of "N" items most likely of interest to a user, predictive strategies estimate utility values for the recommendation. In this case, computing how close a predicted rating is to an "actual" rating a user would give an item is more effective as a measure than only determining whether the item is relevant or not.

Using the aforementioned datasets and metric, we conducted a number of empirical studies to verify the correctness of EasyEx's recommendation strategy and compare its performance with a number of baseline and (sampled) state-of-the-art recommenders. The results of these studies are discussed below.

***EasyEx's Hybrid Recommendation Strategy.*** To verify EasyEx's ability to predict reliable ratings for a candidate book to be received by a given user as part of a exchange transaction and demonstrate the need for adopting a hybrid recommendation strategy, we conducted an empirical study using BkXing-Dev.

Based on a 5-fold cross-validation approach, using a random split that ensured that a user was present at least once in each fold, we employed 4 folds of BkXing-Dev for training the PersMean and MF-FunkSVD algorithms and the remaining fold for testing. Since EasyEx's regression strategy depends upon PersMean and MF-FunkSVD predictions, we created a counterpart instance of the form $<u, b, p_1, p_2, t>$ for each instance in the corresponding fold, where $<u, b, t>$ is extracted from BkXing-Dev such that $t$ is the target rating assigned by user $u$ on book $b$, and $p_1$ and $p_2$ are the ratings predicted for $u$ on $b$ using PersMean and MF-FunkSVD, respectively. Using these extended tuples and the same training and testing folds, we also evaluated EasyEx's regression-based prediction strategy.

As shown in Table 1, EasyEx achieves a lower RMSE than PersMean and MF-FunkSVD, the individual prediction strategies simultaneously considered by EasyEx. These differences in RMSE achieved by EasyEx are statistically significant, which were determined using a Wilcoxon signed-rank test ($p < 0.05$). The lower RMSE demonstrates the correctness of the recommendation strategy adopted by EasyEx in determining the degree of appeal of a user on a book to be received as part of the exchange. As we adopt the popular 5-fold cross validation strategy, the results depicted in Table 1 correspond to the average RMSE generated by each strategy based on the five repetitions of the experiments.

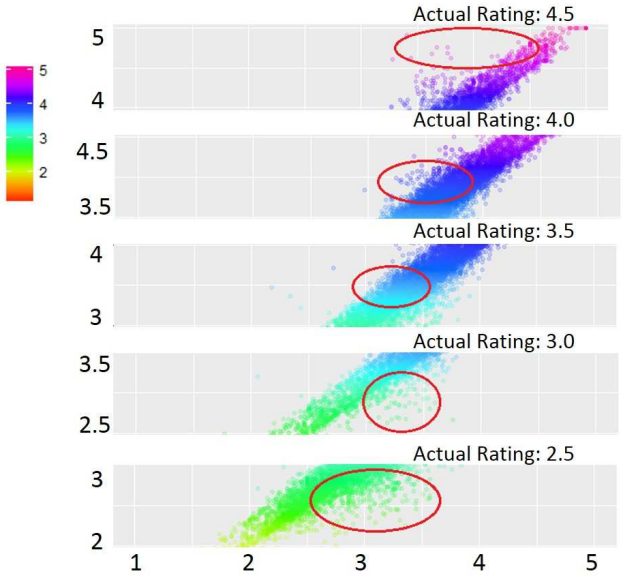


Figure 2: Sample distribution of ratings predicted using MF-FunkSVD (Y-axis), PersMean (X-axis), and a Regression-based strategy on the BkXing-Dev dataset. The palette on the left indicates the value of the predicted score for the corresponding instance in BkXing-Dev based on EasyEx's regression analysis

To further demonstrate the correctness of EasyEx's strategy in predicting exchange appeal scores, we conducted an in-depth analysis of each of the training instances in BkXing-Dev. As illustrated in Figure 2, we observe that MF-FunkSVD and PersMean (depicted on the Y-axis and X-axis of Figure 2, respectively) yield similar prediction scores for a number of instances, which tends to be caused by the bias in the dataset towards medium to high ratings. There are instances, such as the ones shown within the circled portion of the scattered plot illustration depicted in Figure 2, that clearly highlight why considering a single recommendation strategy would lead to erroneous predicted ratings, which does not happen when MF-FunkSVD and PersMean are considered in-tandem, as indicated by the color used to represent these data points in the graph. This type of instances can be handled more favorably by EasyEx's regression-based model which compensates for the discrepancy in performing the prediction task using either MF-FunkSVD or PersMean by itself, since the former achieves an overall *lower* error rate on the BkXing-Dev dataset. The decrease in error rate using EasyEx's regression strategy is also consistent on the dataset used for testing purposes, i.e., the BkXing-Test dataset, as depicted in Figure 4.

