WHEN THE SYSTEM BECOMES YOUR PERSONAL DOCENT: CURATED BOOK RECOMMENDATIONS

by

Nevena Dragovic

A thesis

submitted in partial fulfillment

of the requirements for the degree of

Master of Science in Computer Science

Boise State University

December 2016

BOISE STATE UNIVERSITY GRADUATE COLLEGE

DEFENSE COMMITTEE AND FINAL READING APPROVALS

of the thesis submitted by

Nevena Dragovic

Thesis Title: When the System Becomes Your Personal Docent: Curated Book Recommendations

Date of Final Oral Examination: 20th August 2016

The following individuals read and discussed the thesis submitted by student Nevena Dragovic, and they evaluated the presentation and response to questions during the final oral examination. They found that the student passed the final oral examination.

Maria Soledad Pera, Ph.D.	Chair, Supervisory Committee
Marissa Schmidt, MS	Member, Supervisory Committee
Edoardo Serra, Ph.D.	Member, Supervisory Committee

The final reading approval of the thesis was granted by Maria Soledad Pera, Ph.D., Chair of the Supervisory Committee. The thesis was approved by the Graduate College.

Dedicated to my mom

ACKNOWLEDGMENTS

I would like to express my enormous appreciation for my committee chair and advisor, Dr. Maria Sole Pera and her guidance. This work would not have the same quality if it was not for her exceptional availability, dedication, encouragement and patience. Likewise, I would like to express a special gratitude to her for introducing me to Information Retrieval research and providing me with endless number of opportunities to learn and grow as a student and a person. I would also like to thank the entire faculty staff at Boise State University Computer Science Department for sharing their knowledge and providing me with the highest level of education. I would like to thank the committee members for taking time to read the manuscript and providing me with their feedback. I would like to express a special gratitude for Marissa Schmidt and her encouragement, as well as Dr. Michael Ekstrand for all his help during this thesis work.

In addition, I would like to express my sincere appreciation for everyone who guided and supported me throughout my life. First and foremost, I would like to thank my family: Father, Mother, Sisters, Brother and Grandparents for their support and love. I would like to thank Nick for his support and belief in me. Furthermore, I want to thank my friends around the world for their support.

I would like to thank entire Computer Science Student Research Lab for being extremely supportive and for sharing their knowledge and experience. I owe special thanks for the unconditional support from many colleagues, especially Ion Madrazo,

Anup Shrestha, Jesse Lovitt and Mikel Joaristi. Lastly, I would like to acknowledge everyone mentioned above for their help and impact on my life and education.

ABSTRACT

Curation is the act of selecting, organizing, and presenting content most often guided by professional or expert knowledge. While many popular applications have attempted to emulate this process by turning users into curators, we put an accent on a recommendation system which can leverage multiple data sources to accomplish the curation task. We introduce *QBook*, a recommender that acts as a personal docent by identifying and suggesting books tailored to the various preferences of each individual user. The goal of the designed system is to address several limitations often associated with recommenders in order to provide diverse and personalized book recommendations that can foster trust, effectiveness of the system, and improve the decision making process. *QBook* considers multiple perspectives, from analyzing user reviews, user historical data, and items' metadata, to considering experts' reviews and constantly evolving users' preferences, to enhance the recommendation process, as well as quality and usability of the suggestions. *QBook* pairs each generated suggestion with an explanation that (i) showcases why a particular book was recommended and (ii) helps users decide which items, among the ones recommended, will best suit their individual interests. Empirical studies conducted using the Amazon/LibraryThing benchmark corpus demonstrate the correctness of the proposed methodology and *QBook*'s ability to outperform baseline and state-of-the-art methodologies for book recommendations.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

When visiting a museum, do you have a feeling that each exhibit is trying to tell you a story or do you instead think that every item is just displayed randomly to showcase what the museum holds? The truth is, there is a discipline behind the scenes that studies how to build associations between exhibits and visitors. *Curators* are professionals who know how to best create a display that relates to a specific audience. The process of curation can be used to improve other application areas. One of them is *recommendation systems*. Based on research conducted in understanding recommenders, it is clear that one of the fundamental goals of these systems is to suggest items to present to users that satisfy their individual needs [49, 91]. Recommendation systems aid users in locating items (either products or services) of interest [66]. Regardless of the domain, from shopping websites (e.g., Amazon [2], e-bay [8]), to news related sites (e.g., Yahoo [28], CNN [6]), and hotels or restaurants (e.g., Yelp [29], hotels.com [10]), recommenders have a huge influence on business success and user satisfaction. From a commercial standpoint, existing recommenders enable companies to advertise items by offering them to potential buyers. From a user prospective, these systems enhance the user's experience by assisting them in finding information pertaining to their particular preferences, thus addressing the information overload concerns that web users have to deal with on a daily basis. This

can be done by understanding each individual user, his interests and behaviors, and thus provide him with valuable, appealing and pertinent suggestions. Now, imagine a recommender that could act as your *personal docent* in the process of suggesting which books to read, movies to watch, or even which stocks to invest in. It would get to know and understand you and provide a personalized, curated set of recommendations tailored only to you.

While recommenders based on information retrieval, natural language processing, machine learning and data mining techniques have been studied for the last two decades [33, 43, 53, 60, 69], they are still affected by a number of limitations and have not managed to emulate the curation process. Among these problems, the most common ones and yet to be solved are: cold start, data sparsity, lack of personalization and diversity. Cold start and data sparsity occur when the system is unable to create recommendations due to unavailable historical data for new users or items or the lack of sufficient ratings for most items. Many existing recommendation systems do not provide sufficiently diverse suggestions which limits users' exposure to new, prospective items of interest [101]. This is due to the fact that a common alternative for generating recommendations is to rely primarily on existing community data, thus suggesting the same items to similar users within the community, which can be very vague and impersonal [68]. Some researchers believe that data sparsity is the reason why recommenders cannot be more successful in creating personalized suggestions and linking items to users [96]. Even when recommendation systems accumulate and use large amounts of knowledge extracted from users' historical data, considering only this type of data is insufficient, because the real meaning of these ratings cannot always be fully understood (e.g., by giving 5 starts to a restaurant, what did a user like the most? Was it the food, ambient, service, or everything?).

Furthermore, historical data can be very noisy, which limits the accuracy of recommendation systems due to its natural variability, i.e., the meaning or intention behind the same numerical rating can vary for different users [35]. Collecting new information from users could help solve this problem. However, it has to be done without imposing extra effort on the users [96]. In an attempt to more adequately tailor recommendations towards each user, researchers take advantage of various data sources, such as reviews, metadata and personal tags, to learn user preferences [35, 40]. The pursuit of further personalized strategies continues to be of importance to improve both the performance and perceived usability of recommendation systems. Another challenge faced by recommenders is to get users to trust them, as they still operate as "black boxes" without providing additional information to the users to increase their satisfaction and understanding of the system [70]. To increase their perceived trust on the corresponding recommender, many users need to see more details related to suggestions than just "dry" recommendations [75]. To address this issue, recent research works focus on explaining the generated recommendations [66]. Unfortunately, justifying reasons why an item has been suggested to a user is not an easy task. Thanks to the growth of online sites that archive reviews, researchers have suggested leveraging this data source to enhance the recommendation process [135]. Nonetheless, a better understanding of the aspects or features of a particular item that appeal the most to each individual user (e.g., *price* in the case of restaurants or *genre* in the case of a movie), which can inform the recommendation justification process, is yet to be accomplished.

Recommendation is not the only area that has a need for a strategy that solves each of the described issues, since as stated in [35] areas such as: *"e-commerce, search, Internet music and video, gaming or even online dating make use of similar techniques*

to mine large volumes of data to better match their users' needs in a personalized fashion". A simple solution would be to let the user deal with information overload and choose among multiple options and decide which ones suit him the best at any given moment. This can be accomplished by turning to a recommender capable of understanding the user as a whole, not only based on the ratings the user provided for movies he has seen or recently bookmarked books. This recommender could provide suggestions suitable for different occasions and areas of interest of a user. A number of newly-developed applications, such as Bundlr [5], Paper.li [20], Pinterest [21], and Storify [23], let the user be the center of the decision making process. Essentially, these applications allow their users to play the role of a curator, who *"selects, organizes, and presents content typically using professional or expert knowledge"* [55] in the same fashion as museum curators, who organize and propose exhibitions for an art show. However, as stated in [46], curation is *"more than reflection of a persons interests. It is scholarship, framing ideas, telling stories showing the edge that exists between the things curated and the rest of us"*. Even the most knowledgeable art connoisseurs do not know of every single piece of art work that exists. Likewise, ordinary visitors do not go to museums to see the exhibits they are familiar with over and over again. Let's be honest, you can only see Leonardo da Vinci's Mona Lisa or Monet's The Water Lily Pond so many times. Eventually, visitors crave for something new. Most of the time, curators can bring new and unexpected pieces into the exhibit which are still relevant to the show. However, curators must be able to fit these new artworks within the story the exhibit is trying to tell and provide information in an unpretentious and natural manner. Therefore, it is natural to think that, given the ability of a recommender to consider and examine large amounts of data from multiple perspectives, the system itself can act as a curator.

One domain that could benefit from this type of recommender is *books*. Books, which constitute a billion dollar industry [73], are the most popular reading material among all generations of readers, both for leisure and educational purposes [26]. Hundreds of thousands of books of different types (e.g., paperback and e-books) and styles (e.g., fiction and non-fiction) are published on a yearly basis, giving readers a variety of options to choose from and satisfy their reading needs.

Book recommendation systems, which are meant to enhance the decision making process, can help users by identifying, among the sometimes overwhelming number of diverse books, the ones that best suit their interests and preferences. These recommenders are not exclusively designed to aid *individuals* in their quest for reading materials. They can also improve the decision making process for *libraries* by suggesting what books to buy in order to maximize the use of library resources by their patrons, and *publishing companies* by advising which books to publish in order to maximize revenue. From a *profit-making point of view*, the benefit would be in understanding the influence of reading patterns on deciding what types of books should be published or acquired. While book recommenders have been studied in the past, we observed that most of existing suggestion strategies centered on books are solely based on either historical data [47, 82, 104] or content [40, 61, 101], and thus are affected affected by (some of the) aforementioned issues common to recommendation systems.

We focus our research efforts on the design and development of techniques and methodologies that lead to *QBook*, a Curated Recommendation System. This *inanimate docent* captures different areas of users' interests, not just the most dominant one, and provides a meaningful and personalized "exhibit" of suggestions coupled with explanations, which enable users to decide which suggestion is the most suitable in a

given moment. *QBook* takes the role of the curator upon itself and makes connections between suggested items and the reasons for their selection. Justifying each suggestion does not only increase the user's trust and satisfaction on the system, but also makes it quicker and easier for him to decide and understand how the system works [123]. As users' overall satisfaction with a recommender is related to the perceived quality of its recommendations and explanations [66], users' confidence on *QBook* also increases.

In designing the proposed curated recommender, we explicitly aimed to enhance the recommendation process by addressing popular issues affecting these systems. Examining historical data allow us to capture suitable candidate items for each individual user. Items' metadata brings another rich knowledge resource to explore, which gives us an opportunity to consider potentially relevant items that otherwise might be ignored by solely using a rating patterns approach. *QBook* also considers reviews generated by users to understand which item features each user cares about and their degree of importance. By incorporating feature preferences into the recommendation strategy, we aim to learn more about the user and understand why he could potentially be interested in each suggested item using other users' and experts' opinions[1] on provided feature preferences and overall quality of books. We also analyze users' genre preferences and their change over time. We use this knowledge in the process of curation to enable *QBook* to provide recommendations from different areas of interest based on a prediction strategy that considers user's reading patterns. By considering a larger number of possible books to be suggested and curating them based on users' personal interests and experts' knowledge, *QBook* increases the chances to create a set of significant (i.e., relevant and valuable) book suggestions that become "a personal

[1]For experts' opinion we use book critiques from well-known web sites, including the New York Times [17].

exhibit" for the user. Even if suggesting items that are relevant, but unfamiliar, to a user increases the serendipity of the system, due to lack of sufficient knowledge, the user may not treat these recommendations as something worth exploring. Hence, *QBook* provides a user not only with relevant recommendations, but also justifies the inclusion of each book in the generated exhibit with a corresponding explanation to provide a user with information he is interested to know. In conducting this work we improve research related to recommenders by proposing a strategy that combines well-known traditional approaches with novel preference matching methods into a single recommendation strategy that provides suggestions containing information related to the interests of each user. We believe our approach is a novelty in the area of recommenders since we created a system that simultaneously considers different data points (such as rankings, users' reviews, experts' reviews) for improving the quality and usability of the recommendations, while offering diverse, relevant and high quality suggestions, and corresponding explanations in terms of users' interest, without involvement of a sentiment. With this, *QBook* aims to increase users' trust, effectiveness, persuasiveness and transparency [123]. Furthermore, by providing explanations that include information about features users care about, we can help users make better and faster decisions in choosing recommended books which increases effectiveness and efficiency [123]. In fact, users do not need to do additional research to find information about book traits that correlate with their individual preferences and needs.

With the amount of information and data available these days, curated recommendation systems can focus on collecting knowledge about each individual user and becoming his personal docent. The newly-developed *QBook* is definitely able to deal with the information overload problem, as traditional recommenders do, but more

importantly, *QBook* is able to create personal exhibits and eliminate all irrelevant informations and documents. Wouldn't it be exciting to have a museum set-up only for yourself? Of course!, but that is not an easy task. The same way you cannot just come up with the words to write a poem by randomly picking them out of a page, you cannot create personalized recommendations that exceed users' expectations. The question is: Can you make a poem by curating a set of all possible words to get

Figure 1.1: **Blackout poetry**

a perfectly matched subset (Figure 1.1)? Can you curate a perfect recommendation from the sea of information and available resources? Challenge accepted.

To better understand the magnitude of the limitations affecting existing recommenders, we describe background work in Chapter 2. We detail how we completed the challenge in Chapter 3, while the discussion of the evaluation we conducted to verify the correctness of our solution is provided in Chapter 4. Conclusions and directions for future work are described in Chapter 5.

CHAPTER 2

RELATED WORK

Recommendation systems is an area of research commonly associated with Information Retrieval that has greatly evolved over the last two decades [107]. The area focuses on the design and development of tools and techniques that identify items potentially of interest to users [108]. Machine learning, information retrieval, natural language processing, and probabilistic models have been adopted for developing systems that recommend (web) documents [69], songs [51], videos [84], and movies [78], to name a few. Examples of popular recommenders we use on a daily basis can be found in different domains, including music (Spotify [22], Pandora [19]), movies (Netflix [16]), recipes (Pinterest [21]), people to follow (Twitter [25], Facebook [9]) and videos (Youtube[30]).

2.1 Recommendation Strategies: Groundwork and Obstacles

Among many strategies adopted nowadays to generate recommendations, two of the most well-known and traditional are content-based filtering [61, 108] and collaborative-based filtering [104, 108, 112]. Mentioned techniques consider different input data to identify relevant items to be suggested to users. Content-based strategies analyze metadata, such as textual comments appended to online news articles [40] or tags generated by users in social sites [40], related to items previously rated by a user

to find similar ones to suggest [108]. Collaborative filtering methods, on the other hand, rely on historical data generated by users (e.g. rated items) as well as items' ratings assigned by other users who have a similar rating history to predict the rating of a new, unseen item [108]. Collaborative filtering methods can be classified into memory-based and neighborhood-based [31, 47]. The former predicts the rating of an item for a given user based on a rating collection of items previously rated by similar users [82], whereas the latter learns a model (such as a latent factor or neighborhood model [78]) from a collection of data consisted of rated items, which is used to predict the rating of an item for a given user [31, 113]. To further improve performance of recommendation systems and overcome known limitations, hybrid strategies [49], which usually combine traditional strategies like content-based and collaborative-filtering [92, 39], or incorporate other available information, such as demographic data [38] or knowledge-based filters [125], have been studied. For the past few years researchers have focused their efforts on matrix factorization algorithm (SVD)[108], developing state-of-the-art systems based on latent-factor models. SVD models map users and items to a joint latent factor space, which will generate suggestions for each individual user describing both products' and users' factors obtained by user's feedback. Recently, algorithms based on SVD became very popular (especially in combination with other available user data making another form of hybrid recommender [34]) due to their accuracy and scalability [80, 106, 108].

As mentioned in the Chapter 1, regardless of the strategy considered, there are limitations and common issues affecting recommendation systems; including cold start, data sparsity, lack of personalization and diversity. Other shortcomings of recommendation systems include (i) restricting a user to access items similar to those already rated, since content-based recommendations systems can recommend only

items scoring highly against the profile of a user, and (ii) scalability, since algorithms based on collaborative filtering tend to be time consuming and scale poorly.

2.2 Book Recommenders

Based on statistics that showcase the results of a survey conducted to investigate the reading habits of Americans, as shown in Figure 2.1, books take an important place in human lives and culture. Consequentially, books constitute a very popular and large domain containing billions of titles that continuously increase. Therefore, it is not surprising that there are a number of recommendation systems that have been tailored specifically for generating suggestions that will help users in selecting the most suitable books to read [81, 100, 101, 103, 132].

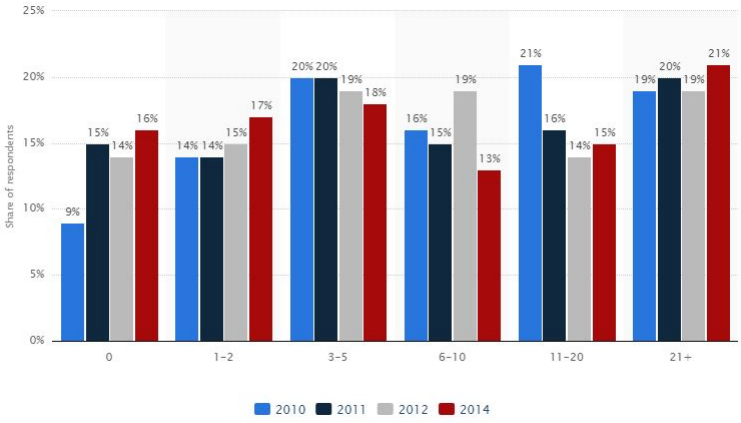

Figure 2.1: **Number of books read by American adults per year 2010-2014 [4]**

The most popular recommenders in this domain, i.e., Amazon [81], suggest books based on the purchasing patterns among users. Yang et al. [132] rely on a collaborative filtering approach with ranking, which considers users' preferences on library resources extracted from their access logs to recommend library materials. This

approach overcomes the problem that arises due to the sparseness of explicit users' ratings, i.e., lack of initial information to perform the recommendation task. Park and Chang [100] create a user-profile based on individual and group behavior, such as clicks and shopping habits, compute the Euclidean distance between a profile and each product profile, and recommend the products for which their Euclidean distances are closest to the user profile. Givon and Lavrenko [67] combine collaborative filtering and social tags to capture the content of books for recommendation. Sieg et al. [117] rely on the standard user-based collaborative filtering framework together with a domain ontology to capture the topics of interest to a user. However, the hybrid-based book recommenders in [67, 117, 132] require (i) historical data on the users in the form of ratings, which may not always be available, or (ii) an ontology, which can be labor-intensive and time-consuming to construct.

