

UNIVERSITY OF MINNESOTA

This is to certify that I have examined this copy of a doctoral dissertation by

Sean Michael McNee

and have found that it is complete and satisfactory in all respects,
and that any and all revisions required by the final
examining committee have been made.

Joseph A. Konstan

Name of Faculty Advisor

Signature of Faculty Advisor

Date

GRADUATE SCHOOL

Meeting User Information Needs in Recommender Systems

A DISSERTATION
SUBMITTED TO THE FACULTY OF THE GRADUATE SCHOOL
OF THE UNIVERSITY OF MINNESOTA
BY

Sean Michael McNee

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR THE DEGREE OF
DOCTOR OF PHILOSOPHY

Joseph A. Konstan, Advisor

June 2006

Public Pre-print
Copyright © Sean Michael McNee 2006

ACKNOWLEDGEMENTS

Many people helped make this dissertation a reality. First, I would like to thank my thesis committee, John Riedl, John Carlis, Paul Garrett, and Joseph Konstan. I would especially like to thank Joe for being a fantastic advisor; I have learned more from you than you realize. Thank you to my paper writing colleagues, especially Roberto Torres, Jr. and Cai-Nicholas Ziegler, it was wonderful working with you. Thank you to everyone in GroupLens Research, especially Dan Cosley, Al Mamunur (Mamun) Rashid, and Shyong (Tony) Lam. Because of your support, encouragement, and friendship, GroupLens Research is a special place. Thanks also those who've already gotten their Ph.D. from Minnesota for their support and encouragement, including Jon Herlocker, Brad Miller, J. Ben Schafer, Brian Bailey, Harini Veeraraghavan, and Doug Perrin. I would also like to thank Ed H. Chi for his support and advice during my graduate school career. I have been blessed with many great teachers and professors, especially Jack Goldfeather and Jeff Ondich from Carleton College who showed me the love of computer science, and my sophomore year high school English teacher, Mrs. Skoy. As far as I know, I still have extra credit "in her heart". Finally, I want to thank my wonderful group of friends, my family, and everyone who helped and supported me through this process. I could not have done it otherwise.

In a different set of thanks, I want to thank the University of Minnesota Computer Science Department Systems Staff for keeping all of the machines running. Thanks also to NEC Research for the ResearchIndex data, thanks to BookCrossing.com for allowing us to collect data and run experiments, and thanks to the ACM for providing us with data from the ACM Digital Library.

For my parents, who always believed

ABSTRACT

In order to build relevant, useful, and effective recommender systems, researchers need to understand why users come to these systems and how users judge recommendation lists. Today, researchers use accuracy-based metrics for judging goodness. Yet these metrics cannot capture users' criteria for judging recommendation usefulness. We need to rethink recommenders from a user's perspective: they help users find new information. Thus, not only do we need to know about the user, we need to know what the user is looking for. In this dissertation, we explore how to tailor recommendation lists not just to a user, but to the user's current *information seeking task*. We argue that each recommender algorithm has specific strengths and weaknesses, different from other algorithms. Thus, different recommender algorithms are better suited for specific users and their information seeking tasks. A recommender system should, then, select and tune the appropriate recommender algorithm (or algorithms) for a given user/information seeking task combination. To support this, we present results in three areas. First, we apply recommender systems in the domain of peer-reviewed computer science research papers, a domain where users have external criteria for selecting items to consume. The effectiveness of our approach is validated through several sets of experiments. Second, we argue that current recommender systems research is not focused on user needs, but rather on algorithm design and performance. To bring users back into focus, we reflect on how users perceive recommenders and the recommendation process, and present Human-Recommender Interaction theory, a framework and language for describing recommenders and the recommendation lists they generate. Third, we look to different ways of evaluating recommender systems algorithms. To this end, we propose a new set of recommender metrics, run experiments on several recommender algorithms using these metrics, and categorize the differences we discovered. Through Human-Recommender Interaction and these new metrics, we can bridge users and their needs with recommender algorithms to generate more useful recommendation lists.

TABLE OF CONTENTS

CHAPTER 1 INTRODUCTION	1
THESIS STATEMENT.....	3
<i>External Validation</i>	4
<i>The Curse of Accuracy</i>	5
BUILDING BRIDGES	7
RESEARCH APPROACH AND CONTRIBUTIONS	9
<i>Recommending Research Papers</i>	10
<i>A User-Centric Approach to the Recommendation Process</i>	11
<i>Understanding Recommender Algorithms</i>	11
DISSERTATION ORGANIZATION	12
CHAPTER 2 RELATED AND PREVIOUS WORK.....	14
RECOMMENDER SYSTEMS AND PERSONALIZATION	14
<i>Content-Based Recommenders</i>	15
<i>Collaborative Filtering</i>	18
<i>Case-Based Reasoning and Conversational Recommenders</i>	21
A SUMMARY OF COMMON RECOMMENDER PROBLEMS.....	23
<i>Common Problems when Evaluating Recommender Algorithms</i>	23
<i>Problems with Collaborative Recommender Algorithms</i>	24
<i>Problems with Content-based Recommender Algorithms</i>	25
<i>Problems with Knowledge-based Recommender Algorithms</i>	26
CITATION INDEXING AND RECOMMENDING RESEARCH PAPERS	27
THEORIES OF INFORMATION SEEKING	28
SEARCH, SEARCH ENGINES, AND DIGITAL LIBRARIES	30
CONCLUSION	32
CHAPTER 3 CONCERNING RECOMMENDING, RECOMMENDER ALGORITHMS, AND RECOMMENDER METRICS	33
BACKGROUND AND DEFINITIONS	33
<i>The Ratings Matrix</i>	34
<i>The Recommendation Process</i>	36
USER-BASED COLLABORATIVE FILTERING	38
ITEM-BASED COLLABORATIVE FILTERING.....	43
EXTENSIONS TO COLLABORATIVE FILTERING ALGORITHMS	46
<i>Denser Collaborative Filtering</i>	47
<i>Symmetric Collaborative Filtering</i>	48
NAÏVE BAYES CLASSIFIER.....	49
PROBABILISTIC LATENT SEMANTIC ANALYSIS.....	51
TF/IDF CONTENT-BASED FILTERING	55
PREDICTIVE ACCURACY AND DECISION SUPPORT METRICS	59
CONCLUSION	63
CHAPTER 4 RECOMMENDATION LISTS AND INTRA-LIST SIMILARITY	64
INTRA-LIST SIMILARITY	65
EXPERIMENTS.....	66
<i>Offline Experiments</i>	67

<i>Online Experiments</i>	71
DISCUSSION AND IMPLICATIONS.....	77
CHAPTER 5 RECOMMENDING CITATIONS FOR RESEARCH PAPERS	79
INTEGRATING RECOMMENDERS INTO THE DOMAIN OF RESEARCH PAPERS	80
PURE RECOMMENDER ALGORITHMS FOR RESEARCH PAPERS.....	83
<i>Collaborative Algorithms</i>	84
<i>Baseline Algorithms</i>	85
PURE ALGORITHM EXPERIMENTS	86
<i>Offline Experiment</i>	86
<i>Online Experiment</i>	90
<i>Pure Algorithm Discussion</i>	98
COMBINING CONTENT AND COLLABORATIVE FILTERING.....	100
HYBRID RECOMMENDER ALGORITHMS IN THIS DOMAIN.....	102
HYBRID RECOMMENDER EXPERIMENTS	106
<i>Offline Experiment</i>	109
<i>Online Experiment</i>	112
<i>Hybrid Algorithm Discussion</i>	122
CONCLUSION (A FUNNY THING HAPPENED...).....	124
CHAPTER 6 HUMAN-RECOMMENDER INTERACTION THEORY	126
THE GULF OF INTENTION.....	127
HUMAN RECOMMENDER INTERACTION: A USER-CENTRIC PERSPECTIVE.....	128
THE PILLARS OF HRI.....	130
<i>The Recommendation Dialog</i>	130
<i>The Recommender Personality</i>	131
<i>The User Information Seeking Task</i>	132
THE ASPECTS OF HRI.....	133
<i>Aspects of the Recommendation Dialog</i>	134
<i>Aspects of the Recommender Personality</i>	139
<i>Aspects of the User Information Seeking Task</i>	145
THE HRI ANALYTIC PROCESS MODEL	148
APPLYING HRI AND THE PROCESS MODEL TO RECOMMENDER DESIGN.....	150
LIMITATIONS OF HRI AND THE PROCESS MODEL	151
CONCLUSIONS	153
CHAPTER 7 RECOMMENDER TASKS IN A DIGITAL LIBRARY.....	154
FOUR KINDS OF USERS	154
USER TASKS	156
RELATIONSHIPS BETWEEN TASKS.....	159
<i>The Large Set Phenomenon</i>	160
<i>The Fluidity of "Find More References"</i>	160
PERSONAS IN THIS DOMAIN	161
APPLYING HRI TO THE DOMAIN OF DIGITAL LIBRARIES	163
CONCLUSIONS	167
CHAPTER 8 UNDERSTANDING RECOMMENDER ALGORITHMS, PART I: DESIGNING METRICS, RUNNING BENCHMARKS	169
NEW RECOMMENDER METRICS	169
PREVIOUSLY EXISTING RECOMMENDER METRICS.....	177
METRIC DISCUSSION	178
BENCHMARKING RECOMMENDER ALGORITHMS	178
<i>Research Questions</i>	178

<i>Experimental Design and Setup</i>	179
<i>Experiment Results</i>	180
<i>Experiment Discussion</i>	192
HRI AND RECOMMENDER ALGORITHMS	195
APPLYING HRI TO THE DOMAIN OF DIGITAL LIBRARIES, REVISITED	196
<i>Discussion and Limitations</i>	198
CONCLUSION	198
CHAPTER 9 UNDERSTANDING RECOMMENDER ALGORITHMS, PART II: A USER	
EVALUATION	199
OUR RECOMMENDER ALGORITHMS.....	199
USER STUDY.....	204
<i>Experimental Design</i>	205
<i>Experiment Walkthrough</i>	207
<i>Results</i>	210
<i>Analysis and Discussion</i>	213
IMPLICATIONS AND FUTURE WORK	217
CONCLUSION	219
CHAPTER 10 IMPLICATIONS, FUTURE WORK, AND CONCLUSIONS	221
SUMMARY OF CONTRIBUTIONS AND IMPLICATIONS	221
<i>Recommending Research Papers</i>	221
<i>Re-examining the Recommendation Process</i>	222
<i>Understanding Recommender Algorithms</i>	223
FUTURE WORK	224
CONCLUSION	226

LIST OF TABLES

TABLE 4-1: HYPOTHETICAL RECOMMENDATION LISTS FOR ROBERT HEINLEIN	65
TABLE 4-2: PRECISION AND RECALL FOR NON-DIVERSIFIED CF	68
TABLE 4-3: MULTIPLE LINEAR REGRESSION RESULTS	76
TABLE 5-1: THE NUMBER OF SUBJECTS FOR EACH RECOMMENDER ALGORITHM	92
TABLE 5-2: OVERALL USER OPINION, IN PERCENTAGES	94
TABLE 5-3: USER PROFILING ALTERNATIVES	108
TABLE 5-4: NUMBER OF USERS PER ALGORITHM	116
TABLE 5-5: OVERALL USER SATISFACTION	117
TABLE 5-6: RECOMMENDED ALGORITHM BY PAPER CLASS	118
TABLE 5-7: DISTRIBUTION OF USERS IN ONLINE EXPERIMENT	119
TABLE 7-1: THE FLUIDITY OF 'FIND MORE REFERENCES'	161
TABLE 7-2: EXAMPLE USER TYPE/USER TASK MATRIX	163
TABLE 7-3: EXAMPLE HRI ASPECT MAPPINGS FOR USER TYPES	166
TABLE 7-4: EXAMPLE HRI ASPECT MAPPING FOR USER TASKS	167
TABLE 8-1: ALGORITHM AND METRIC LISTINGS	179
TABLE 8-2: SUMMARY OF RESULTS BY ALGORITHM	186
TABLE 8-3: HRI MAPPINGS FOR RECOMMENDER METRICS IN THE DOMAIN OF RESEARCH PAPERS	196
TABLE 9-1: SUMMARY OF RECOMMENDER ALGORITHM PROPERTIES	200
TABLE 9-2: AVAILABLE INFORMATION SEEKING TASKS	207
TABLE 9-3: SUMMARY OF ALL SURVEY QUESTIONS	209
TABLE 9-4: NUMBER OF USERS PER EXPERIMENTAL CONDITION	210
TABLE 9-5: OVERLAP OF TOP-10 RECOMMENDATION LISTS, BY BASKET SIZE	214
TABLE 9-6: ORDER EFFECT FOR QUESTION 2-2	215

LIST OF FIGURES

FIGURE 1-1: THE INTENTION GAP BETWEEN USERS AND RECOMMENDERS	8
FIGURE 1-2: THE HUMAN-RECOMMENDER INTERACTION PROCESS MODEL	9
FIGURE 3-1: A CONCEPTUAL RATINGS MATRIX.....	35
FIGURE 3-2: THE RECOMMENDATION PROCESS MODEL.....	37
FIGURE 3-3: AN INSTANCE OF THE RECOMMENDATION PROCESS MODEL.....	38
FIGURE 3-4: RATING MATRIX EXAMPLE WITH CARS AND PAINT	47
FIGURE 3-5: SELF-REFLECTIVE RATINGS MATRIX AS DIRECTED GRAPH	48
FIGURE 4-1: PRECISION (A) AND RECALL (B) FOR INCREASING DIVERSITY	69
FIGURE 4-2: INTRA-LIST SIMILARITY BEHAVIOR (A) AND OVERLAP WITH PURE CF LISTS (B)	70
FIGURE 4-3: RESULTS FOR SINGLE-VOTE AVERAGES (A), COVERED RANGE IF INTERESTS (B), AND OVERALL SATISFACTION (C).....	73
FIGURE 5-1: OUR RATINGS MATRIX FOR RESEARCH PAPERS.....	82
FIGURE 5-2: FOR REMOVED CITATIONS THAT AN ALGORITHM WAS ABLE TO RECOMMEND, THE PERCENTAGE OF CITATIONS RECOMMENDED FIRST, AND IN THE TOP 10, 20, 30, OR 40 BY EACH ALGORITHM	89
FIGURE 5-3: FOR ALL REMOVED CITATIONS, THE PERCENTAGE OF CITATIONS RECOMMENDED FIRST, AND IN THE TOP 10, 20, 30, OR 40 BY EACH ALGORITHM.....	89
FIGURE 5-4: QUALITY OF INDIVIDUAL RECOMMENDATIONS	93
FIGURE 5-5: NOVELTY OF INDIVIDUAL RECOMMENDATIONS	94
FIGURE 5-6: FINDING RELATED WORK RESULTS	95
FIGURE 5-7: FINDING PAPERS TO READ RESULTS	96
FIGURE 5-8: MODEL INSTANCE FOR HYBRID RECOMMENDER ALGORITHMS	101
FIGURE 5-9: THE CBF-COMBINED AND CBF-SEPARATED ALGORITHMS	104
FIGURE 5-10: THE FUSION ALGORITHM	106
FIGURE 5-11: OFFLINE SUMMARY FOR TOP FIVE ALGORITHMS	110
FIGURE 5-12: A SCREENSHOT FROM TECHLENS+	113
FIGURE 5-13: OVERALL USER SATISFACTION BY ALGORITHM.....	117
FIGURE 6-1: THE PILLARS AND ASPECTS OF HRI	134
FIGURE 6-2: THE HRI ANALYTIC PROCESS MODEL	149
FIGURE 8-1: OVERVIEW OF THE ADAPTABILITY METRIC	176
FIGURE 8-2: POPULARITY RESULTS.....	181
FIGURE 8-3: RATABILITY RESULTS	182
FIGURE 8-4: ADJUSTED RANK RESULTS	183
FIGURE 8-5: ADAPTABILITY, EQUAL-SPLIT RESULTS.....	184
FIGURE 8-6: ADAPTABILITY, ADAPT-HEAVY RESULTS	185
FIGURE 8-7: POPULARITY NEIGHBORHOOD RESULTS.....	188
FIGURE 8-8: HYBRID POPULARITY NEIGHBORHOOD RESULTS	188
FIGURE 8-9: RATABILITY NEIGHBORHOOD RESULTS	189
FIGURE 8-10: HYBRID RATABILITY NEIGHBORHOOD RESULTS	189
FIGURE 8-11: ADAPTABILITY, EQUAL-SPLIT NEIGHBORHOOD RESULTS.....	191
FIGURE 8-12: HYBRID ADAPTABILITY, EQUAL-SPLIT NEIGHBORHOOD RESULTS	192
FIGURE 9-1: THE AUTHOR SELECTION PAGE.....	205
FIGURE 9-2: THE PAPER AND CITATION SELECTION PAGE	205
FIGURE 9-3: THE RECOMMENDATION LIST SCREEN	206
FIGURE 9-4: RESULTS FOR FIRST THREE SURVEY QUESTIONS	211
FIGURE 9-5: RESULTS FOR SECOND THREE SURVEY QUESTIONS	211
FIGURE 9-6: USER OPINION OF SUITABILITY FOR CHOSEN TASK	212

CHAPTER 1

INTRODUCTION

“Every day, approximately 20 million words of technical information are recorded. A reader capable of reading 1000 words per minute would require 1.5 months, reading eight hours every day, to get through one day's output, and at the end of that period he would have fallen 5.5 years behind in his reading.”

Hubert Murray Jr,

Methods for Satisfying the Needs of the Scientist and the Engineer for Scientific and Technical Communication. [100]

In 1996, Reuters performed a survey of IT managers and found many of them suffered health problems from information overload. Other effects found in the study included, “anxiety, poor decision-making, difficulties in memorizing and remembering, and reduced attention span” [56]. Information overload is real, and it is getting to the point where people are becoming physically ill trying to keep up. Hubert Murray's quotation above is even more telling than it first appears. It is from 1966, before the introduction of the personal computer, before the explosion of the World Wide Web, before the ‘Information Age’.¹

Fortunately, there are recommender systems. Recommenders quickly and efficiently sort through vast quantities of information and bring the relevant pieces of information to our attention. In doing so, recommenders help us navigate through complex information spaces, allowing us to be more efficient and well, healthier.

¹ Also, the ACM had around 10,000 members and had just awarded its first Turing Award.

Humans have acted as recommenders for many years: newspaper editors, movie critics, Consumer Reports, and postings on E-pinions.com are all examples of people acting as recommenders for others. People provide their opinions for others to use when making a similar decision. This form of recommendation is called *collaborative filtering*. A target person seeks out the opinions of other people who are similar to her (e.g. collaborators) and uses their opinions to inform her decision making process.

Just as computers exacerbate the information overload problem, they also help solve it. In the last 12 years, collaborative filtering has been enhanced through computer algorithms. By combining the strengths of a computer, sifting through data, with the strengths of a person, interpreting people's opinions, automated collaborative filtering-based recommender systems help users receive high quality personalized recommendations.

Because of these strengths, automated collaborative filtering (a.k.a. CF) is popular with many e-commerce websites, appearing at Amazon.com, Yahoo! Music, and Netflix, among others. In each case, CF recommends products the user might not have known about otherwise, enhancing the user's experience and improving the company's bottom line. However, recommenders, both computers and humans, do not always generate good recommendations; sometimes things can go quite wrong. For example, here is a true anecdote:

John is the hero of our story. On a Friday evening, he was going to take a date to see a movie. Unfortunately, he did not know what movie to see. He asked his closest friend for advice on which movie was the best one playing. His friend immediately answered, "A Clockwork Orange. You will love that movie." John was skeptical but followed his friend's advice. The date went horribly. While he did enjoy the movie himself, his date did not care for it at all.

What went wrong? The problem was not that the recommendation was bad; on the contrary, the recommendation was exactly right. The context was wrong; the friend

assumed John was not going on a date, and John failed to mention it. This kind of miscommunication between recommenders and users still plagues us today. As computers have become recommenders, this problem has gotten worse. While it would have been easy for John to correct his friend, it is more difficult, for example, to correct your TiVo when it incorrectly thinks you are gay, or when Amazon.com incorrectly thinks you love all cartoons [160].

Yet, recommenders have enjoyed great success in many domains, including movies, books, music, and jokes. It is not an issue of quality; recommenders make accurate recommendations. It is an issue of context and purpose—*why* does the user want a recommendation? What is he going to do with it? This problem has been discussed in the recommender systems literature. Herlocker et al. described it best in [51] when they state: “There is an emerging understanding that good recommendation accuracy alone does not give users of recommender systems an effective and satisfying experience. Recommender systems must provide not just accuracy, but also *usefulness*.” (Emphasis in original)

We believe that a deeper understanding of ‘why’ can only help improve the quality and usefulness of recommenders. Much like the friend in our story, if computer recommenders knew why someone wanted a movie recommendation, they could tailor recommendations not just to the person, but also to the person’s current need. Of course, understanding why someone wants a recommendation can be elusive.

Thesis Statement

We believe *information seeking theory* provides a framework to answer this question of *why* [20]. We assume that the user has an information need and has come to a recommender as a part of her information seeking behavior. Given this assumption, we state our thesis as the following:

In a recommender system, selecting and tuning the appropriate recommender algorithm for both the user and the user's current information seeking task will generate a more useful recommendation list than a generic or un-tuned algorithm.

In this dissertation, we develop new theoretical models, run offline simulation experiments, and conduct user studies in support of this thesis. Before outlining our research approach, there are two points we must first consider:

- 1. Importance of external validation criteria**

As recommenders move into domains with additional validation criteria, the importance of considering the user's information seeking task will increase

- 2. The current state of recommender metrics**

Current predictive accuracy and decision support metrics are poor at evaluating the suitability of a recommender algorithm for an information seeking task [86]

External Validation

One of the criticisms of recommender systems up until this point is that they have mostly appeared in 'low-cost' entertainment domains where decisions are based on taste, such as movies, music, television, books, jokes, etc. This is a valid criticism, if recommenders are going to be accepted by a larger audience, they need to generate high quality recommendations in domains where taste is not the main deciding factor of consuming an item.

We posit that users come to a recommender as part of an information seeking task. More important, we believe the importance of the information seeking task varies from domain to domain. In entertainment-based domains, the decision of whether to consume a recommendation is a *tasted-based* decision. The user determines if she likes the item, and then acts accordingly. The information-seeking task is one-dimensional—how much do I like it?

If we move into a non-entertainment domain, e.g. the domain of peer-reviewed research papers, we claim the decision of whether to consume the item becomes more complex. While taste may be one component, other *non-taste* criteria also must be satisfied. For example, if I am looking to add references to a paper I am writing, high quality paper recommendations from a different research area will not help me, independent of how much I might enjoy reading them. My external, non-taste criteria places constraints on what I choose to consume. Until this point, recommenders have dealt with one criterion at a time, historically, a taste-based criterion. Users may have any number of criteria when visiting a recommender system, independent of domain. For example, a user of a movie recommender may be limited to items released on VHS tape. To understand context, recommenders need to generate recommendations based on all of user's criteria. We claim these external criteria are representations of the user's information seeking task. Thus, as the number and/or importance of these criteria increase, the more important our thesis becomes. If we want to tailor recommendations to a user's information seeking task, we first need to understand what these criteria are for a given user in a given domain.

The Curse of Accuracy

Researchers and practitioners have long used accuracy to judge the goodness of recommender algorithms. Many metrics have been proposed and used to measure accuracy, including ROC curves [55], modifications to precision and recall [131], Breese's Half-Life metric [13], and, most commonly used, Mean Absolute Error (MAE) [13, 55, 135]. For example, [51] provides an analysis of 432 variants of User-User Collaborative Filtering algorithm run against 11 accuracy metrics. While accuracy is an important component of a recommender algorithm, focusing solely on it leads to two different problems.

The first problem comes from the way accuracy-centric analysis views the recommendation process. At a high level, this process is one where a user, either with or without establishing a user model, makes a request of the recommender in the form of a basket of items and ratings. The recommender algorithm performs a computation based

on this basket and returns a recommendation list to this user. In this setting, each request sent to the recommender is an independent event done in isolation of all other recommendation events. While this may be true for the algorithm itself, it is not true for the user.

Users see each recommendation in the context of other recommendations—in a list. Independently good recommendations may create a bad recommendation list. For example, if we assume that Tolkien’s *Lord of the Rings* would be a good book recommendation, would a list containing ten different editions of that book (e.g. hardcover, paperback, split into three volumes, combined into one volume, etc.) be a good recommendation list? As we argue in Chapter 4, we believe recommendation lists need to be evaluated as a single recommendation entity. Moreover, the recommendation process is iterative. One recommendation list is evaluated in the context of previous lists the user has seen. The user will have different opinions of a consistent algorithm compared to an inconsistent one. A metric that judges independent events, even recommendation lists, will miss this temporal component of the recommendation process. The fact that users return to recommenders is an important aspect of recommendation process.

The second problem is an artifact of the standard approach used to measure accuracy. The leave- n -out methodology [13], works by splitting collected ratings data into test and train datasets, removing n ratings from each test line, and recording how well the recommender can predict back the removed ratings. In essence, you hide a portion of the data and check how well the recommender reconstructs it. Leave- n -out is commonly used in machine learning to test the accuracy of classification algorithms [97].

If we approach leave- n -out from the user’s perspective, it is equivalent to recommending items the user has already rated. This is not as useful as it could be. For example, when looking for recommendations on new places to visit on vacation, a travel guide book only containing information on places you have visited before is not helpful. Moreover, if the guide recommended some places you’ve never been—say Beijing and Prague—then the leave- n -out methodology penalizes it for failing to recommend instead

the places you have been, even if you'd like Beijing better than Boston or Prague better than Paris. Even if the book correctly ordered the places such that your favorites were first, this book is useless for planning a vacation. Accuracy metrics which reward algorithms for generating recommendation of withheld items are not measuring how well an algorithm performs on the task users care about: generating recommendations for items they have not seen.

The problem is rooted in the difference between a classifier and a recommender. Classifiers segment spaces into their most probable classes. In recommendation terms, a classifier would find, and recommend, the items the user is most like to rate next. While these items would have high “ratability” for that user, there is no guarantee that these items will help users with their information seeking tasks. For instance, an online music store used a User-based collaborative filtering algorithm to generate recommendations. The most common recommendation was for the Beatle’s “White Album”. From an accuracy perspective, these recommendations were dead-on: most users like that album very much. From a usefulness perspective, though, the recommendations were a complete *failure*: every user either already owned the “White Album”, or had specifically chosen not to own it. Even though it was highly ratable, “White Album” recommendations were almost never acted on by users, because they added almost no value.

In order to tailor recommendation lists not only to a user, but to a user’s information seeking task, we will need to judge recommender algorithms using a variety of metrics, each of which measure a different property of that algorithm and each of which correspond to different aspects of a user’s information seeking task. When we have that information, we can select and tune the appropriate algorithm(s) for each user and task. In short, we need to re-think how to generate a ‘good’ recommendation list.