Upon closer analysis of the results, we observed that while MF-FunkSVD outperforms PersMean for users who have rated multiple books, PersMean shows a better predictive performance for users who have only provided one or two ratings (see Figure 3). This is anticipated, since for users who lack historical data, Matrix Factorization algorithms have been known to perform less adequately. This discovery also helps explain the improvement observed on EasyEx's regression-based recommendation performance over either PersMean or MF-FunkSVD applied separately.

***EasyEx versus Other Recommendation Strategies.*** We further demonstrate the correctness of EasyEx's recommendation strategy by comparing its performance with a number of baseline and state-of-the-art recommenders. For baseline measures, in addition to *PersMean* and

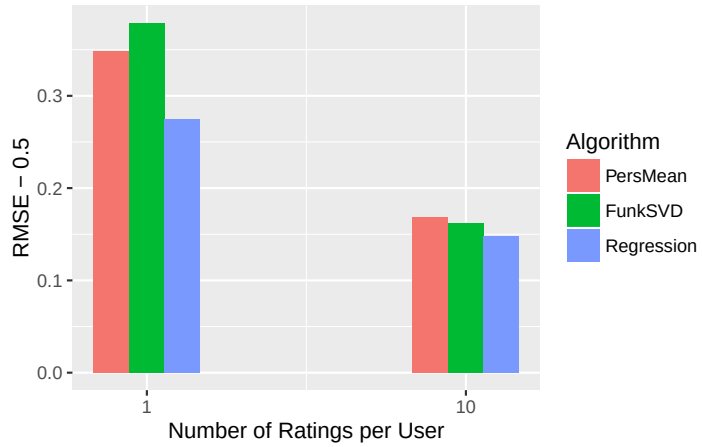


Figure 3: RMSEs grouped by the number of ratings per user computed by using BkXing-Dev for different prediction strategies. For easier visualization, RMSE values are offset by 0.50

*MF-FunkSVD* (125 iterations, 0.002 learning rate, and 40 features), we have chosen *Item-based* (neighborhood size = 20) and *User-based* (neighborhood size = 30) collaborative filtering (CF) strategies on LensKit. These strategies are based on the well-known and effective CF approach for rating prediction (Ricci, Rokach, Shapira and Kantor, 2011). The remaining recommendation strategies considered for comparison purposes, which are introduced below and implemented as in (Qumsiyeh and Ng, 2012)), rely on matrix factorization, machine learning, and metadata-analysis methodologies. These approaches were chosen because they have achieved high accuracy in rating prediction on items in various domains.

*Non-parametric matrix factorization (NPMF)*. Yu et al. (2009) propose a non-parametric matrix factorization method in predicting ratings on items, which include books, without requiring the model dimensionality, i.e., users, features, and ratings, to be specified a priori (for the comparison purpose, we did set the number of iterations to be 50). Rather, the dimensionality is determined from previously rated items instead.

*Collective matrix factorization (CMF)*. Similar to NPMF, the recommender developed by Singh and Gordon (2008) is also based on matrix factorization.[16] However, Singh et al.'s approach incorporates relational learning, which predicts unknown values of a relation among entities of an item using a given database of entities and observed relations among entities.

*Machine Learning (ML)*. Probabilistic frameworks have been introduced for rating predictions in the past. Shi et al. (2010) apply a supervised machine learning approach to automatically construct a ranking model/function from training data for rating prediction.[17] The machine learning approach applies a rank-oriented strategy, which exploits pairwise preferences between items and users to generate a list of ranked items corresponding to the ratings such that the higher an item is ranked, the higher its rating is.

*Multimedia Recommendation System (MudRecS)*. MudRecS (Qumsiyeh and Ng, 2012) is one of the most effective strategies that considers item metadata in making recommendations. It simultaneously analyses users' ratings and readability levels, in addition to the genres, reviews,

[16] We used default parameters, with $\alpha = 0.20$ and genre extracted from the Library of Congress catalog records.