Some book recommenders are tailored towards specific group of users, as it is the case with Rabbit [101] and K3Rec [103], targeting K-12 and K-3 students, respectively. By emulating the readers' advisory service popular at libraries, the authors in [101] describe the process of recommending suitable books for children, by identifying the topics, contents, and literary elements appealing to each individual user. K3Rec [103] uses information about grade levels, contents, illustrations, and topics together with length and writing style, to generate relevant suggestions for emergent readers. Similar to *QBook*, Garrido and Illarri [64] rely on content-based data for making book suggestions. Their proposed TMR [64] examines items' descriptions and reviews and takes advantage of lexical and semantic resources to infer users' preferences. However, TMR can only work if textual descriptions and reviews of the books are available, unlike *QBook* for which descriptions and reviews are 2 of the multiple data points considered in the recommendation process. To enhance the book recommendation

process, the authors on [116] discuss the use of an adjusted Eucledian distance measure which can be used to better determine the degree of similarity among users and in turn improve collaborative-filtering-based strategies that depend on that information to generate suggestions. This strategy focuses on the dimensionality of vectors that capture user preferences, as opposed to their direction (as cosine and correlation-based strategies do). The authors in [42] present a strategy based on graph analysis and PageRank to perform the recommendation task by exploiting both clicks and purchase patterns as well as book metadata, including a title and a description. However, their recommender is constrained to the existence of a priori pairwise similarity between items, e.g. "similar products", which might not always be available and this is not a requirement for *QBook*. The authors in [122] highlight the importance of books recommenders as library services (besides the more traditional e-commerce applications), and thus propose a fuzzy analytical hierarchy process based on a priori rules mining that depends upon the existence of book-loan histories. The empirical analysis presented in [122] is based on a limited and private dataset, which difficults the task of verifying its applicability on large-scale benchmark datasets. More importantly, the proposed strategy is contingent on book-loan historical information that due to privacy concerns libraries rarely, if at all, make available.

To address some of the aforementioned limitations, existing book recommenders use different techniques which allow them to learn and explore relationships between users, items, as well as users and items, with the goal of generating more appealing recommendations for their users [108]. In the following subsections, we describe different data sources and techniques considered to overcome these obstacles.

2.2.1 Extracting Information from Reviews

The performance of recommendation systems and their ability to satisfy users' needs can be improved by taking advantage of different users' generated data to better identify user preferences in an attempt to further personalize recommendations [66]. Many relevant information sources, based on user-generated data, users' reviews are the ones that spark the most interest among researchers for recommendation purposes [34, 63, 135]. A product review describes products' actual features, while specifically for the book domain, a book review is a form of criticism where the work is analyzed based on its content, style, and merit [115]. Numerous approaches have been developed to identify and extract either features, opinions, or feature-opinion pairs from reviews based on bootstrapping, natural language processing, machine learning, extraction rules, latent semantic analysis, statistical analysis, and information retrieval [99]. These approaches can be used to infer users' preferences, which can be considered as part of the recommendation process and thus increase users' satisfaction with provided suggestions.

The authors in [34] propose a number of review models that are used to improve the performance of a prediction rating model, whereas the authors in other strategies for generating recommendation explanations focus on analyzing the sentiment, i.e., positive or negative, of feature descriptions [135] and understanding informal language used in reviews in order to improve rating prediction [63]. Ling et. al. [83] combine content-based with collaborative filtering, while taking advantage of both ratings and reviews by applying topic modeling techniques on the review text and align the topics with ratings to improve prediction accuracy. Instead, we take advantage of reviews written by each user to learn a set of features the user cares about.

2.2.2 Exploring Expert Opinions

To further address issues related to data sparsity and cold-start, researchers have lately turned towards using experts opinions [36, 67]. Among the few existing research works, the one conducted by Amatrain et. al. [36] uses experts ratings within a collaborative filtering approach to overcome the weaknesses of this traditional strategy. The proposed strategy finds experts that are similar to a specific group of users and takes advantage of their opinions as relevant in creating recommendations for a corresponding population [36]. While the study in [36] considers professional raters in a given domain as experts, the authors in [67] consider that users with more experience gain a certain level of expertise. McAuley et. al. [67] believe that suggestion generation process should be influenced by the amount of knowledge and experience users have in a specific domain.

QBook's strategy for extracting experts' opinions is closer to the one described in [36], in the sense that professional book reviewers are treated as experts. However, unlike the aforementioned works, *QBook* examines textual reviews written by experts to capture their sentiment (e.g., positive or negative) and uses it to determine the quality of books to be suggested.

2.2.3 Time Component as Part of the Recommendation Process

To better capture change in user's preferences, including a time dimension has became of interest to the research community [50, 56, 79, 131]. Even though many studies considered time as a global component shared by all users [118, 119], a recent study focuses on individual users with a goal to better personalize suggestions [79, 131]. Xiang et. al. [131] proposed an approach which considers and models the changes in

short and long term preferences of users' over time, as well as a new recommendation strategy and extended personalized Random Walk with a time component [131]. In [79], the authors created a matrix factorization model that includes time factors for each individual user. Similarly, the authors in [56] included a time component to improve collaborative filtering approach, with an idea to weight more recently generated ratings versus the ones that represent older data.

QBook considers a change of book genre over time, and its influence on the recommendation process. A considerable number of studies examine the importance of book genre on readers' activity [32, 37, 97]. However, to the best of our knowledge, research based on past genre distribution and time series analysis to influence the recommendation process has not been conducted. Genre was used as a part of a cross-domain collaborative filtering approach to recommend books based on users' genre preferences [111], without considering time component. You et al. [134] propose a clustering method based on users' ratings and genre interests extracted from social networks to solve the cold-start problem affecting collaborative filtering approaches [114]. Unlike the proposed method, *QBook* uses time series to predict the genres most likely of interest to each individual user.

2.2.4 Explaining Recommendations

A powerful way to build a successful relationship between users and the recommendation system is by providing more information on how the system works and why items are recommended to each user [123]. An attempt to address the lack of personalization and increase users' acceptance of the system involves including explanations that justify the generated suggestions. Unfortunately, explanations on suggested items is a less explored research area [108]. Recent research works focus on

explaining the generated recommendations and how different types of explanations influence users while making decisions [66]. Among the most common strategies to generate explanations we should highlight those based on exploring previous activity of a user [44], information collected from user reviews [135], content-based tag cloud explanations (which were shown to have high level of acceptance within users) [66], and the ones developed specifically for traditional recommenders, system as in the case of explanation generation for the collaborative filtering method [70]. Scientists believe explanations, which reveal reasoning and data behind the process of a recommendation and offer information about suggested items, has been known to provide transparency and enhance trust on the system [123]. Many researchers consider sentiment-based explanations as more effective, trustworthy, and persuasive than the ones that capture relationship between previous activity of the user and suggested items [52]. However, unlike existing explanation-generation strategies, our explanation generation process does not include sentiment in order to make *QBook* look honest in the eyes of users. In doing so, *QBook* is designed to provide explanations which contain other users' and experts' (objective) opinions on features that are of a specific interest for each individual user, regardless of the polarity of the opinions.

2.3 Curation in the Recommendation Process

Recommendation systems can leverage community data to emulate the curation process in the same manner art museums "outsource" the curations of their exhibits to the public by processing user-generated data create the art shows [62]. Unfortunately, few research works focus on simulating such a task [76, 77, 109]. In [76], the authors

discussed the development of an application that learns from users' interactions with the system while they swipe through the news and like them. In this research, the authors use social networks and users' browsing history to create and recommend crowd curated content, but using users as curators. The work conducted by Saaya et al. [109], on the other hand, relies on a content-based strategy that considers information authors collect from users' profiles, which are then managed by the users themselves. The most recent study conducted by Kislyuk at al. [77], combines a user-based curation method along with a traditional collaborative filtering strategy to improve Pinterest's recommendation process. The aforementioned alternatives simply let users organize suggested content based on their personal preferences, since all three studies treat users as curators and based on their interests suggest them new items. However, no recommendation system takes the role of the curator upon itself. We take a different approach and allow the system to take the curator role using existing user and item data.

Unlike existing book recommendation strategies, *QBook* simultaneously considers multiple data sources, such as books metadata, users' ratings and reviews, experts' reviews, and genre distribution over time, to create curated recommendations tailored to individual users. We discuss the curation of books to be recommended to each individual user in Chapter 3.

CHAPTER 3

THE ANATOMY OF QBOOK

In this chapter we detail the design of *QBook*, which is based on a novel methodology that relies on a number of data sources. *QBook* generates book suggestions by presenting each user with a set of curated books, which are tailored to the varied interest of the user and paired with the corresponding explanations. In designing *QBook* we follow the steps illustrated in Figure 3.1. These steps include (i) selecting candidate books to be recommended, (ii) exploring reviews to gain knowledge about users and books, (iii) considering experts' opinions to determine the quality of books, (iv) predicting future genres of interest to a user, (v) curating possible suggestions by simultaneously considering multiple data points, and (vi) creating explanations for the suggested items. We discuss below the design methodology of each individual step of *QBook*, which addresses a particular research problem on its own: Can item metadata complement the lack of historical data (and vice versa)? How can a time component influence recommendation systems?, Can experts' opinions align with readers' preferences?, Are users' features of interests a valuable asset to a recommendation process?, How does curation work as a part of a recommendation process? Can personalized explanation aid user in selecting the most relevant suggestions from recommended ones?

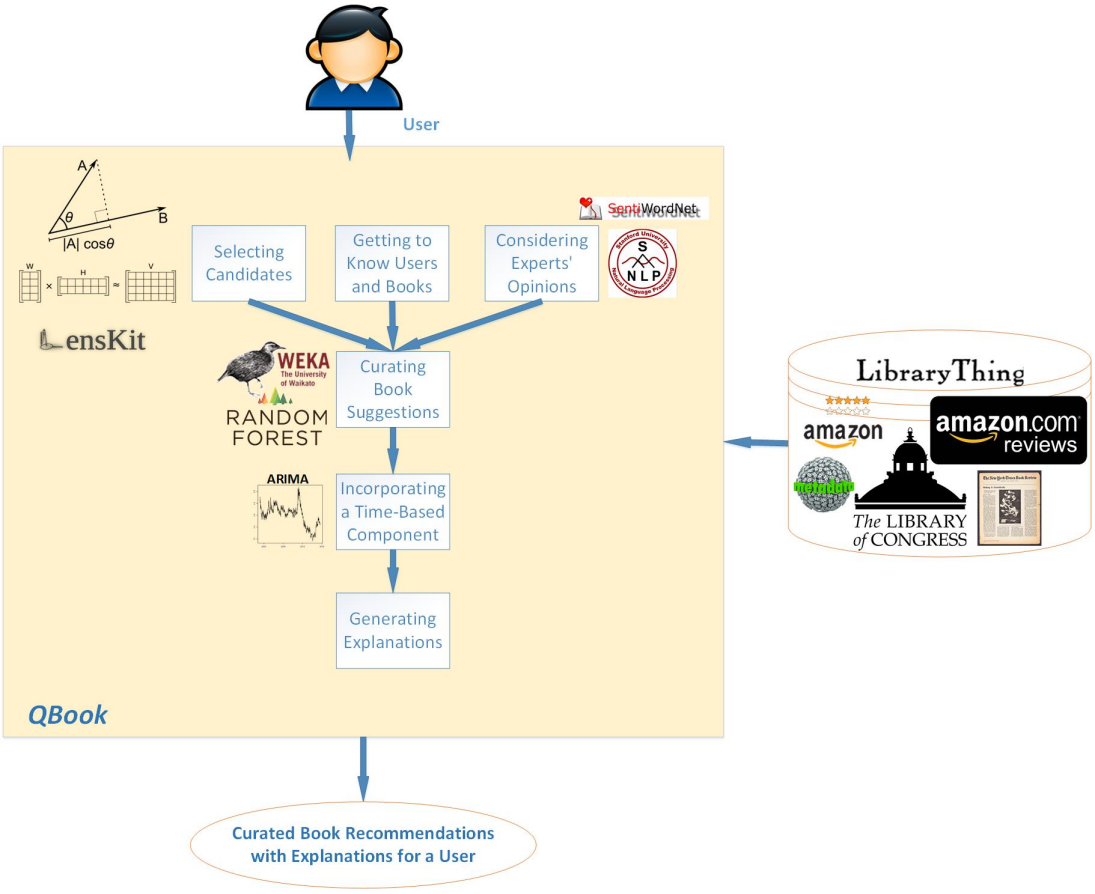

Figure 3.1: **Overview of *QBook***

3.1 Selecting Candidate Books

To initiate the recommendation process, *QBook* needs to examine archived books to identify those potentially of interest to a user U. Unfortunately, due to their volume, *QBook* cannot examine every book to determine which ones are potential candidates to be recommended. For this reason, *QBook* applies filtering strategies to identify a manageable set of books to be further examined and curated. Besides taking advantage of users' historical data (i.e., rated books) to discover latent features underlying the interactions between users and books, *QBook* locates books similar to

the ones users liked in the past by applying a content-similarity strategy.

3.1.1 Candidate Recommendations Based on Matrix Factorization

Matrix factorization [80] is used in some of the most effective latent factor models to learn user and item characteristics in order to predict users' ratings for unseen items [80, 108]. Among many effective latent factor models, strategies based on Singular Value Decomposition (SVD) are very popular [108]. These models have the ability to learn and explore relationships between users, items, as well as users and items [108].

The estimation strategy based on matrix factorization considered by *QBook* represents each item i and user U as n-dimensional vectors of the form q_i and $p_u \in R^n$, where the vector components of q_i represent the degree to which each latent factor applies to the corresponding item i and the vector components of p_u show the degree of interest of U on items [108]. The rating predicted for U on i ($r_{U,i}$) is calculated using Equation 3.1.

$$r_{U,i} = \mu + b_i + b_u + q_i^T p_u \tag{3.1}$$

where $q_i^T p_u$ represents 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 μ, b_i, and b_u are parameters that denote the overall average rating and the observed deviations of U and i from the average, respectively. These parameters are estimated using Equation 3.2, which aims to minimize the regularized squared error on a set of known ratings.

$$min_{b*,p*,q*} \sum_{U,i \in K} (r_{U,i} - \mu - b_i - b_u - q_i^T p_u)^2 + \lambda(\|q_i\|^2 + \|p_u\|^2 + b_i^2 + b_u^2) \tag{3.2}$$

where μ, b_i, b_u, q_i, p_u, $q_i^T p_u$, and $r_{U,i}$ are as defined in Equation 3.1, K is a set

of training instances, and λ is a constant that controls the extent of regularization determined by using cross-validation. Minimization is performed by gradient decent [58].

Matrix factorization methods provide greater prediction accuracy, memory efficient compact models, and easier implementation compared to other collaborative filtering strategies [108], therefore we deemed it as an adequate approach for generating relevant candidate items for the recommendation process. *QBook* depends upon the LensKit's [13] implementation of FunkSVD algorithm for book candidate generation based on generated score. *QBook* only treats as candidate books for U those for which their corresponding $r_{U,i}$ is above 3 (on Likert-type scale). In doing so, *QBook* ensures that items that are known to be not likely appealing to U are not further considered during the curation process.

3.1.2 Candidate Recommendation Based on Content-Based Filtering

Content-based filtering methodologies create suggestions based on a comparison between items' characteristics and users' preferences. Available content representations (e.g., metadata) are used to describe items, as well as users' profiles based on items users favored in the past [108]. Among existing content-based models, the *Vector Space Model* [54] is one of the most popular ones, which represents items and user profiles as vectors of weighted terms. To estimate the degree of similarity between P, the vector representation of U's profile, and I, the vector representation of i content profile, the model generates a score based on the cosine correlation similarity measure [54]. This score, computed as in Equation 3.3, reflects how likely a given item is to be relevant to the corresponding user.

$$sim(I, P) = \frac{I \cdot P}{\|I\| \cdot \|P\|} \tag{3.3}$$

where $\cdot$ indicates vector dot product, while $\|I\|$ and $\|P\|$ are norms of vectors I and P, respectively. Normalizing the result of the dot product of the item and profile vectors by considering their lengths prevents larger vectors from producing higher scores only because they have a higher chance of containing similar terms.

In this approach, the weight of the terms considered in the vector representations w_i, where each w_i showcases the importance of each term (t_i) in capturing the content of i or the preference profile of U based on a popular $TF - IDF$ weighting scheme [54] as shown in Equation 3.4.

$$w_i = tf_i * idf_i \tag{3.4}$$

where *term frequency*, tf_i, reflects the importance of t_i in a corresponding item description, D (as shown in Equation 3.5) and *inverse term frequency*, idf_i, reflects the importance of t_i in a set of all item descriptions, C (calculated as shown in Equation 3.6).

$$tf_i = \frac{f_i}{\sum_{j=1}^{t} f_i} \tag{3.5}$$

where f_i is the number of occurrences of t_i in D.

$$idf_i = log(\frac{n}{df_i}) \tag{3.6}$$

where n is the size of C and df_i the number of item descriptions in C in which t_i appears at least once. w_i increases proportionally to the number of times t_i appears in D, but is offset by the frequency of t_i in C, which helps to adjust for the fact that some words appear more frequently in general.

To determine the degree of appeal of each book, b, for U, *QBook* relies on

Figure 3.2: **User-generated tags for Hamlet as extracted from LibraryThing [15]**

Lenskit's implementation of the content-based algorithm [7] to represent b's vector, B, and U's profile vector P. *QBook* takes advantage of user-generated tags from the LibraryThing [15] website to capture the content of books and select ones similar to those for which U has already expressed a preference. Therefore, b's description is represented as sets of tags for a book, while a set of all descriptions is a set of all tags of all books. We choose LibraryThing since it represents the most popular social application for cataloging books. As shown in Figure 3.2, tags are a simple way to categorize books based on users' opinions. Anything can be a tag: words or phrases separated by commas. Therefore, one person can describe $Hamlet$ using $classic$ as a tag while another creates tags such as $fiction$ or $freshman$ for the same book. Tags, which capture books' content from users' perspectives, are useful for searching and sorting (e.g., to list all books for the freshman year of college). Because of the large number of available tags, which amount to 129,320,610 tags generated by more than 2 million members[1], and their mentioned characteristics, it is natural for *QBook* to depend on them for selecting books that may be of interest to users.

To ensure candidate books are potentially appealing to U in terms of their content, *QBook* applies the same filtering approach discussed in Section 3.1.2 and selects only candidate books for U with a respective $CBrank(B, P)$ score (as shown in Equation

[1]According to the Zeitgeist Overview (http://www.librarything.com/zeitgeist)

3.7) above 3.

$$CBrank(B, P) = sim(B, P) * 4 + 1 \tag{3.7}$$

where B, P and $sim(B, P)$ are defined in Equation 3.3 and $CBrank(B, P)$ represents a normalized $sim(B, P)$ value to a 1-5 range.