Building Bridges

In Figure 1-1, we show the current state of recommender systems, the state of the world before this dissertation. Between the user and the recommender is a space we call the

‘gap of intention’. We argue that not only is the information channel between user and recommender too narrow, but that the two sides may not understand the cues currently transmitted across this channel. Information the user wishes to send concerning her intentions for using the recommender is lost to the gap. In much the same way, contextual information may not come back to the user; the recommender’s intentions also fall into the gap.²

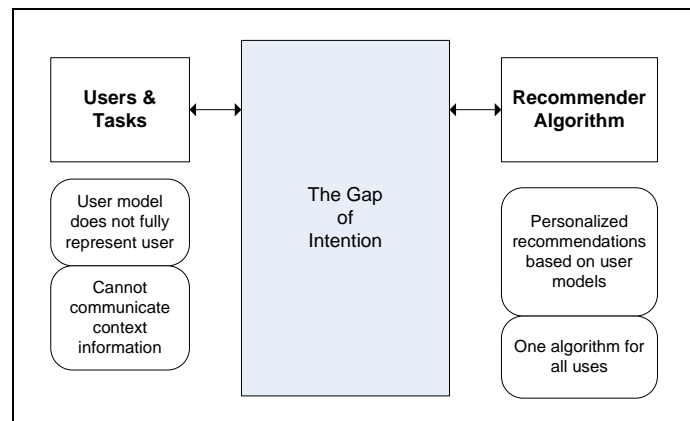


Figure 1-1: The Intention Gap between Users and Recommenders

Our solution to this problem is to build a bridge: provide an organizing language the two sides can use to communicate their intentions, and categorize recommender algorithms in terms of this language abstraction. As shown in Figure 1-2, our bridge is a process model connecting users and their needs to recommender algorithms. It adds two nodes: Human Recommender Interaction theory (HRI), and a new set of recommender metrics, as well as a set of processes connecting those nodes.

HRI is a new framework and methodology for analyzing both user information seeking tasks and recommendation algorithms in the context of a recommender system with the end goal of generating useful recommendation lists. HRI was developed by re-examining the recommendation process from the end user’s perspective and categorizing

² Adding transparency in the recommendation process has been long advocated [137], but few existing recommenders provide insight as to how or why items were recommended.

various aspects of the interaction that are important to the user. This categorization provides a common language for users and recommenders to communicate intention, thus providing a key linking algorithms to users' needs.

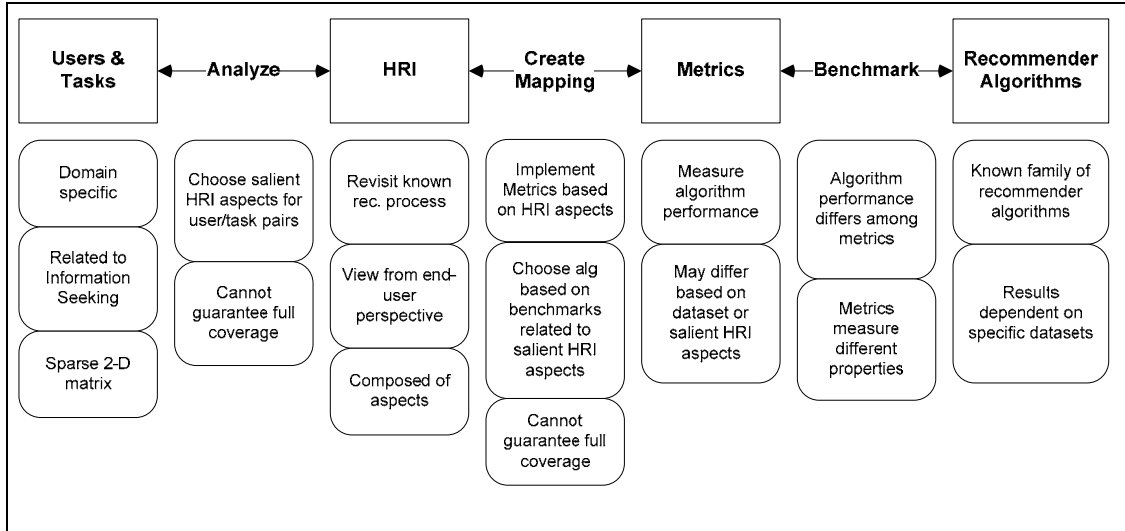


Figure 1-2: The Human-Recommender Interaction Process Model

To connect HRI theory to algorithms, we designed a set of metrics to benchmark wide families of recommender algorithms. These metrics work with existing accuracy metrics to provide a richer understanding of the strengths and weaknesses of recommender algorithms. By linking the meaning of these metrics to the language of HRI, we have a two-way path connecting user needs to recommender algorithms.

Research Approach and Contributions

To provide support for our thesis and its two related points, we focus our approach in three areas. First, we apply recommender systems in the domain of peer-reviewed research papers, a domain where users have external criteria for selecting items to consume. As far as we are aware, this is the first work to propose, design, build, and perform a complete experimental evaluation of recommenders in this domain. Second, we argue that current recommender systems research is not focused on user needs, but rather on algorithm design and performance. To bring users back into focus, we propose

to shift perception from ‘recommenders giving recommendations’ to ‘users receiving recommendations’ by reflecting on how users perceive recommenders and the recommendation process. Third, to escape the curse of accuracy, we look to different ways of evaluating recommender systems algorithms. To this end, we propose a new set of recommender metrics and perform detailed experimental evaluation of several recommender algorithms using these metrics.

Therefore, our research approach has three prongs: *Recommending Research Papers*, *Re-examining the Recommendation Process*, and *Understanding Recommender Algorithms*. For each prong, we will list the specific research contributions this prong provides to our thesis and the relevant chapters in this dissertation.

Recommending Research Papers

- We propose a novel approach to use collaborative-filtering and machine learning algorithms in the domain of peer reviewed research papers by mining the citation network between papers (Chapter 5)
- We propose hybrid recommender algorithms combining content-based and collaborative filtering algorithms for use in this domain following Burke’s established hierarchy (Chapter 5)
- We propose two new extensions to collaborative filtering algorithms in this domain: Denser Collaborative Filtering and Symmetric Collaborative Filtering (Chapter 3)
- We demonstrate the value of our new approach, our new algorithm extensions, and our hybrid algorithms against other approaches mimicking how users previously searched for research papers through a series of offline simulation experiments and online user studies (Chapter 5, Chapter 8)
- We present an analysis of users in the digital library domain, presenting a list of user types, a list of user information seeking tasks, and a set of personas, all of which are used to demonstrate the effectiveness of our overall research approach (Chapter 7)

A User-Centric Approach to the Recommendation Process

- We argue that a recommendation list, not an individual item, should be the important unit of measure when determining the usefulness or the user satisfaction with a recommender system (Chapter 4)
- We propose a new metric to measure the similarity of a recommendation list by comparing the pair-wise similarities of all items in the list: the Intra-List Similarity Metric (ILSM) (Chapter 4)
- Through a series of offline simulation experiments and online user studies, we demonstrate the value of ILSM and the effect of topic diversification on recommendation lists (Chapter 4)
- We propose Human-Recommender Interaction theory (HRI) as a way to view the recommendation process from an end user's perspective and as a language for describing user expectations and perceptions of recommendations and the recommender system itself (Chapter 6)
- We introduce the HRI Analytical Process Model, a process model to bridge user types and information seeking tasks to recommender algorithms through the language of HRI and a classification of recommender algorithms by a series of metrics based on HRI (Chapter 6)

Understanding Recommender Algorithms

- We argue for the creation of a new set of recommender metrics, stating that the current predictive accuracy and decision support metrics were not designed to, and thus unable to classify recommender algorithms among the dimensions needed for HRI (Chapter 1, Chapter 3)
- We propose a set of new recommender metrics to categorize recommender algorithms based on the differences in recommendation lists they generate, and use this information as part of the HRI Process Model (Chapter 8)
- We conduct a series of offline simulation experiments benchmarking the differences across several recommender algorithms, including collaborative filtering, content-based filtering, and hybrid algorithms, using this new set of

metrics (including ILSM) in the domain of peer-reviewed research papers (Chapter 8)

- We run an online user study confirming the differences in recommender algorithms found by our set of new recommender metrics (Chapter 9)

Dissertation Organization

This dissertation is organized as follows:

In Chapter 2, we review previous and related work, paying close attention to work from library and information sciences as well as from recommender systems.

In Chapter 3, we discuss the recommendation process, implementation details of the recommender algorithms used in this dissertation, including a discussion of their strengths and weaknesses, and a summary of current predictive accuracy and decision support metrics.

In Chapter 4, we argue for the use of recommendation lists as the relevant units of comparison between recommender algorithms, introduce the Intra-List Similarity metric, and report on a series of experiments studying the diversity of recommendation lists.

In Chapter 5, we discuss how recommender algorithms can be used in the domain of research papers, including a set of hybrid recommender algorithms for this domain, and two sets of experiments exploring how well these algorithms perform in this domain and how users reacted to the recommendation they received.

In Chapter 6, we introduce Human-Recommender Interaction theory (HRI) and the HRI Process Model.

In Chapter 7, we present a set of user types, a set of user information seeking tasks, and a list of personas for the domain of peer-reviewed research papers in a digital library.

In Chapter 8, we present a set of new recommender metrics and a detailed offline simulation experiment using these metrics to catalog the performance of several recommender algorithms from Chapter 3.

In Chapter 9, we describe the design and review the results of an online user study validating our experimental results from Chapter 8.

Finally, in Chapter 10, we conclude with a summary of our research findings, a discussion of their implications and on potential future work.

CHAPTER 2

RELATED AND PREVIOUS WORK

This dissertation draws on several different fields, most notably recommender systems, information seeking, searching (on the Internet, for example), and citation analysis. In this chapter, we review this related and previous work. A detailed discussion of the recommendation process, recommender algorithms, and recommender metrics appears in Chapter 3.

Recommender Systems and Personalization

Recommender systems help a target user navigate through a complex information space by making suggestions of which bits of information the user should consume (i.e. read, watch, listen to, etc.) based on the system's knowledge of the user, other users in the system, and the information space itself [119]. All recommenders use information about a user (sometimes called a *user profile*, a *user model*, or *user preferences*) to generate recommendations. Recommenders come in three varieties: content-based, collaborative-based (also known as 'social'), and knowledge-based [17]. Moreover, there are hybrid recommenders combining these varieties. Content-based recommenders use domain content to generate recommendations: information and meta-data gathered from items in the domain (e.g. the text of books you previously purchased). Collaborative-based recommenders use people's opinions of items in the domain to generate recommendations (e.g. users with similar item preferences to you liked this book, thus you may like it too). Knowledge-based recommenders use rules, patterns, or connections between items to generate recommendations (e.g. when you are buying a lamp, it suggests that you also buy some light bulbs). After discussing content-based and collaborative recommending, we will discuss one kind of knowledge-based recommending, case-based reasoning.

Content-Based Recommenders

Content-based recommenders use information from the items themselves to generate recommendations. For example, in a research paper recommender, text extracted from the papers could be used to generate recommendations. Such recommenders use information retrieval and filtering algorithms to generate recommendations. A complete review of information retrieval (IR) and information filtering (IF) is beyond the scope of this dissertation, but a high-level overview helps place our research in context. We will first discuss the similarities and differences between IR and IF, review common models used in these systems, and finally discuss the relationship between IR, IF, and content-based recommenders.

In their influential 1992 paper, Belkin and Croft provided a clear argument that information filtering (IF) and information retrieval (IR) were much closer related than had been previously discussed in their respective communities [7]. In it, they argue that both IR and IF share the same five characteristics: A predefined *representation* and organization of documents, a representation of a user's current *information state*, a *comparison* step in which relevant documents are selected, an *evaluation* step where the user reviews the selected documents, and a possible *iteration* on the user's information state. The two key points are the user's information state, and the possibility of iterating on this state. In combined IR/IF systems, this information state must to be translated into a query that the system can parse. Such queries are comprised of keywords describing the user's need [5]. This implied translation could seriously affect the user's ability to find documents that meet her information need. Once given this translation of state, the iteration step becomes essential for helping users meet their information need. These problems also appear in recommender systems and we discuss them in this dissertation.

While there are many similarities between IR and IF, there are differences as well. The differences between IR and IF can be expressed in a few salient points:

- IR is focused on returning relevant documents from a static corpus, whereas IF is focused on selecting relevant documents from an incoming stream of documents (a constantly changing corpus)
- IF is concerned with the timeliness of the documents returned, preferring new documents to older ones, whereas IR wants the most relevant documents no matter what
- IR bases its model on independent queries; IF retains query information into a user information model
- IF assumes to have a richer and more well-defined user information state than IR. IF has a model of the user's information need whereas IR usually only has the user's current query.

There are several kinds of models that IR and IF systems use to generate meaningful recommendations. All models index and categorize information from the documents in the corpus. These indexes are based on content extracted from the documents themselves. The correct data representation is very important as it can limit the kinds of results that IR/IF systems return [5].

With indices in place, one of several models can be employed to return relevant documents. We mention three popular ones.

1. Boolean Retrieval Model

In this model, the user's query is augmented with Boolean operators indicating the user's intention with their query (i.e. "recipes AND hamburger OR recipes AND cheeseburger"). This is an 'exact-match' model where the given query terms must match terms found in relevant documents. This model does not provide for relevance ranking, and it could exclude possible relevant results. It is, however, simple to implement, and is commonly seen in IR systems.

2. The Vector Space Model

In this model, all documents are evaluated using a multi-dimensional vector

space, where each dimension represents a word in the corpus. Similarities between documents can be computed using cosine similarity measures. One extremely popular version of this model weights the different word dimensions based on the frequency of the word in the document and in the corpus and is called TD/IDF similarity [127]. We will discuss using TF/IDF as an IR algorithm inside a content-based recommender in Chapter 3.

3. The Probabilistic Model

The probabilistic model uses probability calculations to determine the relevance of a query to different documents. For example, a Bayesian inference network can be used to model the relationships between documents in the corpus. When a query is presented to the system, the network ‘propagates a signal’ depending on the probabilities between nodes, and returns the nodes representing the documents with highest probability of being related to the query [7]. We will discuss a different probabilistic model, the Naïve Bayes Classifier, for use in recommender systems in Chapter 3.

The relationship between IR/IF systems and content-based recommender systems is one of abstraction and purpose. Recommenders require elements of both kinds of systems, including the ability to search an existing corpus and streams of information, make use of an existing user model, and return the most relevant information. In essence, we argue a content-based recommender is equal to the abstraction above an IR or an IF system—an information processing system with the explicit goal of generating meaningful recommendations to users. This goal, we believe is a different goal from IR or IF systems. In an IR/IF system, the goal is to provide the most relevant documents to meet a user’s information need, either from the corpus or from the incoming streams. In a recommender, the goal is the similar, but instead returning the most relevant documents for a user’s information need, they return the most *salient*. That is, they should return not only relevant documents, but those which have the greatest impact on the user’s

perception of completing her information seeking task. This is a point we will discuss in detail when we present Human-Recommender Interaction theory in Chapter 6.

Collaborative Filtering

Collaborative filtering-based recommenders (CF) work by gathering the opinions of users about items in a domain (e.g. movie ratings) placing this information into a user-item ratings matrix. Algorithms then compare either rows (users) or columns (items) to predict values for empty entries in the matrix. The idea of using the opinions of others to generate item recommendations combined with the phrase “collaborative filtering” was first discussed by Goldberg et al [41]. Their vision was that the opinions of others could help a person manage email and electronic documents by providing opinions and annotations to each item, giving the current user extra information about each of the items.

In 1994, Resnick et al. published “GroupLens: an open architecture for collaborative filtering of netnews” in which they proposed a k -nearest neighbor algorithm for generating recommendations based on user opinions of netnews news articles [120]. This algorithm, now commonly referred to as User-based Collaborative Filtering, was the first algorithm widely used in recommender systems. Herlocker et al. performed a detailed analysis and proposed several important modifications [55]. A more technical discussion will appear in Chapter 3. The first experiments in CF were in netnews news articles [120, 143], but this soon expanded into several other domains, such as: jokes [42], movies [55, 58], and music [135].

The k -nearest neighbor algorithm is considered an instance-based machine learning algorithm used for instance classification [97] before becoming known as ‘the’ CF algorithm. Soon, other machine learning algorithms were also explored as recommenders. The first to be tried were statistical methods such as Bayesian networks [13] and clustering methods [15, 147]. Just as important, these papers brought with them machine learning evaluation methodologies for evaluating recommendation quality, such as k -folding and leave- n -out, the implications of which will be discussed in detail later.

Since then research in collaborative filtering recommenders has moved forward in several directions. The most fruitful has been in creating and evaluating more algorithms. A new algorithm, called Item-based Collaborative Filtering was proposed by Sarwar et al. [131]. Instead of finding users who have similar opinions to each other, Item-based CF finds items that are similar to each other. We will look at this algorithm in detail later. Beyond Item-based CF, several more machine learning algorithms have been applied in this domain. SVD has been applied to reduce the size of the ratings space [129]. Canny et al. proposed using factor analysis [18], and Popescul et al. used latent class models [113] to calculate relationships between users and items. As can be expected, this has led to a proliferation of other probabilistic models for use as recommenders [108, 133]. While they all are interesting for different reasons, we will explore two examples in this dissertation: Naïve Bayes [13], and Probabilistic Latent Semantic Analysis [59]. The first is considered a classic machine learning algorithm and is useful as a baseline, the other is a new algorithm which has many interesting properties. Both will be discussed in detail later.

Recently, researchers have developed and tested new algorithms that can have useful properties for complex, data-rich recommendations. Many of these are hybrid algorithms as classified by [16]. Several researchers have combined collaborative filtering with content-based filtering to generate movie recommendations [92], a personalized newspaper [25], and book recommendations [99, 156]. Taxonomies have also been used to enhance recommenders [93].

Researchers have also moved forward to tackle other problems in recommenders as well. To help relieve sparsity in recommenders, FilterBots are ‘users’ inserted into a recommender who rate items based on a predetermined criteria (e.g. the “Bruce Willis” bot would rate all of his movies very high, and other movies low) [43]. The problem of getting new users into a recommender has also received a lot of attention, especially in determining more efficient ways of bringing new users into the system [89, 117, 133]. It may be possible to cheat a recommender through ‘shilling’, the process of artificially adding ratings to influence predictions in a set way. Luckily, there has been work done to

tackle that problem as well [71, 105]. Finally, there have been discussions of theoretical foundations for collaborative filtering. Pennock et al. propose using Social Choice Theory as such a foundation [107].

Trying to understand a user's context is not new in recommender systems. Herlocker et al. created an 'ephemeral' recommender which generated recommendations using both a user's profile and a selected subset of that profile representing the user's current interest [52]. Yahoo! Music (formerly LaunchCast) has taken this concept one step further by allowing users to create and store multiple lists of items, each one representing a 'mood' [157]. Both of these concepts express a user's need as 'please show me more like this'. In a similar fashion, van Setten et al. demonstrated location-sensitive recommendations using GPS-enabled devices for tourists [149]. Here, the need is 'please show me more near where I am located.' These approaches are similar in spirit to what we propose. We believe, however, that the user should also be able to specify what kind of recommendations she would like to get back (e.g. in Yahoo Music, being able to say, "I'm feeling curious right now, please play less popular and more esoteric music"). So, not only can a user describe a current need, but the user should also have an influence as to how the recommender generates recommendations by also describing context. As we will argue later, this may require continual changes within recommender, but should appear transparent to the end user.

Finally, there are augmented algorithm approaches to incorporate extra contextual information in directly into recommenders. Adomavicius et al. used a multi-dimensional approach [3]. Goal-based structuring systems create networks of recommendation strategies through which the user traverses as he receives recommendations, selecting strategies that match the user's current interests [151]. These networks are usually predefined or linked to content/attributes of items the space [150]. These concepts are related to our work, but we take a content-agnostic approach where we link recommender algorithms to a set of externally derived concepts from which users can select the most appropriate concept(s) for their current need.

Case-Based Reasoning and Conversational Recommenders

Another important recommender methodology related to this work is case-based reasoning (CBR) [49, 122, 124]. CBR is useful for solving constraint-based problems where users specify content-based attributes which limit the returned recommendation set (e.g. I want to see a movie starring Arnold Schwarzenegger released in the 1990s rated either G or PG). Historically, CBR is considered a problem-solving methodology in which previous problem solving episodes are remembered by the system and stored as ‘cases’ [145]. Each case contains a problem component and a solution component. When a new problem is formulated, the system looks for similar problem components and suggests their solutions for this new problem. The recommender view equates the problem to the user model and the potential solution to recommendations.

One of the methodologies used in CBR systems is to create an interactive recommendation experience where the system presents information sequentially, using the information gathered at each stage to further customize the user’s experience. These have come to be known as “conversational recommenders” and come in two varieties: iterating over attribute questions (as in [123]) and iterating over a selection of recommended items (as in [82]). For both types of systems, Pu et al. [115] provides a set of recommended interaction principles to follow suggesting ways in which the user maintains control over their profile during the conversation. As a note of terminology, in a conversational recommender each iteration is a ‘recommendation cycle’ [82].

In the first variety, CBR systems use a wide variety of methods for determining which questions to ask at each cycle. Both entropy and popularity-based approaches have been used finding that a hybrid approach works well [94], similar to the work for bringing new users into a collaborative filtering-based recommender system [117]. When user constraints become too narrow, CBR systems can choose to ‘relax’ a constraint in order to generate recommendations [95]. This method is generalizable into creating a partial ordering expression to express the constraints, such that the knowledge of which constraints can be ‘relaxed’ is embedded in the user profile [14].

In the second variety, lists of items are presented to user, and based on a given selection, new lists are presented until the user chooses to consume (e.g. purchase) a presented item. Such methods promote discovery and exploration by users, especially in domains where the user may be intimidated by jargon or complexity [122]. These systems, however, must allow a user to ‘back out’ of a given selection if he feels he made an incorrect selection, and the system should adapt accordingly [82]. Moreover, such systems can retain both long-term and short-term preference information about a frequently visiting user to make even the first set of recommended items personalized [83, 144].

Because of the interactive nature of CBR systems, researchers have also focused on the diversity of presented recommendation lists. The argument is that if a user is not happy with the mostly highly recommended item in a list, then the chances of her being satisfied with some other item on the list are greater if the list is more diverse [14]. Diversifying lists may reduce accuracy, and researchers have developed algorithms to increase diversity with a minimal loss of accuracy [91]. The argument that a loss of accuracy is always unacceptable is one that we challenge in this thesis. Finally, recent work has provided a framework for determining when to increase and decrease diversity in CBR conversational recommender [82, 122].

Researchers have also suggested ways to combine collaborative filtering with case-based reasoning. The first suggested way is to ‘boost’ CBR by mining similarity knowledge found in cases, and using those similarities in a CF-like manner to generate recommendations [104]. Hayes et al. took the idea further to show that CF can be viewed from a CBR perspective, and it can be implemented using a case retrieval net model [49]. Their unique and novel approach shows how one can encode different levels of similarity and rating information into a neural net and use a spreading activation algorithm to generate recommendations in a conversational recommender context. They also show how this model can be used to add ‘context’ to a CF-based recommender [48]. Here, they re-order recommendation lists generated by traditional CF with similarity

information gathered by a spreading activation algorithm over the users' most recent system usage (considered as their context).

A Summary of Common Recommender Problems

Recommender algorithms face several problems. In this section, we summarize these problems and discuss research in these areas. We will refer back to these problems in future chapters and discuss how our algorithm implementations avoid these problems. We first discuss evaluation problems common across many of the previously published papers on recommender systems. For several systems, these problems were operational assumptions, and we call them out here as we feel these assumptions may have unduly limited our view of recommender systems. After discussing these common problems, we will focus our attention on problems relevant to each of the three varieties of recommender algorithms. Some of these problems may be relevant to more than one variety of recommender, but we will discuss each one in the variety in which we feel it is most severe.

Common Problems when Evaluating Recommender Algorithms

There are several problems common to previous recommender evaluations discussed above. Several of these problems will be re-visited in later chapters.

- 1. A single recommendation cycle**

The entire model is designed around the principle of a single, independent recommendation instance. There is no memory of previous recommendations or of changes to a user's model. As has already been discussed in [82, 122], there are many benefits to an iterative recommendation process, and we will explore them in Human-Recommender Interaction theory.

- 2. No feedback sent back to the user**

Sinha et al. argue that in order to increase trust in recommenders, recommenders need to be more transparent about the recommendations they generate [137].

Transparency includes explaining why specific recommendations appeared [54]

and helping users understand how the recommendation process works [88]. This point will return when we discuss Human-Recommender Interaction in Chapter 6.

3. **Focus on accuracy**

Many previous papers have focused on the accuracy of recommender algorithms (for example, see [13, 55, 59, 131]). The methodologies employed by these papers have stressed the importance of generating a list full of individually accurate recommendations, not an accurate recommendation list. We discuss this issue later in this chapter.

4. **Focus on efficiency**

Previous research has also focused on ways to make the recommendation process more efficient. Methods such as clustering [15, 147] and SVD [129] have been applied to this problem to alleviate sparsity and speed computation time, content-boosting approaches [92] have also been used to alter the ratings matrix, filling in otherwise empty cells. But what effect does making an algorithm more efficient have on the usefulness of the recommendation lists it generates?

Problems with Collaborative Recommender Algorithms

Collaborative algorithms, such as Collaborative Filtering, Naïve Bayes Classifiers, and PLSA, are “domain independent” in that they perform no content analysis of the items in the domain. Rather, they rely on user opinions to generate recommendations. Despite being a successful technique in many domains, collaborative algorithms have their share of shortcomings [6]:

1. **The Cold-Start Problem (a.k.a. the First-Rater Problem)**

When a collaborative system is first created, there are many items in the system, few users, and no ratings. Without ratings, the system cannot generate recommendations and users see no benefit. Without users, there is no way for new ratings to be entered into the system. When applying these algorithms to a new domain, it is valuable to seek preexisting data that can be used to seed such a database of ratings. In the case of MovieLens, the freely available EachMovie

dataset was used to “jump start” the system [55]. This problem has been framed and explored in [120, 133].

2. **The New-User Problem**

Before a user can take advantage of a collaborative recommender system, the user must first provide their opinions. The user has to trust the system that the recommendations will be worth the effort of entering opinions. It is difficult, because this trust is needed before the user starts using the system. Getting new users into a recommender is a fruitful area of research we have explored in other publications [88, 89, 117]. This problem is common to other varieties of recommenders as well, but is more severe for collaborative recommenders since these recommenders cannot rely on content or categories to ‘ease’ a user into the system.