[17] The system was originally designed to predict ratings on movies but was implemented by Qumsiyeh and Ng (2012) on books as well using the default parameter setting. In addition, we set the number of iterations to 950 and $\lambda = 0.01$. As we did for CMF, we supplemented data from BookCrossing with catalog records from the Library of Congress and Epinions.com to ensure MudRecS' high performance.

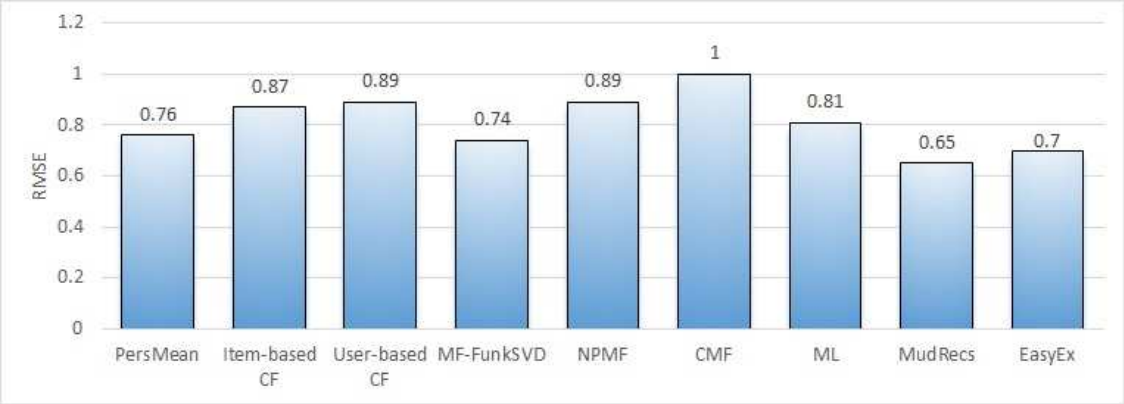


Figure 4: RMSEs of different rating prediction systems based on the BkXing-Test dataset

and popularity of items, to generate personalized rating predictions for multimedia items.

Using BkXing-Dev for training each of the aforementioned strategies (including EasyEx), we computed the corresponding RMSE values based on the predictions generated for instances in BkXing-Test. As shown in Figure 4, EasyEx significantly outperforms (based on the Wilcoxon signed-rank test with $p < 0.05$) majority of the rating prediction methods defined above using BkXing-Test. The predictions achieved by EasyEx and MudRecS are comparable. However, the latter thrives on the presence of data in multiple domains and enhanced item metadata to make recommendations. Given that exchange systems tend to be applied to individual domains, such as books or movies, but not both, the former is preferable than the latter in this regard.

In addition to EasyEx's RMSE as shown in Figure 4, we computed its mean average error, which is 0.68. This error rate indicates that the rating predicted by EasyEx for an item is in general a little over half a star away from the rating provided by the corresponding user on the item.

### Exchange Assessment of EasyEx

As discussed in the Exchange Optimization section, OptaPlanner offers a number of algorithms that can be applied to generate effective solutions for optimizations problems in different application domains, which is book-exchange in our case. These algorithms include Tabu Search (as discussed in the Exchange Optimization section), Hill Climbing, Simulated Annealing, and (Hill Climbing with) Late Acceptance. The last three optimization strategies are discussed in detail in (Opt, 2015).

To determine the most effective algorithm offered by OptaPlanner and to be applied by EasyEx to iteratively identify an optimal set of exchanges among users and books available for swapping, we first created datasets of different sizes. These datasets, denoted Opt_5, Opt_10, Opt_15, Opt_25, Opt_50, and Opt_100, were created by randomly selecting 5, 10, 15, 25, 50 and 100 user-book pairs, respectively, from the BkXing-Test dataset (introduced in the Datasets section) to initiate the recommendation-based exchange process. Each dataset includes tuples $<u, b, a>$, which represent candidate transactions generated for a set of users and books (introduced in the Candidate Transactions section), such that $a$ is an appeal score computed by using EasyEx's recommendation strategy for the corresponding user $u$ and book $b$ (as presented in the Appeal Scores section).