3.1.3 How does QBook Generate Candidates?

The final candidate set of books (CB) to be considered during the recommendation process contains highly rated books that address topics of interest for U. The two strategies considered for candidate selection (i.e., matrix factorization and content-based filtering) complement each other and ensure diversity among candidate books. While the first one relies on examining users' rating patterns, the second one focuses on books characteristics and does not require user-generated data. Moreover, by considering rating patterns and content, serendipity and novelty of the recommendations also increase, because users are exposed to a variety of books to chose from. *QBook* also addresses data sparsity in this step, since some books are missing assigned ratings or user-generated tags or other metadata. Even if books do not have (sufficient) ratings assigned to them, they might still have tag descriptions that can help the recommender determine if they are indeed potentially of interest to a user and vice versa (books with ratings might be selected to pick up other patterns that reflect appeal to a user regardless of the content descriptions). By doing this, *QBook* overcomes limitations of individual strategies used and addresses data sparsity issue.

3.2 Getting to Know Users and Books

QBook aims to provide U with a set of appealing suggestions that are personalized based on the information he values. *QBook* examines reviews written by U and identifies the set of literary elements (features) that he cares the most about[2]. To identify how a candidate book, b (generated in Section 3.1) addresses U's features of interest, *QBook* finds the similarity between features U talked about in his reviews and the ones referenced in book reviews made by other users. To do so, *QBook* needs to capture (i) the most *important* features for each user and (ii) the most *popular* features for each candidate book.

3.2.1 User's Interest

To identify features of interest to U, *QBook* performs semantic analysis on reviews and considers the frequency of occurrence of specific terms U employs in his reviews. By using the set of literary elements described in [101], *QBook* explores diverse features of books such as their *characters*, the *pace* in which the stories unfold, their overall *storyline*, the *tone* of their narrative, the *writing style* of their stories and their *frame*. As defined in [101], each literary element (feature F_i) is associated to with a set of terms used to describe that element, since different words can be used to express similar book elements. A list of sampled literary elements and their corresponding related terms considered by *QBook* is shown in Table 3.1[3].

QBook analyzes each review previously written by U to capture which features he talks about and cares about the most. Whenever *QBook* recognizes a term

[2]If U does not have reviews, then the most popular user features are treated a U's features of importance.

[3]The complete list of literary terms and their associated sets of terms is included in Appendix A.

Table 3.1: **Sample of terms associated with each literary feature used by *QBook***

Literary Element	**Sample of Related Terms**
characters	stereotypes, detailed, distant, dramatic, eccentrice, vocative
pace	fast, slow, leisurely, breakneck, compelling, deliberate
storyline	action-oriented, character-centered, cinematic, complex, conclusive
tone	happy, light, uplifting, dark, ironic, funny, evocative, serious
writing style	austere, candid, classic, colourful, complex, concise
frame	descriptive, minimal, bleak, light-hearted, fun, bitter-sweet

related to a specific literary feature, it assumes that U talks about that feature. For example, if a review contains the sentence *"Some of the questions at the end of the chapter are very complex."*, *QBook* identifies *writing style* as one of the features mentioned by U. Moreover, it is necessary to take into consideration the frequency with which U refers to these features in his reviews. Consequently, for each feature mentioned by U, denoted F_{Un} (where n is a number of mentioned features), *QBook* computes its associated overall frequency of occurrence ($AF_{F_{Un}}$), using Equation 3.8. To ensure that a feature frequently mentioned in a single review, or a few ones, does not overshadow the importance of other features more prominently mentioned in a number of U's reviews, *QBook* normalizes the overall frequency of occurrence of F_{Un} based on the number of reviews generated by U.

$$AF_{F_{Un}} = \frac{\sum_{x \in F_{Un}} freq(x)}{|RU|} \tag{3.8}$$

where RU is the set of all reviews generated by U, $|RU|$ is the total number of reviews made by U, x is a term from a corresponding set of terms for F_{Un}, and $freq(x)$ is the frequency of occurrence of x in RU.

By recognizing the features addressed in U's reviews and calculating their associ-

ated overall frequencies of occurrence, *QBook* acquires knowledge about the interests of U and identifies which literary elements he cares the most about.

3.2.2 Most-Discussed Book Features

QBook also needs a deeper understanding of the literary elements (features) that are often used in reviews to describe b. *QBook* takes advantage of reviews available for b and examines them to identify the features most-commonly addressed in reviews pertaining to b.

The most-discussed features about b (F_{bm}, where m is a number of detected features) are identified by following the process defined for identifying features of interest to U, as described in Section 3.2.1. *QBook* uses the same list of literary elements with associated sets of terms to extract features from all the reviews written for b. *QBook* also uses Equation 3.8 to compute the overall frequency of each book feature, F_{bm}, denoted $AF_{F_{bm}}$, which captures its relative degree of importance with respect to other features discussed in reviews pertaining to b. In this case, the normalization factor is the total number of reviews made for b.

With this step *QBook* gains knowledge related b based on subjective opinions of all users who read b. By capturing their opinions, *QBook* identifies the most discussed literary elements for b, and thus determines if U would be interested in reading b.

3.2.3 How does QBook Incorporate Users' Interests?

QBook uses knowledge about U and each book in CB collected from the processes discussed in Section 3.2.1 and 3.2.2 to determine if b is a likely candidate of interest for U based on their respective reviews. *QBook* learns which book traits U most frequently mentions in his reviews and applies that knowledge to predict which book

would be preferred by U. *QBook* leverages U's preferences in the recommendation process by calculating the degree of similarity between U's feature preferences and b's most-discussed features. For this reason, *QBook* generates n-dimensional vector representations of U and b based on the aforementioned features, i.e., $\vec{UV}$ and $\vec{BV}$, respectively. In this step, *QBook* explores the feature resemblance with similarity measure, which is defined using the VSM model (as defined in Section 3.1.2) and computed using Equation 3.9.

$$Sim(U,b) = \frac{\vec{UV} \cdot \vec{BV}}{\left\|\vec{UV}\right\| \times \left\|\vec{BV}\right\|} \tag{3.9}$$

where $\vec{UV} = < AF_{F_{U1}}, AF_{F_{U2}}, ..., AF_{F_{Un}} >$ and $\vec{BV} = < AF_{F_{b1}}, AF_{F_{b2}}, ..., AF_{F_{bm}} >$, n and m are numbers of distinct features describing U and b, respectively, $AF_{F_{Un}}$ and $AF_{F_{bm}}$ are degrees of importance for F_{Un} and F_{bm} as defined in Equation 3.8.

By using the $Sim(U,b)$ score as one of the data points considered during the recommendation process, *QBook* triggers the generation of personalized books suggestions, since it captures all features of interests for U and compares them with all the most-discussed features of $b \in CB$ to determine how likely b is relevant to U.

3.3 Considering Experts' Opinions

To further analyze $b \in CB$, *QBook* takes advantage of experts' reviews. The goal of this step is to learn unbiased and objective opinions from experts to enable *QBook* to incorporate another data point in the recommendation process and increase its accuracy and U's satisfaction by providing high quality books. *QBook* explores publicly available book critiques to determine experts' opinions on candidate books

by performing semantic analysis to examine which books experts valued more.

To capture sentiment at a word and sentence level, *QBook* examines expert reviews from two different perspectives and takes advantage of popular sentiment analysis tools for natural language processing (NLP)[53]: CoreNLP [88] and SentiWordNet [59].

3.3.1 Sentiment Analysis at Word Level

SentiWordNet [59] is a lexical resource for opinion mining that is used for assigning one of three sentiment scores, positive, negative, objective, to each synset[4] of WordNet [27], as shown on Figure 3.3.

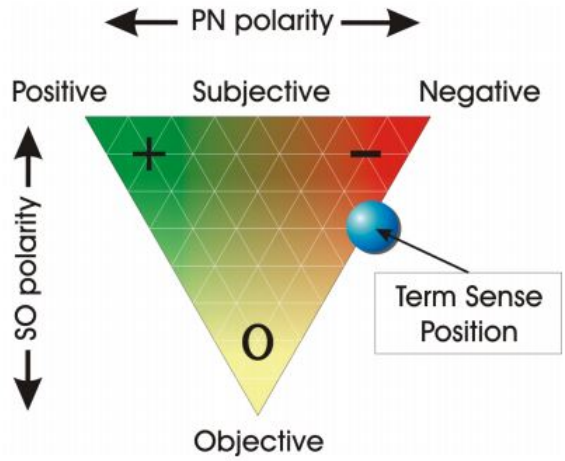

Figure 3.3: **The graphical representation adopted by SentiWordNet for representing the opinion-related properties of a term sense [59].**

WordNet is a large lexical database of English words [27]. In order to use SentiWordNet, each sentence must be tokenized and each token must be assigned a part-of-speech (POS) tag. The POS tag can be: *noun*, *verb*, *adjective* and *adverb*. Since a word can have multiple meanings, a word sense disambiguation process is

[4]Synsets represent a sets of cognitive synonyms expressing a distinct concept.

needed. SentiWordNet uses WordNet, which provides various senses for a word and each sense's definition. For example, for the word *good* as a noun, WordNet contains senses with corresponding definitions as shown in Table 3.2.

Table 3.2: **WordNet senses and definitions for the word *good***

Sense	**Description**
good.n.01	benefit
good.n.02	moral excellence or admirableness
good.n.03	that which is pleasing or valuable or useful
commodity.n.01	articles of commerce

To determine the sentiment of each word from an expert review (ER), SentiWordNet checks all possible meanings of the word and selects the one whose definition most closely matches the context of the word in the ER.

As shown in Equation 3.10, while using SentiWordNet, $QBook$ determines ER's overall sentiment by the calculating an average score ($sentiWNr(ER)$) based on the sentiment of each word included in ER.

$$sentiWNr(ER) = \frac{\sum_{j=1}^{m} sentiWN(x_j))}{m} \tag{3.10}$$

where x_j is the j^{th} word in ER, m represents the number of words in ER, and $sentiWN(x_j)$ is a sentiment score of x_j. The range of $sentiWNr(ER)$ is -1 to 1, where 1 is considered very positive, -1 very negative, and 0 neutral. For example, "important" is an adverb with sentiment score *0.5*, and "horror" is recognized as a noun with sentiments core *-0.159*.

Based on a conducted empirical analysis[5] and study described in [98] where the authors note that "the end of the document where closing remarks would reflect

[5]For purpose of this study we analyzed critiques for 85 different randomly books from NRP website [18].

the general author view", we noticed that reviewers give their overall thoughts in the last sentence of their review. For this reason, $QBook$ also analyses the sentiment of the last sentence to ensure the real tone of review is captured. This score, denoted $sentiWNs(ER)$, is also considered by $QBook$ in the curation process. $sentiWNs(ER)$ is computed as in Equation 3.10, but only on ER's last sentence.

3.3.2 Sentiment Analysis at Sentence Level

$SentiWordNet$ and some of the other popular sentiment prediction systems look only at words in isolation to determine their sentiment and then use them to classify the sentiment of the whole text. In some cases, the polarity of a word on its own does not properly capture the intended polarity of a sentence. With that in mind, $QBook$ is designed to also consider the polarity of sentences in their entirety. In doing so, $Qbook$ uses Stanford $CoreNLP$ [88] builds up a representation of whole sentences based on their structure which is why $QBook$ is using incorporating $CoreNLP$ to estimate the sentiment of the objective opinions of experts in their reviews. $CoreNLP$ is a java natural language analysis library that contains a set of tools that can provide: the base forms of words, their parts of speech, if they are entities (e.g., names of companies or people), normalized forms of numeric quantities and dates, and determine the structure of sentences in terms of phrases and word dependencies and their sentiment, etc. It contains models for performing text analysis on a number of different languages, such as English, Chinese, French, German, and Spanish. Its sentiment analysis tool uses deep learning techniques trained on 215,000 phrases extracted from 12,000 sentences. $QBook$ uses the library to compute the sentiment of each sentence of a review based on the composition of word meanings in the longer phrase. For example, $funny$ and $witty$ are captured as positive, but the following

sentence has a negative connotation:

This movie was actually neither that funny, nor super witty.

QBook takes advantage of CoreNLP by using the provided parser to extract each sentence, s_j, from ER and calculates a sentiment score, $coreNLP(s_j)$, for each sentence in ER. These scores are combined into a single score, $coreNLPr(ER)$ that captures the overall sentiment of ER based on the sentiment of its individual sentences, as shown in Equation 3.11.

$$coreNLPr(ER) = \frac{\sum_{j=1}^{p} coreNLP(s_j))}{p} \tag{3.11}$$

where ER represents a review, p is a number of sentences in a ER, s_j is j^{th} sentence in ER and $coreNLP(s_j)$ is a sentiment score of s_j. The estimation of the review's sentiment score is done by averaging the scores of the individual sentences in the whole review. As previously stated, the last sentence carries an important weight in terms of the overall sentiment of the review, thus *QBook* also considers the sentiment of the last sentence ($coreNLPs(ER)$). This score is calculated using Equation 3.11 but on the last sentence.

$coreNLPr(ER)$ and $coreNLPs(ER)$ scores are in range of 0 (very negative) to 1 (very positive) with 0.5 being neutral, 0.25 negative and 0.75 positive.

3.3.3 How does QBook Quantify Experts' Opinions?

QBook applies the two aforementioned complementary strategies to several expert reviews to capture an unbiased opinion on the quality of a book. The four generated data points (from Sections 3.3.1 and 3.3.2) bring objectivity to *QBook*'s recommendation process, especially since scores are based on an average of multiple

expert reviews[6]. Furthermore, involving experts' reviews in the recommendation process can help overcome the data sparsity issue, since some books do not have sufficient user-generated data, but have professional critiques which provide valuable information.

3.4 Incorporating a Time-Based Component

To better serve their stakeholders, recommenders must be able to predict readers' interest at any given time. However, given that users' preferences may evolve over time, it is crucial to consider a time component in the process of predicting suitable suggestions [130]. As previously stated, there are a number of data points based on user-generated ratings, reviews or book metadata, that can inform the book recommendation process. However, the influence of a change in users' reading activity over time is often not considered. Furthermore, many avenues can be explored from a time-sensitive stand point in order to generate better predictions of books that will correspond to readers' interests. One of them that is often overlooked is *genre*. By definition, genre (e.g., drama, comedy)[7] is a category of literary composition, determined by literary technique, tone, content, or even length. While genre has been studied as a part of the recommendation process [111], the influence of its distribution over time on suggesting suitable books for individual or groups of users has not been explored. Change of genre over time is a significant dimension to improve the genre prediction process that consequentially influences process performed by book recommenders. Including this component provides the likelihood of reader(s) interest

[6]Popular books tend to be reviewed by more than one expert, while less popular ones may not have any.

[7]The full list of main literal genre is given in Appendix B

in each genre based on its occurrences at a specific point of time, not only the most recent or the most frequently read one. As an answer to this need, *QBook* uses a genre prediction strategy that examines a genre distribution over time and applies a time series analysis model. The goal of our strategy is to discover different areas of user interest, not only the most dominant or recent ones. To do so, *QBook* analyzes user reading history and book genre to not only determine the most dominant areas of U's interest, but also the degree of a U's interest in each of them (e.g., we can predict that a user is 40% likely interested in reading fiction books, 30% fantasy books, and 30% drama books, and thus provide recommendations accordingly).

From the user's point of view, explicitly including time based analysis to inform the recommendation process leads to relevant suggestions that satisfy his specific reading needs. At the same time, the recommendation process improves by providing more effective and relevant suggestions. In the remainder of this section we discuss a novel time-based genre prediction strategy we developed to inform the recommendation process.

3.4.1 Time Series Analysis

To include the time component into *QBook*'s recommendation process, we use a time series analysis model to better predict users' reading interests. By its definition, a univariate time series is a sequence of measurments of the same variable collected over time [3]. Unlike the standard linear regression, data used in time series models are not necessarily independent and identically distributed, and they represent a list of observations where the ordering matters since changing the order could influence the meaning of the data [3]. The goal of time series analysis is to identify a model that describes the pattern of the time series and explain how the past affects the

future. Auto-Regressive Integrated Moving Average ($ARIMA$) [12] includes the most popular models that use time series for prediction purposes. A specific time-series model, denoted $ARIMA(p, d, q)$ is defined and built based on the values of p, d, and q. These parameters represent important factors of time-series models: p shows that a value of an observed variable in one period is related to the number of its values in previous periods (autoregressive term), d represents a differencing operation between terms to produce a stationary process[8], and q represents the possibility of a relationship between a variable and the number of residuals from previous periods (moving average term). Therefore, to identify the appropriate $ARIMA$ model for a given variable Y, it is necessary to iteratively determine values of p, d and q, where $p, q \geq 1$.

The $ARIMA$ forecasting equation for a stationary time series is a linear regression-type equation in which the predictors consist of intervals of the dependent variable Y and periods of the prediction errors. That is: "*predicted value of* Y = *a constant and a weighted sum of one or more recent values of* Y, *and a weighted sum of one or more recent values of the errors*" [12]. To determine the d^{th} value of Y noted as y, over t periods of time, $ARIMA$ calculates the value of y_t when using the following equations:

If $d = 0$: $y_t = Y_t$

If $d = 1$: $y_t = Y_t - Y_{t-1}$

If $d = 2$: $y_t = (Y_t - Y_{t-1}) - (Y_{t-1} - Y_{t-2}) = Y_t - 2Y_{t-1} + Y_{t-2}$

The general forecasting equation is provided in Equation 3.12.

$$\hat{y}_t = \mu + \phi_1 y_{t-1} + ... + \phi_p y_{t-p} - \theta_1 e_{t-1} - ... - \theta_q e_{t-q} \tag{3.12}$$

[8]If statistical properties of a random variable are all constant over time, its time series is stationary.

where the moving average parameters are represented with ϕ's so that their signs are negative in the equation.

3.4.2 How does QBook Use Time-Based Genre Prediction?

Predicting genre to inform the recommendation process involves examining genres read by a specific user. While a simple genre distribution analysis yields probabilities or weights that determine the most favored genres, it lacks the ability to consider genre preference evolution over time. To overcome this drawback, *QBook* uses a time-based genre examination which requires information on reading activities among readers. We first explore U's reading activity to obtain the distribution of his genre interest during continuous periods of time and a significance of each genre for U as described below.

To explicitly consider the change of genre preference distribution over time, our genre prediction strategy takes advantage of $ARIMA$. Specifically, *QBook* uses the implementation of $ARIMA$ model in R language. Since changes in reading activity between fixed and known periods of time are not constant, *QBook* applies non-seasonal $ARIMA$ models. By using $ARIMA$, *QBook* is able to determine a model tailored to each genre distribution to predict its importance for a corresponding user in real time based on its previous occurrences. $ARIMA$ forecasting model uses a specific genre distribution to predict the likelihood of future occurrence of that genre based on its importance in previous periods of time. An interesting characteristic we have discovered is that one book may belong to several genres. For example, *Hamlet* (by William Shakespeare) belongs to number of different genres, such as drama, tragedy and fiction. Therefore, *QBook* considers all books read in a given time period with all corresponding genres. *QBook* calculates $imp(g_{n,t,U})$, as in Equation

3.13, in order to determine the importance of a given genre, g_n, for U at a specific time period, t.

$$imp(g_{n,t,U}) = \frac{\sum_{b=1}^{m} |g_{n,t,b}|}{|G_t|} \tag{3.13}$$

where $|g_{n,t,b}|$ represents the frequency of occurrences of a specific genre among the $m \geq 1$ books read during t, G_t is the set of all genres read in t, and $|G_t|$ is a size of G_t.