3. **The Sparsity Problem**

In many domains, a user is likely to rate only a very small percentage of the available items. This can make it difficult to find agreement among individuals, since they may have little overlap. Different recommender algorithms deal with this problem in various ways. Item-based CF uses similarity measures between items. If we assume there are fewer items than users in a recommender (as is commonly the case), then Item-based CF reduces the impact of sparsity on the same dataset than algorithms using a user-item ratings matrix. Statistical-based or latent analysis algorithms, such as Naïve Bayes and PLSI, also work in sparse situations, mining all connection data to generate recommendations [63, 75, 158].

Problems with Content-based Recommender Algorithms

Content-Based Filtering (CBF) is also commonly used in recommender systems. Applied mostly in textual domains, such as news [9], CBF recommends items to a user if these items are similar in content to items the user has liked in the past. Many algorithms, most notably TF-IDF [127], have been used in CBF systems. CBF has many strengths, including the ability to generate recommendations over all items in the domain. CBF also has its shortcomings [6]:

1. **Content limitation in domains**

In non-textual domains like movies and audio, many content algorithms cannot successfully and reliably analyze item contents. Rich metadata, such as actors, directors, artists, etc., has been improving recommendations in this area, but does not attack the problem of analyzing non-textual content.

2. **Analysis of quality and taste**

Subjective aspects of the content in an item, such as style and quality of writing or authoritativeness of the author are hard to analyze. Writing samples can be grammatically analyzed, and thus some level of quality can be achieved. But this is not a semantic analysis; the meaning of the content cannot be easily determined through automatic methods.

3. **Narrow content analysis**

CBF recommends items similar in content to previous items, and cannot produce recommendations for items that may have different but related content. Recently, lexicons and advanced algorithms have made improvements in this area, but are still costly to use [5, 8].

Problems with Knowledge-based Recommender Algorithms

Knowledge-based recommenders such as Case-Based Reasoning also have their share of problems. By using both content information and content-independent knowledge rules, these kinds of recommenders have unique qualities [17].

1. **Focus on Domain Attributes**

Not all domains have a rich set of attributes. In such domains, a knowledge-based recommender would have a difficult time gathering and processing knowledge states.

2. **The Constraint Satisfaction Problem**

By using knowledge rules to select recommendation items, it is possible to be in a

state where no items satisfy the constraints imposed by the user and the rules.

Learning how to relax some rules is an area of active research [14].

Citation Indexing and Recommending Research Papers

There have been many recommender and personalization systems targeted at recommending research papers or research colleagues. Referral Web used collaborative filtering to recommend professional referrals mined from citations and other aspects of a social network [68]. In addition to automatic citation indexing, CiteSeer provides citation analysis and recommendations for papers in their system [10, 28, 72, 73]. Recently, Relescope created personalized conference schedules for attendees based on their publication history and an analysis of the conference program [36]. While the system was well received, users had varying opinions of usefulness. Specifically, inexperienced users (users new the field) found it quite helpful for locating important and interesting people, but experienced users found it of little practical value even though they strongly liked the idea—not all users want the same information.

Lately, researchers are turning to search engines such as Google Scholar [45] for papers. Such interfaces are great when searching for particular items. Others have suggested using recommendation as a tool within digital libraries [40], and have even demonstrated limited implementations [64]. While we agree with the principles of this work, we note that such systems need to support a variety of users and tasks; something we are proposing in this work. Finally, as digital libraries become more a part of our lives, people will use them to share information [79], suggesting the importance of group-support applications.

It is often interesting to measure the impact that a particular citation or journal has in its field [11, 31, 159]; ISI's Web of Knowledge uses one such example metric [38, 57], but not everyone agrees that it is the best metric to use [47]. Citation analysis and bibliometrics [12] can be used to create social networks between authors [134]. Not only are the patterns of authors interesting [103], there are parallels between friendship and co-citation patterns [152], and they can be used to generate recommendations [81]. These

findings agree with our ideas that citation patterns can be successfully used to generate research paper recommendations.

Theories of Information Seeking

Information seeking theory provides us with a framework to understand user information needs and context. Models of information seeking behavior, including Taylor's four stages of information need, Wilson's Mechanisms and Motivations model, Dervin's theory of Sense Making, and Kuhlthau's Information Search Process, reveal the ways in which emotion, uncertainty, and compromise affect the quality and nature of a user's information search and its results [20, 22, 70].

In his influential paper, Taylor proposed four stages at which people perceive information needs: a visceral need, a conscious need, a formalized need, and a compromised need [142]. We will focus on the last stage: it suggests that a user will tailor (compromise) their need when explaining it to an information repository. The user will use a language they think will be better for the repository. With a human librarian, this compromise could be easily mitigated by insight and pointed question asking [153], but it is unclear how a computer can do that.

Expanding to the entire search process, Kuhlthau's model has six stages: Initiation, Selection, Exploration, Formulation, Collection, and Presentation [70]. Her model is important as it suggests there is an exploration aspect to all information seeking processes that happens before the detailed search begins (in Formulation and Collection). Further, she suggests that each stage has different feelings attached to it. For example, when starting a search process, many users feel nervous or concerned, but develop confidence during exploration.

Adding to this idea is Dervin's Sense Making theory that suggests that information seeking comes from a need to make sense of the world. As such, users will settle for answers that satisfy their emotional concern even at the expense of accuracy [20, 22]. For example, if a user is frustrated with a search, it could be because there is dissonance between the user's understanding of the world and the reality provided by her

search. In such cases, a user may select an information source not because it is the most relevant, but because it reinforces her worldview and thus makes her feel better about the search process.

Wilson provides a different view of information seeking; in the 1996 version of his Mechanisms and Motivations model, he states that there are ‘activating mechanisms’ through which a user determines whether or not to pursue an information need [155]. These mechanisms come in many forms, but many are psychological, environment, or interpersonal, suggesting that outside influences has as much as effect on the information seeking process as the need for the information itself. These theories ground our analysis of what forms of information seeking behavior appear in a digital library environment: users come to an information repository, such as a digital library, looking for research information—information for a need they may not be able to succinctly express—but their confidence, emotional state, and comfort/familiarity with the system affect their information seeking behavior as much as their intellectual desire to solve their information need.

Lately, research has moved into new directions in information seeking. Mokros et al. suggest looking at it from a communications perspective: modeling information seeking through meaning engagement practice to analyze not just the seeking process but also the products and by-products in a way that provides for reflection and deeper theoretical understanding [98]. Narayanan et al. argue for the creation of adaptive interfaces for digital libraries as a way to mitigate concerns raised when applying Taylor’s and Kuhlthau’s models to digital libraries [101]. Finally, information foraging is a new theory for understanding how users perform information seeking tasks, suggesting that users look through information spaces by seeking promising-looking ‘patches of information’ [112].

One commonality among all of these theories is the importance of understanding a user’s information context to increase the confidence and relevance of results, a point emphasized by recent work on uncertainty reduction [39, 50, 154]. In traditional libraries, a librarian would reduce uncertainty through a ‘reference interview’ to clarify

both a user's information need and information context [142, 153]. Online, it is difficult to express this context, so users rely on hints from other researchers (e.g., by e-mailing colleagues for advice) to reduce uncertainty both before and during searching [32]; this interaction creates a feedback cycle between users and the system, refining a search until they fill their information need. In the context of recommender systems, this work suggests that an *iterative* process between the user and the system can both clarify the user's information need and reduce the uncertainty of the recommended items (i.e. generate with greater confidence). This iterative process forms a foundation for Human- Recommender Interaction.

Search, Search Engines, and Digital Libraries

Search is an important access method for digital libraries. Search engines, such as Google [44], have become a way of life for web-surfers and have permanently changed the landscape of the World Wide Web [21]. Even so, search engine effectiveness varies widely depending of discipline, systems responsiveness, and user effectiveness at entering search terms [136, 139, 140]. Researchers studying the information-seeking in the home have identified location context as an important factor in search strategy—users in a familiar environment have different strategies from those in other environments [125]. Others have suggested that user tasks can be inferred from log analysis [126] or interviews [30] to improve search results. Finally, both cognitive style and previous search engine experience independently affect how users perceive as well as use search engines [69].

More important than understanding search engines, however, is understanding the *search process*, especially how search has changed with the introduction of the Internet. There are multiple classes of searching: “search”, looking for a known object or object with known properties, and “discovery”, exploring promising information spaces for underspecified or unknown objects [20, 78]. Experts argue about the importance of intention when searching, as people encounter information on a daily basis, not just when

in the middle of an information seeking task [20]. We assume intention, as a user has chosen to interact with a recommender, but we do not know the user's class of searching!

Marchionini argues that in order to be useful, information must be presented to a user along "aggregate thematic indexes" to align the information seeking to her "school of thought" [77]. Current information systems, including search engines, are good for the "search" class, but poor for the "discovery" task. Discovery, he argued, required "recognizing relationships between objects" [77]. Since then, many ways have been found to mine such information, including our novel method for generating collaborative filtering-based recommendations based on citation information. His end argument, however, was electronic systems should align the presented information to a user's current perspective. We address this lofty goal in the domain of computer science research papers.

The number of items available from digital libraries is staggering: ACM's Guide to Computing Literature indexes over 750,000 articles, and its Digital Library has over thirty years worth of ACM-published articles [1], the National Library of Medicine's PubMed contains over 15 million citations spanning fifty years [102], and JSTOR currently archives hundreds of journals across arts, sciences, and other fields [65]. The list grows larger every day and finding information could prove taxing. End user search skills may be further tested as digital libraries add to the complexity by interlinking with each other [96]. Also, users are not perfect information processing machines; they are human. They will mix together different reasons for making the choices they do when seeking information—they will mix relevance judgments and decision-making to best suit their current state as a person, even if they choose less than optimal results [23, 37]. Knowing this, we need to design DL interfaces accordingly by personalizing the display of information so that users can find information faster [101, 121]. We argue that recommender systems are tools to drive this personalization, but need to be applied with care: a mis-personalized recommendation list could severely limit a user's ability to find information. In this dissertation, we demonstrate how recommenders drive

personalization and present results detailing the benefits of high quality personalization as well as the drawbacks of mis-personalization.

Conclusion

In this chapter, we reviewed previous and related work. We discussed the history of work in personalization, recommender systems, and collaborative filtering, ending with a detailed discussion of common problems in recommender systems. Next, we reviewed work in citation analysis and in the categorization and recommendation of research papers. From there we discussed several theories of information seeking and then reviewed literature on search and the search process. Building on this foundation, we discuss the recommendation process and various recommender algorithms in Chapter 3.

CHAPTER 3

CONCERNING RECOMMENDING, RECOMMENDER ALGORITHMS, AND RECOMMENDER METRICS

In this chapter, we discuss how to generate recommendations and how to evaluate their accuracy. We begin by reviewing the recommendation process, an algorithm-centric view of how recommendations are generated. Next, we will review the recommender algorithms used in this dissertation, including a discussion of various parameters and settings for each algorithm as well as an analysis of benefits and weaknesses. Finally, we will review the most popular predictive accuracy and decision support metrics used in recommender systems.

In Chapter 2, we discussed three variations of recommender algorithms. In this dissertation, we choose to examine collaborative filtering and content-based recommenders in detail. The principles discussed concerning case-based reasoning, especially that of conversational recommendations, are significant to this dissertation, but we chose not to include CBR algorithms in our experiments. This was done so we could focus on an analysis of the differences between content-agnostic recommenders and content-centric recommenders.

Background and Definitions

The person for whom the recommender is generating recommendations for is called the *active user*, also known as the target user. The recommender algorithm performs a series of calculations over a *ratings matrix*, a matrix containing the preferences of all users in the system (also known as a ratings database or a preference database), to generate a top- n list of recommended items. When a user follows the advice of the recommender, we say that the user *consumes* the item. This may mean different things in different domains. It could mean ‘purchase’ for a retail recommender, or ‘read’ from a book recommender, or ‘view’ from a movie recommender.

There are two ways for the user's opinion of an item to be recorded in a recommender: explicitly or implicitly. If the information is recorded implicitly, it is usually gathered at the point of consumption, (e.g. when the item is purchased, but before the user uses the item). Since the opinion of the user is never recorded, this unary information is may not be as reliable as explicitly gathered preference information—only the user's initial perceptions are recorded, not their final opinion. If explicit ratings are gathered, the active user returns to the recommender to 'rate' the item, encoding her opinion of the item into the ratings matrix. Before we can discuss the algorithms in detail, we first must discuss the ratings matrix.

The Ratings Matrix

Figure 3-1 shows an example of a ratings matrix. In such a matrix, the columns of the matrix represent items in the domain, whereas the rows represent users. Each entry in the matrix represents the opinion that one user has about one item. There are several ways to encode this opinion. In the MovieLens movie recommender, for example, movie opinions are encoded on a 0.5- to 5.0-star scale. Online retailers, on the other hand, might only record whether a user has purchased an item or not as unary ratings. Many cells in the matrix will be empty. That is, not every user will have an opinion on every item, and not every item will have been consumed by every user. In practice, a ratings matrix may be stored in a sparse representation or an otherwise compressed format to save space.

A recommender algorithm has two possible tasks: the prediction task and the recommendation task. In the prediction task, the recommender is to determine what value should appear in any given empty box. In the recommendation task, the recommender is to determine a list of empty boxes that the active user will fill in with the highest possible opinion; that is, which items the user will like the best. When a user asks a recommender about a particular item (e.g. will I enjoy *Hitchhiker's Guide to the Galaxy?*), the recommender makes a prediction about that item. When the recommender generates a top-*n* list of books the active user should read, the recommender makes a recommendation. While similar, the algorithms for generating predictions and

recommendations are different. Moreover, the metrics used to evaluate success and the underlying user tasks are different. In this dissertation, we deal exclusively with *recommendations*.

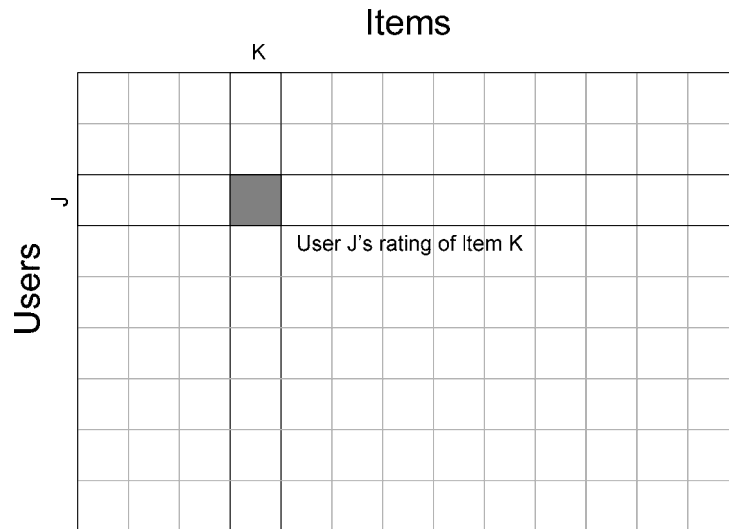


Figure 3-1: A Conceptual Ratings Matrix

Collaborative filtering and content-based recommenders make very different use of the ratings matrix. In a content-based recommender, the ratings matrix acts as a historical record of the items a user has previously accessed. This information may or not be used to generate future recommendations. That is, content-based recommenders make use of external information gathered from the items themselves to generate recommendations. The matrix plays a more active role for collaborative algorithms.

Collaborative filtering recommender algorithms are called as such because they filter information for a target user based on the opinions of other users—an implicit collaboration. They operate on a ‘word of mouth’ principle—opinions of items spread through groups of people as friends interact with each other, dynamically deciding if a particular friend would like a given item. Here, the recommender algorithm acts as the spreading mechanism by mining the opinions of a very large group of people. The preferences of this ‘very large group of people’ are stored in a ratings matrix, and

collaborative algorithms use the information stored in other rows or other columns to generate recommendations for the target user.

To formalize our discussion in the following sections, we will define the following. Let:

R = the ratings matrix

$U = \{u_1, u_2, u_3 \dots u_n\}$ is the set of all users u_i

$I = \{i_1, i_2, i_3, \dots i_n\}$ is the set of all items i_j

r_{ij} = the rating of user u_i for item i_j . r_{ij} = null when user u_i hasn't rated item i_j

p_{ij} = the prediction for item i_j for user u_i

a = the active user

The Recommendation Process

Figure 3-2 depicts our generic recommendation model. The heart of the model is the Recommendation Generator; for any given recommendation calculation, it may contain several recommender and post-processing algorithms. The model can represent applications that have a single algorithm, a hybrid algorithm, or many different algorithms at once. Flow in the model moves from left to right, starting with the user model and ending with the recommendation list. The Recommendation Generator takes in a representation of the user's current state of knowledge (the user model), the user's Information Need, and any specified Settings and Contextual Parameters. The settings and parameters may come from several different sources, including from the user via the user interface, gathered or calculated from user's model, from the application hosting the recommender, or from some internal state of the recommender itself. The Recommender Generator applies the algorithms designated for a given user model and information need to recommend and filter a set of results from a single data repository.

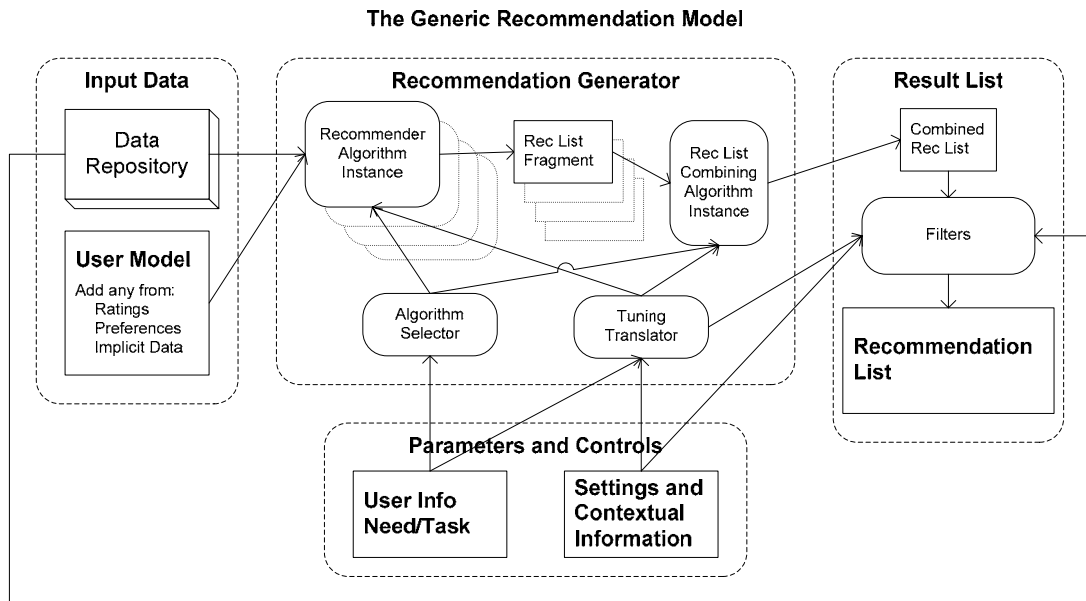


Figure 3-2: The Recommendation Process Model

To better understand the model, consider Figure 3-3, which shows the model as instantiated for the collaborative filtering (CF) recommender from Chapter 5. Subjects were to select a single paper with which they were familiar and answer questions about recommendations we generated for that paper. Users only went through the system once, receiving one recommendation list. The imposed information need was to find relevant citations for an existing research paper (for details see Chapter 5). The citations from that paper are used as the user model for the recommender. The recommender used an untuned User-based Collaborative Filtering algorithm. The information context specified a filter—not to recommend papers by the authors of the selected paper. After all, authors are usually quite good at citing their own work!

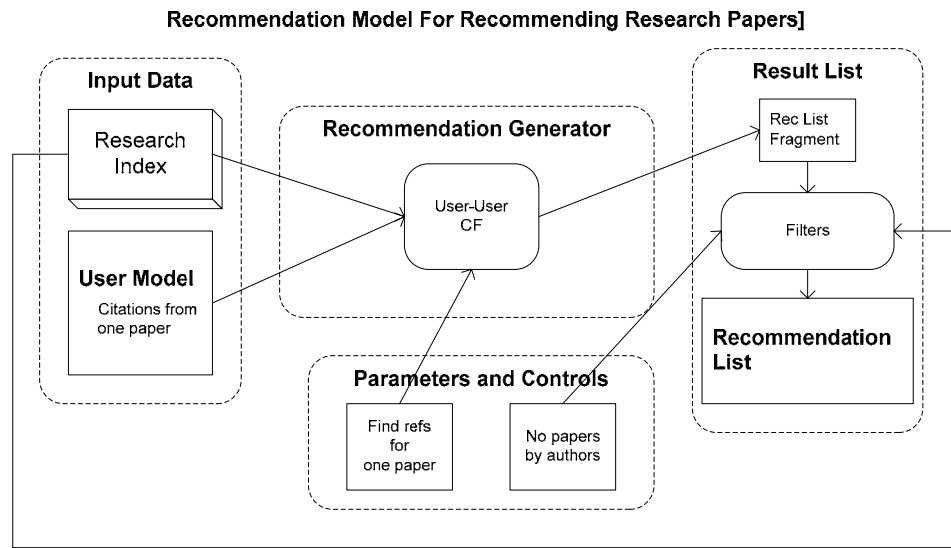


Figure 3-3: An Instance of the Recommendation Process Model

Any given algorithm can only use one ratings matrix to generate recommendations. Suppose that we created several different ratings matrices. In the movie domain, for example, we could link user ratings to movies, to actors, to genres, etc. Each one would have a different ratings matrix based on the same data repository. Each algorithm instance could be running on a different matrix, and the combination algorithm and/or filters would combine the lists together appropriately to create one recommendation list presented to the end user. The entire model can be chained (results to inputs) to create an iterative process with each step having a different context and need. The point of this model is to outline possibilities of interactive assembly of novel recommenders. We do not mean to imply that the model is necessary for creating our applications, only than it provides a way of conceptualizing them.

User-based Collaborative Filtering

User-based Collaborative Filtering (CF), also known as User-User CF, *k*-Nearest Neighbor CF, and as ‘Resnick’s algorithm’, was the original automated collaborative filtering algorithm [120]. Its core assumption: people who previously have had similar opinions on items in a domain will continue to have similar opinions in the future. To

generate recommendations, User-based CF searches the ratings matrix for the k users most similar to the active user, forming a ‘neighborhood’; the neighbors’ ratings are used as recommendations for the target user. User-based CF runs in three phases: computing similarities between users, finding k neighbors, and aggregating neighborhood ratings to return a top- n list.

As discussed in [53, 55, 120], there are several methods for computing similarity between two users. The most commonly used method is Pearson Correlation. For an active user a , and another user u , the Pearson Correlation, $w(a, u)$, is

$$w(a, u) = \frac{\sum_{k=0}^n (r_{ak} - \bar{r}_a) * (r_{uk} - \bar{r}_u)}{\sigma_a * \sigma_u}$$

where we are summing over all items co-rated by users a and u ,

\bar{r}_f is the average rating of user u_f and

σ_f is the standard deviation of user u_f ’s ratings.

After calculating similarities across all users, the k most similar are chosen as neighbors. Once we have neighbors, we aggregate their ratings. We gather the set of all items that the neighbors have rated which the active user has not, and calculate a prediction score for each item using a deviation from mean approach. The prediction p_{ai} for active user a and item i is

$$p_{ai} = \bar{r}_a + \frac{\sum_{u=1}^k (r_{ui} - \bar{r}_u) * w(a, u)}{\sum_{u=1}^k w(a, u)}$$

where we are summing over all k neighbors of the active user a .

Finally, we sort the predictions and return the top- n predicted items to the active user.

Unary vs. Explicit Scale Ratings

The equations expressed above are for explicit ratings on a predetermined scale. User-based CF also works for unary data, with the assumption that all ratings are positive. Instead of performing Pearson correlation, other measures, like a cosine similarity, may

be used. Once calculated, the candidate set derived from the neighbors contains all possible items to be recommended. They need to be sorted using a variant of p_{ai} where $\bar{r}_a = 0$ and $(r_{ui} - \bar{r}_u)$ equals 1 if user u has rated item i and equals 0 otherwise. In essence, the similarity values are used as weights, and the items are ranking from largest to smallest. We will revisit this weighting algorithm in Item-based CF.

Parameters, Settings, and Modifications

User-based CF depends on three settings: the similarity measure, $w()$, k , and n . Pearson correlation has traditionally been used to compute similarity, but is not the only option. Other options include Spearman correlation, mean-squared differences, or other Euclidian-based measures [53]. As mentioned, a vector cosine (dot product) calculation may be appropriate for unary data. Neighborhood size (k) also plays a large factor in recommendation quality. A small k can lead to unpredictable recommendations, whereas very large values can lead to recommendations tending towards the mean. Previous work suggests that neighborhood sizes between 30 and 100 tend to work well [13, 53, 55], but these values will be dataset dependent. Finally, n , the number of recommendations to return, can affect expected results. Specifically, a recommender may not be able to return a recommendation list for very large values of n . If an active user's neighbors have not rated many items, the recommendation list will not be that long.

There are three common modifications to User-based CF we will explore.

1. Negative Correlations

Some users will have a strong negative correlation. For example, if user a and user b are negatively correlated, then most items that user a likes, user b dislikes and vice-versa. Then, should user b should be in user a 's neighborhood (accounted for correctly, of course)? To rephrase a traditional Irish toast, are you friends with your enemies' enemies?³ The answer to this question is, unfortunately, dataset dependent. In a domain such as movies, where many

³ "We drink to the health of the enemies of your enemies", is a traditional Irish toast to a long life.

ratings are positive, negative correlations are not strong and can have undesired effects. Even more so, the answer depends on the purpose of the recommender. In many domains, taste can be determined as much by what a person hates as by what a person likes. This is especially true in domains with a strong self-selection for rating items. For example, two travelers may be good recommenders for each other if they both hate the same airports. Depending on rating patterns, negative correlations could be stronger than positive correlations! Traditionally, negative correlations have not been used, but that does not mean they could not be, especially in sparser data environments where it may be difficult to find neighbors or in environments with a wide variety of ratings.

2. **Similarity Thresholds**

Instead of choosing the k closest neighbors, it is possible to select only neighbors above a specific similarity threshold [55, 135]. A few neighbors with high similarity are more important than many neighbors with low similarity. Users with similarity near zero have little in common and tend to generate noise. Experimentally, using a high threshold does not greatly improve accuracy but does reduce coverage, only a smaller percentage of the dataset is available to be recommended to a target user [55]. A common hybrid approach is to set both a minimum correlation and a maximum number of neighbors. For example, a typical setting for User-based CF is k set at 50, and a minimum correlation set at 0.1. Thus, each neighborhood could have up to 50 neighbors, each of whom have at an absolute correlation of at least 0.1 with the active user.