Table 2 shows OptaPlanner's processing time (in seconds) in generating an optimal solu-

Table 2: The effectiveness and processing time of solutions generated by OptaPlanner on diverse optimization approaches and datasets of different sizes

| Datasets | Exchange Pairs | Solution Score | | | | | Processing Time (in Seconds) | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | | Late Acceptance | Simulated Annealing | Hill Climbing | Tabu Search | Ideal | Late Acceptance | Simulated Annealing | Hill Climbing | Tabu Search |
| Opt_5 | 20 | 1.36 | 1.36 | 2.68 | 2.68 | 3.63 | 62.9 | 61.3 | 61.4 | 61.4 |
| Opt_10 | 90 | 1.31 | 1.66 | 3.04 | 3.04 | 3.76 | 62.9 | 61.3 | 61.4 | 61.4 |
| Opt_15 | 210 | 2.05 | 2.73 | 3.73 | 3.73 | 4.29 | 69.1 | 64.4 | 61.4 | 61.3 |
| Opt_25 | 600 | 1.12 | 1.89 | 2.87 | 3.15 | 3.75 | 67.5 | 108.0 | 62.9 | 66.8 |
| Opt_50 | 2450 | 0.59 | 0.74 | 2.10 | 3.39 | 3.83 | 97.0 | 114.0 | 85.2 | 67.6 |
| Opt_100 | 9900 | 0.36 | 0.39 | 1.71 | 3.41 | 3.86 | 229.0 | 654.0 | 135.2 | 117.6 |

tion for the corresponding data-set, in addition to the solution score[18] achieved by each of the optimization algorithms mentioned above. As shown in Table 2, Tabu Search (i) consistently outperforms or generates solutions as good as its counterparts as reflected by its solution scores, and (ii) minimizes the time required for locating optimal solutions for datasets of different sizes, especially for large numbers of user-book pairs. The results presented in Table 2 provide a convincing justification in choosing Tabu Search for generating EasyEx's optimal exchanges both for performance and scalability reasons.

To establish a context on the effectiveness of the different optimization strategies considered by OptaPlanner, we have included in Table 2 the "ideal" solution based on the corresponding dataset. An *ideal solution* is defined as the average degree of appeal of each user on each book received as part of an exchange, assuming that every user was given the book that is the most appealing to him among the ones available for swapping, regardless whether the book is also the most appealing to other users involved in the exchange cycle. Of course this scenario is highly unlikely to occur in reality, since within an exchange cycle, a book can only be assigned to one of the users involved in the exchange regardless the number of users who want it. This explains why the solution is called *ideal*. The difference between the solution score generated by using Tabu Search and the ideal solution score is the *lowest* among the optimization approaches shown in Table 2, regardless of the dataset examined.

## Overall Assessment of EasyEx

In this section, we discuss the overall assessment of EasyEx in terms of its ability to suggest book exchanges for heterogeneous groups of users and its efficiency.

### Assessment based on User Cohesiveness

To further assess EasyEx, we analyze its ability to generate recommended-based book exchanges for users regardless of the degrees of cohesiveness, i.e., *highly-similar*, *dissimilar*, and *intermediate*, among them. The assessment based on cohesiveness, which we adopted from the evaluation framework presented in (Sihem, Roy, Chawlat, Das and Yu, 2009), verifies that EasyEx is capable of recommending books of interest to its users, even if the books to be considered for swapping belong to users who are not similar in terms of their preferences on books determined by their rating patterns. This task is more challenging than simply interchanging books among users sharing mutual interests. *Highly-similar* users refer to users with similar rating patterns, *dissimilar* ones are users who have disparate rating patterns, and *intermediate* users are the ones with

[18]The solution score quantifies the effectiveness of a solution and is calculated by averaging the degree of appeal of each user with the book received as part of the exchange, which is determined by the solution being evaluated.

Table 3: Optimal Solution scores obtained by EasyEx on users with diverse degrees of cohesiveness, where 'Diff' stands for "Difference"

| Cohesiveness Among Users | Average Optimal Solution Score | Average Ideal Score | Difference |
|---|---|---|---|
| Highly-Similar | 3.24 | 4.37 | 1.13 |
| Intermediate | 2.78 | 4.16 | 1.38 |
| Dissimilar | 2.56 | 4.02 | 1.46 |

neither similar nor dissimilar rating patterns. To determine the users who should be included in each group, we calculated the user-to-user similarity, denoted *UserSim*, on each pair of users $u$ and $u'$ in BkXing-Test as follows.

$$UserSim(u, u') = \frac{|\{i \mid i \in I_u \bigwedge i \in I_{u'} \bigwedge |r_{u,i} - r_{u',i}| \leq 2\}|}{|\{i \mid i \in I_u \bigvee i \in I_{u'}\}|} \tag{9}$$

where $r_{u,i}$, $r_{u',i}$, $i$, $I_u$ and $I_{u'}$ are as defined in Equation 2, and $|r_{u,i} - r_{u',i}| \leq 2$ indicates that $i$ is a book rated similarly by $u$ and $u'$, given that both users rated $i$ within 2 units of each other on the scale of 1 to 5. This constraint guarantees that $u$ and $u'$ assigned a similarly high/low rating to $i$, i.e., $u$ and $u'$ share a similar preference on $i$.