Our prediction strategy is able to conduct a search over possible $ARIMA$ models and choose the one with a best fit for a specific genre distribution in time. As illustrated in Figure 3.4, a recent study done by Pew [1] on reading habits in the USA, shows that 28 percent of readers read 11 books per year, and that the average 18-to-29 year old finishes 9 books per year, compared to 13 among older American [24]. Based on this data, we establish one month long "windows" of time in which each user is expected to read at least one book. Our strategy uses one month time frames from the first book log (either bookmarked or rated book) to last. As illustrated in Table 3.3, *QBook* creates a matrix where predicted likelihood of occurrence of a given genre g_n, denoted $p(g_n)$ at time frame, T_W, is based on: its occurrences in the past (in T_{W-1} time frames), a specific time when it occurred and its importance for a specific user ($imp(g_{n,W-1})$). With the described time series genre prediction strategy, *QBook* is able to prioritize the recommendation of $fantasy$ books for U who recently started reading more fantasy books, and less *comedy* ones, which was a genre known to be favored by U in the past. The described prediction approach provides an additional data point to further personalize the recommendation process.

The typical American read five books in the last 12 months

Among all American adults ages 18+ (including non-readers), the mean (average) and median (midpoint) number read by each group

		Mean	Median
	Total (All adults 18+)	**12**	**5**
	Sex		
a	Male	10	4
b	Female	14[a]	6
	Race/ethnicity		
a	White	13[c]	5
b	Black	12[c]	4
c	Hispanic	7	3
	Age group		
a	18-29	9	5
b	30-49	13	5
c	50-64	13	5
d	65+	12	4
	Education level		
a	High school grad or less	9	3
b	Some college	13[a]	5
c	College graduate	16[a]	8
	Household income		
a	Less than $30,000 per year	9	3
b	$30,000-$49,999 per year	10	5
c	$50,000-$74,999 per year	18[ab]	6
d	$75,000 or more per year	16[ab]	8
	Community type		
a	Urban	13	5
b	Suburban	12	5
c	Rural	14	5

Source: Pew Research Center's Internet Project Omnibus Survey, January 2-5, 2014. N= 1005 American adults ages 18 and older. Interviews were conducted on landlines and cell phones, in English and Spanish.

PEW RESEARCH CENTER

Figure 3.4: **Pew statistics on reading activity among Americans**

Table 3.3: **Genre prediction model for U**

$TimePeriod$	$Genre_1$	$Genre_2$	$Genre_3$	**...**	$Genre_n$
T_1	$imp(g_{1,1,U})$	$imp(g_{2,1,U})$	$imp(g_{3,1,U})$	...	$imp(g_{n,1,U})$
T_2	$imp(g_{1,2,U})$	$imp(g_{2,2,U})$	$imp(g_{3,2,U})$	...	$imp(g_{n,2,U})$
T_3	$imp(g_{1,3,U})$	$imp(g_{2,3,U})$	$imp(g_{3,3,U})$	...	$imp(g_{n,3,U})$
...	...	...	...	...	...
T_{W-1}	$imp(g_{1,W-1,U})$	$imp(g_{2,W-1,U})$	$imp(g_{3,W-1,U})$	...	$imp(g_{n,W-1,U})$
T_W	$p(g_1)$	$p(g_2)$	$p(g_3)$	...	$p(g_n)$

3.5 Curating Book Suggestions

The most important step of *QBook*'s recommendation process focuses on curating CB to generate $top-K$ suggestions tailored to U. In this step, *QBook*'s goal is to emulate the curation process (as defined in [46]) and become a personal docent that

understands U and provides relevant books to read that appeal to the diverse, yet unique, preferences and interests of U. To do so, *QBook* simultaneously considers different data points generated in Sections 3.1-3.3 and builds a model that creates a single score that quantifies the degree to which U prefers $b \in CB$. Based on generated scores for each data point, *QBook* selects $top-K$ suggestions to present to U. *QBook* constructs a model that simultaneously considers different perspectives, captured based on the generated scores for each book, into a single score that represents the degree to which U prefers the corresponding book. For model generation, *QBook* adopts the Random Forests algorithm [48], which is a *"classifier consisting of a collection of tree-structured classifiers* $\{h(x, \Theta_k), k = 1, ...)\}$ *where the* $\{\Theta_k\}$ *are independent identically distributed random vectors and each tree casts a unit vote for the most popular class at input* x*."*

Intuitively, random forest is an ensemble learning method for classification and regression that combines a number of decision trees in order to reduce the risk of overfitting. A decision tree is a predictive modeling approach used in machine learning, data mining and statistics that maps observations about an item to conclusions about the item's target value. Like decision trees, random forests handle categorical features, extend to the multiclass classification setting, do not require feature scaling, and are able to capture non-linearities and feature interactions. Random Forest is one of the most successful machine learning models which outputs the class or mean prediction of the individual trees [46, 128]. As reported in [46, 128], among the benefits of Random Forest models we should highlight that they:

- correct decision trees' habits of overfitting to their training set,
- are one of the most accurate learning algorithms available,

- run efficiently on large datasets,
- handle thousands of input variables without variable deletion,
- estimate which variables are important in the classification,
- generate an internal unbiased estimate of the generalization error as the forest building progresses,
- have an effective method for estimating missing data and maintaining accuracy when a large proportion of the data are missing,
- have methods for balancing error in class population unbalanced data sets,
- compute information about the relation between the variables and the classification,
- compute proximities between pairs of cases that can be used in clustering, locating outliers, or (by scaling) give interesting views of the data,
- extend to unlabeled data, leading to unsupervised clustering, data views and outlier detection, and
- offer an experimental method for detecting variable interactions.

Random forests train a set of decision trees separately where the algorithm injects randomness into the training process so that each decision tree is slightly different. By combining the predictions from each tree, the model reduces the variance of the predictions and improve the performance on test data. The randomness injected into the training process includes: subsampling the original dataset on each iteration to get a different training set and considering different random subsets of features to

split on at each tree node. To make a prediction on a new instance, i.e., generate the score that represents a degree to which U would like to read b, a random forest must aggregate the predictions from its set of decision trees. Each model finds a prediction by averaging the predictions of individual trees.

As part of curation process, $QBook$ uses vectors Ub_j (which correspond to each $b \in CB$ and U) to build a random forest model through training by using multiple data points and fully capturing the relationship between each candidate book (b_j) and a user (U) (as shown in Equation 3.14).

$$Ub_j =< r_{U,i}, CBrank(B_j, P_i), Sim(U, b_j), sentiWNr(b_jr), sentiWNs(b_jr), coreNLPr(b_jr), coreNLPr(b_jr) > \quad (3.14)$$

where $r_{U,i}$ is defined in Equation 3.1, $CBrank(B_j, P_i)$ is defined in Equation 3.7, $Sim(U, b_jV)$ in Equation 3.9, $sentiWNr(b_jr)$ and $sentiWNs(b_jr)$ are generated using Equation 3.10 for review and last sentence sentiment, and $coreNLPr(b_jr)$ and $coreNLPs(b_jr)$ are also defined in Equation 3.11 (review and sentence level).

To demonstrate the correctness of selecting the random forest algorithm, we conducted an empirical study using 10-fold cross validation on a disjoint subset of data from the one used for experimental purposes. In this study, we evaluated the performance of a number of multiple machine learning techniques to determine the one best suited to our duration task using popular prediction metrics: Mean Average Error (MAE) and Root Mean Square Error ($RMSE$) [87]. Based on our assessment of Gaussian Process, Decision Tree, Linear Regression and Random Forest [95], we concluded that the model that best fits $QBook$ task is $RandomForest$ (see Table 3.4 for a detailed report of the results of our study).

Table 3.4: **Machine Learning Techniques Comparison**

Name	Gaussian Process	Decision Tree	Linear Regression	Random Forest
MAE	0.502	0.317	0.502	**0.253**
RMSE	0.671	0.571	0.671	**0.432**

The reasons behind the better performance of $RandomForest$ are due to the fact that it: (i) builds many independent classifiers and then chooses or favors the one that happens to do best, (ii) builds trees on good features and favors them over trees that were built on noise features and (iii) has decision rules at the leaves that are not sensitive to outliers or small noise.

3.5.1 Creating a Personalized Exhibit

As mentioned in Section 3.5, $QBook$ assigns a score for $b \in CB$ using the trained random forest model that showcases the degree of which U prefers b_j. By doing this, $QBook$ identifies the $top-K$ suggestions for U. As discussed in Section 3.4.2, reading activity among users varies, which causes $QBook$ to have lack of information for curation purposes corresponded to "nonreaders", who are users who rated less than 35 books. The lack of available information can hinder the process of further curation of personalized recommendations, which is why the final set of recommendations for these users is based on the scores generated using the random forest model. However, for active readers (who read at least 35 books), it is important to identify motives for their reading selections, which can be different among different readers; some are biased by experts other by authors preferences. Consequently, $QBook$ explores and considers these reasons in the book selection process for each active reader to further personalize suggestions. To do so, $QBook$ captures correlations among different data points involved in the process of generating Ub_j for U. $QBook$ uses a commonly-

adopted correlation strategy, Pearson correlation [133], which is a statistical measure that indicates the extent to which two or more variables fluctuate together calculated using Equation 3.15.

$$corr.coefficient(XY) = \frac{n\left(\sum xy\right) - \left(\sum x\right)\left(\sum y\right)}{\sqrt{\left[g\sum x^2 - \left(\sum x\right)^2\right]\left[g\sum y^2 - \left(\sum y\right)^2\right]}} \tag{3.15}$$

where X and Y are specific data points, x and y are variables that belong to X and Y, respectively, and g is a number tuples of variable considered. The positive correlation coefficient between any two variables, which are data points considered by $QBook$, in our case, means that for every positive increase of one variable, there is a positive increase in the other. If a coefficient is negative, it means that for every positive increase of one variable, there is a negative decrease of the other one. Otherwise, the two variables are not related.

To better understand dependencies among data points, their overall correlations for all users are presented in Figure 3.5 where the blue bubbles represent positive correlations and the red ones are negative. The size of the bubbles represent the strength of the correlation, i.e., the bigger the bubble is, the higher the correlation between the corresponding data points.

Consider a user, U_1, and the degree of correlation each of the perspective data points considered by *Qbook* has in influencing his reading selections. As shown in Figure 3.6, $CBrank(U_1, book)$ (obtained in Section 3.1.2) has the highest correlation score among the data points, so for U_1 the content of books to be recommended has the highest importance for him at the time of deciding what to read. Now consider U_2, where (as showcased in Figure 3.7) the data point with a highest influence in the process of a book selection is $Sim(U_2, book)$, calculated in Section 3.2. This means

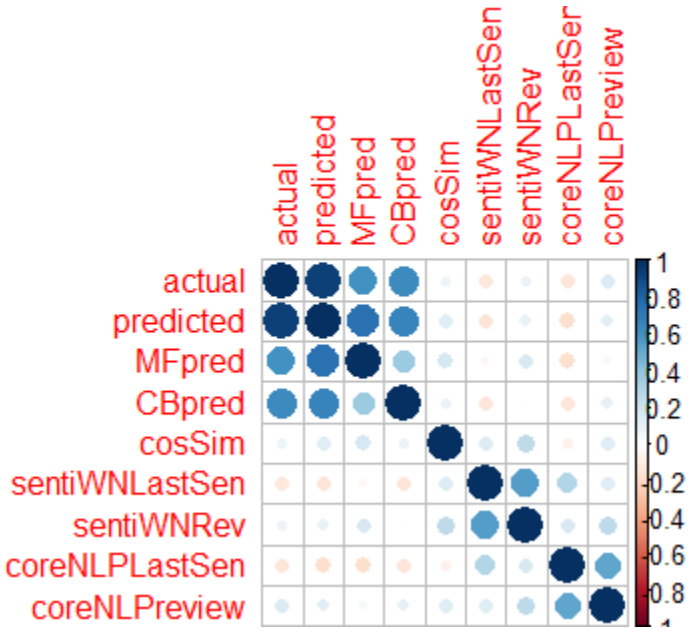

Figure 3.5: **Overall data points correlations computed on the *QDevel* dataset, where actual is real rating value assigned by a user *U* to a given book *b* and predicted is rating value predicted during *QBook* recommendation process.**

that unlike U_1, book selections made by U_2 are driven by the similarity between literal elements U_2 cares about and the ones that are used in b.

QBook determines which data point has the most influence on U in the process of selecting books to read in the past and re-ranks the top recommendations generated using random forests model based on the importance (i.e., computed score) of the corresponding data point for each book in CB for U. If the selected data points has positive correlation with a predicted score, *QBook* considers that data point as the most important one in U's process of selecting books to read, so it re-orders the $top - K$ recommendations generated on Section 3.5 based on the score of the most influential data point for U. For example, recommendations for U_1 are re-ordered based on $CBrank$ values, while suggestions for U_2 are ranked based on the values of

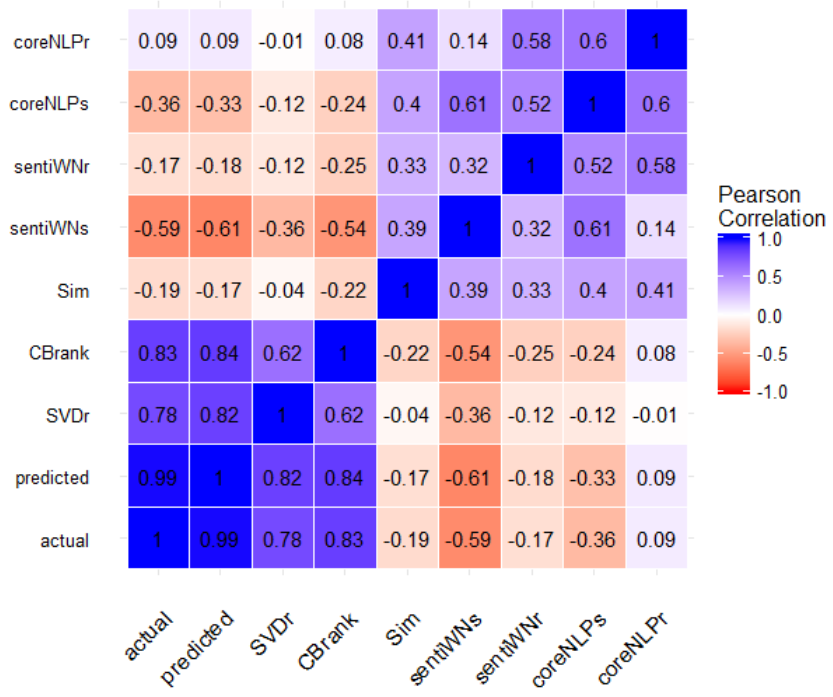

Figure 3.6: **Feature Correlation for a given** U_1

Sim. Consequentially, *QBook* provides personalized suggestions for U, since choosing a book to read differs among the users.

In the case of active readers, the final step of *QBook* for curating their suggestions considers the genre preference of U. This is accomplished using prediction strategy (described in Section 3.4). To determine the number of representative candidates from each genre that should be part of the final exhibit presented to U, *QBook* uses p_{g_n} predictions of how likely U will be interested in each genre at a given point of time. Consequently, books from the previously generated $top-K$ set are selected to become a part of the final list of $top-7$ recommendations that are provided to U based on the most influential data point and predicted genre preferences. Note that number 7 is determined based on a popular psychology paper [93], where authors

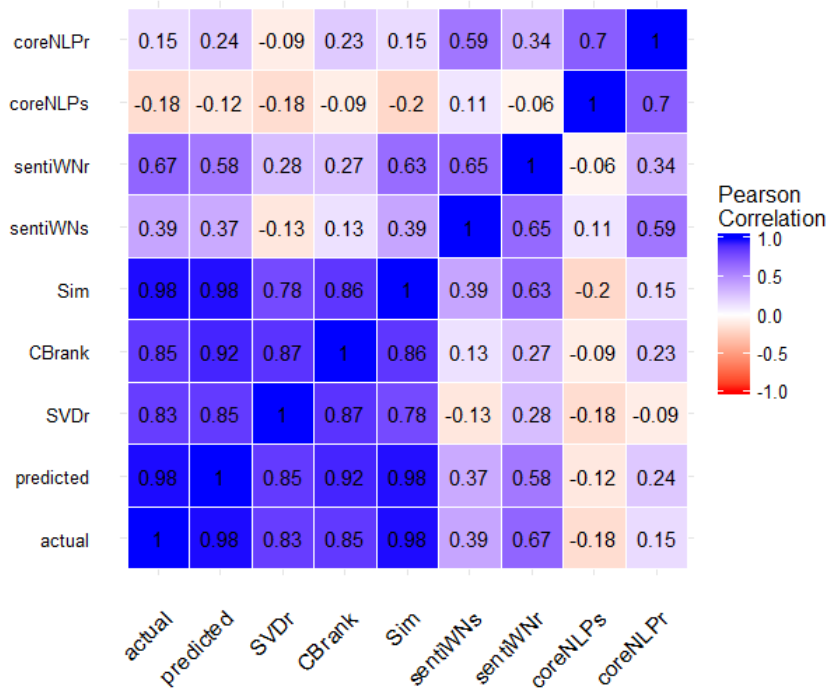

Figure 3.7: **Feature Correlation for a given user U_2.**

argue that a number of objects an average human can hold in working memory is 7 $\pm$ 2.

By performing this curation step, *QBook* increases diversity among the suggestions by including books from all different areas of users' interests and improves personalization of recommendations by ordering suggestions based on a specific feature for U. Consequently, *QBook* enables U to chose from exhibit of books tailored solely to him in order to satisfy reading needs in a given time.

3.6 Generating Explanations

QBook cannot be a personal docent if it does not justify why the suggestions were generated. To do this, *QBook* pairs each book recommendation with an explanation

giving U data that allows him to make the most informed and fastest decisions in terms of selecting a single item among the suggested ones. The purpose of providing an explanation for each of the curated book suggestions generated in Section 3.5 is to help U in the process of selecting the one that will better satisfy his reading needs at a particular moment. To generate the explanations, *QBook* uses archived book reviews provided by other users, experts' reviews, and the set of steps taken to generate curated recommendations.

3.6.1 What Do Other Readers Say About a Book?

QBook creates explanations for a curated book, b_j, by extracting from archived reviews on b descriptions related to U's preferred features (literary elements determined in Section 3.2). In other words, *QBook* identifies sentences in reviews pertaining to b that include literary elements of U's interest. In doing so, *QBook* provides the user with sufficient information about the recommendations without overwhelming him with too much information to read related to the recommended books. *QBook* does not emphasize the sentiment of the features, since *QBook*'s intent is not to make U like one option more than another, but to save time on identifying information important for him in the process of choosing the book to read.

3.6.2 What Do Experts Say About the Book?

Along with the other users' opinions, *QBook* generates explanations based on the experts' opinions on the book's quality. As described in Section 3.3, *QBook* is able to analyze expert's reviews. Moreover, *QBook* includes in the corresponding explanations sentences from experts' reviews that reference users' top feature (literary element) of interest. This way, U is provided with objective opinions by extracting

sentences from experts' reviews pertaining to the feature of U (denoted F_{U_1}). This increases U's trust in *QBook*, since U can read unbiased opinions that help him determine if he would like to read a particular recommendation or not.