3. **Significance weighting**

In significance weighting, the goal is to prevent neighbors who are highly correlated with the active user due to only a few co-rated items from having too large an influence on the recommendation list [53]. For example, a user could have perfect correlation with the active user on 3 items, another user could have an excellent correlation based on 58 items. In traditional User-based CF, the neighbor with 3 correlation items would have a stronger influence than the

neighbor with 58. To account for this, an extra weighting factor based on the number of co-rated items was added to the similarity calculations between users. In [55], it was implemented as an $n/50$ weighting, where potential neighbors with less than 50 ratings in common with the active user were devalued. Applying such a weighting technique has proved to increase accuracy. Implementation specifics should take into account the average number of ratings per user, and it is more important to devalue neighbors with a few correlations than reward neighbors with many correlations.

Strengths and Weaknesses

User-User CF has many strengths. First, it is one of the most well-known and studied CF algorithms. It is easy to understand conceptually and easy to develop. It has proven to generate high quality recommendations, and has become the ‘gold standard’ for recommenders against which all other algorithms are compared. Moreover, User-User has high serendipity, generating unexpected, yet high quality, recommendations. It is this property more than any other that is the real strength of User-based CF.

User-based CF also suffers from several limitations. It is expected that recommender algorithms operate in pseudo-real time—we must assume a human being is waiting for the recommendation list—and the algorithm must deliver, usually in under a second. For a ratings matrix of m users and n items, User-based CF is $O(m^2n)$: scalability is a real concern for User-User CF. As is sparsity. In many real world datasets, there may be millions of items and millions of users. The overlap of ratings between users could be very small, and neighborhood quality can suffer as a result. This problem becomes even worse when the only overlap is for popular items. For example, enjoying the movie “Casablanca” does not imply you will enjoy “Dude, Where’s My Car?”

When User-based CF is implemented as described, computing similarity scores for each recommendation list, we refer to it as being a *memory-based* recommender algorithm. CF algorithms can also be *model-based* in which part of the computations are pre-computed in a model and used at run time. For a User-based CF algorithm, the user-to-user correlations can be computed and looked up when needed. By splitting the

computations into two parts, the real-time component of the algorithm can be quite fast. Historically, User-based CF has been a memory-based algorithm, but there is no reason why it cannot be a model-based algorithm. The terms memory-based and model-based first appeared in [13].

Item-based Collaborative Filtering

Item-based collaborative filtering, also known as Item-Item CF, was first proposed by Sarwar et al. in [131] and has received further analysis in [33, 130]. As opposed to User-based CF, when Item-Item CF was proposed, it was proposed as a model-based algorithm.

In Item-based CF, the core assumption is that there is a relationship between items themselves. In retail for example, there is a relationship between products frequently purchased together (e.g. peanut butter and jelly). Thus, when a user is interested in one product, it is likely the other should be recommended. Item-based CF runs in three phases: first, pre-compute a model of the k -nearest item similarities; second, use the model to calculate an item neighborhood for all items rated by the active user; third, aggregate and compute a top- n recommendation list.

First, we must create the item similarity model. To do so, the similarity between all pairs of items is computed with the k -closest similar items to each other item stored in the model. A ratings matrix with n items will have kn entries in its corresponding item model. Just as before, a variety similarity measures can be used. Previous research has been conducted mostly with unary data. As such, cosine-based similarity and conditional probability-based similarity have been commonly used [33]. Once built, this model can then be used for all recommendations.

When the active user requests a recommendation list, an Item-based CF algorithm performs the following steps:

1. The algorithm creates an ‘active list’ containing all of the items the active user has rated.

2. For each item on the active list, the algorithm looks up similar items from the model and stores a reference to the item and its similarity score in a candidate list. Items appearing in active list are not recorded in the candidate list.
3. If an item already appears on the candidate list, the new similarity score is added to the previous value.
4. After iterating through the active list, the candidate list is sorted by similarity score; only the top n items are returned as recommendations.

Unary vs. Explicit Ratings Scales

When generating recommendations with explicit ratings data, the steps vary slightly with only two changes from above. First, in step 1, we need to determine which items appear on the active list. Is it all items the user has rated? Or is it only items the user has liked? This difference has large implications on the meaning associated with the recommendations generated.

Second, there is a different step 4:

5. (*in place of step 4 above*) Prediction values are generated for all items on the candidate list, the list is sorted by prediction, and the top n items are returned to the user as the recommendation list

The prediction value is calculated in a similar way as it is for User-based CF, except the similarity between items is used. The calculation uses a difference from mean approach using a weighted average of the active user's ratings, using the similarity measure between items as weights:

$$P_{aj} = \bar{r}_a + \frac{\sum_{u=k}^n (r_{ak} - \bar{r}_a) * w(i_j, i_k)}{\sum_{u=k}^n w(i_j, i_k)}$$

Where:

a is the active user,

i_j is the current item,

\bar{r}_a is the average of the active user's ratings, and

$w(i_j, i_k)$ is the similarity between items i_j and i_k .

Parameters, Settings, and Modifications

Just as with User-based CF, Item-based CF is dependent on the values for k , n , and the chosen similarity metric. Similarities are based on user ratings, but they are computed across all users in Item-based CF. This leads to one potential drawback: users have different internal rating scales (i.e. what one user thinks is a 5.0 might only be a 4.5 to someone else). To account for these differences, Sarwar et al. proposed an Adjusted Cosine Similarity [131], and they found that this adjustment increased the accuracy of Item-based recommenders. The other modifications discussed for User-based CF, such as negative correlation and similarity thresholds have similar affects for Item-Item. One further discussion point is the use of symmetric vs. asymmetric similarity functions for building the item model. As discussed in [33], each approach has their advantages and disadvantages, the most notable being that symmetric similarity functions will 'punish' highly popular items unless there are many other popular items in the active list.

Modifying the size of k in the item-to-item similarity model will also alter the recommendations Item-based CF generates. k determines how many other items each item is connected to. Increasing k will increase computation time for the model, but will also do two other things: first, it will increase coverage, as a larger number of items will be added to the candidate list. Second, increasing k will increase 'voting power' of each item in the active list—each active item gets k 'votes' of varying strengths to apply to items in the candidate list. Increasing k will increase the spread of potentially recommended items, strengthening votes for candidate items similar to items on the active list. Thus, increasing k should improve the quality of recommendations. Existing experimental results on this point are mixed, suggesting that increasing k beyond some dataset-dependent threshold does not to continue to improve recommendation quality [33, 130].

Strengths and Weaknesses

The model-based nature of Item-based CF is both a strength and weakness. The runtime for Item-based CF is quite faster than for User-based CF. Since the item similarities are pre-computed in the model, the runtime is only dependent on the number of items the active user has rated and value of k from the model. However, recommendations are only as fresh as the model is. If a dataset continuously receives new ratings data, this data cannot be used until the model is rebuilt, a potentially non-trivial computational task.

Items are first judged based on similarity, then by predicted value. Thus, Item-based CF is not guaranteed to return all highly predicted items, but the ones that are have a high similarity to items the active user has already rated. If certain similarities are exceptionally strong, they could overpower other parts of the calculation. In previous experiments, we have demonstrated that Item-Item can fall into a ‘similarity well’ where the addition of certain items into an active list will dominate the recommendations received [117]. A fun example is the Star Wars example from MovieLens: all six Star Wars movies are very highly correlated to each other. If a user has seen one, the user has seen all six. Once you place one Star Wars movie in your active list, forever will the other five dominate your destiny (or at least your top- n recommendation list).

Extensions to Collaborative Filtering Algorithms

We now present two new extensions to collaborative filtering: *denser collaborative filtering*, and *symmetric collaborative filtering*. Both extensions can only be implemented when the ratings matrix meets specific requirements.

Earlier, we defined the ratings matrix as matrix listing the preferences users have for items. In a collaborative filtering recommender, the following statement always holds true: “Users rate items; items are recommended to users.” We will call this the *collaborative filtering recommendation statement*. The key insight to the statement is not that it involves users and items, but rather how users and items are related to the ratings matrix itself. If we abstract above users and items, the statement becomes, “Rows rate columns; columns are recommended to rows”. We can place *any pair* of items into the matrix and have a recommender. The question is whether the recommendation statement

makes sense. For example, automobiles are painted colors. If we gather colors as car preference information, our recommendation statement becomes, “Cars rate colors [as paint]; colors [as paint] are recommended to cars”, see Figure 3-4.

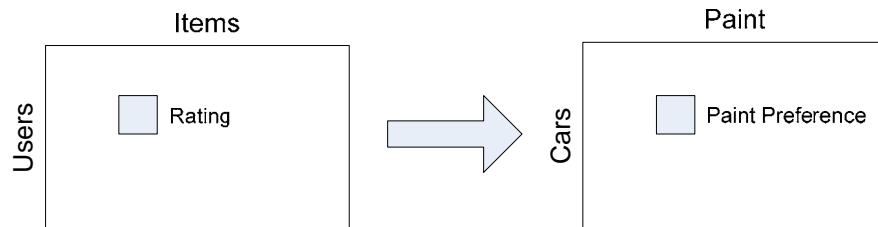


Figure 3-4: Rating Matrix Example with Cars and Paint

What happens when items refer to themselves? How does that affect the ratings matrix? Let’s look at an example: friendships. If we gather friendship preferences into a matrix, we would have, “people rate people [as friends], people [as potential friends] are recommended to people.” Another such domain is research papers; papers make references to other papers. If we gather citation preferences into a matrix, we would have, “papers rate papers [as citations], papers [as potential citations] are recommended to papers”. We claim that when a ratings matrix has this self-referential property, we can apply these two new extensions to collaborative filtering.

Denser Collaborative Filtering

In an environment where users and items are equal in a recommender, we can envision the relationship between items as a directed graph where each item is a node and a link indicates preference information. When we view preference information this way, we notice that items can be related by multiple ‘hops’. In traditional collaborative filtering, the input basket would be all items one hop out from the target user. In Denser Collaborative Filtering, we use the implied preference information from more than one hop out as items placed in the input basket; we make the input basket *denser* with items transitively related to the target user, see Figure 3-5. We evaluate this approach in Chapter 5.

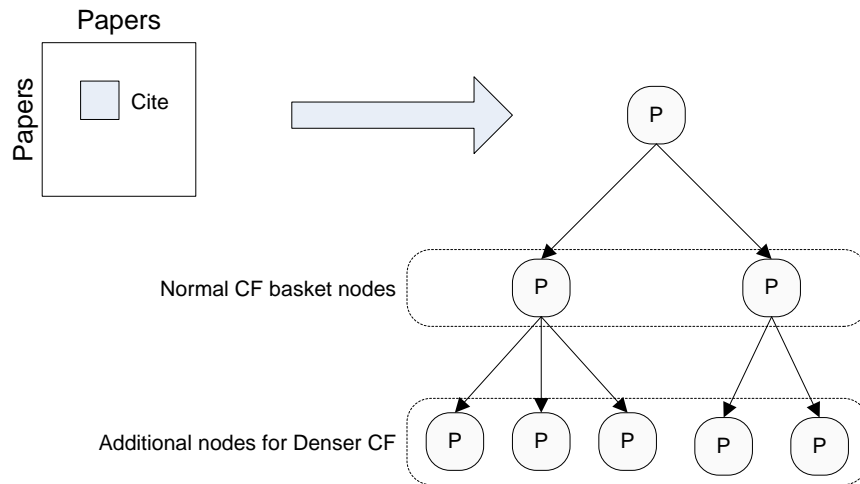


Figure 3-5: Self-Reflective Ratings Matrix as Directed Graph

Symmetric Collaborative Filtering

Even though the act of rating is ‘one way’ in a symmetric ratings matrix, users may perceive such a space as being ‘two way’. With research papers, for example, users may view it as a link between papers—if one paper cites another, then the two papers are related. This view is reinforced when users are presented with browsing interfaces where the current focus of interest can be moved both up and down the citation graph [76]. We propose to augment collaborative filtering to follow this same relaxed view of a citation network. Symmetric Collaborative Filtering is a variant of collaborative filtering where the act of rating is bidirectional. Since the ratings matrix in this domain is square, this equates to adding two entries to the matrix for each voting instance: the paper’s vote for a reference, and the reference’s vote for the paper. The net result is a ratings matrix with twice the number of ratings as a traditional ratings matrix in this domain. The standard CF algorithms run unaltered on this new ratings matrix. As a directed graph, this approach is equal to making all links bi-directional, but still only using one hop for all basket creation. We evaluate the effectiveness of this approach to CF in Chapter 8.

Naïve Bayes Classifier

A Naïve Bayes Classifier is a machine learning classification algorithm for placing items into the most probable partition of a partitioned dataset. The classifier uses both the anterior probabilities of the correct partition for each item as well as the conditional probabilities of each item based on the existence of other items in a partition. The naïve assumption of this classifier is that all of the conditional probabilities are independent of each other; there are no interaction effects between lists of items on the conditional probabilities. In a recommender, we use this classifier to determine the probability of a user rating a given item: the probability of an item belonging to the set of the active user's ratings, given what this user and all other users have rated.

Our Naïve Bayes Classifier is

$$P(t | b_1, b_2, b_3, \dots, b_k) = \frac{P(t) \cdot P(b_1 | t) \cdot P(b_2 | t) \cdot P(b_3 | t) \cdot \dots \cdot P(b_k | t)}{P(b_1) \cdot P(b_2) \cdot P(b_3) \cdot \dots \cdot P(b_k)}$$

Where:

t is our target item to classify

$b_1 \dots b_k$ are the active user's rated items (the active list)

$P(t)$ = the anterior probability for t (the baseline probability of rating this item), usually related to the popularity of the item in the dataset.

$P(b_1 | t)$ = Number of times that t and b_1 have been co-rated together by any user.

To generate a recommendation list, the active list is sent to the classifier. For all other items in the dataset, the classifier calculates the posterior probability for that item being added to the active list. After performing all of the calculations, the top n items, those with the highest probabilities, are returned as calculations.

The classifier is a model-based recommender algorithm. All anterior and conditional probabilities are pre-computed in the model; the runtime consists of looking up the appropriate probabilities, calculating the posterior probabilities, and sorting the results.

Unary vs. Explicit Ratings Scales

The methodology as defined will only calculate the probability of an item being rated by a user, it does not generate a prediction score. Thus, as is, a naïve Bayes classifier works for unary data. There are a few different possibilities for applying naïve Bayes to an explicit rating scale. The first idea is to redefine the classifier into more partitions, each one being a user-rating pair. For example, in MovieLens, where there are 10 possible ratings for an item, each user would have 10 partitions, one for each possible rating. Classifications are calculated for each relevant pair, and the strongest is returned. So, in MovieLens, the target item would be classified 10 times, and the strongest classification would be returned as the prediction.

A second possibility is for ratings events themselves to be considered probability distributions over all possible ratings (i.e. each rating is a Gaussian distribution centered at the given rating). This allows for the possibility for noise in a recommendation and has been discussed in [84, 108]. The recommender would then return the most probable rating distributions; a similar method is used in the next algorithm will discuss, PLSA.

Parameters, Settings, and Modifications

A naïve Bayes classifier does not have many parameters. Defining the anterior and conditional probabilities are the two most important things to do for this algorithm. It is important to make sure the exact definitions match the domain. Here, we have defined the anterior probability as being related to the number of ratings each item has in the dataset; it is possible some other domain dependent measure could be used. The conditional probabilities are related to number of co-ratings two items have together. Again, the conditional measure could be domain dependent.

There is one other important decision to be made for a naïve Bayes classifier: determining how items appear in the candidate set. There are two possibilities: first, as described above all items are considered. This method allows the recommender to have 100% coverage, but is very expensive to calculate. A closer look at the Bayesian equation suggests that only items that are conditionally related to the active set will be preferred to all other items. Thus, a shortcut is to calculate posterior probabilities only

for conditionally related items—create a candidate set. This method however does not have 100% coverage; coverage is proportional to candidate set size. If the candidate set contains only a few items, coverage may greatly suffer. This point will be relevant in our user study of a naïve Bayes classifier-based recommender in Chapter 9.

Strengths and Weaknesses

A naïve Bayes classifier assumes that all conditional events are independent of each other. This is a strong assumption to make in recommendation domains; it is quite likely that rating events are not independent events... the opinion a user has on one item is likely to have an effect on the opinions of other items (e.g. A user has a ‘soft spot’ for John Cleese, and always rates movies he appears in high, no matter what). Domingos et al. analyzed the independence assumption in a related domain, that of text classification, where the probability of two words appearing next to each other is not independent (e.g. the phrase “collaborative filtering”). Even though independence assumption fails, naïve classifiers still performed very at the classification task [34]. Thus, while we know the independence criteria fails, we believe a naïve Bayes classifier will generate high quality recommendations in similar domains. Other more complex Bayesian approaches, such as Bayesian Belief Networks, also work in this domain [13].

The naïve Bayes classifier is a well studied and mathematically rigorous classification algorithm, and has shown to generate high quality classifications in several domains, even when the independence criteria is not fully satisfied [97]. These properties make it an excellent algorithm to use as a comparison. Finally, as another model-based algorithm, it has a fast runtime, especially if it uses a candidate list for generating recommendations.

Probabilistic Latent Semantic Analysis

Thomas Hofmann proposed a new model-based algorithm for collaborative filtering called Probabilistic Latent Semantic Analysis (PLSA) [59, 62]. This algorithm has also been called Probabilistic Latent Semantic Indexing, or PLSI for short. In his co-occurrence data model, Hofmann proposes that each rating instance is independently

generated through hidden variables Z with a set of k states labeled z . Thus, person u rates item i “because of” z [61]. These latent classes are similar to the latent variables from other algorithms [108, 129, 133].

Often, these latent classes do not have obvious meanings, but the *idea* is something we can understand: there may be certain reasons an item appeals to us, but these characteristics are not visible on the surface. For example, in the domain of movies, it is possible for people to be sensitive to the amount of brightness in the picture. Some people prefer bright and colorful movies, and others preferring darker films. This characteristic cuts across all movies, regardless of genre, and could be considered a latent variable. In PLSA, the latent classes are used probabilistically to determine the relationship between users and items.

In a similar fashion to the Naïve Bayes Classifier, PLSA calculates the probability that a user will rate a particular item, but PLSA uses latent classes to make these calculations. Further, membership in a class is determined at the per-rating level; a user can be a member of several different latent classes at once. To generate recommendations, PLSA predicts which items a user will rate and what the rating will be: instead of only calculating $P(r_{ui} | u, i)$, PLSA calculates $P(r_{ui}, i | u)$.

Since $P(r_{ui}, i | u) = P(r_{ui} | u, i)P(i | u)$, PLSA needs to predict which item and then predict the rating, once given that item. It does so by computing the following:

$$P(i | u) = \sum_z P(i | z)P(z | u)$$

$$P(r_{ui} | u, i) = P(z | u)P(r_{ui}; \mu_{i,z}, \sigma_{iz})$$

$$P(r_{ui}; \mu, \sigma) = \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{(r_{ui} - \mu)^2}{2\sigma^2}\right)$$

Where:

$\mu_{i,z} \in \mathfrak{R}$ is a ‘location parameter’ and

$\sigma_{iz} \in \mathfrak{R}^+$ is a ‘scale parameter’

such that $\mu_{i,z}, \sigma_{iz}$ define a Gaussian mixture model where the observed individual ratings are noisy, based on a typical value but corrupted by normally distributed noise. The prediction value is calculated by performing a summation over all possible rating values: $\sum_z P(z | u) \mu_{iz}$.

How do we perform the model fitting? PLSA uses a variant of the EM algorithm that maximizes a negative conditional log-likelihood function based on the triplets (u, i, z) [60]. Since the states of the latent variables are not known when starting, Hoffman uses a variational probability distribution, Q , for every observed rating [59]. Thus, PLSA's E-step is:

$$Q^*(z; u, i, r_{ui}, \hat{\theta}) = \frac{\hat{P}(z | u) \hat{P}(r_{ui} | i, z) \hat{P}(y | z)}{\sum_z \hat{P}(z' | u) \hat{P}(r_{ui} | y, z') \hat{P}(z | z')}$$

Where:

- Q^* is the optimal variational probability distribution at this step,
- $\hat{\theta}$ represents the model parameters from the previous M-step, and
- $\hat{P}()$ are the probabilities associated with those parameters.

PLSA's M-step is:

$$P(r_{ui} | i, z) \propto \sum_{\langle u, r'_{ui}, i' \rangle: r'_{ui}=r, i'=i} Q^*(z; u, r_{ui}, i, \hat{\theta})$$

Details, including derivations of these equations, can be found in [59, 60].

Unary vs. Explicit Ratings Scales

As discussed, this PLSA model uses an explicit ratings scale. When using PLSA for unary data the definition of $P(r_{ui}, i | u)$ changes and the equations become simpler.

$P(r_{ui}, i | u)$ becomes $P(i | u)$, where

$$P(i | u) = \sum_z P(i | z) P(z | u)$$

The E-M steps also simplify accordingly. For details, see [59, 60].

Parameters, Settings, and Modifications

There are only a few settings to modify for this algorithm, the most important one being k , the number of latent classes to choose. It is helpful to think of PLSA as a soft-clustering algorithm, where both users and items can be described as a weighted combination of latent classes. The more latent classes, the wider variety of options for describing items in the system. Too few classes, and some latent classes could become ‘junk’ classes, limiting the effectiveness of this algorithm. Because of how new this algorithm is, there are not many published results measuring algorithm sensitivity to the number of classes, but our limited testing suggests it is highly dataset dependent.

MovieLens requires between 40 and 100 classes, but the ACM Digital Library needed over 1000 classes. In any case, the number of classes should be much smaller than either the number of users or items in the system. A good rule of thumb is first try the square root of either the number of users or number of items in the system, whichever is larger. Most likely, this value will be too small for the number of latent classes, but is a good place to begin.

As discussed above, users do not make comparable use of the ratings space. Some users tend to rate high, other low. To account for this, PLSA can perform a user normalization step by which user rated are converted to differences from the user’s mean and the variance renormalized to one. For users with only a few ratings, Hofmann performed a smoothing step when estimating the variances [59]. Normalizing user ratings dramatically improved algorithm accuracy, and we suggest using it, not just for PLSA, but also for all recommender algorithms.

Finally, performing the EM algorithm from a potentially limited amount of data could lead to overfitting. To overcome this, Hofmann proposed tempered EM, which acts to smear out the exactness of the calculations. This is accomplished by adding a β parameter both to the risk function and to the E-step. The E-step would be more like:

$$Q^*(z; u, i, r_{ui}, \hat{\theta}) \propto \left(\hat{P}(z | u) \hat{P}(r_{ui} | i, z) \hat{P}(y | z) \right)^\beta$$

Strengths and Weaknesses

Similar in spirit to the Bayes classifier, PLSA has similar benefits and drawbacks. It is mathematically rigorous, using latent classes to find probabilistic relationships between items. Model creation takes a very long time, requiring exceptional computing resources. In practice, we have found constructing PLSA models to take 3 to 5 times longer than the comparable Bayes model. Using these models at runtime, however, is exceptionally fast, depending only on the number of latent classes.

As mentioned before, the latent classes are the *key feature* of this algorithm, thus performance and quality are highly dependent on the number of latent classes used. PLSA has 100% coverage; it will always generate a recommendation list. If there are not enough latent classes, the recommendations may be of poor quality, as we will see in Chapter 9.

This is a relatively new algorithm in this domain. We believe the latent aspect of this algorithm makes it more like User-based CF being able to generate interesting and serendipitous recommendations. By performing local maximizations, the EM nature of the algorithm could reinforce more “popular” connections between items at the expense of “further reaching” (and potentially more interesting) connections, thus recommendations could be interesting locally, finding unexpected items closely related to the input, but not interesting for finding a broad set of items from across the dataset.

TF/IDF Content-based Filtering

We now shift from collaborative recommenders to a content-based recommender. As a reminder, these algorithms use information calculated from within the items themselves to generate recommendations, they do not compare information between rows or columns of a ratings matrix for recommendations. In this domain, the ratings matrix becomes a preferences matrix, storing users’ preference information for items the user has already consumed. Thus, a content-based recommender will only make use of the active user’s row in the matrix. Traditionally, in the literature of content-based filtering, items are referred to as ‘documents’, and the domain of items is referred to as a ‘corpus of documents’ [5]. We follow that tradition here.

Term Frequency/Inverse Document Frequency content filtering (TF/IDF) is considered the classic vector-space (a.k.a. bag-of-words) analysis algorithm in Information Filtering and Information Retrieval [127, 128]. Based on a vector model of information retrieval, TF/IDF applies weights to a vector containing entries for each word appearing in the document. Documents can be directly compared against each other, or more importantly, words can be compared between documents. As its name suggests, TF/IDF depends on computing the relative frequency of a word appearing in one document compared to the frequency with which the word appears in the entire corpus of documents. Thus, unpopular words appearing in two documents will provide strong evidence that these two documents are related.

As an information retrieval algorithm, TF/IDF works as follows. We first create a data model, calculating the popularities of all words appearing in the corpus. If we let:

N = total number of documents in the corpus

d_i = document i in the corpus

k_i = term i appearing in the corpus

n_i = the number of times term k_i appears in the corpus

$freq_{ij}$ is the number of times term k_i appears in document d_j .

$maxfreq_{Mj}$ is the maximum frequency of all terms appearing in the text of the document j (i.e. the value of computing $\max(freq_{ij})$ for all terms in the document)

Then, the inverse document frequency for term k_i is:

$$idf_i = \log_2 \frac{N}{n_i}$$

In a similar fashion, the term frequency for each term k_i can be calculated for each document d_j :

$$tf_{ij} = \frac{freq_{ij}}{maxfreq_{Mj}}$$

Finally, the TF/IDF weight score is calculated as:

$$tfidf_{ij} = tf_{ij} \cdot idf_i$$

Thus, we can calculate the TF/IDF score for each (word, document) pair. If a term did not appear in the document, the score is 0. These calculations are performed over the entire corpus as a model-building step, just as with Item-based CF. Except that in Item-based CF, the model was built with information from the ratings matrix, here it is built with information from the documents themselves.

An information retrieval system based on TF/IDF would work as follows: the user would submit a text query to the system. This query is viewed as term vector, just like all other documents. The system would then apply the TF/IDF weighting scores for all terms in the query, and run similarity calculations against all documents in the system. As discussed in Item-based CF, a cosine similarity score is used but could be replaced with a more sophisticated metric. The top matching documents, those with the highest similarity score, would be returned to the user.