Following the strategy presented in (Sihem, Roy, Chawlat, Das and Yu, 2009), we relied on the distribution of *UserSim* scores computed for pairs of users in BkXing-Test and treated the 33% of user-pairs with the highest *UserSim* scores as *highly-similar* users and the 33% with the lowest *UserSim* scores as *dissimilar* users. The remaining 33% are treated as *intermediate* users. Based on this group formation protocol, we created five distinct groups of five users in each of the categories, i.e., highly-similar, dissimilar, and intermediate. As shown in Table 3, the best average optimal solution score obtained for book exchanges is among the highly-similar users, which is expected. However, even when the exchanges occur among dissimilar or intermediate users, the average optimal solution scores remain close to 3, which showcases that EasyEx generates appealing book-swapping suggestions among its users, regardless of their rating patterns on books.

Table 3 also includes the average ideal solution score as defined earlier. Given that it is more likely that most of the books available for swapping are equally appealing to users who share similar rating patterns, it is anticipated that the *difference* between EasyEx's optimal solution score and the ideal one is the *lowest* for groups of highly-similar users. On the other hand, it is less likely that books among dissimilar users are highly of interest to each other, and thus the *difference* between EasyEx's average optimal solution score for groups of dissimilar users and the corresponding average ideal solution score is *higher*.

**Efficiency of EasyEx**

In addition to validating the effectiveness of EasyEx, we also analyze the efficiency of its design methodology, starting with predicting users' appeal scores on candidate books to be exchanged and ending with generating an optimized exchange cycle. Using the BkXing-Test dataset, we created random sets of user-book pairs of various sizes, from 5 to 500. For each group, we generated the corresponding candidate transactions (from 20 to 249,500, respectively, as illustrated in Figure 5) and determined the optimized exchange solution among the users in each group. As shown in Figure 5, even though the increase in the number of candidate transactions to be examined follows an exponential trend, the increase in EasyEx's processing time is polynomial,

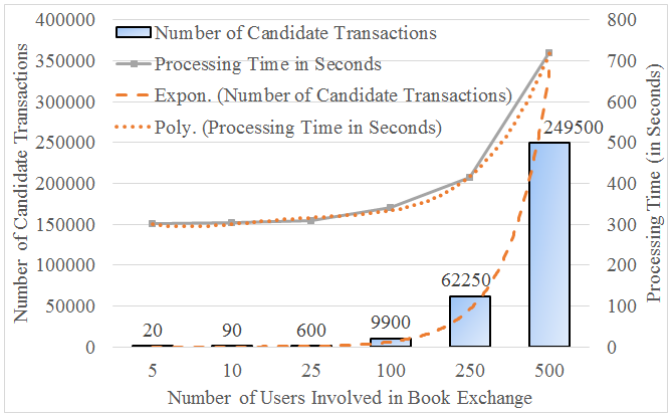


Figure 5: Efficiency assessment of EasyEx

which demonstrates the efficiency of EasyEx.[19]

## Conclusions and Future Work

To further enhance existing online book-exchange mechanisms and facilitate the exchange cycles that increase serendipity on books received by users, we have proposed a recommendation-based book-exchange system, called EasyEx. EasyEx relies on widely-used recommendation algorithms provided by LensKit and multiple regression to make suggestions on books to be received as part of the exchange process and counts on optimal book swapping cycles generated by OptaPlanner to handle user requests on exchanging books online. The proposed recommendation system is novel, since unlike online exchange modules used by existing book-exchange websites, it (i) does not require users to create and maintain wish lists, (ii) generates recommendation-based exchanges among users to offer books of interests users might not otherwise be aware of, and (iii) proposes suitable exchanges for more than a handful of users at the time. EasyEx is unique, since it is the first recommendation-based system on book exchanges, which can easily be adopted for exchanging other items besides books online. Experimental results conducted using the popular BookCrossing dataset have verified the *correctness* of the recommendation and optimization process, as well as the *efficiency* of EasyEx and its *effectiveness* as a recommendation-based book-exchange system.