3.6.3 Why Was Each Curated Book Selected?

As the final part of the explanation generation process, *QBook* looks into the steps taken in curating each book suggestion. If a book was selected based on users' historical data, an explanation contains information that a given book was selected since similar users gave it a higher rating score. In the case of book from content based filtering, the explanation includes information that suggested book is similar to the ones the user liked in the past. If experts' opinion had a strong influence in the curation process, *QBook* makes sure that the user is aware of it, as well as the fact that a recommended book belongs to the most likely genre of interest to the user or not.

3.6.4 Why Should Reader Care About Suggested Books?

Explanation generation is an important part of the *QBook* recommendation process. In doing so, *QBook* provides U with sufficient information about the recommendations without overwhelming U with too much information to read related to the recommended books. As previously stated, *QBook* does not emphasize the sentiment of the features, since its intent is not to make U like one option more than another, but save U's time in identifying information important for him. To provide U with more information about each suggestion, *QBook* selects the sentences from other users reviews for a suggested book that describe the most frequently mentioned feature F_{U_1}, since F_{U_1} represents the literary element of highest interest for U. If there are multiple

sentence describing the same feature, *QBook* arbitrarily selects one to be shown to U. Furthermore, duplicate sentences are excluded, since *QBook* never selects a sentence to be part of the explanation for b, if that sentence is previously selected. To make suggestion's explanation more informative and complete, *QBook* includes a sentence from experts' reviews (selected as described in Section 3.6.2) that describes F_{U_1} and thus offers an objective opinion about the corresponding literary element, as well as a sentence that describes the most important perspective that influenced reading selections for U detected during the curation process (as described in Section 3.6.3). The explanation paired with b includes three sentences specifically selected for U to provide him with personalized and valuable information. The number of sentences is chosen based on a study [120] that proves that users prefer to see at most 7% of the document as the summary to make the best decisions if documents are of their interest or not. Since *QBook* provides "summarized" reviews (in the form of explanations), often there are at least 35 sentences in a book critique[9], we chose to show 7% of 35 sentences as part of a book suggestion's explanation. To further justify the chosen number of sentences to include in the explanation, we rely on a user-study conducted in [57], where 80% of the appraisers preferred explanations generated of the same number of sentences over remaining two strategies for generating explanations, since appraisers indicated that explanations provided the most useful information about suggestions.

Most importantly, by providing *personalized explanations*, *QBook* is able to *tell a story* about U and how each suggestion is related to U, which increases users' trust in the system, as well as the system's transparency [124]. Unlike the majority of

[9]We evaluated a random sample of reviews posted on NPR books website [18] to determine the frequent length of expert reviews.

existing recommendation strategies, *QBook* is not a "black box" since it provides all information regarding the selection and curation of the final exhibit of books that are suggested. By doing this, *QBook* increases users' trust and transparency of the system since they know *QBook* is making decisions tailored towards each individual user. Therefore, by providing the most complete explanations *QBook* acts as a personal docent for U.

CHAPTER 4

EVALUATION

In this chapter, we discuss the framework we adopted for evaluating *QBook* and the results of the experimental studies conducted to validate its performance and design methodology. These studies include (i) examining individual strategies considered by *QBook*, (ii) demonstrating the importance of the generated explanations, (iii) analyzing *QBook*'s effectiveness and scalability, (iv) comparing the performance of *QBook* with baselines and existing approaches focused on recommending books and (v) how *QBook* addresses current recommendation issues.

4.1 Framework

In this section we discuss the dataset, evaluation metrics and assessment strategy considered in this study.

4.1.1 Dataset

To the best of our knowledge, there is no existing benchmark dataset that can be used for evaluating the performance of a curation recommendation system. For this reason, we created our own dataset based on well-known resources. For user historical data, i.e. ratings and reviews, we used the INEX Amazon/LibraryThing Book Corpus [11],

which consists of 2.7 million book titles[1], each with combined book metadata from Amazon [2] and LibraryThing [15]. Each book record in the dataset is an XML file (an example of which is shown in Figure 4.1), that includes: isbn, title, author, publisher, number of pages, publication date, reviews, and user-generated tags. Each review tag

<?xml version="1.0" encoding="UTF-8" standalone="no"?><!-- version 1.0 / 2009-11-10T13:40:56+01:00 --><!DOCTYPE book SYSTEM "books.dtd">
<book><isbn>0711947430</isbn><title>Best of Sade</title><ean>9780711947436</ean><binding>Paperback</binding><label>Hal Leonard Corporation</label><listprice>$16.95</listprice><manufacturer>Hal Leonard Corporation</manufacturer><publisher>Hal Leonard Corporation</publisher><readinglevel/><releasedate/><publicationdate>1994-01-01</publicationdate><studio>Hal Leonard Corporation</studio><edition/><dewey>016</dewey><numberofpages>80</numberofpages><dimensions><height>31</height><width>882</width><length>1173</length><weight>66</weight></dimensions><reviews><review><authorid>A2D4BSQC032LXB</authorid><date>2006-08-17</date><summary>Sade...How can I describe?</summary><content>Her music is like sugar (with a salty sweetness).
So this album is like someone laying out before you a taste of 16 different sugars; this one home-grown; this one from a far off place; this one raw & pure; this one melting into you...just that way....

--Vu</content><rating>5</rating><totalvotes>0</totalvotes><helpfulvotes>0</helpfulvotes></review><review><authorid>ADBS99FQMSVXX</authorid><date>2007-10-03</date><summary>Great!</summary><content>I love this collection, although it would be nice if a few other favorites such as "Cherry Pie" were included. Some parts of the songs have been deleted in the sheetmusic, like the solo sax in "Jezebel" and the jazz jam in "Hang on to your love". But it can probably be improvised well, probably not to hard to pick up on it.</content><rating>5</rating><totalvotes>0</totalvotes><helpfulvotes>0</helpfulvotes></review></reviews><editorialreviews><editorialreview><source>Product Description</source><content>16 of her best, including: No Ordinary Love · Smooth Operator · The Sweetest Taboo · and more.</content></editorialreview></editorialreviews><images><image><url>http://ecx.images-amazon.com/images/I/51C466Q103L._SL160_.jpg</url><height>160</height><width>120</width><imageCategories><imagecategory>primary</imagecategory></imageCategories></image><image><url>http://ecx.images-amazon.com/images/I/51C466Q103L._SL30_.jpg</url><height>30</height><width>22</width><imageCategories><imagecategory>primary</imagecategory></imageCategories></image><image><url>http://ecx.images-amazon.com/images/I/51C466Q103L._SL500_.jpg</url><height>500</height><width>375</width><imageCategories><imagecategory>primary</imagecategory></imageCategories></image><image><url>http://ecx.images-amazon.com/images/I/51C466Q103L._SL75_.jpg</url><height>75</height><width>56</width><imageCategories><imagecategory>primary</imagecategory></imageCategories></image></images><creators><creator><name>Sade</name><role>

Figure 4.1: **Sample of XML file that represents a book record.**

contains author ID, the content of a review, numerical rating value and corresponding timestamp, so we were able to create a dataset that captures connections among books' metadata and rating distribution over time. The Amazon/LibraryThing collection is complemented with library catalog records from the Library of Congress (LoC) [14], which includes 1.25 million titles. Lastly, experts' reviews are collected

[1]Note that some books can have multiple editions, therefore multiple titles refer to the same book

from known book critiques' websites, such as The New York Times [17], NPR Books [18] and Editorial Reviews from Amazon/LibraryThing dataset.

To better understand the reading activity of users in the dataset, in Figure 4.2 we present the distribution of read books among all users from Amazon/LibraryThing dataset. Based on the provided distribution, we notice two extremes: a large number of users read a very small number of books, while a small number of readers rated a large number of books. Further analysis of users based on their ratings and reviews is illustrated in Figure 4.3, which showcases that 11% of users read more than 100 books per year among all users. This means that the majority of readers tend to read less than 100 books on a yearly basis. With that in mind, for evaluating the performance of $QBook$, we considered users who read less than 100 books per year, while the others are considered spammers and thus treated as outliers. For example, as shown in Figure 4.2, there are some users in the dataset who rated around 47000 books in a couple of years, which is clearly not possible. The remaining set of 1,728,756 users constitute the dataset used to validate the performance of $QBook$, denoted $QData$.

We split $QData$ in three parts, where 1,727,200 users were used for training — denoted $QTrain$, 778 users for development — denoted $QDevel$, and the remaining 781 users for evaluation — denoted $QEval$. To ensure a representative distribution for development and evaluation purposes, we first clustered users from $QData$ based on their corresponding number of books read. Thereafter, we created the development and evaluation partitions by randomly selecting the same percentage of users from clusters to "simulate" real representation of $QData$ in each of the partitions.

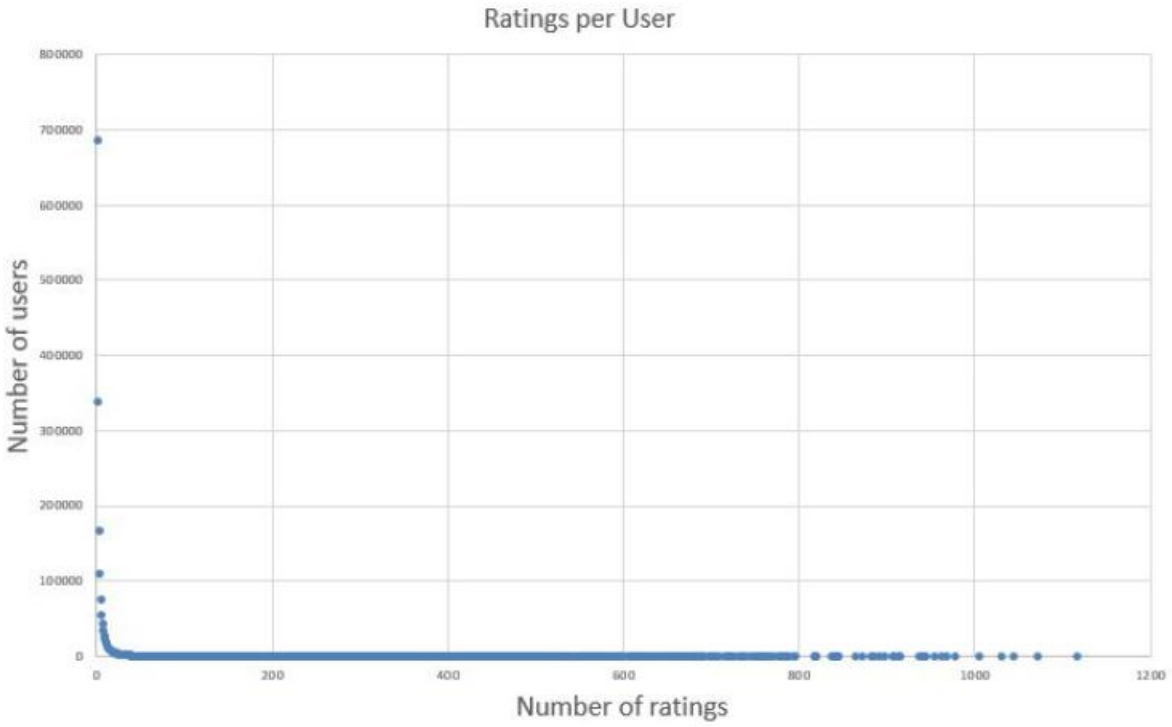

Figure 4.2: **Number of users vs. number of books on INEX/LibrayThing Dataset.**

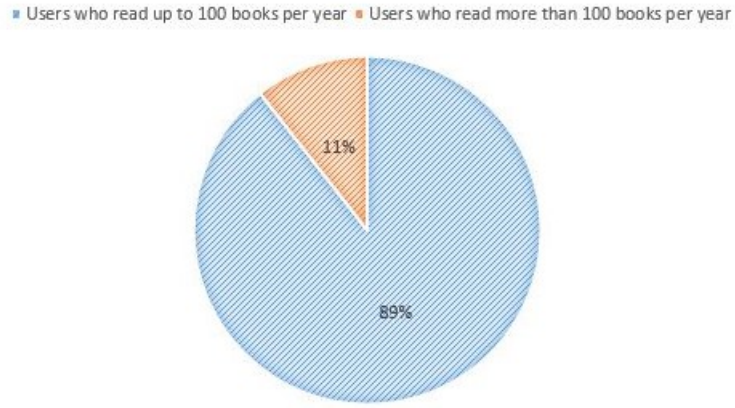

Figure 4.3: **Reading activity among users on INEX/LibrayThing Dataset.**

4.1.2 Metrics

For **recommendations validation** purposes, we used the well-known metrics[2] discussed below:

[2]Note that all corresponding equations are given for calculating metrics of an individual user. The metrics of the system are obtained by averaging individual scores of all users who make a dataset.

- Normalized Discounted Cumulative Gain ($NDCG$), which measures the correctness of the generated recommendations and penalizes relevant recommendations positioned lower in the ranking calculated using Equation 4.1 [71, 108].

$$nDCG = \frac{DCG}{IDCG} \tag{4.1}$$

DCG is calculated using Equation 4.2 and $IDCG$ is an ideal DCG used as normalization factor.

$$DCG = rel_1 + \sum_{i=2}^{pos} \frac{rel_i}{log_2(i)} \tag{4.2}$$

where $log_2(i)$ is penalization factor, pos is a rank position and rel_i is the graded relevance of the book at rank i (rel_i is 1 if the corresponding book is relevant and 0 otherwise).

- Mean Reciprocal Rank (MRR), which captures the average number of suggested items a user has to scan through to identify a relevant one, as shown in Equation 4.3 [71, 108, 112].

$$MRR = \frac{1}{|Q|} \sum_{i=1}^{|Q|} rank_i \tag{4.3}$$

where Q is the number of books recommended to a user, $rank_i$ is the ranking position of the book, and $\frac{1}{|Q|}$ is normalization factor.

- Mean Absolute Error (MAE), which is a measure of the deviation of recommendations from their true user-specified values [87]. For a given set of n (>1) $< p_i, q_i >$ pairs for a user, this metric computes their absolute error, where p is real user-generated rating and q is predicted value. For each rating-prediction pair MAE treats the absolute error between them i.e., $|p_i - q_i|$ equally. The

MAE is computed by first summing the absolute errors of the n rating-prediction pairs and then computing the average. MAE is computed using Equation 4.4.

$$MAE = \frac{1}{n}\sum_{i=1}^{n}|p_i - q_i| \tag{4.4}$$

- Mean Squared Error (MSE) is an estimator that measures the average of the squares of the errors, that is, the difference between the estimator and what is estimated [87]. MSE assesses the quality of a predictor using Equation 4.5.

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(p_i - q_i)^2 \tag{4.5}$$

where p_i and q_i are as defined in Equation 4.4, and n is as defined in Equation 4.4.

- Root-Mean-Square Error ($RMSE$) represents a quadratic scoring rule which measures the average magnitude of the error, the difference between predicted scores and user-specific values are each squared and then averaged over the sample [87]. Since the errors are squared before they are averaged, the RMSE gives a relatively high weight to large errors. RMSE is calculated using Equation 4.6.

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(p_i - q_i)^2} \tag{4.6}$$

where p_i and q_i are as defined in Equation 4.5 and n is as defined in Equation 4.4.

- Accuracy showcases the percentage of right and wrong predictions [87]. The accuracy of a measurement system is the degree of closeness of measurements

of a quantity to that quantity's true value. In the case of *QBook*, we treat book recommendations on a binary fashion, in such a way that they are treated as relevant if they have been rated or bookmarked by the corresponding user on a given dataset, and non-relevant otherwise.

- Coverage shows how many of the items from the dataset are being recommended to all users who get recommendations, as shown in Equation 4.7 [65]. This measure aims to quantify if all items of users interests are recommended, thus the larger coverage value is better in a recommendation context.

$$coverage = \frac{|K \bigcap R \bigcap A|}{|K \bigcap R|} \tag{4.7}$$

where K is the set of books of the collection known to a given user, R is the set of relevant books to a user and A is the set of recommended books. Therefore, $|K \bigcap R \bigcap A|$ is a set composed of the relevant books known to a user that have been recommended.

- Serendipity measures how surprising the successful recommendations are to the user [65]. It is the amount of relevant, but new, information offered to a user in a recommendation. To calculate the unexpected set of recommendations, we use Equation 4.8.

$$UNEXP = RC/PM \tag{4.8}$$

where PM is a set of recommendations generated by a primitive recommendation model and RS is the recommendations generated by *QBook*. However, unexpected recommendations are not necessarily successful recommendations [65], so we use Equation 4.9 to calculate serendipity.

$$SRDP = u\left(RS_k\right) \tag{4.9}$$

where RS_k is k^th element in $UNEXP$ and $u\left(RS_k\right)$ describe the usefulness of the unexpected recommendations. The $u\left(RS_k\right) = 1$ if a recommendation is useful to a user, and $u\left(RS_k\right) = 0$ otherwise.

- Novelty captures how different a recommendation is with respect to what the user has already seen along with the relevance of the recommended item [126]. Novelty is the fraction of the relevant books in the recommended set that are not known to a user. This metric is calculated using Equation 4.10.

$$novelty = \frac{|(R \bigcap A) - K|}{|R \bigcap A|} \tag{4.10}$$

where K and R are defined as in Equation 4.7.

For demonstrating the usefulness of the **explanations** generated by $QBook$, we rely on the criteria defined in [123] which outline the "characteristics" of good explanations for recommendation systems. These criteria are: Transparency, Scrutability, Trust, Effectiveness, Persuasiveness, Efficiency and Satisfaction. Table 4.1, initially introduced in [123], includes detailed definitions of each of the aforementioned goals.

Table 4.1: **Aims of explanations in a recommender system**

Aim	**Definition**
Effectivness	Help users make good decisions
Efficiency	Help users make decisions faster
Persuasiveness	Convince users to try or buy
Satisfaction	Increase the ease of usability or enjoyment
Scrutability	Allow users to tell the system it is wrong
Transparency	Explain how the system works
Trust	Increase users' confidence in the system

4.1.3 Test of Statistical Significance

To verify the performance of our recommendation strategy we use the well-known t-test of **statistical significance** [87]. A t-test of statistical significance shows if the difference between the averages of two samples most likely reflects a real difference in the population from which the samples were taken.

In the case of recommendation systems, this test aims to demonstrate that the improvement in performance of a given system or recommendation strategy (regardless of the metric considered) is consistent.

4.1.4 Offline Assessment

Offline evaluation strategies are very popular in the recommendation domain [110, 86], given that they do not require any direct interaction with real users, and thus are not costly. Unfortunately, this type of evaluation can only answer a narrow set of questions, mostly based on the prediction power of algorithms. In conducting offline evaluations, we assume users' behaviors and their interactions with the system, thus we are not able to measure the real influence of a system to its users or account for relevant suggestions on items users are yet to rate [108]. Therefore, the results of the offline empirical studies are undervalued [108]. However, this limitation affects all recommenders evaluated with such strategies, which is why the measures reported in this manuscript are consistent [102].