As a content-based recommender algorithm, a similar process would happen. The input basket sent to the recommender would be converted into a query and sent to the algorithm. But there may also be extra translation or augmentation steps. The user may provide terms or other pieces of information to alter the query or weights of terms in the query. The user's model (a.k.a. row in the preferences matrix), could alter the query. For example, in a research paper recommender, if a user provided one document as input, the system might use all of the document's text as a query. Or the system could select only the terms with the highest *idf* weight as a query. Finally, the system could add additional terms related to the user's known interests. The result matching documents would be returned as described above.⁴

Parameters, Settings, and Modifications

While the TF/IDF algorithm itself has no meaningful parameters, there have been several enhancements to the algorithm. First is the use of a stopwords list [46, 127]. This list

⁴ There are many implications of a system changing queries without user knowledge or approval. It is more likely that the system may generate a 'suggested query' for the user to review or change as required. This is an example of the importance of transparency in recommenders, a subject we will discuss later.

contains very common words that the TF/IDF engine should ignore (e.g. ‘the’, ‘of’, ‘because’, ‘in’, etc.). Second is Porter Stemming [114] in which each word is stripped of any suffixes so that words in different tenses can be correlated with each other (e.g. a stripped ‘stripped’ would be ‘strip’). All current implementations of TF/IDF make use of these enhancements. Finally, it is worth noting that exact calculation of TF/IDF weights can be adjusted based on properties of the dataset. Salton et al. discuss several possible alterations [127]; their final recommendation for a generic TF/IDF approach is to temper the *tf* step by doing the following:

$$tf_{ij} = 0.5 + \left(0.5 \frac{freq_{ij}}{maxfreq_{Mj}} \right)$$

Strengths and Weaknesses

TF/IDF is well known and straightforward to implement. Documents returned are usually highly relevant to the input basket. It is also another model-based algorithm, and is subject to all of the same strengths and weaknesses. Speed of runtime depends on size of the active set and size of the corpus, but it can be improved through caching. The most common complaints are lack of semantic meaning and inability to understand synonyms. Several newer content analysis algorithms perform calculations over sets of words (bigrams or trigrams), or perform a semantic analysis on the content in order to make more relevant matches [46]. Words in TF/IDF are points on a vector; they are assumed mutually independent of each other. Following a similar argument to the Naïve Bayes Classifier, the dependences are local to each document and might have different meanings across documents. Since these dependences occur for all documents and are largely independent between documents, we can ignore them for all documents. Finally, TF/IDF cannot equate words together with related meanings (e.g. “car” is not equated to “automobile”). One option is the use of an external lexicon or knowledge base. Another option is to use a semantic analysis technique (including PLSA!) to discover these relationships. These calculations could be quite expensive. These steps may not need to be performed when using TF/IDF in a restricted domain with a known vocabulary (such as computer science research papers). Even with all of these shortcomings, TF/IDF is

still considered a core information retrieval algorithm and can be found at the heart of many search and retrieval applications.

Predictive Accuracy and Decision Support Metrics

Traditionally, recommender algorithms have been judged for ‘goodness’ based on a small set of accuracy and decision support metrics. We will review the most common metrics along with the leave- n -out methodology used when evaluating recommenders.

The Leave- n -Out Methodology

While several variants of this methodology have appeared in recommender systems papers, the most commonly cited and followed version is from Breese et al. [13]. In general, this experimental methodology involves withholding some of the ratings for each user in the dataset, generating recommendations, and comparing the generated results to the withheld items. This approach has roots in the information retrieval and machine learning literature [46, 97]. In their work, Breese et al. describe several implementations of this methodology: *All but 1*, *Given 2*, *Given 5* and *Given 10*. In *All but 1*, only a single, randomly selected rating is withheld from each user. In the *Given* set, all users are reduced to an equal number of randomly selected ratings—all of the rest are withheld. Most commonly, the *All but n* variant is used, for various values of n .

This methodology is usually accompanied with a division of the dataset into test/train datasets along with an x -fold cross validation. For example, many papers used a 10%-90% test/train ratio where 10% of the ratings were test, and the rest train. This would be repeated 10 times (and thus all experiments run 10 times) for a 10-fold cross validation. To accomplish this, the entire dataset was randomly partitioned into 10 partitions; each partition becomes the test set for one of the folds. Thus, each user was tested exactly once. Another way to accomplish this is to select the test set for each fold randomly, allowing a user to be a test user in multiple folds.

Mean Absolute Error

Mean Absolute Error (MAE) is the most common predictive accuracy metric used in recommender systems. It has appeared in a wide variety of papers on recommenders

going back 10 years, such as [13, 89, 99, 108, 135]. As its name suggests, MAE is calculated by determining the mean of the errors between a prediction and the associated actual rating. There are two variations on MAE: calculating it per user, averaging over all users, or per item, averaging over all items. These two ways report different views into the error of an algorithm. We choose to calculate MAE per user, as it is more closely reports a user's view of the algorithm. MAE per user is calculated as follows:

$$MAE(u_i) = \frac{\sum_{j \in R_i} |r_{ij} - p_{ij}|}{|R_i|}$$

Where:

R_i is the set of all ratings for user u_i .

There are a couple of variations on MAE to be aware of. First is Mean Squared Error (MSE), where the square of the error is used instead of the absolute value. This variant adds extra weight to an error the larger it is. Second is Root Mean Squared error (RMS), also known as the quadratic mean, takes the square root of MSE value, and is commonly used in the physical sciences and electrical engineering for calculating error values. Finally, Normalized MAE (NMAE) re-normalizes MAE to be from a 0 to 1 scale so that MAE values can be compared across datasets [42].

While it seems there may be many possibilities for measuring error, we only need one. Herlocker et al. performed a detailed study on predictive accuracy metrics and found that many accuracy metrics correlated with MAE [51], so we can use MAE (normalized, if you wish) as our one metric when measuring predictive accuracy.

Precision, Recall, and the F1 Metric

Precision and Recall are two popular decision support metrics from information retrieval. In recommender systems, they judge relevance by counting recommendation 'hits': if a withheld item appears in a top- n recommendation list. These measures are also done per-user and averaged over all users. For one user, recall measures how many of the withheld items appeared in the recommendation list:

$$\text{Recall} = \frac{\text{Size of hit set}}{\text{Size of withheld set}} = \frac{|w_u \cap rec_u|}{|w_u|}$$

Where:

w_u is the set of withheld items for user u in the current experimental fold

rec_u is the top- n recommendation list for user u for the current fold

Precision, on the other hand, measures how many of the recommended items appear in the list of withheld items.

$$\text{Precision} = \frac{\text{Size of hit set}}{\text{Size of recommendation list}} = \frac{|w_u \cap rec_u|}{|rec_u|} = \frac{|w_u \cap rec_u|}{n}$$

These two measures compliment each other by measuring the relevance from both the point of view of the items being recommended and the items known to be good. The F1 measure [46] brings these complementary nature together by combining precision and recall together into one metric:

$$F1 = \frac{2 * \text{Precision} * \text{Recall}}{(\text{Precision} + \text{Recall})}$$

These decision support measures are used either in tandem with MAE, when there are explicit ratings, or, by themselves, when there are implicit or unary ratings. In the explicit ratings case, we must define the difference between a good hit from a bad hit, usually by partitioning explicit rating space into a good set and a bad set.

Recommendations are only hits if it falls in the good set. Just like with MAE, these metrics make use of withheld items as a ‘ground truth’ from which to base recommendations. In this thesis, we argue that the ‘ground truth’ must be defined in terms of a user’s information seeking task. Current metric definitions are linked to ground truth as expressed in and related to the dataset. We would say they check for ‘dataset consistency’, judging how well an algorithm performs over a dataset. While useful and important, these measures need to be augmented with metrics which measure other properties of recommender algorithms. We explore such metrics in Chapter 8.

Breese's Half-Life Metric

One final metric we review here is Breese's Half-Life Metric, also known as the "Breese Score" [13]. In this metric, the location of an item in the recommendation list is taken into account. It is assumed that the lower an item appears on the list, the less likely a user will see it; it is implemented as an exponential decay function on the utility of an item in the list. The expected utility metric is:

$$EU(a) = \sum_{i \in w_u \cap rec_a} \frac{\max(r_{ai} - d, 0)}{2^{(i-1)/(\alpha-1)}}$$

Where:

w_u is the set of withheld items for user u in the current experimental fold

rec_u is the top- n recommendation list for user u for the current fold

d is the "neutral rating". This can be either a default value for all users (i.e. 50 on a 100-point scale) or the average rating for the active user

α is the half-life value, the rank of the item on the top- n list that has a 50% chance of being seen by the user; its value should be less than n .

As written, this metric assumes explicit ratings. For implicit or unary ratings, the max value is 1 for all hits.

To calculate the Breese Half-Life Score for an algorithm, the value is summed over all users as follows:

$$BreeseScore = 100 \frac{\sum_a EU(a)}{\sum_a EU^{Max}(a)}$$

Where:

$EU^{Max}(a)$ is the maximum utility score, computed by pretending all hits appeared at the top rank in their respective recommendation lists.

Thus, the Breese Score computes the utility of all recommendation lists compared to the best possible scenario involving the same hit sets and top- n lists.

The Breese Score is interesting because of its implicit assumption that order in recommendation list matters. The previous metrics only cared if the withheld item appeared anywhere in the recommendation list. In the next chapter, we take this concept further and argue that the composition of the recommendation list as a whole has a strong influence on the perceived quality and usefulness of items appearing on that list.

Conclusion

In this chapter, we reviewed the recommendation process, details concerning collaborative filtering and content-based recommenders, and current recommender metrics. In addition, we proposed two extensions to collaborative filtering: denser CF, and symmetric CF. In the coming chapters, we will argue for the use of recommendation lists as salient unit of a recommender algorithm, see how recommender algorithms perform in the domain of research papers, and explore new metrics for comparing recommender algorithms.

CHAPTER 4

RECOMMENDATION LISTS AND INTRA-LIST SIMILARITY

Though predictive accuracy and decision-support metrics are an important facet of usefulness, there are traits of user satisfaction they are unable to capture. The most notable is the user perception of the recommendation list. When users perceive recommendations, they do so by reviewing a list. Each item is judged not only on its own merit, but also in the context of items surrounding it. Accuracy metrics are only able to judge the utility of a single item on the list, they fail to understand list context. The closest to do so is the Breese Score, but it only accounts for the position of an item in the list, not the content of the item's neighbors.

How important is this context? We claim this context is extremely important in generating meaningful and useful recommendations. We will revisit this claim and provide evidence to support it in future chapters. Here, we will focus on one aspect of a recommendation list that previous metrics have largely ignored: the diversity of the items in a recommendation list.

For example, suppose you enjoyed reading your first Robert Heinlein novel, *Stranger in a Strange Land*, and you are ready to read another book. You turn to a book recommender. The recommender could generate one of many different recommendation lists; Table 4-1 lists two hypothetical lists. In the first list, we see that all of the recommendations are for other Heinlein books. In the second example, we see a variety of books ranging from science fiction to romance to philosophy and religion. Which recommendation list is better? We argue that the answer is, "It depends". There are many external factors which could influence your decision, such as why you read the book, how the book affected you, etc.

Neither recommendation list is bad, but they are very different from each other. In particular, we say that List B is more diverse than List A. We claim the difference in list diversity will have a large effect on the usefulness of the recommendation list. Before

we study the effect that diversity has on a recommendation list, we will define it more carefully; we do so by defining a metric to measure the similarity between items in a list.⁵

Table 4-1: Hypothetical recommendation lists for Robert Heinlein

List A	List B
<i>The Moon Is a Harsh Mistress</i> by Robert A. Heinlein	<i>Dune</i> by Frank Herbert
<i>Starship Troopers</i> by Robert A. Heinlein	<i>The Time Traveler's Wife</i> by Audrey Niffenegger
<i>The Door into Summer</i> by Robert A. Heinlein	<i>Crimes Against Logic</i> by Jamie Whyte
<i>The Cat Who Walks Through Walls</i> by Robert A. Heinlein	<i>Ender's Game</i> by Orson Scott Card
<i>Have Spacesuit-Will Travel</i> by Robert A. Heinlein	<i>Misquoting Jesus : The Story Behind Who Changed the Bible and Why</i> by Bart D. Ehrman

Intra-List Similarity

We present a new metric to capture the diversity of a recommendation list. Diversity may refer to all kinds of features, e.g., genre, author, and other discerning characteristics. We measure the pair-wise similarity of all items on the list to determine the intra-list similarity of the entire list. For a two items x, y from recommendation list L and similarity metric $w(x, y)$, the Intra-List Similarity (ILSM) for L is:

$$\text{ILSM}(L) = \frac{\sum_{x \in L} \sum_{y \in L, x \neq y} w(x, y)}{|L| * (|L| - 1)}$$

A higher score on this metric implies a higher degree of similarity. The score is scaled to appear between 0 and 1. ILSM is designed so that any external measure of similarity which returns a value between -1 and 1 can be used as $w()$. For example, we could use Pearson Correlation as described above in User-based CF.

For the experiments reviewed in this section, we used the taxonomy-based similarity metric from [163] for $w()$. Given an external taxonomy, each item can be

⁵ All research described in this chapter was performed with Cai-Nicholas Ziegler. It was first published in [162]. A revised version also appears as a part of Cai Ziegler's Ph.D. dissertation [161].

represented as a list of taxonomy entries that describe that item. Further, this list is augmented to recursively include all relevant parent entries from the taxonomy. The similarity metric works in two phases. First, it calculates a weighted score to all taxonomy entries on each item’s list by looking at the ratio of sibling nodes from the taxonomy to nodes in the item’s list. After creating such a score for all items, the metric computes a Pearson correlation calculation between items using their lists as weighted vectors. For details on this metric, see [163].

In our experiments, we also made use of a new Topic Diversification algorithm [161, 162]. This algorithm constructs recommendation lists by selecting items that are maximally different from each other according to some similarity metric $w()$. The algorithm works as follows:

The algorithm starts with an undiversified recommendation list; a value n , denoting how long the diversified list is to be; a similarity measure $w()$, and a diversification factor $\Theta_F, 0 < \Theta_F \leq 1$. The algorithm builds the final recommendation list incrementally. The top recommendation is moved to the final list. Then the similarity between the final list and each of the items on the old list is calculated using $w()$, creating a dissimilarity ranking for each item. This dissimilarity ranking and the original recommendation ranking are merged together, using Θ_F to determine the relative weighting. The item with the best rank is moved to the final recommendation list, and the process repeats. Note that at each step, the similarity measure for a candidate item is calculated against the set of items already on the new recommendation list. The algorithm ends when n items have been selected.

Experiments

We conducted offline evaluations to understand the ramifications of topic diversification on accuracy metrics, and online analysis to investigate how our method affects actual user satisfaction. We applied topic diversification with $\Theta_F \{0, 0.1, 0.2, \dots 0.9\}$ to lists generated by both User-based CF and Item-based CF, observing effects that occur when steadily increasing Θ_F and analyzing how both approaches respond to diversification.

In a 4-week crawl, we collected data on 278,858 members of BookCrossing and 1,157,112 ratings, both implicit and explicit, referring to 271,379 distinct ISBNs.⁶ Invalid ISBNs were excluded from the outset. Next, we mined Amazon.com’s book taxonomy, comprising 13,525 distinct topics. In order to be able to apply topic diversification, we mined content information, focusing on taxonomic descriptions that relate books to taxonomy nodes from Amazon.com. Since many books on BookCrossing refer to rare, non-English books, or outdated titles not in print anymore, we were able to garner background knowledge for only 175,721 books. In total, 466,573 topic descriptors were found, giving an average of 2.66 topics per book.

Owing to the BookCrossing dataset’s extreme sparsity, we decided to further condense the set in order to obtain more meaningful results from CF algorithms when computing recommendations. Hence, we discarded all books missing taxonomic descriptions, along with all ratings referring to them. Next, we also removed book titles with fewer than 20 overall mentions. Only community members with at least five ratings each were kept. The resulting dataset’s dimensions were considerably more moderate, featuring 10,339 users, 6,708 books, and 361,349 book ratings.

Offline Experiments

We performed offline experiments comparing precision, recall, and intra-list similarity scores for 20 different recommendation list setups. Half these recommendation lists were based upon User-based CF with different degrees of diversification, the others on Item-based CF. We did not compute MAE metric values since we are dealing with implicit rather than explicit ratings.

For cross-validation of precision and recall metrics of all 10,339 users, we adopted 4-folding cross validation, splitting the dataset into 4 sections and using each one in turn as the test set. For each of the 41,356 different training sets, we computed 20 top-10 recommendation lists. To generate the diversified lists, we computed top-50 lists

⁶ The complete BookCrossing dataset, featuring fully anonymized information, is available from Cai Ziegler’s website at <http://www.informatik.uni-freiburg.de/~ctieglar/>.

based upon pure, i.e., non-diversified, Item-based CF and pure User-based CF. The high-performance Suggest recommender engine was used to compute these base case lists [67]. Next, we applied the diversification algorithm to both base cases, applying Θ_F factors ranging from 10% up to 90%. For evaluation, all lists were truncated to contain 10 books only.

Results

We were interested in seeing how accuracy, captured by precision and recall, behaves when increasing Θ_F from 0.1 up to 0.9. Since topic diversification may make books with high predicted accuracy trickle down the list, we hypothesized that accuracy will deteriorate as Θ_F approaches 0.9. Moreover, in order to find out if our novel algorithm has any significant, positive effects on the diversity of items featured, we also applied the intra-list similarity metric. An overlap analysis for diversified lists, $\Theta_F \geq 0.1$, versus their respective nondiversified pendants indicates how many items stayed the same for increasing diversification factors.

Table 4-2: Precision and Recall for Non-diversified CF

	User-based CF	Item-based CF
Precision	3.69	3.64
Recall	5.76	7.32

First, we analyzed precision and recall scores for both nondiversified base cases, i.e., when $\Theta_F = 0$. Table 4-2 states that User-based and Item-based CF exhibit almost identical accuracy, indicated by precision values. Their recall values differ considerably, hinting at deviating behavior with respect to the types of users they are scoring for.

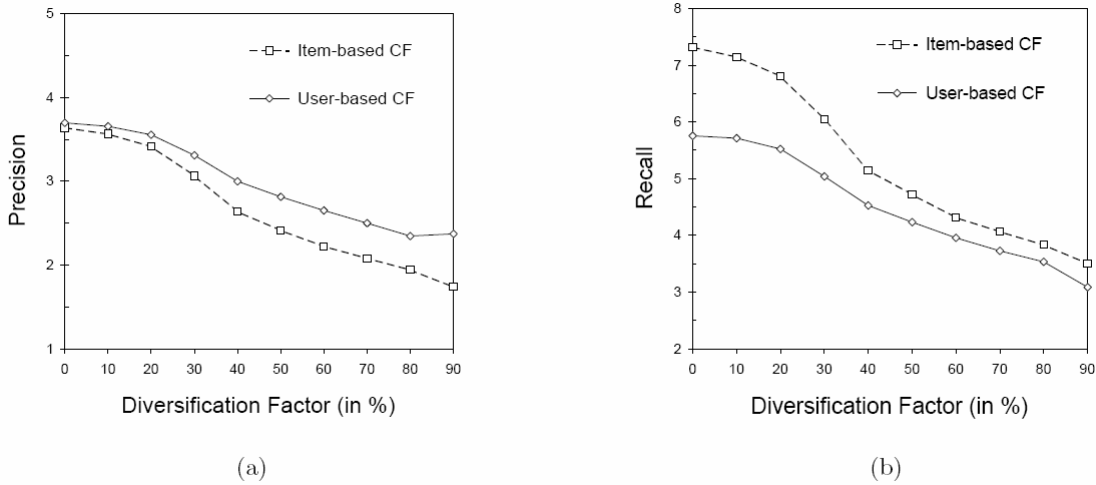


Figure 4-1: Precision (a) and Recall (b) for Increasing Diversity

Next, we analyzed the behavior of User-based and Item-based CF when steadily increasing Θ_F by increments of 10%, depicted in Figure 4-1. The two charts reveal that diversification has detrimental effects on both metrics and on both CF algorithms. Interestingly, corresponding precision and recall curves have almost identical shape. The loss in accuracy is more pronounced for Item-based CF than for User-based CF. Furthermore, the drop is most distinctive for Θ_F between 0.2 and 0.4. For lower Θ_F , negative impacts on accuracy are marginal. We believe this last observation because precision and recall are permutation-insensitive, i.e., the mere order of recommendations within a top-N list does not influence the metric value, as opposed to Breese score [13, 51]. However, for low Θ_F , the pressure that the dissimilarity rank exerts on the top-N list's makeup is still too weak to make many new items diffuse into the top-N list. Hence, we conjecture that rather the positions of current top-N items change, which does not affect either precision or recall.

Knowing that our diversification method exerts a significant, negative impact on accuracy metrics, we wanted to know how our approach affected the intra-list similarity measure. Similar to the precision and recall experiments, we computed metric values for User-based and Item-based CF with Θ_F running from 0.0 to 0.9. Hereby, we instantiated

the intra-list similarity metric function $w()$ with Cai-Nicholas Ziegler’s taxonomy-driven metric. Results obtained from intra-list similarity analysis are given in Figure 4-2(a).

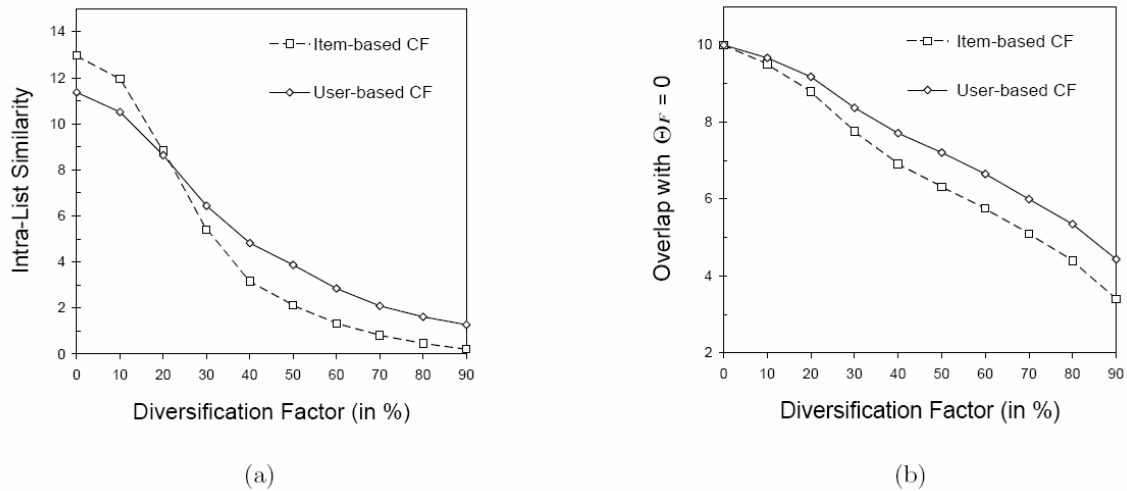


Figure 4-2: Intra-List Similarity Behavior (a) and Overlap with Pure CF Lists (b)

The topic diversification method considerably lowers the pair-wise similarity between list items, thus making top-N recommendation lists more diverse. Diversification appears to affect Item-based CF stronger than its User-based counterpart, in line with our findings about precision and recall. For lower Θ_F , curves are less steep than for between 0.2 and 0.4, which also well aligns with precision and recall analysis. Again, the latter phenomenon can be explained by one of the metric’s inherent features, i.e., like precision and recall, intra-list similarity is permutation-insensitive.

Figure 4-2(b) shows the number of recommended items staying the same when increasing Θ_F with respect to the original list’s content. Both curves exhibit roughly linear shapes, being less steep for low Θ_F , though. Interestingly, for factors $\Theta_F \leq 0.4$, at most three recommendations change on average.

We found that diversification appears detrimental to both User-based and Item-based CF along precision and recall metrics. In fact, this outcome aligns with our expectations, considering the nature of those two accuracy metrics and the way that the

topic diversification method works. Moreover, we found that Item-based CF seems more susceptible to topic diversification than User-based CF, backed by results from precision, recall, and intra-list similarity metric analysis.

Online Experiments

Offline experiments helped us in understanding the implications of topic diversification on both CF algorithms. We could also observe that the effects of our approach are different on different algorithms. However, knowing about the deficiencies of accuracy metrics, we wanted to assess actual user satisfaction for various degrees of diversification, thus necessitating an online survey.

For the online study, we computed each recommendation list type anew for users in the denser BookCrossing dataset, though without K-folding. In cooperation with BookCrossing, we emailed all eligible users via the community mailing system, asking them to participate in our online study. Each email contained a personal link that would direct the user to our online survey pages. In order to make sure that only the users themselves would complete their survey, links contained unique, encrypted access codes. During the 3-week survey phase, 2,125 users participated and completed the study.

The survey consisted of several screens that would tell the prospective participant about this study's nature and his task, show all his ratings used for making recommendations, and finally present a top-10 recommendation list, asking several questions thereafter. For each book, users could state their interest on a 5-point rating scale. Scales ranged from "not much" to "very much", mapped to values 1 to 4, and allowed the user to indicate that he had already read the book, mapped to value 5. In order to successfully complete the study, users were not required to rate all their top-10 recommendations. Neutral values were assumed for non-votes. However, we required users to answer all further questions, concerning the list as a whole rather than its single recommendations, before submitting their results. We embedded those questions we were actually keen about knowing into ones of lesser importance, in order to conceal our intentions and not bias users.

The one top-10 recommendation list for each user was chosen among 12 candidate lists, either User-based CF or Item-based CF with $\Theta_F \in \{0, 0.3, 0.4, 0.5, 0.7, 0.9\}$ each. We opted for those 12 instead of all 20 list types in order to acquire enough users completing the survey for each slot. The assignment of a specific list to the current user was done dynamically, at the time of the participant entering the survey, and in a round-robin fashion. Thus, we could guarantee that the number of users per list type was roughly identical.

Results

For the analysis of our inter-subject survey, we were mostly interested in the following three aspects. First, the average rating users gave to their 10 single recommendations. We expected results to roughly align with scores obtained from precision and recall, owing to the very nature of these metrics. Second, we wanted to know if users perceived their list as well-diversified, asking them to tell whether the lists reflected rather a broad or narrow range of their reading interests. Referring to the intra-list similarity metric, we expected users' perceived range of topics, i.e., the list's diversity, to increase with increasing Θ_F . Third, we were curious about the overall satisfaction of users with their recommendation lists in their entirety, the measure to compare performance. Both latter-mentioned questions were answered by each user on a 5-point likert scale, higher scores denoting better performance, and we averaged the eventual results by the number of users. Statistical significance of all mean values was measured by parametric one-factor ANOVA, where $p < 0.05$ if not indicated otherwise.

Users perceived recommendations made by User-based CF systems on average as more accurate than those made by Item-based CF systems, as depicted in Figure 4-3(a). At each featured diversification level Θ_F , differences between the two CF types are statistically significant, $p < 0.01$. Moreover, for each algorithm, higher diversification factors obviously entail lower single-vote average scores, which confirms our hypothesis stated before. The Item-based CF's cusp at Θ_F between 0.3 and 0.5 appears as a notable outlier, opposed to the trend, but differences between those three means are not

statistically significant, $p > 0.15$. Contrarily, differences between all factors Θ_F are significant for Item-based CF, $p \ll 0.01$, and for User-based CF, $p < 0.1$. Hence, topic diversification *negatively* correlates with pure accuracy. Besides, users perceived the performance of User-based CF as significantly better than Item-based CF for all corresponding levels Θ_F .