For future work, we plan to investigate whether EasyEx can further be enhanced by (i) clustering users based on their reading preferences inferred by the content or reviews of previously-rated books and (ii) excluding users from exchange cycles whose appeal score on a recommended book is below a certain threshold. Furthermore, since generating recommendation-based exchanges is the merit of EasyEx, it is important to explicitly address the cold-start issue that often affects recommendation strategies. With that in mind, we plan to explore popularity bias and content-based strategies, which are two of the alternatives that can be incorporated into EasyEx's recommendation process to address the cold-start problem. Considering the popularity bias approach could help EasyEx prioritize exchanges of popular books among users for whom no historical rating data is available. Applying the content-based strategy could enhance EasyEx, since it would be able to identify books matching the theme or topic of the book a user

[19]The processing times reported in Figure 5 correspond to the average processing times computed over different groups with the same number of users. For each group size, i.e., groups of 5, 10, 25, 100, 250, and 500 users involved in book exchange, we repeated the experiment 5 times.

should be willing to provide as part of the exchange, even if no other information is available about the user. Besides the rating-based strategies, other approaches, such as those based on demographic information, could also be examined to determine if they can positively influence the overall quality of EasyEx's predictive scores.

Given that the current offline assessment based on simulated exchanges does not account for the fact that users can reject the suggested exchange, we plan to conduct other assessments to further verify the effectiveness, efficiency, and scalability of EasyEx by simulating rejected exchanges (offline validation) and conducting evaluations on real-life exchange setting (online validation). We would also like to redesign EasyEx to target K-12 readers. This audience is special, since it is essential to provide young readers with books that not only appeal to their interests, but also they can comprehend. Anticipated challenges in modifying EasyEx for K-12 readers include ensuring that suggested books explicitly take into account the reading skills of these young readers and handling the lack of personal user ratings among K-12 users.

## References


Abbassi, Z., & Lakshmanan, L. (2009). On Efficient Recommendations for Online Exchange Markets. In Proceedings of the IEEE International Conference on Data Engineering (ICDE), pp. 712–723.

Abbassi, Z., Lakshmanan, L., & Xie, M. (2013). Fair Recommendations for Online Barter Exchange Networks. In Proceedings of the ACM International Workshop on the Web and Databases (WebDB 2013), pp. 43–48.

Abraham, D., Blum, A., & Sandholm, T. (2007). Clearing Algorithms for Barter Exchange Markets: Enabling Nationwide Kidney Exchanges. In Proceedings of the 8th ACM Conference on Electronic Commerce, pp. 295–304.

Anderson, R., Ashlagi, I., Gamarnik, D., & Roth, A. (2015). Finding Long Chains in Kidney Exchange Using the Traveling Salesman Problem. *Proceedings of the National Academy of Sciences of the United States of America (PNAS)* **112**(3), 663–668.

Bauer, J., & Nanopoulos, A. (2014). A Framework for Matrix Factorization Based on General Distributions. In Proceedings of the ACM Conference on Recommender Systems (RecSys), pp. 249–256.

Blei, D., Ng, A., & Jordan, M. (2003). Latent Dirichlet Allocation. *Journal of Machine Learning Research* **3**, 933–1022.

Bogers, T., Hendrickx, I., Koolen, M., & Verberne, S. (2016). Overview of the SBS 2016 Mining Track. Available at http://ceur-ws.org/Vol-1609/16091053.pdf.

Cremonesi, P., & Quadrana, M. (2014). Cross-Domain Recommendations without Overlapping Data: Myth or Reality?. In Proceedings of the 8th ACM Conference on Recommender Systems (RecSys), pp. 297–300.

DeSmet, G. (2015), *Visualizing Vehicle Routing with Leaflet and Google Maps*, Retrieved from http://goo.gl/9ox4LM.

Ekstrand, M., Ludwig, M., Konstan, J., & Riedl, J. (2011). Rethinking the Recommender Research Ecosystem: Reproducibility, Openness, and LensKit. In Proceedings of the ACM Conference on Recommender Systems (RecSys), pp. 133–140.