4.2 Experimental Results

In this section, besides evaluating *QBook*'s overall methodology by using the dataset described in Section 4.1.1 and metrics described in Section 4.1.2, we compare its

performance with baseline, yet popular, strategies available on LensKit, including Matrix Factorization (SVD) [13] and Content-Based Filtering (CB) [7], as well as state-of-the-art book recommendations,including the ones [41, 74, 83, 85, 90, 129]. In this section, we also provide an initial assessment based on the scalability and efficiency of *QBook*.

4.2.1 QBook Performance

We evaluate the individual strategies that contribute to *QBook*'s recommendation process and analyze how each of them influences generation of book suggestions. In doing so, we create $top-K$ recommendations for each user in *QDevel* using the individual strategies considered by *Qbook* and defined in Chapter 3. Thereafter, we evaluate effectiveness of the recommendation generated using each strategy based on NDCG.

In Figure 4.4, we illustrate the NDCG scores computed using *QDevel*. Note that both *QDevel* and *QEval* sets provide comparable scores, which showcases that *QBook* performance is consistent without occurrence of overfitting as a result of training.

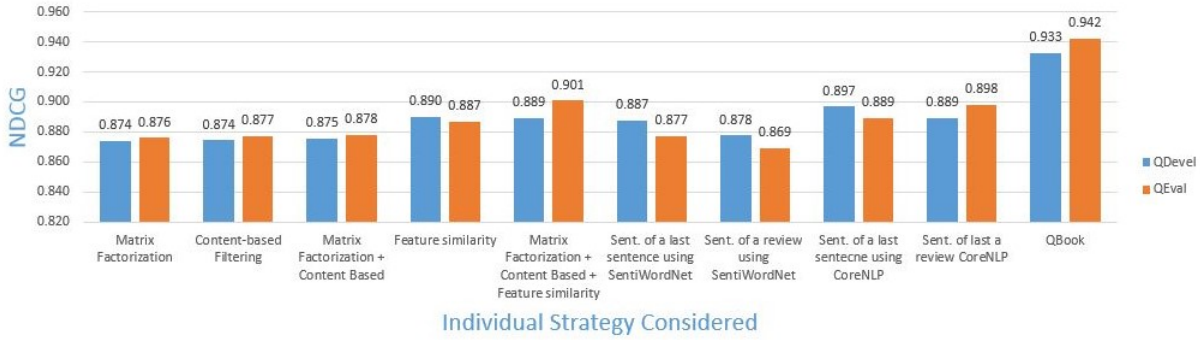

Figure 4.4: **Performance evaluation of individual recommendation strategies considered by *QBook* on *QDevel* and *QEval*.**

As shown in Figure 4.4, matrix factorization and content based approaches are similarly effective, in terms of generating book recommendations. However, when combined they slightly increase the value of NDCG. This improvement is statistically significant ($p < 0.001$), which means that users get, in general, more relevant recommendations, when both strategies are considered. This can be explained with the fact that these two methodologies complement each other, i.e., where the lack of users' ratings can be replaced with a content of the book, and vice versa. Furthermore, we can see that similarity between literary features of a user's interest and book's most mentioned ones, has a positive influence on the recommendation process as it increases NDCG by 2.5 % when explicitly considered as part of the recommendation process. This is anticipated, since user-generated reviews hold a lot of information that can allow us to gain knowledge about each user and personalize suggestions.

The most reliable data points, which do not only achieve relatively high NDCG but also are widely applicable and do not depend on individual users, are the four strategies that analyze sentiment of expert reviews. These strategies rely on information frequently available for books and thus are applicable to the majority of books examined by *QBook*. Based on Figure 4.4, we can see that data points calculated using sentence level sentiment analysis, provide slightly better recommendations comparing to the ones generated using word level sentiment analysis.

Even though all of the individual strategies perform relatively well, we cannot assume that each data point can be calculated for every single book at any moment. *QBook*'s improvements in terms of NDCG we can justify with the fact that *QBook*: (i) simultaneously considers multiple data points, (ii) includes genre-prediction strategy, and (iii) more completely analyzes different perspectives of user-book relations to provide recommendations even when some of the data points are unavailable. This is

important because *QBook* is able to address data sparsity as one of the major issues recommendation systems are facing. With an NDCG of 93% on *QDevel* (and 94% on *QEval*), *QBook* provides the most relevant recommendations to the largest number of users.

This is demonstrated based on the fact that NDCG of *QBook* is statistically significant with respect to the NDCG reported for the individual strategies (for $p < 0.001$).

4.2.2 Evaluating Genre Prediction

Since our time-based genre prediction strategy is novel, we validate it in isolation to prove its correctness. To validate the performance of our proposed time-based genre prediction strategy, we selected a subset of the *QDevel* corpus of books. We used 1214 users[3] along with the books they rated or reviewed. To quantify the assessment, we applied *MAE*, *Accuracy* and *Kullback—Leibler (KL) divergence* [87]. In the context of genre prediction, MAE estimates the difference between the predicted genre importance and the ground truth, i.e., genre distribution for a user at a given time, whereas Accuracy applies a binary strategy that reflects if the predicted genres correspond to the ones read by a user in a given period of time. Furthermore, KL divergence (also known as information gain) measures how well a distribution b generated by a prediction strategy approximates to distribution a, the ground truth, as shown in Equation 4.11.

$$D\left(a \parallel b\right) = \sum_{i} a_i log \frac{a_i}{b_i} \tag{4.11}$$

[3]In our initial assessment, we considered Amazon users who provided ratings for at least 35 books.

where a is real distribution, b is predicted distribution, and a_i and b_i are individual values that correspond to real and predicted distributions, respectively.

In establishing the ground truth for each user considered for evaluation purposes, we adopted the well-known $W-1$ strategy, such that the genre of the books rated by a given user in the W time frame are treated as "relevant" genres for a user, and the genre of the books rated in the previous $W-1$ windows are used for training a user's genre prediction model. As a **baseline** of our initial assessment, we use a traditional prediction strategy that considers the proportion of occurrences of each genre over the total number of occurrences of genres read by a user in $W-1$ periods of time, as the only decider on how likely is for that genre to be of a users' choice in W.

Table 4.2: **Evaluation using the Amazon dataset**

Genre Prediction	MAE	KL divergence	Accuracy
With Time Series	0.143	0.623	0.870
Without Time Series	0.144	0.663	0.826
With Time Series (3+ genre)	0.138	0.660	0.857
Without Time Series (3+ genre)	0.146	0.720	0.810

The reading activity of users from a dataset collection varies: from new users who noted one book to spammers who rated over a thousand of books during long periods of time. As shown in Table 4.2, for W=11[4] the performance of proposed strategy improves with the number of observed windows of time. KL divergence scores showcase that genre distribution predicted using time-series approach better approximates to the ground truth. This can be seen on Figure 4.5, where MAE distribution of proposed genre prediction approach is closer to zero, comparing to the baseline. Figure 4.5 depicts that a probability of occurrence of each considered genre is closer to the ground truth when a time component is included in the prediction

[4]We empirically verified that for 6≥N≥11 the results are comparable to the ones for W=11.

process. As a further assessment, we observed differences in genre predictions among

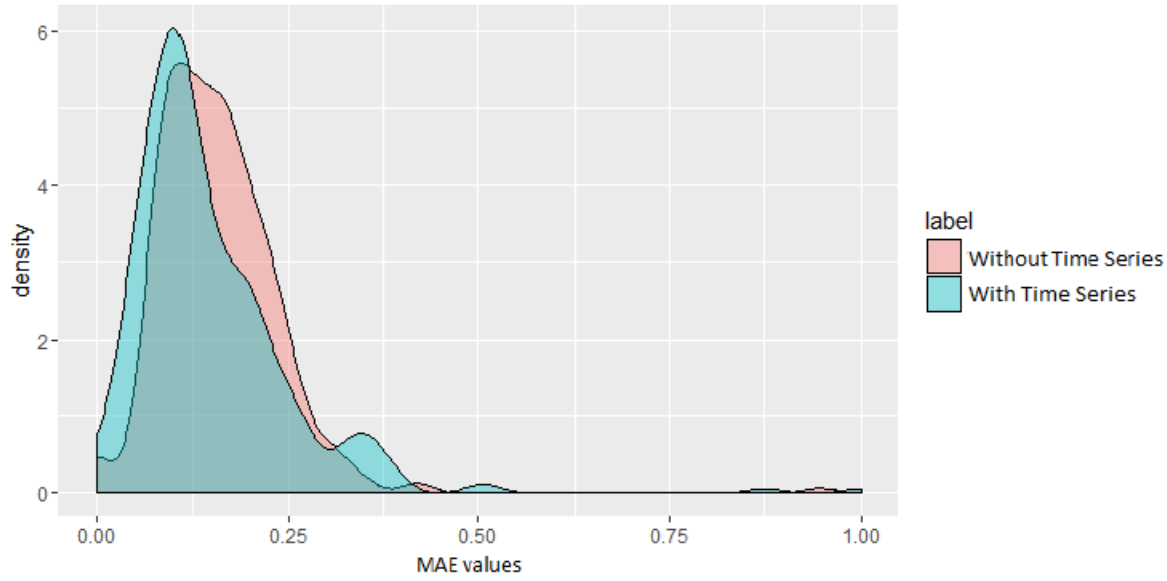

Figure 4.5: **Comparison of density based distributions of MAE.**

users who read different number of distinct genres. As shown in Figure 4.6, for users who read only one to two genres, the time-based prediction strategy does not outperform the baseline. Note that the graph contains two axes with discrete values. If we do not apply jittering, we would only see a lot of points on top of each other, making it impossible to see the color of those points which is an important aspect of the graph. Therefore, we apply jittering to move those data points slightly so that they can be better visualized. This juttering will never change the result since the jittering applied is less than 1, which is smaller than the gap between each point. This way, jittering points never overlap and with points of other groups. However, if a user reads three or more genres, our time-based genre prediction strategy outperforms the baseline in all three metrics. This is not surprising, as it is not hard to determine area(s) of interest for a user who constantly reads only one or two book genres, which

is why the baseline performs as well as the time-based prediction strategy. In Figure 4.6, we can observe a higher percentage of bubbles with a color (shades of blue, purple) that represents smaller MAE comparing to the baseline occur more when number of read genres increases. Given that users that read three or more genres represent 91% of the users in our sampled dataset, the proposed strategy provides significant improvements in predicting preferred genre for each reader. Furthermore, KL divergence improvement when using time-series prediction approach is statistically significant with respect to prediction without an involvement of a time series, for $p < 0.001$.

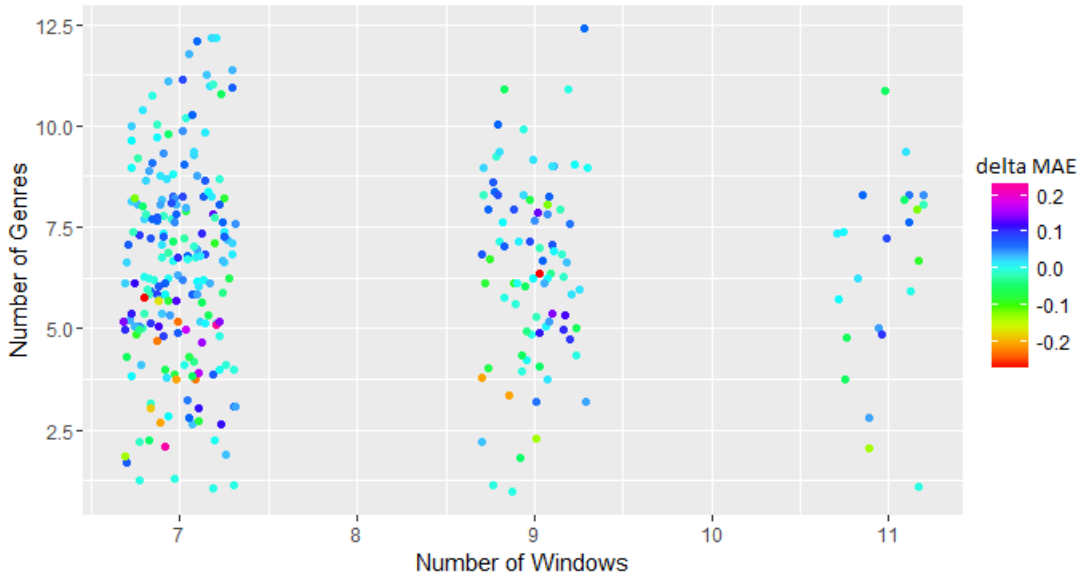

Figure 4.6: **Influence of the number of read genres and the number of window frames on genre predictions, where delta MAE = baseline prediction - prediction using time series.**

4.2.3 Quantifying QBook's Explanations

Based on the criteria described in Table 4.1 and the design strategy of *QBook* (presented in Chapter 3), we can conclude that *QBook* achieves five out of the seven defined criteria expected for explanations generated by recommenders, including: transparency, trust, satisfaction, effectivness and efficiency. By suggesting curated books which are described based on users' preferred features of interest, showcasing opinions of other users on those features and describing curation steps, *QBook* addresses transparency. *QBook* inspires trust of its users in the system, since it does not consider the sentiment, i.e., positive or negative, connotation of the features to determine if they should be included in the corresponding explanations. Instead, *QBook* provides unbiased recommendations and explanations. With that, users' confidence increases knowing that *QBook* provides a real picture of each suggested book. Users are also able to make good and fast decisions, in terms of selecting books among the suggested ones, since they know which books, based on provided explanations, match their preferences. With this, *QBook* increases its effectiveness. Given that users' overall satisfaction with a recommender is related to the perceived quality of its recommendations and explanations [66], *QBook* users appreciate the fact they do not need to spend more time on researching books with characteristics important to them.

According to the studies presented in [123] and assessments on a number of current strategies for generating recommendations with the corresponding explanations [72, 94, 121, 127, 135], we can report that, on average, only two (out of the seven) criteria are satisfied by any given recommender system that provides justified recommendations. The only system that is comparable with *QBook* is the

one developed by Zha et al. [135] that fulfilled five of the aforementioned criteria with its provided explanations. Given that the proposed system involves sentiment in the generation of the explanations, we argue that users' trust in *QBook* is more than its counterpart presented in [135], since *QBook* makes unbiased decision when generating users' features of preferences and making decisions which sentences to select to address the corresponding features.

4.2.4 Understanding how QBook Addresses Recommendation Issues

In this section, we showcase the benefits of *QBook* and its ability to address popular recommendation issues. To quantify *QBook*'s performance in these aspects, we considered MAE, novelty, coverage and serendipity (previously described in Section 4.1.2). Furthermore, while using the evaluation framework presented in [86], we can simulate online evaluation using offline metrics: coverage, serendipity and novelty.

Table 4.3: ***QBook* and Recommendation Issues**

Book Recommenders	MAE	Novelty	Coverage	Serendipity
QBook	0.42	0.73	0.92	0.68

Based on the results of our conducted empirical study, we conclude that the recommendation prediction accuracy of *QBook* is consistent, regardless of the presence or absence of some data points used in the recommendation process. As shown in Table 4.3, low MAE of 0.42 generated by *QBook* showcases that recommendation strategy is able to successfully predict the user's degree of preference for each book, no matter if there is available data for that book.

As reflected by the coverage score in Table 4.3, *QBook* is able to consider a vast number of diverse books in generating recommendations, as opposed to popular ones.

Based on this assessment, we can conclude that *QBook* addresses data sparsity and cold-start problem, since it considers high volumes of books to recommend.

The novelty score of 73% depicts that a user is provided with book suggestions that are diverse based on what he already saw. This characteristic of *QBook*, together with relatively high serendipity score, shows that new and unexpected, but relevant suggestions are provided to a user. We can determine that books suggestions *QBook* provides to each user are not only novel and serendipitous, but also relevant, which is an important ability for recommendation systems.

4.2.5 Efficiency and Scalability of QBook

Besides assessing the effectiveness of *QBook* and its individual strategies considered on making book recommendations, we have also conducted the initial study and validated the overall efficiency of *QBook*. We consider 10 randomly selected users from *QEval* who rated a different numbers of books in the past (users with 5, 10, 20, 30, 40, 50, 60, 70, 80 and 90 rated books) and compute the processing time of *QBook* in generating recommendations for each of them. The average time in milliseconds required by *QBook* to generate book recommendations is 607.3 milliseconds which proves the efficiency of *QBook*. As determined by the curve created using the Microsoft Excel Trend/Regression tool (shown in Figure 4.7), the processing time of *QBook* as the number of analyzed books increases follows a linear trend, which demonstrates the scalability of *QBook*.

4.2.6 QBook versus Others

In this section, we demonstrate the correctness of *QBook* by comparing its performance with existing book recommendation methodologies, introduced in [41, 74, 83,

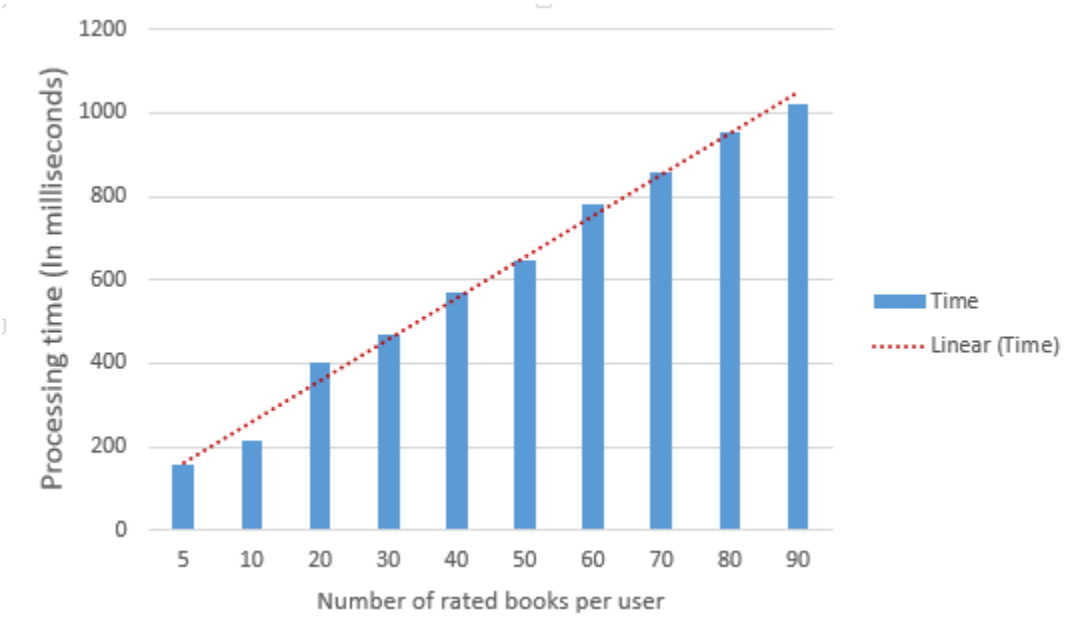

Figure 4.7: **(Average) Processing time of *QBook* for generating book recommendations for users who rated different number books in the past.**

85, 90, 129]. The empirical analyses reported in this section, are based on *QEval* dataset. A brief description of the baseline and state-of-the-art approaches considered in our evaluation is provided below:

- Matrix Factorization (SVD) is a popular baseline recommendation strategy as explained in Chapter 2.