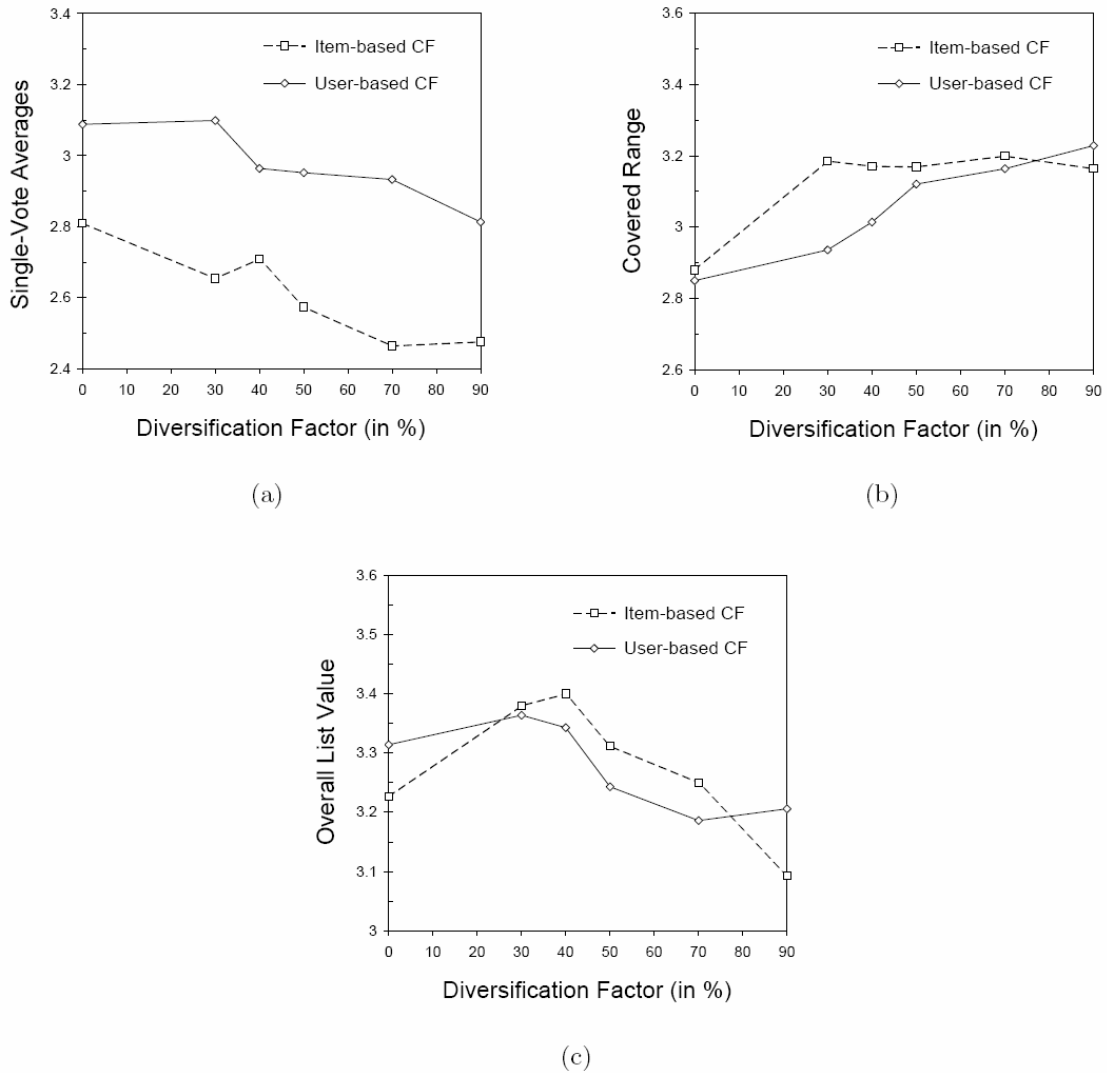


Figure 4-3: Results for Single-Vote Averages (a), Covered Range of Interests (b), and Overall Satisfaction (c)

Next, we analyzed whether users actually perceived the variety-augmenting effects caused by topic diversification, illustrated before through the measurement of intra-list similarity. Users' reactions to steadily incrementing Θ_F are illustrated in Figure 4-3(b). Between both algorithms on corresponding Θ_F levels, only the difference of means at $\Theta_F = 0.3$ shows statistical significance.

Studying the trend of User-based CF for increasing Θ_F , we notice that the perceived range of reading interests covered by users' recommendation lists also increases. Hereby, the curve's first derivative maintains an approximately constant level, exhibiting slight peaks between Θ_F at 0.4 and 0.5. Statistical significance holds for User-based CF between means at $\Theta_F = 0$ and $\Theta_F > 0.5$, and between $\Theta_F = 0.3$ and $\Theta_F = 0.9$.

On the contrary, the Item-based curve exhibits a drastically different behavior. While soaring at $\Theta_F = 0.3$ to 3.186, reaching a score almost identical to the User-based CF's peak at $\Theta_F = 0.9$, the curve barely rises for $\Theta_F \geq 0.4$, remaining rather stable and showing a slight, though insignificant, upward trend. Statistical significance was shown for $\Theta_F = 0$ with respect to all other samples taken from $\Theta_F \geq 0.3$. Hence, our online results do not perfectly align with findings obtained from offline analysis. While the intra-list similarity chart in Figure 4-2 indicates that diversity increases when increasing Θ_F , the Item-based CF chart defies this trend, first soaring then flattening.

We conjecture that the following two factors account for these peculiarities:

- **Diversification factor impact.** Our offline analysis of the intra-list similarity already suggested that the effect of topic diversification on Item-based CF is much stronger than on User-based CF. Thus, the Item-based CF's user-perceived interest coverage is significantly higher at $\Theta_F = 0.3$ than the User-based CF's.
- **Interaction with accuracy.** Analyzing results obtained, bear in mind that covered range scores are not fully independent from single-vote averages.

When accuracy is poor, i.e., the user feels unable to identify recommendations that are interesting to him, chances are high his discontentment will also negatively affect his diversity rating. For $\Theta_F \geq 0.5$, single-vote averages are remarkably low, which might explain why perceived coverage scores do not improve for increasing Θ_F .

However, we may conclude that users do perceive the application of topic diversification as an overall positive effect on reading interest coverage.

The third feature variable we were evaluating, the overall value users assigned to their personal recommendation list, effectively represents the target value of our studies, measuring actual user satisfaction. Owing to our conjecture that user satisfaction is a mere composite of accuracy and other influential factors, such as the list's diversity, we hypothesized that the application of topic diversification would increase satisfaction. At the same time, considering the downward trend of precision and recall for increasing Θ_F , in accordance with declining single-vote averages, we expected user satisfaction to drop off for large Θ_F . Hence, we supposed an arc-shaped curve for both algorithms.

Results for overall list value are given in Figure 4-3(c). Analyzing User-based CF, we observe that the curve does not follow our hypothesis. Slightly improving at $\Theta_F = 0.3$ over the non-diversified case, scores drop for Θ_F between 0.4 and 0.7, eventually culminating in a slight but visible upturn at $\Theta_F = 0.9$. While lacking reasonable explanations and being opposed to our hypothesis, the curve's data-points bear no statistical significance for $p < 0.1$. Hence, we conclude that topic diversification has a marginal, largely negligible impact on overall user satisfaction, initial positive effects eventually being offset by declining accuracy.

On the contrary, for Item-based CF, results obtained look different. In compliance with our previous hypothesis, the curve's shape roughly follows an arc, peaking at $\Theta_F = 0.4$. Taking the three data-points defining the arc, we obtain statistical significance ($p < 0.1$). Since the endpoint's score at $\Theta_F = 0.9$ is inferior to the non-

diversified case's, we observe that too much diversification appears detrimental, perhaps owing to substantial interactions with accuracy. Eventually, for overall list value analysis, we come to conclude that topic diversification has no measurable effects on User-based CF, but significantly improves Item-based CF performance for diversification factors Θ_F around 40%.

Multiple Linear Regression

Results obtained from analyzing user feedback along various feature axes already indicated that users' overall satisfaction with recommendation lists not only depends on accuracy, but also on the range of reading interests covered. In order to more rigidly assess that indication by means of statistical methods, we applied multiple linear regression to our survey results, choosing the overall list value as dependent variable. As independent input variables, we provided single-vote averages (SVA) and covered range (CR), both appearing as first-order and second-order polynomials, i.e., SVA and CR, and SVA^2 and CR^2 , respectively. We also tried several other, more complex models, without achieving significantly better model fitting.

Table 4-3: Multiple Linear Regression Results

	Estimate	Error	t-Value	Pr(> t)
(const)	3.27	0.023	139.56	< 2e -16
SVA	12.42	0.973	12.78	< 2e -16
SVA^2	-6.11	0.976	-6.26	4.76e -10
CR	19.19	0.982	19.54	< 2e -16
CR^2	-3.27	0.966	-3.39	0.000727

Analyzing multiple linear regression results, shown in Table 4-3, confidence values $Pr(> |t|)$ clearly indicate that statistically significant correlations for accuracy and covered range with user satisfaction exist. Since statistical significance also holds for their respective second-order polynomials, i.e., CR^2 and SVA^2 , we conclude that these relationships are non-linear and more complex, though. In fact, linear regression delivers

a strong indication that the intrinsic utility of a list of recommended items is more than just the average value of accuracy votes for all single items, but also depends on the perceived diversity.

Limitations

There are limitations in the way topic diversification was implemented. Though the Amazon.com taxonomies were human-created, there may still be some mismatch between what the topic diversification algorithm perceives as “diversified” and what humans do. The issue is effectively inherent to the taxonomy’s structure, which has been designed with browsing tasks and ease of searching rather than with interest profile generation in mind. For instance, the taxonomy features topic nodes labeled with letters for alphabetical ordering of authors from the same genre, e.g., *Books > Fiction > . . . > Authors, A-Z > G*. Hence, two Sci-Fi books from two different authors with the same initial of their last name would be classified under the same node, while another Sci-Fi book from an author with a different last-name initial would not. Though the problem’s impact is largely marginal, owing to the relatively deep level of nesting where such branchings occur, the taxonomy does diverge from intuitive measures of diversity.

Discussion and Implications

Contrasting precision and recall metrics, computed both for User-based and Item-based CF and featuring different levels of diversification, with results obtained from a large-scale user survey, we showed that the user’s overall liking of recommendation lists goes beyond accuracy and involves other factors, e.g., the users’ perceived list diversity. We were thus able to provide empirical evidence that lists are more than mere aggregations of single recommendations, but bear an intrinsic benefit. This is a profound result, providing evidence that the value of a list to a user cannot be adequately expressed by the aggregation of measures of its elements. While this experiment only tested diversity, we hypothesize there are a variety of factors important to users at different points in time. To be highly effective, we argue, recommenders need to understand these factors and tailor recommendation lists accordingly.

Though effects of diversification were largely marginal on User-based CF, Item-based CF performance improved significantly, an indication that there are some behavioral differences between both CF classes. Moreover, while pure Item-based CF appeared inferior to pure User-based CF in overall satisfaction, diversified Item-based CF outperformed User-based CF. Interestingly, for $\Theta_F < 0.5$, no more than three items tend to change with respect to the original list, shown in Figure 4-2(b). Small changes thus have high impact. This could be a “basking in the glory” effect: if a few items are great recommendations, it makes the others look good as well. The opposite effect is also possible; a few horrible recommendations could poison the rest of a list for a user.

CHAPTER 5

RECOMMENDING CITATIONS FOR RESEARCH PAPERS

Collaborative filtering uses relationships between people to generate recommendations, an approach that fits nicely with the idea that humans are fundamentally social creatures. One consequence of our sociability is that in our interactions with other people, we create social networks. These networks can be personal, professional, or political, can have varying importance on people's lives, and have been studied in detail by sociologists, psychologists, and physicists, among others [134]. While studying direct human-human social networks is extremely important, it is also worthwhile to study the social networks of *artifacts*, especially when these networks directly relate people and their artifacts together.

We are interested in how CF can be applied in one of these social networks of artifacts: the network of research papers.⁷ The citations between research papers form a graph that can be viewed as a social network, known as a citation web. For any given paper, it is possible to follow the citation web to see what papers cite it and what papers are cited by it.

Searching for related work can be tedious and it is possible to miss important developments in areas outside a researcher's specialty. Many searches for related work consist of starting with a small number of initial papers and navigating the citation web near those papers. For example, ResearchIndex provides an interface to the citation web for computer science research papers [74]. Search engines such as Google, Yahoo, and MSN also help researchers find related work.

With these strategies, however, it is likely that some important related work will be missed: ResearchIndex is unable to scan the entire citation web effectively to find

⁷ All research described in this chapter were previously published in [90, 146]. The analysis of hybrid recommender algorithms in this domain was performed with Roberto Torres. A revised version of [146] appears in Torres' Masters Thesis.

connections between papers. Search engines, while able to scan all documents for relevant text, do not use semantic information, such as paper citations, in their results. Those that do, such as Google Scholar [45], do not support any browsing features, thus making it easy to find a paper only when exact information is already known. We hypothesize that recommender systems, especially those based on collaborative filtering, will discover more powerful relationships among the citation web of research papers than the above methods for searching.

ResearchIndex is a digital library in which the content is not created, selected, or edited by professional staff, but by automatic algorithms. We call a digital library of this type an *emergent digital library*. Emergent digital libraries are similar to other digital libraries in many respects: they have a well-defined corpus of material; that corpus is digital in nature, and hence may be distributed by digital means; and they have powerful cataloging and search technologies appropriate to their content. However, emergent digital libraries have a key difference: the quality of their content varies more widely than the quality of content in most digital libraries, because professional staff is not involved in collection and appraisal. Therefore, exploration and search techniques are needed that can seek quality and relevance of results beyond what keyword similarity can provide. Our research seeks to explore such techniques.

We chose to use ResearchIndex as a test bed for our exploration as it provides several advantages. ResearchIndex has spent years developing automatic citation indexing techniques that work on the domain of research papers [73]. They have demonstrated effective citation processing heuristics both for extracting citations from research papers, and for resolving multiple versions of the same citation in other research papers. They have become a frequently visited and valuable resource to the research community.

Integrating Recommenders into the Domain of Research Papers

We draw a subtle but important distinction between a citation and a paper. A *citation* represents a research paper for which we have a reference (i.e., a paper has cited this

citation, but we may or may not possess the paper which corresponds to the citation). A *paper* is a *citation* for which we have access to the full text, including the paper's citation list. Thus, for a paper we have the citations that it references, some of which may also be papers in our dataset but all of which must be citations in our dataset. It should be noted that a paper could exist without a citation. It would imply that the full text of the paper is in the system, but there is insufficient knowledge of its publication status, authors, etc. to create a citation.

As discussed in Chapter 3, standard CF algorithms work by viewing a dataset as a ratings matrix. Columns of the matrix represent items in the CF environment, while rows represent users. Each entry in the matrix is the user's rating of a particular item. These ratings are gathered from users either implicitly, such as through purchase records and browsing history, or explicitly, by asking users to rate the items. Collaborative filtering predicts which values would appear in blank spots in this matrix by comparing the similarities of either the rows (users) or the columns (items) in the matrix. In order to perform collaborative filtering on the domain of research papers, we need to map the citation network onto a collaborative filtering ratings matrix. There are several ways to do so.

The first way is to consider an analogy to MovieLens and other current collaborative filtering systems. In such a system, citations would be the 'items' in the matrix, while real people would be the 'users' who rate citations. This approach does not use the network of citations between papers and suffers from many problems, such as the Cold Start problem discussed in Chapter 3. Even worse, social factors might inhibit or bias the ratings of colleagues' work. We instead chose to mine the citation web directly.

An alternative approach is to make paper authors the 'users' and keep citations as 'items.' In this ratings matrix, each author would "vote" for the papers that she has cited. By using the citation web to populate the matrix, this mapping does not suffer from the Cold Start problem. This is the method that has been explored before by Kautz et al. [68] and Newman [103].

This approach suffers from a generality problem. Many authors have written papers in different fields over the course of their careers. For example, Ben Shneiderman has written many papers about human computer interaction and user interface design. However, he started his career with papers about FORTRAN and has written several papers about Jewish heritage and history. With many prolific authors in the system such as Ben Shneiderman, it might be difficult to find a set of authors that would describe your information need. Imagine a user submitting as input a list of papers on user interfaces, and receiving recommendations for papers on Jewish history. This serendipity is useful in many domains, exposing users to items they might not otherwise consider, but may not be as useful in the context of research papers.

Instead of using an author’s list of citations, we chose a mapping that uses the citation lists from individual papers. As shown in Figure 5-1, each paper represents a user in the matrix and a citation represents an item. Each paper votes for the citations found in its references list. This mapping uses the citation network to populate the ratings matrix and therefore does not suffer from the Cold Start problem. This mapping also does not suffer from the loss of specificity that could occur with the author-citation mapping.

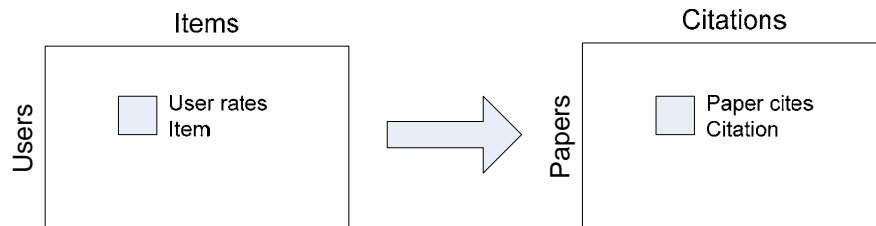


Figure 5-1: Our Ratings Matrix for Research Papers

By having papers as users, we are guaranteed that every user in the system provides ratings information in the form of its votes. This allows us to create our ratings matrix without any startup problems. These users will never add more votes after the paper is first entered. This is in contrast to most CF environments where it is expected that users will add more ratings information over time. As such, in this domain

recommendation performance can be enhanced using model-based approaches. See Chapter 3 for a discussion. It is also expected that the citation lists will be of high quality as the dataset represents research publications designed for public dissemination, which frees the system from ratings consistency and ‘rogue user’ issues. This is the mapping will use.

There are other mappings worth considering. For instance, there is the mapping where both the ‘users’ and ‘items’ in the ratings matrix are citations. In this matrix, a rating between the two would be some measure of co-citation between the two elements. A basic co-citation metric is a count of the number of times the two citations appear together in a references list on a paper. This measure of co-citation is related to the item-item collaborative filtering algorithm [131], which has been very successful in other domains. We include a co-citation algorithm based on this idea as an alternative in our experiments.

Pure Recommender Algorithms for Research Papers

For our first set of experiments, we created six recommender algorithms: two collaborative filtering, two data mining or machine learning, and two baseline algorithms representing ways users currently search for papers. Each of the six algorithms receives a “basket” of citations as input and returns a ranked list of recommended citations. In our experiments, the basket consists of the citations made by a “target paper” for which we recommend other citations the author may wish to know about. This design is based on the following scenario, “I have read this paper, and I have looked at its citations. I would like recommendations for more citations to review”. The algorithms do not recommend items in the input basket. The Google and Citation Graph Search algorithms use additional information (the title and abstract of the target paper) not used by the other methods.

Collaborative Algorithms

We use the following collaborative filtering, machine learning, and data mining algorithms in our experiments. Additional details on some of the algorithms are available in Chapter 3.

Co-Citation Matching

Co-citation Matching works by counting co-citations. For each citation in the basket, the algorithm counts the number of times other citations were co-cited with it. The algorithm recommends citations with the highest total co-citations summed over all of the basket items. This algorithm uses raw co-citation counts and is not normalized; it is a representation of ‘popularity’ in the co-citation space and is used as a baseline.

User-based CF

User-based CF is the original k-Nearest Neighbor CF algorithm. Given the ratings matrix discussed earlier, the User-based CF algorithm compares papers (rows) in the matrix to create a neighborhood of the most similar papers to the target paper. Since citations are binary (either a paper is cited or not), the algorithm uses a cosine similarity metric. The algorithm counts the number of times neighbors make a citation, with the count weighted by the similarity of each neighbor to the target paper. The algorithm recommends citations with the highest weighted counts. We used the freely available Suggest recommendation engine for our implementation of this algorithm [67]. For our experiments, k was set at 30.

Item-based CF

Instead of building neighborhoods among users, Item-based CF ‘flips’ User-User collaborative filtering by comparing similar items. The Item-Item algorithm compares citations (columns) in the ratings matrix to create a neighborhood, an ‘item’ neighborhood, of the closest citations to each item in the basket. Again, we use Suggest for the implementation with a cosine similarity metric. For our experiments, k was set at 50.

Naïve Bayes Classifier

The Naïve Bayes Classifier calculates probabilities that any given citation in the dataset is related to the input basket. The algorithm sorts the citations by probability and recommends citations from highest to lowest probability. The classifier is trained on citation lists from the dataset. Even in domains where the naïve Bayes principle does not hold, Bayesian methods still work remarkably well [13]. We used our own implementation of this classifier in our experiments.

Baseline Algorithms

We could use many other algorithms as baselines. We chose two algorithms that we feel approximated how researchers look for relevant citations: browsing the citations around the given paper, and using a search engine.

Localized Citation Graph Search

The Citation Graph Search uses keyword similarity between the target paper’s abstract and the titles of papers nearby in the citation graph. For a target paper, this algorithm makes a list of papers that cite the target, papers co-cited with the target, and papers cited by items in the basket. Using the analogy of a ‘family tree’, this algorithm grabs the target paper’s parents, the target paper’s siblings, and the target paper’s grandchildren. The algorithm uses TF/IDF term weighting to match the titles of these nearby citations against the title and abstract of the paper in question, and returns recommendations sorted by relevance. It is worth noting that this algorithm assumes the target paper exists in the citation network. Often the ‘real world’ equivalent of our single paper scenario is in the review of a paper submitted to a journal or conference. Thus, this is a ‘best case’ algorithm.

Keyword Search (‘Google’ Baseline)

The title of the paper in question is sent as a query to the Google search engine [44], restricted to results that appear in ResearchIndex. The algorithm recommends papers in the order returned by Google.

Pure Algorithm Experiments

We performed two different experiments to see how the six algorithms would perform in the domain of research papers. In the offline experiment, we were interested in studying the ability of each algorithm to find missing citations using a leave-one-out methodology. In the online experiment, we used human subjects and focused on collecting user opinion on the quality and usefulness of predictions made by our algorithms.

Offline Experiment

The offline experiment tests whether our algorithms are useful for predicting specific relevant citations for a given paper.

We started with a dataset based on the over 500,000 research papers available through ResearchIndex. In order to make that data more manageable for our algorithms, we chose to reduce this dataset by dropping papers that contained fewer than two citations and by dropping citations that were cited fewer than two times. The motivation for doing so was that these papers and citations do not contain enough ties to the rest of the dataset and thus only introduce noise. About 186,000 papers and 485,000 citations remained after removing these poorly connected papers. Papers that remained had an average of 16 connections to other items, either papers or citations, in the dataset.

Experimental Design

This experiment consisted of removing citations from the test dataset and then attempting to predict those missing citations. For each paper in the test dataset, we randomly removed one citation from that paper's reference list. The remaining citations were used to generate a list of recommended citations. We then found the rank of the removed citation in the list of recommendations. This protocol is similar to the "All but one" protocol in [13] but we also include coverage when calculating the prediction accuracy.

We divided our dataset into training and test datasets at a 90% to 10% ratio in the following manner: For 90% of the papers we wrote their citations into the training dataset. For the remaining 10% of the papers, we removed one citation to create a test case for the test dataset. We included the incomplete citation lists from the test cases in

the training dataset. We performed a 10-fold cross validation for this experiment; where in each fold we randomly assigned data into either the test or training datasets. Each recommendation algorithm was run with all of the datasets.

The experiment has some limitations. The most significant one was that our recommendation algorithms might discover citations that are more relevant than the one held out. Such citations may have not been included in the paper's references list because of limits on space or because they overlapped with other references, possibly the one left out. However, we feel that authors generally do a good job selecting citations for their papers, so we expected the removed citation to be recommended.

Another factor to consider was that our recommendation algorithms could recommend citations that did not exist when the paper in question was written. To deal with this problem, we filtered out recommendations with a publication year later than that of the target paper.

Offline Experiment Algorithms

Five of our six algorithms were used for the offline experiments. We did not include Google Baseline because the volume of searching would have greatly exceeded Google's search limits and appropriate usage policy.

Offline Experiment Metrics: Rank, Coverage, and Effective Rank

We examine three metrics for this experiment: rank, coverage, and effective rank. We define rank as the position of the test citation in the filtered recommendation list. Thus, the lower the rank, the higher the item appears on the recommendation list, with the best rank defined as '1'. Rank is a proxy for user utility, since users prefer to find relevant results earlier. In this context, rank is also similar to a precision metric from information retrieval applications. To aggregate the results, we measure the percentage of the time that the rank of the withheld item appears within a given threshold (top-10, top-20, top-30, top-40, etc.). Thus, a "20%" score at Top-10 on this metric means that 20% of the time, when a withheld item was found in the recommendation list, it was found at rank of 10 or better (lower). We define Rank at top- n as

$$Rank(n) = \frac{H_n}{H}$$

where H_n is the number of “hits” at rank n or better (lower) and H is the total number of “hits” the algorithm generated over all test cases.

Coverage is the percentage of the time that an algorithm is able to make a recommendation successfully. Normally, this means that the algorithm was able to make some recommendations. In this experiment, we define a “successful” recommendation as one that *includes* the test citation. As a complement to rank, coverage is similar to a recall metric from information retrieval. So we define coverage as,

$$Coverage = \frac{H}{T}$$

where H is the number of “hits” the algorithm generated over all test cases, and T is the number of test cases.

Since users are most interested in how much value they will get from an algorithm in practice, we calculate an effective rank metric that scales the recommendation percentage by coverage for each rank tested. When measuring the percentage of times a recommender returned an item below a certain rank, effective rank penalizes recommenders for those situations it was unable to recommend any item at all. Thus, the effective rank metric, as a combination of rank and coverage, is similar to the F1 or other combination metrics found in information retrieval. It has the same properties as the Rank score. For a top- n , Adjusted Rank is defined as

$$AdjRank(n) = Coverage * Rank(n) = \frac{H_n}{T}$$

with H_n and T as defined above.

Offline Experiment Results

Figure 5-2 shows the performance of each recommender at recommending the removed citation within its top 1, 10, 20, 30, or 40 recommendations. Item-Item, User-User, and Graph Search all perform substantially better than either the Co-citation or Bayesian recommenders.