Glover, F. (1989). Tabu Search, Part 1. *ORSA Journal on Computing* **1**, 190–206.

Glover, F. (1990). Tabu Search: A Tutorial. *Interfaces* **20**(4), 74–94.

Hariri, N., Mobasher, B., & Burke, R. (2014). Context Adaptation in Interactive Recommender Systems. In Proceedings of the ACM Conference on Recommender Systems (RecSys), pp. 41–48.

Htait, A., Fournier, S., & Bellot, P. (2016). SBS 2016: Combining Query Expansion Result and Books Information Score for Book Recommendation. Available at http://ceur-ws.org/Vol-1609/16091053.pdf.

Jamali, M., & Ester, M. (2010). A Matrix Factorization Technique with Trust Propagation for Recommendation in Social Networks. In Proceedings of the ACM Conference on Recommender Systems (RecSys), pp. 135–142.

Jiang, T., Ren, G., & Zhao, X. (2013). Evacuation Route Optimization based on Tabu Search Algorithm and Hill-climbing Algorithm. *Procedia - Social and Behavioral Sciences* **96**(6), 865–872.

Ling, G., Lyu, M., & King, I. (2014). Ratings Meet Reviews, A Combined Approach to Recommend. In Proceedings of the ACM Conference on Recommender Systems (RecSys), pp. 105–112.

Opt (2015), *OptaPlanner User Guide*. Available at http://goo.gl/hTd46P.

Park, Y. (2013). The Adaptive Clustering Method for the Long Tail Problem of Recommender Systems. *IEEE Transactions on Knowledge and Data Engineering (TKDE)* **25**(8), 1904–1915.

Pera, M., & Ng, Y.-K. (2015). A Recommendation-based Book-Exchange System Without Using Wish Lists (Poster Paper). In Proceedings of the ACM Conference on Recommender Systems (RecSys).

Qumsiyeh, R., & Ng, Y.-K. (2012). Predicting the Ratings of Multimedia Items for Making Personalized Recommendations. In Proceedings of the International Conference on Research and Development in Information Retrieval, pp. 475–484.

Ricci, F., Rokach, L., Shapira, B., & Kantor, P. (2011). *Recommender Systems Handbook*, Springer.

Shi, Y., Larson, M., & Hanjalic, A. (2010). List-wise Learning to Rank with Matrix Factorization for Collaborative Filtering. In Proceedings of the ACM Conference on Recommender Systems (RecSys), pp. 269–272.

Sihem, A., Roy, S., Chawlat, A., Das, G., & Yu, C. (2009). Group recommendation: Semantics and Efficiency. In Proceedings of the International Conference on Very Large Data Bases (VLDB), pp. 754–765.

Simic, D., Simic, S., Mulitinovic, D., & Djordjevic, J. (2014). Challenges for Nurse Rostering Problem and Opportunities in Hospital Logistics. *Journal of Medical Informatics and Technologies* **23**, 195–202.

Singh, A., & Gordon, G. (2008). Relational Learning via Collective Matrix Factorization. In Proceedings of the ACM SIGKDD Conference on Knowledge Discovery and Data Mining, pp. 650–658.

Su, Z., Tung, A., & Zhang, Z. (2012). Supporting Top-K Item Exchange Recommendations in Large Online Communities. In Proceedings of the 15th International Conference on Extending Database Technology (EDBT), pp. 97–108.

Subbian, K., Aggarwal, C., & Hegde, K. (2016). Recommendations for Streaming Data. In Proceedings of the 25th ACM International on Conference on Information and Knowledge Management (CIKM), pp. 2185–2190.

Tan, H., Zhao, Y., & Zhang, H. (2009). Conceptual Data Model-based Software Size Estimation for Information Systems. *ACM Transactions on Software Engineering and Methodology (TOSEM)* **19**(2), Article 4.

Wooldridge, J. (2009). *Introductory Econometrics: A Modern Approach*, South-Western Pub.

Yu, K., Zhu, S., Lafferty, J., & Gong, Y. (2009). Fast Nonparametric Matrix Factorization for Large-Scale Collaborative Filtering. In Proceedings of the International Conference on Research and Development in Information Retrieval, pp. 211–218.

Ziak, H., Rexha, A. & Kern, Roman (2016). KNOW at The Social Book Search Lab 2016 Mining Track. Available at http://ceur-ws.org/Vol-1609/16091190.pdf.