- Content-based Filtering (CB) is another popular baseline approach used in generating suggestions, detailed in Chapter 2.

- LDAMF model [90] tries to harness the information in the review text by fitting an LDA model on the review text. The model uses topic distribution on items (or users), learned by gradient descent methods, as the latent factors in matrix factorization models.

- CTR (Collaborative Topic Regression) is a state-of-the-art method that recommends scientific articles to potential interested readers [129]. Even though the approach is not specifically created for book recommendations, it is comonly used in for the comparison purposes since it contains all characteristics of the books, except maybe the number of pages. The CTR model focuses on the one-class collaborative filtering problem by using different precision parameters. For comparison purposes, we used the results obtained in [83] where authors used the same precision parameters as well as LDA-C [45] for pre-training purposes. CTR also utilizes both ratings and reviews information to inform the recommendation process.

- HFT (Hidden Factors and Hidden Topics) is a state-of-the-art method that combines reviews with ratings [90]. This approach models the ratings using a matrix factorization model with an exponential transformation function to link the stochastic topic distribution in modeling the review text and the latent vector in modeling the ratings.

- The URP (User Rating Profile) model described in [89] is an extension of LDA for collaborative filtering. In this model, each user is represented as a mixture of so-called user attitudes, the rating for each item is generated by selecting a user attitude for the item, and then sampling a rating according to the preference pattern associated with that attitude.

- SVD++ [78] refers to a matrix factorization model which makes use of implicit feedback information. In general, implicit feedback can refer to any kind of users' history information that can help indicate users' preference. As a varia-

tion of SVD algorithm, and commonly used in recommendation community, we compare its performance with our approach.

- The authors in [74] proposed a Bayesian model, called User Rating and Review Profile (URRP), by combining collaborative filtering and content-based filtering to more accurately learn user rating and review preferences. To address cold-start problem, URRP links User Rating Profile with a topic model (Latent Dirichlet Allocation [45]) with the review text generated by users.

- 'Free Lunch' [85] leverages clusters based on information that is present in the user-item matrix, but not directly exploited during matrix factorization. The authors in [85] observe users and items in terms of their overall rating patterns and introduce an approach that uses Factorization Machines [105] to improve recommendation performance.

- Another model that combines content-based filtering with collaborative filtering by using the information of both ratings and reviews RMR (Ratings Meet Reviews) to improve prediction accuracy in a recommendation process is described in [83]. Ling et al. [83], apply topic modeling techniques on the review text and align the topics with rating dimensions to learn latent topics and explore the prior knowledge on items or users and recommend new items.

To better understand the correctness of the design methodology of *QBook*, we compare its performance with baselines, yet popular, algorithms: Matrix Factorization (SVD) and Content-based filtering (CB). While details of algorithms were discussed in Section 2, for their implementation we rely on LensKit. For this purpose, we calculated NDCG and MRR metrics. As shown in Table 4.4, *QBook* outperforms

Table 4.4: ***QBook* versus Baseline evaluating top-K recommendations**

Baseline Recommenders	NDCG	MRR
QBook	**0.933**	**0.670**
SVD	0.874	0.612
Content Based	0.874	0.585

both of the baselines. The significant ($p < 0.01$) NDCG improvement of *QBook*, with respect to SVD and CB, demonstrates that in general, recommendations provided by *QBook* are preferred over the ones provided by either baseline, which either consider ratings patterns or content, but not both, as part of the recommendation process.

To further prove the correctness of *QBook*, we evaluated its performance in terms of MRR. Thereafter, with compared *QBook*'s MRR with respect to the MRR of existing state-of-the-art recommendation strategies. As reported in Table 4.4, QBOok outperforms baselines in terms of MRR metrics, which means that a position of the first relevant book among suggested ones is higher than in the case with baseline recommendations. Furthermore, a higher NDCG indicates that the relevant recommendations generated by *QBook* are positioned higher in the ranking of generated recommendations than the ones provided by *QBook*'s counterparts.

Even if *QBook* outperforms traditional recommendations strategies (SVD and CB), in order to prove its true recommendation strength, we compare *QBook*'s performance with state-of-the-art methodologies. In Table 4.5 we summarized the results of the evaluation conducted using INEX Amazon/LibraryThing dataset in Table 4.5 in terms of RMSE and MSE.

As shown is Table 4.5, *QBook* outperforms existing state-of-the-art book recommenders considered in this empirical study in terms of predicting the degree of which a user would like to read each recommended book. The difference on RMSE and MAE scores computed for *QBook* with respect to aforementioned state-of-the-art book

Table 4.5: **Performance assessment of *QBook* versus state-of-the-art recommendation strategies, in terms of RMSE and MSE**

Book Recommenders	RMSE	MSE
QBook	**0.632**	**0.795**
MF	1.052	1.107
LDAMF	1.053	1.109
CTR	1.052	1.106
HFT	1.066	1.138
RMR	1.055	1.113
Free Lunch	0.933	0.986
Free Lunch with Clustering	0.825	0.681
SVD++	0.908	1.282
URRP	1.104	1.220

recommenders are statistically significant with $p > 0.001$, which means that *QBook* is able to provide recommendations that users find of use, i.e., they are relevant to their reading needs.

To assess *QBook*'s performance, we measure how successfully *QBook* is able to predict *QEval* rating scores based on recommendation method explained in Chapter 3. The prediction power of *QBook* is demonstrated with having the root mean square errors twice as low as in case of state-of-the-art approaches. Lower MSE and RMSE mean that predicted scores (quantify the degree to which the rating predicted for recommended book differs from the actual degree of interest (rating) of a given user on the recommended book) are closer to the ground truth. When analyzing the performance of different strategies in more detail, we can see that Matrix Factorization strategies perform better, as in the case of Free Lunch (with and without clustering) and SVD++. However, *QBook* not only considers MF approach in the process of selecting book candidates, but also complements its limitations when using content based approach during the candidate selection, as well as involving different perspectives including other users' and experts reviews.

CHAPTER 5

CONCLUSIONS

In this study we presented *QBook*, a novel book recommendation system. *QBook* acts as a personal docent and curates books to present them to each individual user and satisfy his reading needs. To create book exhibits, *QBooks* creates suggestions based on different areas of interest to a user, not only the most dominant or recent ones. Instead of gaining knowledge during the education process to inform the curation process, as in the case with professional curators, *QBook* takes advantage of users' historical data (such as ratings and reviews), items' metadata, experts' knowledge and distribution of interest patterns related to each individual user. *QBook* justifies each suggestions with a corresponding explanation, to provide a user with reasons for including each book in the generated exhibit.

The novelty of *QBook* consists of involving user-generated reviews, among other data sources, into its recommendation process but without considering the sentiment expressed in the reviews. *QBook* recommendation strategy is based on the well-known matrix factorization and content-based filtering methods and complemented by reviews, which are considered to infer the literary elements (features) each individual user cares about while reading a book. Even though users' reviews are generated by variety of users and therefore include different subjective opinions, *QBook* complements them by considering knowledge extracted from objective experts opinions which

positively influence curation of recommendations and generation of explanations.

One of the important contributions of *QBook* is the proposed genre prediction strategy that estimates possible interests of each individual user. The novelty of this genre-prediction strategy consists of incorporating an explicit time component to generate genre distribution. To the best of our knowledge, this is the first time that the well-known time series $ARIMA$ model is used to predict book genre of readers' interests. The described strategy provides successful predictions and outperforms the baseline for 77% of users based on the presented initial evaluation, while for the remaining users it provides predictions comparable to the baseline. By considering all the genre a user read in the past, *QBook* achieves diversity of provided suggestions.This is important since a user would not receive the same type of recommendation over time. This way *QBook* aims to satisfy different areas of user's interest and give a greater range of options to select from.

Another contribution of *QBook* lies in the explanation generation process that analyses other users' opinions on extracted features of interest for a given user and experts' objective critique to include them in the generated explanations associated with each suggested book. *QBook* also captures which data points have the highest influence in the process of selecting a book to read for each individual user and thus include that information in this explanations. The generated justifications help each user choose the best book to read, among the recommended ones, without any need to locate further information pertaining to the book.

We conducted a number of offline experiments to validate the performance of *QBook* using popular Amazon/LibraryThing and Library of Congress datasets. Based on the experimental results, we can conclude that *QBook* outperforms baselines, as well as a number of state-of-the-art book recommendation strategies, which further

demonstrates the importance of considering a number of different data sources, beyond solely users' ratings or book content to enhance the recommendation process. We also conducted initial experiments that verified the correctness and usefulness of the proposed explanation-generation strategy by comparing it with existing approaches. By conducting an in-depth assessment based on diversity, coverage and serendipity of provided recommendations, we showed that *QBook* addresses already known recommendation problems, including cold-start, data sparsity, personalization and diversity. Lastly, we analyzed the processing time of *QBook* and were able to demonstrate that *QBook* is efficient and scales well.

5.1 Applicability

While we used books as a case study, in terms of sample items to be recommended, we developed domain independent techniques and methodologies so that our system can be used to recommend any type of items (e.g. songs, movies or events) as long as we can provide it with necessary data points to generate recommendations. However, it is important to note that future assessments on other items are out of the scope of this thesis work.

5.2 Future Work

Even though we successfully completed the challenge presented in Chapter 1 and created a personal docent, *QBook*, which outperforms baseline and state-of-the-art approaches, we are aware of a need to further extend validation and demonstrate that *QBook* not only generates relevant suggestions but also helps users in making appropriate choices among provided suggestions.

As in the case of measuring explanations, offline metrics are not enough to showcase a performance of *QBook* as a personal docent. Consequentially, we propose in the future to rely on the popular, online $A - B$ test [87]. This type of evaluation is performed by including two groups of users directed towards two system, which differ in precisely one thing, and collect their results to observe changes in their behaviors [87]. We propose to conduct $A - B$ testing by asking groups of users to provide us with a number of books with their ranks and reviews, and use it as an input to generate curated suggestions and explanations. One group of users will be provided with curated suggestions, while the other group will receive non-curated recommendations. We will collect users' feedback through a questionnaire to measure the performance differences between tested systems and see the level of satisfaction of users after providing them new, unseen items.

Because of the scope of this study, the conducted evaluation showcases the genre prediction performance for a single user. We plan to conduct further assessments in terms of quantitatively determining the degree to which the proposed strategy provides successful genre predictions for libraries and publishing companies and influences the recommendation process to assist all three stakeholders. Furthermore, we believe that presented genre-prediction strategy can be applied on other metadata that could benefit from an in-depth analysis of preference change over time, as in the case of subject headings. This would aid the process of identifying more concretely user preferences and enable recommendation systems to provide diverse, but personalized suggestions, which is why we will also explore this in future work.

Given the domain-independent nature of our strategies and methodologies, we also plan to validate *QBook* on datasets other than books and demonstrate the applicability of the proposed recommendation strategy in domains other than books.

The evaluations presented in this manuscript are from a user perspective, however, we are interested in enhancing, if needed, the design of *QBook* to ensure that it can satisfy the needs for other stakeholders, such as libraries and publishing companies.

Our goal is to go even one step further and enable our personal curator to generate suggestions in multiple domains, based on a complete virtual footprint available for a user. For example, this recommender will be able to recommend restaurants based on the songs we liked or movies we watched, or even recommend events we should attend as a consequence of clothes we recently rated and reviewed. In that case, the recommendation system would know a user better than anyone else.

REFERENCES

[1] A snapshot of reading in America in 2013 pew research center. http://www.pewinternet.org/2014/01/16/a-snapshot-of-reading-in-america-in-2013/. Accessed: 2016-07-07.

[2] Amazon. https://www.amazon.com/. Accessed: 2016-05-06.

[3] Applied time series analysis. penstate eberly college of science stat 510. https://onlinecourses.science.psu.edu/stat510/node/47. Accessed: 2016-07-07.

[4] Book consumption per capita in the U.S. 2010-2014the statistics portal. http://www.statista.com/statistics/262631/number-of-books-read-by-us-adults-per-year/.

[5] Bundlr. http://bundlr.com/.

[6] CNN. http://www.cnn.com/.

[7] Content based recommendation system. http://eugenelin89.github.io/recommender_content_based/.

[8] e-bay. http://www.ebay.com/. Accessed: 2016-05-06.

[9] Facebook. https://www.facebook.com/.

[10] hotels.com. https://www.hotels.com/.

[11] INEX Amazon/LibraryThing book corpus. http://social-book-search.humanities.uva.nl/data/ALT_Nondisclosure_Agreements.html. Accessed: 2016-02-07.

[12] Introduction to ARIMA Models. tribune digital chicago tribune. http://people.duke.edu/~rnau/411arim.htm. Accessed: 2016-05-06.

[13] LensKit open-source tools for recommender systems. https://lenskit.org. Accessed: 2016-07-07.

[14] Library of Thing. https://www.loc.gov/. Accessed: 2016-02-07.

[15] LibraryThing. http://www.librarything.com. Accessed: 2016-05-06.

[16] Netflix. https://www.netflix.com/.

[17] New York Times. http://www.nytimes.com/. Accessed: 2016-05-06.

[18] NPR books. http://www.npr.org/books/. Accessed: 2016-02-07.

[19] Pandora. http://www.pandora.com/.

[20] Paper.li. http://paper.li/.

[21] Pinterest. https://www.pinterest.com/.

[22] Spotify. https://www.spotify.com/us/.

[23] Storify. https://storify.com/.

[24] The decline of the American book lover and why the downturn might be over. the atlantic. atlantic media company. http://www.theatlantic.com/business/archive/2014/01/the-decline-of-the-american-book-lover/283222/. Accessed: 2016-05-06.

[25] Twitter. https://twitter.com/.

[26] Why To Read: 10 Reasons Why Reading Books Will Save Your Life. http://whytoread.com/why-to-read-10-reasons-why-reading-books-will-save-your-life/. Accessed: 2016-08-07.

[27] WordNet princeton university. https://wordnet.princeton.edu. Accessed: 2016-07-07.

[28] Yahoo. https://www.yahoo.com/.

[29] Yelp. http://www.yelp.com/.

[30] Youtube. https://www.youtube.com/.

[31] Gediminas Adomavicius and Alexander Tuzhilin. Toward the next generation of recommender systems: A survey of the state-of-the-art and possible extensions. *IEEE Transactions on Knowledge and Data Engineering*, 17(6):734–749, 2005.

[32] Peter Afflerbach. The influence of prior knowledge and text genre on readers' prediction strategies. *Journal of Literacy Research*, 22(2):131–148, 1990.

[33] Deepak Agarwal and Bee-Chung Chen. flda: matrix factorization through latent dirichlet allocation. In *Proceedings of the Third ACM International Conference on Web Search and Data Mining*, pages 91–100. ACM, 2010.

[34] Amjad Almahairi, Kyle Kastner, Kyunghyun Cho, and Aaron Courville. Learning distributed representations from reviews for collaborative filtering. In *Proceedings of the 9th ACM Conference on Recommender Systems*, pages 147–154. ACM, 2015.

[35] Xavier Amatriain. Big & personal: data and models behind netflix recommendations. In *Proceedings of the 2nd International Workshop on Big Data, Streams and Heterogeneous Source Mining: Algorithms, Systems, Programming Models and Applications*, pages 1–6. ACM, 2013.

[36] Xavier Amatriain, Neal Lathia, Josep M Pujol, Haewoon Kwak, and Nuria Oliver. The wisdom of the few: a collaborative filtering approach based on expert opinions from the web. In *Proceedings of the 32nd International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 532–539. ACM, 2009.

[37] Jim Anderson, Ann Anderson, Jacqueline Lynch, and Jon Shapiro. Examining the effects of gender and genre on interactions in shared book reading. *Literacy Research and Instruction*, 43(4):1–20, 2004.

[38] Liliana Ardissono, Cristina Gena, Pietro Torasso, Fabio Bellifemine, Angelo Difino, and Barbara Negro. User modeling and recommendation techniques for personalized electronic program guides. In *Personalized Digital Television*, pages 3–26. Springer, 2004.

[39] Marko Balabanović and Yoav Shoham. Fab: content-based, collaborative recommendation. *Communications of the ACM*, 40(3):66–72, 1997.

[40] Trapit Bansal, Mrinal Das, and Chiranjib Bhattacharyya. Content driven user profiling for comment-worthy recommendations of news and blog articles. In *Proceedings of the 9th ACM Conference on Recommender Systems*, pages 195–202. ACM, 2015.

[41] Nicola Barbieri. Regularized gibbs sampling for user profiling with soft constraints. In *International Conference on Advances in Social Networks Analysis and Mining (ASONAM)*, pages 129–136. IEEE, 2011.

[42] Chahinez Benkoussas, Anaïs Ollagnier, and Patrice Bellot. Book recommendation using information retrieval methods and graph analysis. In *Working Notes for CLEF 2015 Conference, CLEF*, 2015.

[43] Christopher M Bishop. Pattern recognition. *Machine Learning*, 128, 2006.

[44] Roi Blanco, Diego Ceccarelli, Claudio Lucchese, Raffaele Perego, and Fabrizio Silvestri. You should read this! let me explain you why: explaining news recommendations to users. In *Proceedings of the 21st ACM International Conference on Information and Knowledge Management*, pages 1995–1999. ACM, 2012.

[45] David M Blei, Andrew Y Ng, and Michael I Jordan. Latent dirichlet allocation. *Journal of Machine Learning Research*, 3(Jan):993–1022, 2003.

[46] Christopher Borrelli. "Everybody's a curator." duke university. `http://articles.chicagotribune.com/2013-10-04/entertainment/ct-ae-1006-borrelli-curation-20131004_1_curator-fake-shore-drive-kristin-cavallari`. Accessed: 2015-12-06.

[47] John S Breese, David Heckerman, and Carl Kadie. Empirical analysis of predictive algorithms for collaborative filtering. In *Proceedings of the Fourteenth Conference on Uncertainty in Artificial Intelligence*, pages 43–52. Morgan Kaufmann Publishers Inc., 1998.

[48] Leo Breiman. Random forests. *Machine Learning*, 45(1):5–32, 2001.

[49] Robin Burke. Hybrid recommender systems: Survey and experiments. *User Modeling and User-Adapted Interaction*, 12(4):331–370, 2002.

[50] Pedro G Campos, Fernando Díez, and Iván Cantador. Time-aware recommender systems: a comprehensive survey and analysis of existing evaluation protocols. *User Modeling and User-Adapted Interaction*, 24(1-2):67–119, 2014.

[51] Hung-Chen Chen and Arbee LP Chen. A music recommendation system based on music data grouping and user interests. In *Proceedings of the Tenth International Conference on Information and Knowledge Management*, pages 231–238. ACM, 2001.

[52] Li Chen and Feng Wang. Sentiment-enhanced explanation of product recommendations. In *Proceedings of the 23rd International Conference on World Wide Web*, pages 239–240. ACM, 2014.

[53] Gobinda G Chowdhury. Natural language processing. *Annual Review of Information Science and Technology*, 37(1):51–89, 2003.

[54] W Bruce Croft, Donald Metzler, and Trevor Strohmann. *Search engines*. Pearson Education, 2010.