Recommendable Removed Citations Recommended Before or at Rank

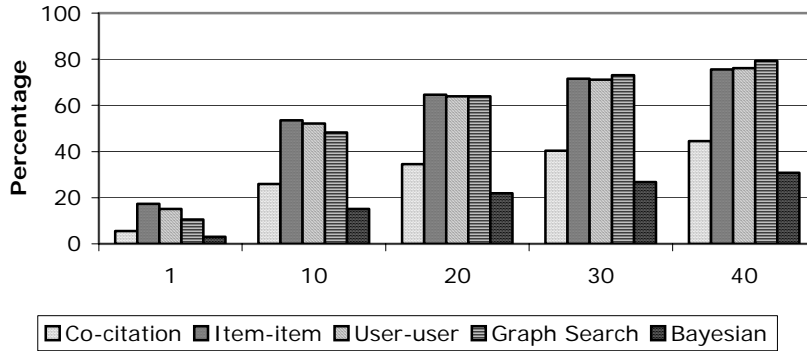


Figure 5-2: For removed citations that an algorithm was able to recommend, the percentage of citations recommended first, and in the top 10, 20, 30, or 40 by each algorithm

However, Figure 5-2 does not consider coverage. The five recommenders vary greatly in their coverage. The Bayesian recommender can recommend 100 percent of the removed citations. Localized Graph Search, on the other hand, only searches a small part of the citation graph, generating predictions for only 28.1% of these citations. Item-item (51.3%), Co-Citation (62.4%), and User-User (67.5%) fall between these two extremes.

Test Citations Recommended Below or at Rank

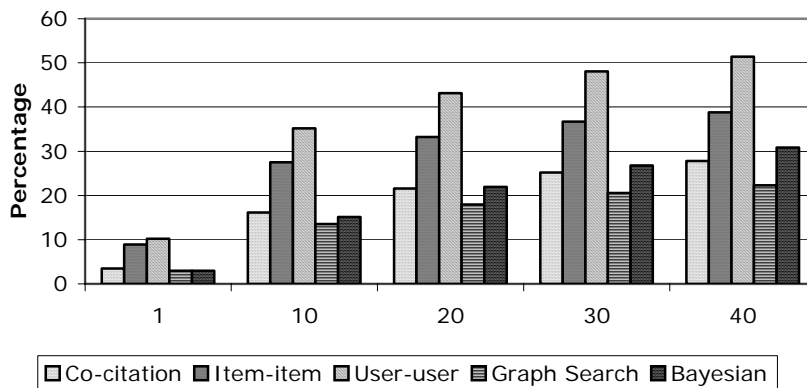


Figure 5-3: For all removed citations, the percentage of citations recommended first, and in the top 10, 20, 30, or 40 by each algorithm

Figure 5-3 reveals that considering coverage using the effective rank metric changes the relative performance of the recommenders. Graph Search falls to the bottom, while Naïve Bayes improves to the middle of the pack. User-User and Item-Item continue to lead all recommenders, with User-User's greater coverage putting it above the rest. The differences between the User-User and Item-Item recommenders versus the other three were statistically significant at all ranks ($p < 0.05$).

Offline Experiment Discussion

Graph Search performed well on the papers it was able to recommend for, but based on its coverage, it was “cherry-picking” easy-to-recommend test cases. This emphasizes the importance of the effective rank results. Recommendation algorithms must both provide high quality recommendations, and be able to provide them over as much of the dataset as possible.

Our test cases were slightly biased toward oft-cited papers, because papers that were cited more often were more likely to be test cases. Algorithms that are better at recommending papers that are cited many times had an advantage. This is part of the natural limitations of a leave- n -out methodology. We also conducted a user study to assess the value of novel recommendations in this domain.

Online Experiment

While the offline experiments allowed us to test the effectiveness of our algorithms at a coarse level, it could not assess the value of the recommended citations. What if an algorithm recommended better or equivalent citations? We felt that only an experiment with real people could provide us with that information. We used the offline experiment dataset for the online experiment as well.

Online Experimental Design

We created an online survey and recommendation architecture called TechLens where authors rate the relevance of citations recommended for a paper they had written. We focused on authors because we wanted people who were intimately familiar with a paper to make the judgments about our recommendations. We did not attempt to verify that a

subject in the study was actually the author they claimed to be, as subjects were unpaid volunteers recruited online, we felt there was little incentive for people to lie.

Subjects were invited to our survey through a link from ResearchIndex. After consenting to participate, we asked each user for her name, as it would usually appear in a paper. We then searched for papers in our test dataset that the author had written and asked the author to choose a paper. Each author was randomly assigned to one of our six recommendation algorithms.

The algorithms generated a list of five recommended citations that were not written by the subject. We felt that recommending an author's own papers would not be helpful for this experiment since few authors forget to cite their own work.

Sometimes an algorithm could not produce a recommendation list for a given paper. If this happened, we dynamically reassigned the subject to other recommendation algorithms until we were able to generate recommendations. Data from these subjects is not included below, as we were afraid that an algorithm's inability to produce recommendations for a paper might indicate that it was a "hard" paper to recommend for.

The Survey

For each recommended citation, the subject answered the following two questions: "How relevant is this citation to your paper and its related work?" with the scale: "Relevant and a good addition", "Relevant but redundant", "In the same field but not relevant", "In a different field and not relevant", and "No Answer", and "How familiar are you with this citation?" with the scale: "I have cited this myself", "I have read this but not cited it", "I have heard of this but not read it", "I do not know this at all", and "No answer".

The survey provided a link to the subject's original paper and links to the ResearchIndex entry for each of the recommended citations so that authors could investigate citations with which they were not already familiar.

After evaluating all five citations, we asked the subject about the citations as a whole. These questions were on a five-point scale of "Excellent", "Good", "Fair", "Poor", and "Terrible". We asked about the overall quality, usefulness, and novelty of

the recommendations. We also asked for an overall rating of the recommendations, “all things considered”.

Finally, we asked two questions about whether the recommendations would be suitable for finding related work and generating reading lists. These questions used a scale consisting of “Definitely Yes”, “Probably Yes”, “Maybe”, “Probably Not”, and “Definitely Not”. Subjects were also given an opportunity to provide extra comments they might have had about the experiment.

Online Experiment Algorithms

We used all six algorithms in our online experiment.

Online Experiment Metrics

The answers for each question were tallied and expressed as percentages. For some questions, we merged answers from multiple categories into baskets. For example, a ‘relevance’ basket would include both ‘relevant and a good addition’ and ‘relevant but redundant’, but not include the other possible answers for that question.

Table 5-1: The Number of Subjects for Each Recommender Algorithm

Algorithm	Users
Co-Citation	23
User-Item CF	31
Item-Item CF	30
Naive Bayesian	19
Local Graph Search	26
Google	28

Online Experiment Results

Table 5-1 shows the number of users who received recommendations from each recommendation algorithm. Table 5-2 shows the overall user opinion from the online experiments.

Figure 5-4 and Figure 5-5 show users’ judgment of the relevance and novelty of recommendations they received. Both figures use the basketing approach described in

the metrics section. Since we excluded “no answer” responses in both figures, the percentages in a figure for a given recommender do not add up to 100 percent.

Figure 5-4 shows the percentage of recommendations users considered relevant and irrelevant for each recommender. We considered that a recommendation was relevant if the user chose either the “relevant and a good addition” or the “relevant but redundant” response. The Google, Naïve Bayesian, and Graph Search algorithms tend to do better than the other algorithms, with one in two citations judged as relevant. The lowest-rated algorithm, Item-Item, returns a relevant citation 25% of the time. Graph Search and Google dominate the three least relevant, and the difference between them and those three is statistically significant.

Figure 5-5 shows the percentage of recommendations users considered familiar and novel for each recommender. We considered a recommendation familiar if the user chose either the “I have cited this myself” or the “I have read this, but not cited it” response. The User-User and Item-Item algorithms tend to produce more novel recommendations than the others. Google produced the least novel recommendations. The differences between Google and all of the other algorithms were statistically significant, as was the difference between both Item-Item and User-User against all of the other algorithms.

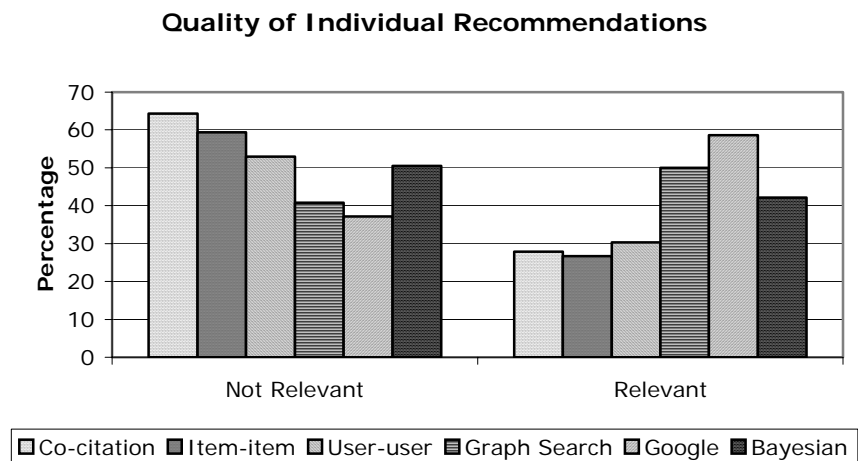


Figure 5-4: Quality of Individual Recommendations

Novelty of Individual Recommendations

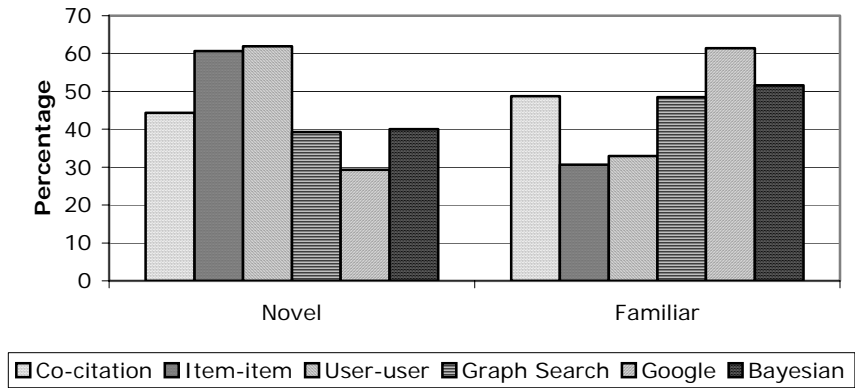


Figure 5-5: Novelty of Individual Recommendations

Table 5-2: Overall User Opinion, in Percentages

	Terrible	Poor	Fair	Good	Excellent
Overall Quality	11	29	31	26	3
Overall Usefulness	16	32	31	20	1
Overall Novelty	7	28	35	25	5
All Things Considered	12	32	32	25	1
	Definitely Not	Probably Not	Maybe	Probably Yes	Definitely Yes
Helpful Finding Related Work	8	24	18	34	17
Helpful Finding Papers to Read	8	15	21	38	19

Google produces both the most relevant and familiar citations, while Item-Item produces the least familiar and least relevant citations. We measured the correlation between a user’s relevance and novelty ratings to be -0.51 . This means that as novelty goes up, relevance tends to go down.

Most users rated the overall, quality, usefulness, and novelty of the recommendations they received in the middle of the range, with more “poor” (43%) than “good” (25%) ratings. The best quality was Graph Search, the best usefulness was Google, the best novelty was User-User, and the best “overall” was Graph Search. On

the questions about the helpfulness of the recommendations for research-related tasks, users also rated toward the middle of the range, but were more likely to say “probably yes” (50-56%) than “probably no” (22-32%).

Users’ perception of the usefulness of the recommendations for research tasks differed based on the recommender that generated the user’s recommendations. Figure 5-6 and Figure 5-7 show results for the task-related questions broken down by recommendation algorithm. In both figures, we counted “Definitely Yes” and “Probably Yes” responses as helpful and “Definitely No” and “Probably No” responses as unhelpful. We did not count “Maybe” or blank responses.

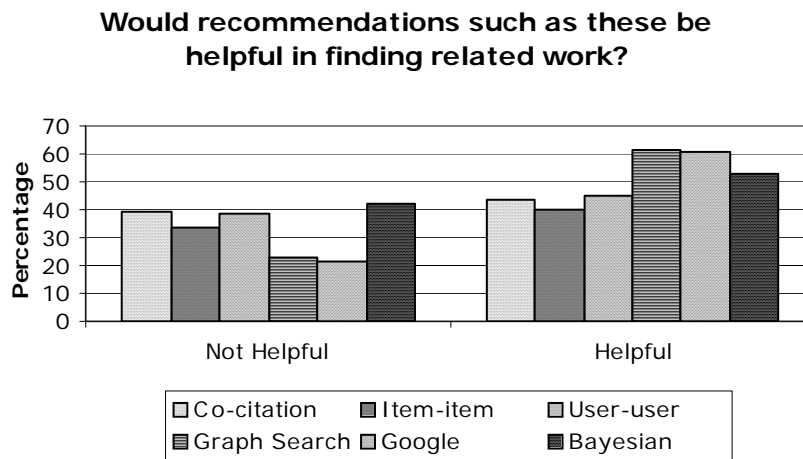


Figure 5-6: Finding Related Work Results

Figure 5-6 shows the percentage of users who thought that a system that generated recommendations like those they received would be helpful for finding related work. The Graph Search algorithm, which produced the most relevant recommendations, was ranked most helpful for this task, and Google, the most likely to generate familiar and relevant citations was considered very helpful.

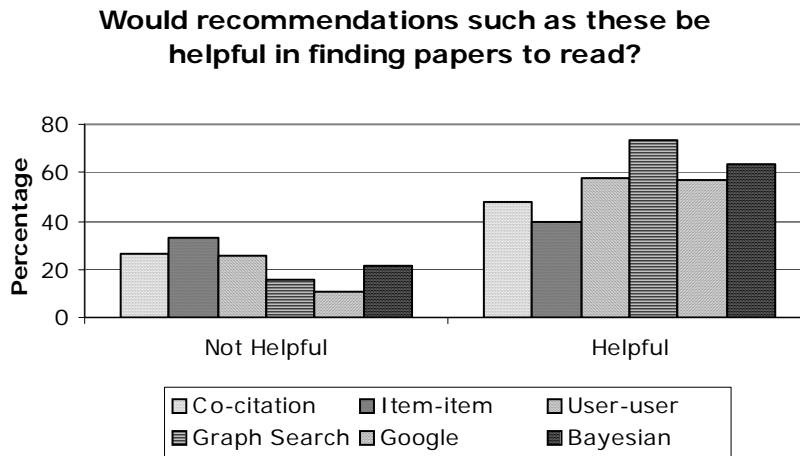


Figure 5-7: Finding Papers to Read Results

Figure 5-7 shows the percentage of users who thought that their recommendations might be useful in finding papers to read. The Graph Search algorithm also performs well here, while the User-User and Naïve Bayesian algorithms move up relative to other algorithms. The difference between both Graph Search and Google to Item-Item was statistically significant.

Online Experiment Discussion

More users thought that the algorithms would be helpful than not helpful, both for finding related work and for locating papers to read. This was true both on average over all algorithms and for each individual algorithm on every task.

This result seems at odds with users’ judgments on the relevance and novelty of individual recommendations. However, consider that even the worst algorithm in terms of quality generated one relevant recommendation in four. Over 60% of users received one or more recommendations that they considered to be a good addition. Another 16% received at least one that was relevant but redundant.

All four CF-based algorithms were able to make recommendations for every paper. The baseline algorithms were not, with Graph Search failing on 19% and Google failing on 39% of recommendation requests. Scaling users’ perception of the helpfulness

of these algorithms by their ability to make recommendations makes Google fall to the bottom, and makes Graph Search roughly comparable to User-User.

The importance of this “cherry-picking” behavior for the baseline algorithms, algorithms that are modeled after how people currently search for related work, cannot be stressed enough. It is important for two reasons. First, these algorithms work well only for the subset of the papers they could search over. This limitation over the coverage of the space greatly reduced the number of possible recommended citations. Second, both of these algorithms scored very high on familiarity. Neither of these algorithms could be effectively used to find novel recommendations.

In order to find both relevant and novel recommendations, no one single algorithm stands out. One interpretation is that an effective system might incorporate several algorithms. This could be done in several ways:

- Try the algorithms in descending order of quality until one produces recommendations.
- Make recommendations from all algorithms, and combine them, perhaps by learning which users like which algorithms and weigh the algorithms’ recommendations.
- Use different recommenders for different tasks. CF-based algorithms produced more novel recommendations that might be more useful for finding reading lists, while the baseline algorithms might be more useful for finding related work.

The inverse correlation between novelty and quality was also interesting. Many novel recommendations are indeed not relevant. However, we received user comments stating that it was difficult to judge relevance accurately only from the information provided by a citation. This means that systems for exploring citation webs such as ResearchIndex must make it easy for users to judge a citation’s relevance quickly by providing as much information as possible about novel citations.

Pure Algorithm Discussion

The User-User and Item-Item algorithms performed very well in the offline experiments, but did not do nearly as well at locating relevant citations in the online experiments.

There are several possible reasons for this discrepancy. Since authors often cite themselves, the offline experiment would often test whether an algorithm could find other relevant papers by the same author. In the online experiment, we specifically prohibited algorithms from recommending citations by the author of the paper. Since these self-citations are presumably relevant and not novel, excluding them would tend to decrease relevance and increase novelty.

In many usage scenarios, it would be appropriate to recommend citations by the same author. We believe that the User-User and Item-Item algorithms will perform better online in such scenarios. Further, the offline experiment constrained the recommenders to only recommend older references, while the online experiments allowed all references. There may be properties of the citation graph that lead some algorithms to perform better for discovering older papers, while other algorithms do better at finding newer papers.

Different algorithms performed better along different dimensions of the tasks. User-User and Item-Item were the dominant algorithms in the offline experiments by the effective rank measure. On the other hand, real users rated Item-Item weak at relevance, and User-User was in the middle of the pack at relevance. Users much preferred User-User for novelty, though, scoring Google, Graph Search, and Bayesian as the least novel. In general, algorithms that performed well at relevance performed poorly at novelty.

Rather than identify one algorithm as the single best for all applications, our experiments lead us to believe that different usage scenarios would likely lead to different best algorithms. For instance, consider the application of finding a few references to complete a reference list for a nearly complete paper. This scenario was most like our online experiment. In general, our users preferred highly relevant references for this application, and were skeptical of the novel references suggested to them. One of our users said, “My paper’s references were much more relevant (as to be expected)”. There might be benefits to our community if users sought more novel references, but our

surveys suggest that users would not use a recommender that pushed them in the direction of novelty. Therefore, the best recommenders for this application would be one that produced the most relevant recommendations for users, which was the Google recommender. Since it fails 2/5 of the time, falling back to the second most relevant algorithm, Naïve Bayesian, would be best in these cases.

A weekly reading group seeking interesting papers to expand their horizons might have very different goals. Google and Graph Search would tend to keep them within a narrow area of closely related papers. For instance, one user said of Graph Search, “I have cited all works recommended by your system in some paper. They are certainly very related. However, they lacked novelty in this case.” This application would do better with an algorithm that scored high in novelty, such as User-User, about which one user said, “They would be somewhat useful if I was trying to get a broader understanding but not if I was trying to get a better understanding of the problem my paper was solving”. Thus, User-User would be less suitable for finding closely related references, but perhaps more suitable for finding novel references for a reading group.

Some usage scenarios fall in between these two. A researcher entering a new research area in which he has read a small number of papers might like a recommender that can help him find a larger collection of papers that span the new area. Here, relevance is important to find papers that are within the new field. But, novelty is important too, since otherwise the papers may be too narrowly focused. We have observed this problem in earlier research in which the Item-Item algorithm tended to recommend movies that were very closely related to previously seen movies, resulting in too narrow a view of a user’s tastes [117]. For these applications, the best recommenders are those that do well at both relevance and novelty. The User-User algorithm was best at relevance in the offline experiment, fourth best at relevance in the online experiment, and best at novelty in the online experiment, so it should do well in this scenario. An even better solution might be to combine User-User with Graph Search or Naïve Bayes, the recommenders users judged more relevant, to make a hybrid algorithm that did well on both relevance and novelty.

Our experiments show that the choice of algorithm affects the type of recommendations produced. The best algorithms we studied can provide either very relevant recommendations or very novel recommendations, though we found no single algorithm that provided both at the same time. The use of the citation network to avoid the startup problems that are a challenge for collaborative filtering was effective. The algorithms all ran with very little data from the user: just one paper that we could look up in ResearchIndex.

One application domain that would be interesting would be to find people who are relevant to a paper. For instance, a recommender could generate a list of people rather than citations. An editor might use this recommender to locate good reviewers for a paper. Editors sometimes search for reviewers among the authors of the cited papers. Since these are often closer colleagues of the author, that list may not be broad enough. Using such a recommender would broaden the pool of potential reviewers.

Another idea comes from one of our users, who said: “Suppose I were working on a sequel (much research is, to some extent, a sequel of earlier work). I am vitally interested to know about LATER papers that either cite my earlier work (or should have or might have or are somehow closely related)”. The recommenders mentioned here could easily be tuned to produce later papers, rather than earlier papers as they did in the offline experiment. One possible way to achieve this tuning is to create hybrid recommender algorithms.

Combining Content and Collaborative Filtering

In Chapter 3, we discussed the Recommendation Process. As a part of that discussion, we presented an instance of the process for generating recommendations for research papers (Figure 3-3). One property of this model is its ability to use multiple algorithms. We are interested now in extending our previous model to hybrid recommender algorithms. In particular, we are interested in how collaborative filtering algorithms (CF) could be combined with content-based filtering algorithms (CBF). Our extended model is shown in Figure 5-8.

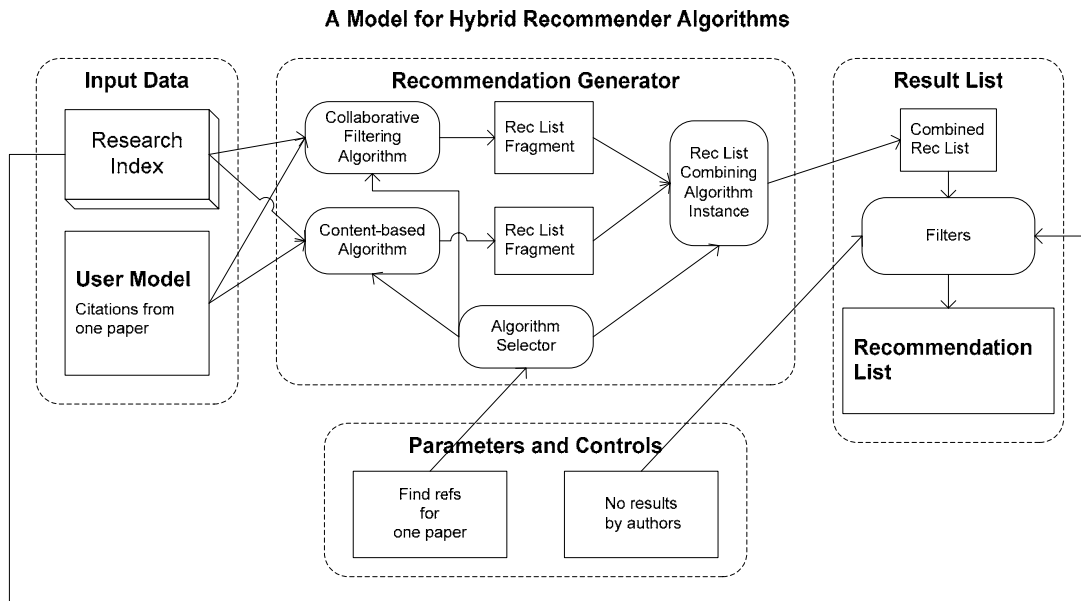


Figure 5-8: Model Instance for Hybrid Recommender Algorithms

The characteristics of collaborative filtering and content-based filtering are complementary. For example, CBF does not suffer from the first-rater problem as long as the content of a new item can be compared against all existing user profiles. In addition, CBF also does not suffer from sparsity, since every item in the system can be related to every user profile. On the other hand, CF does not suffer from content-dependency, since it can be applied to every domain in which humans can evaluate items. In addition, CF uses quality and taste when recommending items. Finally, the serendipitous nature of CF guarantees that there is no over-specialization problem.

We propose a set of hybrid algorithms that combine a TF/IDF algorithm with a User-based Collaborative Filtering algorithm. We hypothesize that CF and CBF can be successfully combined to produce recommendations of research papers. We also hypothesize that different hybrid algorithms might be more suitable for recommending different kinds of papers. Finally, we hypothesize that users with different levels of experience perceive recommendations differently due to their own background, needs, and expectations.

Many hybrid recommender systems have been successfully built in the past. P-Tango recommended news items by combining recommendations from CBF and CF recommenders together using a weighted-average function [25]. Fab recommended Web pages by choosing neighbors for CF-based recommendations using CBF-based user profiles [6]. A ‘boosted’ combination of CF and CBF was proposed in [92], where CBF calculations were used to augment the CF ratings matrix. PTV recommended TV Guides using CF and CBF in parallel [29]. Good et al. took a different view of a hybrid recommender by adding ‘FilterBots’ to a recommender [43]. These bots would rate items based on a set of criteria, many of which were content-based. Finally, Woodruff developed six hybrid recommender algorithms that combined textual and citation information in order to recommend the next paper a user should read from within a single digital book [156]. In our approach, we developed a set of hybrid recommender algorithms and compared their performance to non-hybrid baseline recommenders over a corpus of computer science research papers in both online and offline experiments.

Our algorithms follow the taxonomy of hybrid recommender algorithms first proposed by Burke [16]. In particular, we implemented algorithms that follow the two most straightforward and appealing of Burke’s categories: “feature augmentation” and “mixed” recommenders. In both categories, the hybrid recommender is composed of two standalone non-hybrid recommender algorithms. In ‘feature augmentation’ hybrid recommenders, the results generated from the first algorithm are fed as inputs to the second algorithm. In ‘mixed’ hybrid recommenders, the two algorithms are run independently with the same input and their results are mixed together.

Hybrid Recommender Algorithms in this Domain

Each hybrid algorithm is composed of two independent modules: A CF algorithm and a CBF algorithm. We selected the following two algorithms as the *algorithm modules* for our hybrid algorithms:

1. User-based collaborative filtering with $k = 50$; as with the pure algorithm experiments above, we used the SUGGEST engine [67]. We call this algorithm ‘Pure CF’.
2. A version of Term Frequency/Inverse Document Frequency (TF/IDF); the freely available Bow toolkit [80] was used as our CBF implementation. We call this algorithm ‘Pure CBF’.