[55] Oxford English Dictionary. Oxford: Oxford university press, 1989.

[56] Yi Ding and Xue Li. Time weight collaborative filtering. In *Proceedings of the 14th ACM International Conference on Information and Knowledge Management*, pages 485–492. ACM, 2005.

[57] Nevena Dragovic and Maria Soledad Pera. Polarity-neutral assessment of reviews to inform the recommendation process. In press.

[58] Michael D Ekstrand, Michael Ludwig, Joseph A Konstan, and John T Riedl. Rethinking the recommender research ecosystem: reproducibility, openness, and lenskit. In *Proceedings of the Fifth ACM Conference on Recommender Systems*, pages 133–140. ACM, 2011.

[59] Andrea Esuli and Fabrizio Sebastiani. Sentiwordnet: A publicly available lexical resource for opinion mining. In *Proceedings of The International Conference on Language Resources and Evaluation*, volume 6, pages 417–422. Citeseer, 2006.

[60] Usama Fayyad, Gregory Piatetsky-Shapiro, and Padhraic Smyth. From data mining to knowledge discovery in databases. *AI Magazine*, 17(3):37, 1996.

[61] A Mufit Ferman, James H Errico, Peter van Beek, and M Ibrahim Sezan. Content-based filtering and personalization using structured metadata. In *Proceedings of the 2nd ACM/IEEE-CS Joint Conference on Digital Libraries*, pages 393–393. ACM, 2002.

[62] Ellen Gamerman. Everybody's an art curator. `http://www.wsj.com/articles/everybodys-an-art-curator-1414102402`.

[63] Gayatree Ganu, Noemie Elhadad, and Amélie Marian. Beyond the stars: Improving rating predictions using review text content. In *WebDB*, volume 9, pages 1–6. Citeseer, 2009.

[64] Angel Luis Garrido and Sergio Ilarri. Tmr: a semantic recommender system using topic maps on the items descriptions. In *European Semantic Web Conference*, pages 213–217. Springer, 2014.

[65] Mouzhi Ge, Carla Delgado-Battenfeld, and Dietmar Jannach. Beyond accuracy: evaluating recommender systems by coverage and serendipity. In *Proceedings of the Fourth ACM Conference on Recommender Systems*, pages 257–260. ACM, 2010.

[66] Fatih Gedikli, Dietmar Jannach, and Mouzhi Ge. How should i explain? a comparison of different explanation types for recommender systems. *International Journal of Human-Computer Studies*, 72(4):367–382, 2014.

[67] Sharon Givon and Victor Lavrenko. Predicting social-tags for cold start book recommendations. In *Proceedings of the Third ACM Conference on Recommender Systems*, pages 333–336. ACM, 2009.

[68] Nathaniel Good, J Ben Schafer, Joseph A Konstan, Al Borchers, Badrul Sarwar, Jon Herlocker, John Riedl, et al. Combining collaborative filtering with personal agents for better recommendations. In *Association for the Advancement of Artificial Intelligence/Innovative Applications of Artificial Intelligence Conference*, pages 439–446, 1999.

[69] Ziyu Guan, Can Wang, Jiajun Bu, Chun Chen, Kun Yang, Deng Cai, and Xiaofei He. Document recommendation in social tagging services. In *Proceedings of the 19th International Conference on World Wide Web*, pages 391–400. ACM, 2010.

[70] Jonathan L Herlocker, Joseph A Konstan, and John Riedl. Explaining collaborative filtering recommendations. In *Proceedings of the 2000 ACM Conference on Computer Supported Cooperative Work*, pages 241–250. ACM, 2000.

[71] Jonathan L Herlocker, Joseph A Konstan, Loren G Terveen, and John T Riedl. Evaluating collaborative filtering recommender systems. *ACM Transactions on Information Systems (TOIS)*, 22(1):5–53, 2004.

[72] Antonio Hernando, Jesús Bobadilla, Fernando Ortega, and Abraham GutiéRrez. Trees for explaining recommendations made through collaborative filtering. *Information Sciences*, 239:1–17, 2013.

[73] Sydney Jarrard. Retail sales at bookstores up in April american booksellers association. `http://www.bookweb.org/news/retail-sales-bookstores-april-33897`. Accessed: 2016-05-06.

[74] Mingming Jiang, Dandan Song, Lejian Liao, and Feida Zhu. A bayesian recommender model for user rating and review profiling. *Tsinghua Science and Technology*, 20(6):634–643, 2015.

[75] Salil Kanetkar, Akshay Nayak, Sridhar Swamy, and Gresha Bhatia. Web-based personalized hybrid book recommendation system. In *Advances in Engineering and Technology Research (ICAETR), 2014 International Conference on*, pages 1–5. IEEE, 2014.

[76] Gabriella Kazai, Daoud Clarke, Iskander Yusof, and Matteo Venanzi. A personalised reader for crowd curated content. In *Proceedings of the 9th ACM Conference on Recommender Systems*, pages 325–326. ACM, 2015.

[77] Dmitry Kislyuk, Yuchen Liu, David Liu, Eric Tzeng, and Yushi Jing. Human curation and convnets: Powering item-to-item recommendations on pinterest. *Computer Vision and Pattern Recognition*, 2015.

[78] Yehuda Koren. Factorization meets the neighborhood: a multifaceted collaborative filtering model. In *Proceedings of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 426–434. ACM, 2008.

[79] Yehuda Koren. Collaborative filtering with temporal dynamics. *Communications of the ACM*, 53(4):89–97, 2010.

[80] Yehuda Koren, Robert Bell, Chris Volinsky, et al. Matrix factorization techniques for recommender systems. *Computer*, 42(8):30–37, 2009.

[81] Greg Linden, Brent Smith, and Jeremy York. Amazon.com recommendations: Item-to-item collaborative filtering. *IEEE Internet computing*, 7(1):76–80, 2003.

[82] Gregory D Linden, Jennifer A Jacobi, and Eric A Benson. Collaborative recommendations using item-to-item similarity mappings. US Patent 6,266,649.

[83] Guang Ling, Michael R Lyu, and Irwin King. Ratings meet reviews, a combined approach to recommend. In *Proceedings of the 8th ACM Conference on Recommender systems*, pages 105–112. ACM, 2014.

[84] Chunxi Liu, Shuqiang Jiang, and Qingming Huang. Personalized online video recommendation by neighborhood score propagation based global ranking. In *Proceedings of the First International Conference on Internet Multimedia Computing and Service*, pages 244–253. ACM, 2009.

[85] Babak Loni, Alan Said, Martha Larson, and Alan Hanjalic. 'free lunch'enhancement for collaborative filtering with factorization machines. In *Proceedings of the 8th ACM Conference on Recommender systems*, pages 281–284. ACM, 2014.

[86] Andrii Maksai, Florent Garcin, and Boi Faltings. Predicting online performance of news recommender systems through richer evaluation metrics. In *Proceedings of the 9th ACM Conference on Recommender Systems*, pages 179–186. ACM, 2015.

[87] Christopher D Manning and Hinrich Schütze. *Foundations of statistical natural language processing*, volume 999. MIT Press, 1999.

[88] Christopher D Manning, Mihai Surdeanu, John Bauer, Jenny Rose Finkel, Steven Bethard, and David McClosky. The stanford corenlp natural language processing toolkit. In *ACL (System Demonstrations)*, pages 55–60, 2014.

[89] Benjamin M Marlin. Modeling user rating profiles for collaborative filtering. In *Advances in Neural Information Processing Systems*, page None, 2003.

[90] Julian McAuley and Jure Leskovec. Hidden factors and hidden topics: understanding rating dimensions with review text. In *Proceedings of the 7th ACM Conference on Recommender Systems*, pages 165–172. ACM, 2013.

[91] Sean Michael Mcnee. *Meeting user information needs in recommender systems*. Proquest, 2006.

[92] Prem Melville, Raymond J Mooney, and Ramadass Nagarajan. Content-boosted collaborative filtering for improved recommendations. In *Association for the Advancement of Artificial Intelligence/Innovative Applications of Artificial Intelligence Conference*, pages 187–192, 2002.

[93] George A Miller. The magical number seven, plus or minus two: Some limits on our capacity for processing information. *Psychological Review*, 63(2):81, 1956.

[94] Joanna Misztal and Bipin Indurkhya. Explaining contextual recommendations: Interaction design study and prototype implementation. In *Proceedings of the Joint Workshop on Interfaces and Human Decision Making for Recommender Systems (9th ACM Conference on Recommender Systems), Vienna (Austria)*, 2015.

[95] Tom M Mitchell. Machine learning book, 1997.

[96] C-C Musat, Yizhong Liang, and Boi Faltings. Recommendation using textual opinions. In *IJCAI International Joint Conference on Artificial Intelligence*, number EPFL-CONF-197487, pages 2684–2690, 2013.

[97] Angela Nyhout and Daniela K O'Neill. Mothers' complex talk when sharing books with their toddlers: Book genre matters. *First Language*, pages 115–131, 2013.

[98] Bruno Ohana and Brendan Tierney. Sentiment classification of reviews using sentiwordnet. In *9th. IT & T Conference*, page 13, 2009.

[99] Bo Pang and Lillian Lee. Opinion mining and sentiment analysis. *Foundations and Trends in Information Retrieval*, 2(1-2):1–135, 2008.

[100] You-Jin Park and Kun-Nyeong Chang. Individual and group behavior-based customer profile model for personalized product recommendation. *Expert Systems with Applications*, 36(2):1932–1939, 2009.

[101] Maria Soledad Pera and Yiu-Kai Ng. Automating readers' advisory to make book recommendations for k-12 readers. In *Proceedings of the 8th ACM Conference on Recommender Systems*, pages 9–16. ACM, 2014.

[102] Maria Soledad Pera and Yiu-Kai Ng. Exploiting the wisdom of social connections to make personalized recommendations on scholarly articles. *Journal of Intelligent Information Systems*, 42(3):371–391, 2014.

[103] Maria Soledad Pera and Yiu-Kai Ng. Analyzing book-related features to recommend books for emergent readers. In *Proceedings of the 26th ACM Conference on Hypertext & Social Media*, pages 221–230. ACM, 2015.

[104] Al Mamunur Rashid, Istvan Albert, Dan Cosley, Shyong K Lam, Sean M McNee, Joseph A Konstan, and John Riedl. Getting to know you: learning new user preferences in recommender systems. In *Proceedings of the 7th International Conference on Intelligent User Interfaces*, pages 127–134. ACM, 2002.

[105] Steffen Rendle. Factorization machines with libfm. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 3(3):57, 2012.

[106] Steffen Rendle and Lars Schmidt-Thieme. Online-updating regularized kernel matrix factorization models for large-scale recommender systems. In *Proceedings of the 2008 ACM Conference on Recommender Systems*, pages 251–258. ACM, 2008.

[107] Paul Resnick and Hal R Varian. Recommender systems. *Communications of the ACM*, 40(3):56–58, 1997.

[108] Francesco Ricci, Lior Rokach, and Bracha Shapira. *Introduction to recommender systems handbook*. Springer, 2011.

[109] Zurina Saaya, Rachael Rafter, Markus Schaal, and Barry Smyth. The curated web: a recommendation challenge. In *Proceedings of the 7th ACM conference on Recommender systems*, pages 101–104. ACM, 2013.

[110] Alan Said and Alejandro Bellogín. Comparative recommender system evaluation: benchmarking recommendation frameworks. In *Proceedings of the 8th ACM Conference on Recommender systems*, pages 129–136. ACM, 2014.

[111] Sunita Sarawagi and Sree Hari Nagaralu. Data mining models as services on the internet. *ACM SIGKDD Explorations Newsletter*, 2(1):24–28, 2000.

[112] Badrul Sarwar, George Karypis, Joseph Konstan, and John Riedl. Item-based collaborative filtering recommendation algorithms. In *Proceedings of the 10th International Conference on World Wide Web*, pages 285–295. ACM, 2001.

[113] J Ben Schafer, Dan Frankowski, Jon Herlocker, and Shilad Sen. Collaborative filtering recommender systems. In *The Adaptive Web*, pages 291–324. Springer, 2007.

[114] Andrew I Schein, Alexandrin Popescul, Lyle H Ungar, and David M Pennock. Methods and metrics for cold-start recommendations. In *Proceedings of the 25th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 253–260. ACM, 2002.

[115] Ramon Shaban. A guide to writing book reviews. *Australasian Journal of Paramedicine*, 4(3), 2015.

[116] Tianfeng Shang, Qing He, Fuzhen Zhuang, and Zhongzhi Shi. A new similarity measure based on preference sequences for collaborative filtering. In *Asia-Pacific Web Conference*, pages 384–391. Springer, 2013.

[117] Ahu Sieg, Bamshad Mobasher, and Robin Burke. Improving the effectiveness of collaborative recommendation with ontology-based user profiles. In *Proceedings of the 1st International Workshop on Information Heterogeneity and Fusion in Recommender Systems*, pages 39–46. ACM, 2010.

[118] Jimeng Sun, Christos Faloutsos, Spiros Papadimitriou, and Philip S Yu. Graphscope: parameter-free mining of large time-evolving graphs. In *Proceedings of the 13th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 687–696. ACM, 2007.

[119] Jimeng Sun, Dacheng Tao, and Christos Faloutsos. Beyond streams and graphs: dynamic tensor analysis. In *Proceedings of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 374–383. ACM, 2006.

[120] Simon Sweeney and Fabio Crestani. Effective search results summary size and device screen size: Is there a relationship? *Information Processing & Management*, 42(4):1056–1074, 2006.

[121] Panagiotis Symeonidis, Antonis Krinis, and Yannis Manolopoulos. Geosocialrec: Explaining recommendations in location-based social networks. In *East European Conference on Advances in Databases and Information Systems*, pages 84–97. Springer, 2013.

[122] Yining Teng, Lanshan Zhang, Ye Tian, and Xiang Li. A novel fahp based book recommendation method by fusing apriori rule mining. In *2015 10th International Conference on Intelligent Systems and Knowledge Engineering (ISKE)*, pages 237–243. IEEE, 2015.

[123] Nava Tintarev and Judith Masthoff. A survey of explanations in recommender systems. In *In the Data Engineering Workshop at 2007 IEEE 23rd International Conference*, pages 801–810. IEEE, 2007.

[124] Nava Tintarev and Judith Masthoff. Evaluating recommender explanations: Problems experienced and lessons learned for evaluation of adaptive systems. In *In the Workshop on User-Centred Design and Evaluation of Adaptive Systems in Association with UMAP*. Citeseer, 2009.

[125] Brendon Towle and Clark Quinn. Knowledge based recommender systems using explicit user models. In *Proceedings of the Association for the Advancement of Artificial Intelligence Workshop on Knowledge-Based Electronic Markets*, pages 74–77, 2000.

[126] Saúl Vargas and Pablo Castells. Rank and relevance in novelty and diversity metrics for recommender systems. In *Proceedings of the Fifth ACM Conference on Recommender systems*, pages 109–116. ACM, 2011.

[127] Jesse Vig, Shilad Sen, and John Riedl. Tagsplanations: explaining recommendations using tags. In *Proceedings of the 14th International Conference on Intelligent User Interfaces*, pages 47–56. ACM, 2009.

[128] Michael Walker. Random forests algorithm. data science central. `http://datasciencecentral.com/profiles/blogs/random-forests-algorithm`. Accessed: 2016-05-06.

[129] Chong Wang and David M Blei. Collaborative topic modeling for recommending scientific articles. In *Proceedings of the 17th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 448–456. ACM, 2011.

[130] Jian Wang and Yi Zhang. Opportunity model for e-commerce recommendation: right product; right time. In *ACM SIGIR*, pages 303–312, 2013.

[131] Liang Xiang, Quan Yuan, Shiwan Zhao, Li Chen, Xiatian Zhang, Qing Yang, and Jimeng Sun. Temporal recommendation on graphs via long-and short-term preference fusion. In *Proceedings of the 16th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 723–732. ACM, 2010.

[132] Chenxing Yang, Baogang Wei, Jiangqin Wu, Yin Zhang, and Liang Zhang. Cares: a ranking-oriented cadal recommender system. In *Proceedings of the 9th ACM/IEEE-CS Joint Conference on Digital Libraries*, pages 203–212. ACM, 2009.

[133] Christopher C Yang. Search engines information retrieval in practice, 2010.

[134] Tithrottanak You, Ahmad Nurzid Rosli, Inay Ha, and Geun-Sik Jo. Clustering method based on genre interest for cold-start problem in movie recommendation. *Journal of Intelligence and Information Systems*, 19(1):57–77, 2013.

[135] Yongfeng Zhang, Guokun Lai, Min Zhang, Yi Zhang, Yiqun Liu, and Shaoping Ma. Explicit factor models for explainable recommendation based on phrase-level sentiment analysis. In *Proceedings of the 37th International ACM SIGIR Conference on Research & Development in Information Retrieval*, pages 83–92. ACM, 2014.

APPENDIX A

LITERARY ELEMENTS

In this section we provide complete sets of related terms for each literary elements used in Section 3.2 as described in [101].

- **Characterization** : stereotypes detailed distant dramatic eccentric evocative faithful familiar introspective laudatory lifelike multiple points of view quirky realistic layered linear literary recognizable sympathetic vivid well-developed well-drawn developed real believable

- **Frame** : descriptive minimal bleak light-hearted fun bitter-sweet bittersweet comfortable contemporary darker detailed setting detailed edgy evocative evangelistic exotic foreboding gritty hard-edged heart-warming historical humorous journalistic literary lush magical melodramatic menacing mystical nightmare philosophical political popular psychological romantic rural scholarly sensual small-town stark suspenseful timeless upbeat urban small town school

- **Writing Style**: austere candid classic colourful complex concise conversational direct dramatic dry elaborate extravagant fervent flamboyant frank graceful homespun jargon laconic metaphorical natural ornate passionate poetic polished prosaic restrained seemly showy simple sophisticated stark thoughtful unaffected unembellished unpretentious unusual

- **Pacing**: fast slow leisurely breakneck compelling deliberate leisurely-paced measured relaxed unhurried dense engrossing easy fast-paced stately
- **Tone**: happy light uplifting dark ironic funny evocative serious old-fashioned traditional modern contemporary sentimental emotional stark realistic safe relaxing suspenseful tense quirky wordplay bizarre surreal humorous
- **Story-line**: action-oriented character-centered cinematic complex conclusive domestic episodic violent family-centered folksy gentle humorous inspirational investigative issue-oriented layered linear literary multiple storylines mystical mythic journalistic literary lush magical melodramatic menacing mystical nightmare philosophical open-ended plot-centered plot twists racy rich romp sexually explicit steamy strong language thought-provoking tragic multiple

APPENDIX B

BOOK GENRES

In this section we provide complete list of major genres of literature types used in genre prediction strategy described in Section 3.4.

- Narrative
- Non fiction
- Essay
- Biography
- Autobiography
- Speech
- History
- Fiction
- Drama
- Poetry
- Fantasy
- Humor

- Fable
- Fairy tale
- Science fiction
- Short story
- Realistic fiction
- Folklore
- Historical fiction
- History
- Horror
- Tall tale
- Legend
- Mystery
- Mythology
- Fiction in verse