Detailed descriptions of both algorithms are available in Chapter 3. User-based CF was run in an identical fashion as with the pure algorithm experiments earlier in this chapter: the ratings matrix is composed of papers voting for the citations on their reference lists, and the citations representing one paper (the ‘active paper’) is sent to the recommender in the ‘active basket’. The CBF module uses the text of the active paper as input; the text is converted into a query and submitted to the Arrow engine that returns a list of the most relevant papers in the dataset. In our implementation, we use the title and abstract of a paper as its text. It is worth noting that the engine also returns a similarity score, but only the rank order of the papers is passed on to the user.

Hybrid algorithms following Burke’s feature augmentation model run these two modules in sequence. The output of the first module (up to 20 papers) is used as input to the second module. For example, if the order was CBF first, CF second, then first the Arrow TF/IDF engine generates a recommendation list of papers for some given input. The top 20 papers are then submitted to the SUGGEST User-based CF engine, and those results are returned to the user.

Because of the linear nature of the augmentation model algorithms, the second part always receives a list of papers from which to generate recommendations. For the CF module, this list becomes the active basket submitted to the recommender. But for the CBF module, we are posed with two ways to deal with this list of papers. Remember, the TF/IDF engine wants a query, a string of words. One way is to look up the text for all papers on the list, combine it together, and submit it to the TF/IDF engine. We call this method *CBF-Combined*. A second way is to look up the text for each paper and submit it

to the TF/IDF engine separately. A voting algorithm using the paper similarity scores would be used to combine the lists together into a final list presented to the user. We call this method *CBF-Separated*. Figure 5-9 highlights the differences between CBF-Combined and CBF-Separated.

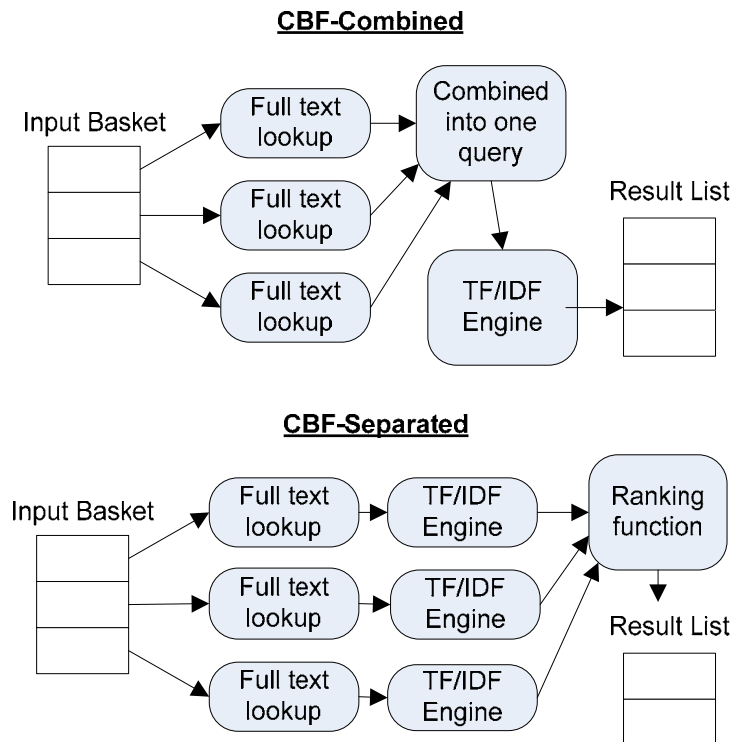


Figure 5-9: The CBF-Combined and CBF-Separated Algorithms

In contrast, hybrid algorithms following Burke’s mixed model runs the two modules in parallel and merges the recommendations together into a final recommended list. This algorithm also uses a voting algorithm, but must base the voting on the relative ranks of the returned items, as the CF module does not produce a similarity score.

CF – CBF Separated

In this algorithm, recommendations from Pure-CF are used as input to CBF-Separated. For every recommendation from CF, the CBF module recommends a set of similar papers (up to 80). Because the recommendations generated by the CF module are ordered, the recommendations generated by the CBF module have to be scaled by this ordering.

Thus, these CBF recommendations are weighted, with the first set (generated from the top CF recommendation) receiving weight 1 and the following sets' weights decreased accordingly according to a linear distribution. For example, if CF returned 20 results, the CBF results returned from the text of the paper from rank 1 would have a weight of 1.00, results from rank 2 would have a weight of 0.95 all the way down to results returned from the paper with rank 20, which would have weight of 0.05. The similarity scores of the CBF recommendations are multiplied by these weights and sorted.

CF – CBF Combined

This algorithm is similar to CF-CBF Separated. However, instead of recommending a set of similar papers for every recommendation received from the CF module, the CBF module aggregates the text of all of the recommendations given by CF and uses this large chunk of text as its input to CBF. The results are sorted by similarity.

CBF Separated – CF

To generate the list of papers CBF Separated needs, we add the list of citations of the active paper to the active basket. Thus, if the active paper has eight citations, the original active list sent to CBF Separated would have nine items: the active paper and all of its citations. The top 20 papers returned from CBF Separated are used as input to Pure-CF to generate the final recommendation list.

CBF Combined – CF

This algorithm is identical to the CBF Separated-CF hybrid algorithm except that CBF Combined is used for the first half of this algorithm.

Fusion

Fusion, our 'mixed' hybrid algorithm, runs the Pure CF and Pure CBF modules in parallel and generates a final recommendation list by merging the results together. The generation of the final recommendation list is as follows: every recommendation that is present in both modules' result lists is added to the final list with a rank score. This score is the summation of the ranks of the recommendation in their original lists. The final recommendation list is sorted based on these scores. Therefore, a paper that was ranked

an online experiment to evaluate users' perceptions about the quality of the recommendations.

Dataset

To test our algorithms we created a dataset with papers extracted from ResearchIndex. This dataset initially had over 500,000 papers and 2 million citations. This was the same initial dataset we started with for our previous experiments. Since these experiments require that we have full text for all items, we were required to perform additional pruning than before. We limited this dataset in two ways. First, we removed papers that cited fewer than three other papers, as we believe these loosely connected papers introduced noise to the dataset. This was the same citation limitation as we imposed before. Second, we removed citations for which we did not have the full text of the paper in our dataset. We performed this trimming so that both CF and CBF would be able to analyze every item in our dataset. The pruned dataset has 102,295 papers with an average of 14 connections per paper, where the number of connections is the number of citations a paper makes plus the number of papers that cite it.

User Profiling

In order to recommend papers to users we need to have a model of the user's interests; we need a user profile. This profile represents the user's tastes and opinions about the papers that she has read. Such a profile could contain both long-term and short-term user interests and the profile could gather data either explicitly or implicitly. See Table 5-3 for a listing of the advantages and disadvantages for creating profiles in these different ways.

An example of gathering information implicitly for long-term interests is to build the profile from all of the papers that the user has read in the past. This can be accomplished by building a system to monitor what papers the user downloads and reads, and silently incorporating all papers into the user's profile. This approach has the benefit of knowing everything a user reads and when the user read it, but it cannot know how closely the user read it. Thus, over time the user's profile could become bloated with

papers that the user may have quickly skimmed and discarded. These false positives can erode the user's profile and reduce recommendation quality.

A way to combat this problem is to have the user explicitly state which papers are to be added to the user profile. Moreover, the user could also provide extra information such as a rating, commentary, or classification of the paper (e.g. by research area, field, etc.). The user could also review her profile to make sure that it accurately represented her. This explicitly gathered information would help the system generate personalized recommendations. It is unlikely, however, that a user would be willing to invest the time and energy needed to maintain such a user profile.

Table 5-3: User Profiling Alternatives

	Gathering Information / Users' Interests	Information Used	Advantages	Disadvantages
All Papers	Implicit and Long-term interests	All papers read in the past	Keep track of user's reading habits over time	Privacy issues, bloated profiles, requires monitoring system
All Papers by Field	Explicit and Long-term interests	All papers read in the past	Filter recommendations based on user's field interests	Requires monitoring system, user must manually adjust profile
One Paper	Explicit and Short-term interests	Only one paper of interest	Does not require a system	Does not keep track of reading habits over time
One Paper	Implicit and Short-term interests	Only one paper of interest	Requires limited system	Privacy issues, does not track habits over time

An example of gathering information implicitly for short-term interests is to build a paper-monitoring system similar to the ones previously described, with one large difference: this system would only remember the last paper a user downloaded and read. This one paper would be used as the current user profile and would be the basis for generating recommendations.

Finally, we could gather information explicitly for short-term interests. In such a system, the user would choose one paper to be his short-term profile; the recommender

would use this paper to generate recommendations. We will use this approach in this research as it is not only the most straightforward and simple to understand, but it is also the only approach that does not require creating and installing a system on users' computers to monitor the papers they are downloading and reading.

Offline Experiment

In this experiment, we randomly removed one citation from the active paper and then checked whether our algorithms could recommend that removed citation. As discussed in Chapter 3, this leave-one-out methodology has been frequently used in other offline experiments and is the same methodology we used for our previous offline experiments in this chapter.

For our offline experiments, there were ten algorithms: Pure CF, Pure CBF, Denser CF, CBF Combined, CBF Separated, CF-CBF Combined, CF-CBF Separated, CBF Combined-CF, CBF Separated-CF, and Fusion. We divided the dataset into training and test datasets at a 90% to 10% ratio. Ten different training and testing datasets were created for 10-fold cross validation. For each trial, every paper in the test dataset had one randomly removed citation.

Although being successfully used in other research, this method of experimentation has some limitations. The recommender algorithms could recommend a paper that did not exist at the time the active paper was published. To handle that, we filtered out recommendations with a publication year later than that of the active paper. The algorithms could also recommend papers that are very similar to or even better than the removed citation, possibly “diminishing” the algorithms' performance. Although this is a possibility, we still expect the removed citation to be recommended.

Offline Experiment Metrics

There are two metrics used in the offline experiment: coverage and rank. These have the same definitions as in the previous offline experiments in this chapter. Rank is the position of the removed item on the recommendation list, and coverage is the percentage of tests for which the removed item was found on the list.

To select the best algorithms, we focused on two particular criteria. As we think that users prefer to see the best recommendations first, our first criterion is the algorithm’s performance in the top-10 coverage. The second criterion is the algorithm’s ability to recommend the removed citation independently of its rank. It is measured by the “all” coverage. As a possible tie-breaker, the algorithm’s top-1 performance is also examined.

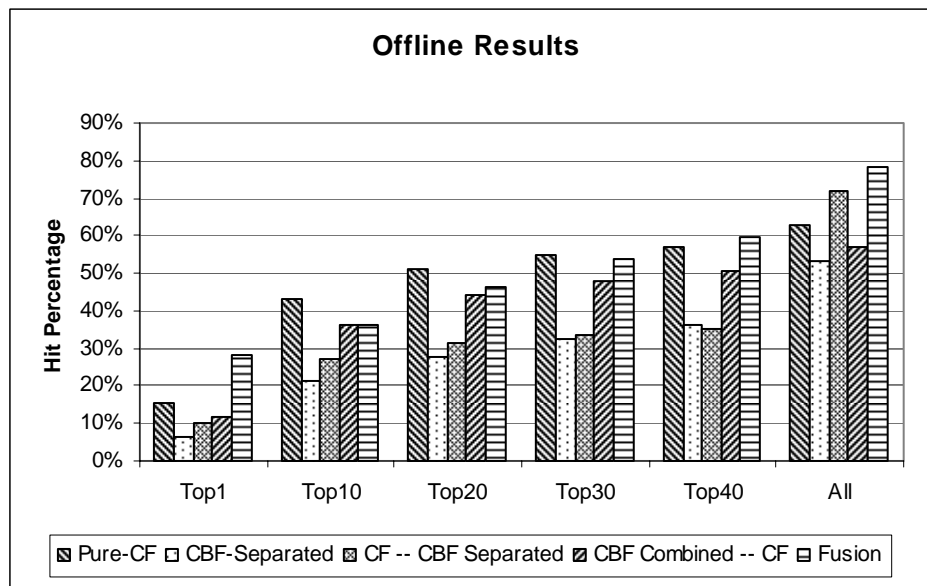


Figure 5-11: Offline Summary for Top Five Algorithms

Offline Experiment Results

Based on the above criteria, the best hybrid algorithms were Fusion, CBF Combined–CF, and CF-CBF Separated. The best non-hybrid algorithms were CBF-Separated and Pure-CF. The results of the five best algorithms are shown in Figure 5-11.

There were both expected results and surprises for the pure algorithms. Both augmented CBF algorithms were expected to perform better than Pure-CBF. Both did, with CBF-Separated performing slightly better than CBF-Combined. We believe this is to the nature of the separated nature of CBF Separated; it widens the search space by using the text of the active paper’s citations individually, retaining the unique nature of each paper when gathering recommendations.

When looking at the collaborative filtering algorithms, Pure-CF was not expected to perform better than Denser-CF. Pure-CF did quite well, usually having the best or second best performance among all bins, whereas Denser-CF had rank percentages of around half the score of Pure-CF at a given top- n level. This unique result provides a glimpse into importance of defining the input basket, a theme we will revisit later.

Moving to the hybrid algorithms, we find that Fusion performed significantly better compared to every other algorithm at both top-1 (28%) and all hit-percentage (78%). The other hybrid algorithms did not perform well. CBF Separated-CF was inferior to the CBF Combined-CF, but the difference is only as a trend. On the other hand, CF-CBF Combined had a *very poor* performance compared to CF-CBF Separated—CF-CBF Combined had a top-1 hit-percentage of 0.1% and never reached above 10% at any top- n level. This second unique result again signals the need for a deeper analysis into how the input basket affects algorithm performance.

Offline Experiment Discussion

The poor performance of Denser-CF was a surprising result because we thought that the denser input used in this algorithm should improve the quality of recommendations, since sparsity is a known CF problem. There are two potential problems with the Denser CF approach. First, the addition of extra items could alter the recommendations. For example, if we assumed each paper had seven citations on average, then Denser-CF would add an average of 49 more citations to the input set; these additions might be not related to the active paper, and only adding noise. Second, some of the items added to the active list might be valuable recommendations themselves. In particular, in a leave-one-out methodology, adding extra items to the active list could negatively impact the metric results, even if the recommendations may have been of high quality.

We believe both of these problems also plagued the CF-CBF Combined hybrid recommender. First, the items returned by the CF module are not seen in the final recommendation list; the CBF module filters out all items appearing on the active list. Any test case in which the Pure-CF algorithm generated a ‘hit’ would automatically be a ‘miss’ for this algorithm. Second, this algorithm grabs the text from all of the papers

recommended by Pure-CF and combines them together into one query. Because of the serendipitous nature of collaborative filtering, this query may contain a wide variety of words; the query may contain noise.

CF-CBF Separated did not suffer from these problems because of the separated nature of the CBF algorithm. The automatic filtering for the CBF module only filtered out the single paper used for each query to the CBF engine. Second, each query sent to the CBF engine came from one paper, thus the queries were each less likely to contain noise. These subtle differences in input baskets and algorithm module behavior can have a large impact on recommendation quality.

To help us understand the behavior of the rest of the algorithms, we also looked at recommendation coverage. Here, coverage is defined as the percentage of items for which the system could generate a recommendation. All of the algorithms had 100% coverage, except Pure-CF, which had coverage of around 93%. This high coverage in the hybrid algorithms is due to the presence of the CBF and is a significant advantage of building a hybrid recommender. We note that this definition of coverage is independent of whether or not the removed item appeared in the recommendation list; it only measures if an algorithm could generate *any* recommendations for a given input basket.

The results from the offline experiment were used to select algorithms for the online experiment. We chose to use the following algorithms online: CBF-Separated and Pure-CF as non-hybrids, and CF-CBF Separated, CBF Combined-CF, and Fusion as the hybrid algorithms.

Online Experiment

The online experiment was aimed to assess users' perceptions about the recommendations they received.

Online Experimental Design

We enhanced our TechLens online experimental system for these online experiments; we call this new system TechLens+. It consists of a six-page Web-based experiment where users evaluated recommendations of research papers from our ResearchIndex-based

dataset. Users were invited to participate anonymously through links at the Penn State version of CiteSeer [110] and EBizSearch [109]. Users were also invited through messages posted in e-mail lists, such as internal lists of the Computer Science departments at: the University of Minnesota, Universidade Federal do Rio Grande do Sul, Georgia Institute of Technology, and UC Berkeley. Additional e-mail invitations were sent to the UC Berkeley Collaborative Filtering Interest List, the User Modeling Interest List, and interest lists of the Brazilian Computer Society. Users ranged from graduate students to professors and professional researchers. Because of the nature of the e-mail lists and websites we chose for recruiting users, we expected to have knowledgeable subjects all of whom would be able to complete the experiment.

The Recommended Papers

Improvements to Collaborative Filtering Algorithms 1 of 5

Anuja Gokhale

[Click here to see the abstract of this paper \(opens a new browser window\)](#)

1. Select the most appropriate response:
"Based on the paper I selected, this paper is a good recommendation".

2. How familiar are you with this recommended paper? (check all that apply)

<input type="checkbox"/> I wrote it	<input type="checkbox"/> I have read it	<input type="checkbox"/> I'm familiar with the author(s)
<input type="checkbox"/> I have cited it	<input type="checkbox"/> I have heard of it	<input type="checkbox"/> I don't know this paper at all

3. How do you describe this recommended paper? (check all that apply)

<input type="checkbox"/> Novel	<input type="checkbox"/> Introductory	<input type="checkbox"/> Survey/Overview
<input type="checkbox"/> Authoritative	<input type="checkbox"/> Specialized	<input type="checkbox"/> I don't know

Would you describe the paper with more labels:
(separated by commas)

Figure 5-12: A Screenshot from TechLens+

There were six pages in the TechLens+ system, an example from one is shown in Figure 5-12. When the user came to the website, she had to consent to participate in the experiment. After that, the user was randomly assigned to one algorithm. The user was asked for the name of an author whose work with which she was familiar. All of the papers with that author's name were retrieved and the user chose a paper for which to receive recommendations. This paper became the user's active paper. The system then generated five recommendations based on the active paper and asked the user questions

about the recommendation she received. The user was able to read the title, author list, and abstract for each recommendation.

The Survey

To gather user opinion about the recommendation she received, the user was asked to answer the following three questions for each recommendation:

1. “Based on the paper I chose, this is a good recommendation”, with the options of “no answer”, “strongly agree”, “agree”, “maybe or unsure”, “disagree”, and “strongly disagree”.
2. “How familiar are you with this recommendation?”, with checkbox options of “I wrote it”, “I have cited it”, “I have read it”, “I have heard of it”, “I’m familiar with author(s)”, and “I don’t know this paper at all”.
3. “How do you describe this recommended paper?”, with checkbox options of “novel”, “authoritative”, “introductory”, “specialized”, “survey/overview”, and “I don’t know”.

Question 1 aims to find out how each algorithm generates high quality recommendations. Question 2 aims to give support for the results found, checking how familiar users were with the recommendations they were evaluating. Question 3 is used to classify papers into different ‘classes’ (e.g. authoritative, specialized, etc.) to find out which algorithms are better for recommending specific classes of papers. The results generated from Question 3 helped us when formulating Human-Recommender Interaction, as we will discuss later.

After reviewing all of the recommendations, we asked a final set of questions to assess the overall quality of the recommendations generated:

4. “For what applications would you be interested in using a research paper recommender system like this one?”, with checkbox options of “weekly/monthly newsletter”, “finding related paper to a chosen paper”,

“finding citations for a current working paper”, “find papers to an unfamiliar area”, “finding reviewers for a paper”, and “finding new papers that build upon previous research”.

5. “Do you think that the overall set of recommendations was good?”, with the options of “strongly agree”, “agree”, “maybe or unsure”, “disagree”, and “strongly disagree”.
6. “Which of the following attributes a recommender system like this one should take into account when generating recommendations?”, with checkbox options of “narrowing the search based on the year of the paper”, “narrowing the search based on authors”, “recommending papers from certain journals or conferences”, and “recommending papers that were cited at least a certain number of times”.
7. “How do you describe yourself?”, with the options of “undergraduate student”, “masters student”, “Ph.D. student”, “researcher”, “professor”, and “professional”. Professor and researcher had a field to enter their years of experience.

Question 4 aims to gather from users what kind of applications they would be interested in using. Question 5 aims to find out if one algorithm can generate a good set of recommendations. This differs from Question 1 because one user can consider a set of recommendations good even if it has only one good recommendation. This provides us with another measure to verify the ability of an algorithm to generate quality recommendations. Question 6 aims to gather which attributes and features might be valuable to take into account when integrating recommenders into digital libraries. Finally, Question 7 provided demographic information about our users.

The questions 3, 4, and 6 had an empty textbox for entering other answers. In addition, at the end of the experiment we provided room for any additional comments.

Online Experiment Results

During the 32-day experimental run, 110 subjects participated in the experiment with 33 from the United States, 43 from Brazil, and 34 from other countries. On average, subjects spent 20 minutes answering our questions. We had 20 Masters students, 33 Ph.D. students, 27 researchers, and 23 professors. Undergraduate and professionals represented six subjects and were not separately analyzed. Table 5-4 shows the number of users per algorithm.

Table 5-4: Number of Users per Algorithm⁸

Algorithm	Number of users
CBF Separated	14
Pure-CF	28
CF-CBF Separated	25
CBF Combined – CF	18
Fusion	25

To evaluate the user’s satisfaction with their recommendations, we categorized their answers. The options strongly agree and agree options are considered as “satisfied” and the strongly disagree and disagree are considered as “dissatisfied” about the recommendations. Non-committal answers (e.g. unsure) were ignored. Table 5-5 shows subjects were satisfied both for each individual recommendation and overall, with satisfaction with two to three times as many people satisfied as dissatisfied. Figure 5-13 goes deeper, showing user satisfaction for each algorithm. It is worth noting that the content-based algorithms were scored higher than the collaborative algorithms, with CBF-Separated, Fusion, and CBF Combined-CF scoring higher than Pure-CF and CF-CBF Separated.

⁸ While seeming improbable, users were randomly assigned to each algorithm! Only three users left the experiment after receiving recommendations.

To evaluate the user’s familiarity with the papers, we broke down our analysis into three groups: a user is considered *very familiar* if he cited or read the paper, *familiar* if he has heard about the paper or is familiar with the authors, and *unfamiliar* if he does not know the recommendations at all. We found a broad spread across these groups. Of all the recommendations, 27% were very familiar to the users, 34% were familiar, and 36% of the papers were unfamiliar. Interestingly, only 2% of the recommendations received were written by the users who were evaluating them.

Recommendation satisfaction also varied by user type. As users became more experienced, satisfaction fell. 75% of the Masters students, 61% of the PhD students, 67% of the researchers, and 52% of the professors said they were satisfied with their recommendations. For this analysis, researchers and professors were ‘professionals’ and Masters and Ph.D. students were ‘students’.

Table 5-5: Overall User Satisfaction

	Individual Recommendations	Overall Set of Recommendations
Satisfied	46%	62%
Dissatisfied	21%	19%

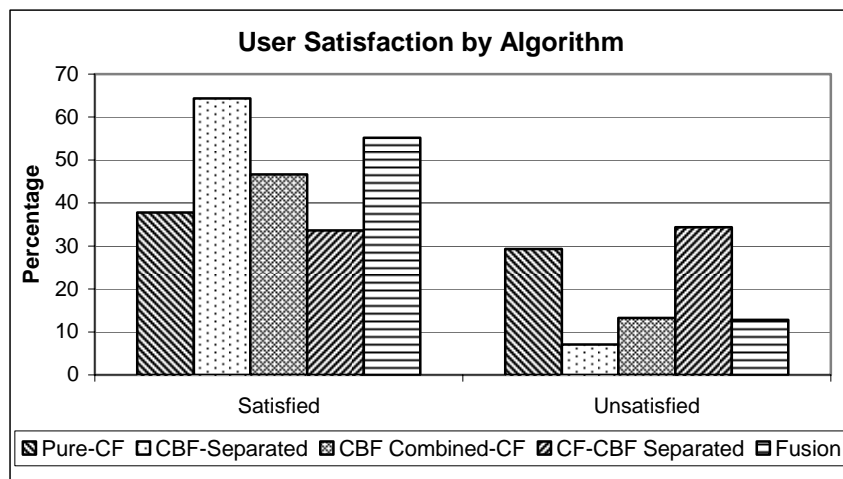


Figure 5-13: Overall User Satisfaction by Algorithm

Paper Class Analysis

Using the results from Question 3, we divided recommendations into different paper classes. In Table 5-6, we review the best and worst algorithms for each class of paper. Pure-CF and Fusion are better than CF-CBF Separated for recommending novel and authoritative papers ($p < 0.05$). Whereas, for introductory papers, CBF-Separated and CF-CBF Separated are better than Pure-CF. Finally, CBF-Separated is better than Pure-CF and Pure-CF was worse than all of the other algorithms to recommend survey papers ($p < 0.1$). It is worth noting the difference between content-based and collaborative filtering algorithms in these results. The collaborative algorithms were good for Novel and Authoritative algorithms, and content-based algorithms were good for Introductory and Survey papers.

Table 5-6: Recommended Algorithm by Paper Class

Class of Papers	Best Algorithms	Worst Algorithms	P Value
Novel	Pure-CF, Fusion	CBF-Sep.	< 0.05
Authoritative	Pure-CF, Fusion	CF-CBF Sep.	< 0.05
Introductory	CBF Sep., CF-CBF Sep.	Pure-CF	< 0.1
Survey/Overview	CBF-Sep.	Pure-CF	< 0.1

Cross-country Analysis

By sending invitations out to users around the world, we have the opportunity to study other possible effects, such as the differences between users in different countries. Approximately 2/3 of the users came from either the United States or Brazil. A user is considered from one of these countries based on the geographic location of her IP address when she accessed the experiment. The breakdown of the subject population is shown in Table 5-7.

We first look at overall user satisfaction. Between countries, user satisfaction with individual recommendations is similar, with 50% satisfaction reported by the

Americans and 49% reported by Brazilians. Dissatisfaction is similar too: Americans at 15% and Brazilians at 17%. On the other hand, satisfaction with the overall set of recommendations varied greatly. Americans were satisfied with 42% and not satisfied with 33% of the recommendations, while the Brazilians were satisfied with 70% and not satisfied with 12% of the recommendations. While satisfaction with individual recommendations remained constant, the perceived value of the over lists was different for Brazilians than for Americans.

There were also strong differences in familiarity. In general, Americans were much more familiar with the recommended papers than the Brazilians. Americans were more familiar with the recommendations, with 31% very familiar, 41% familiar, and 24% unfamiliar. Brazilians, on the other hand, reported being 24% very familiar, 31% familiar, and 44% unfamiliar with the recommendations. It is worth pointing out the large difference in unfamiliar recommendations, with Brazilians being unfamiliar with twice as many papers as the Americans.

Table 5-7: Distribution of Users in Online Experiment

Brazil Eng.	Brazil Port.	Type of User	Total Brazil	Total USA
3	12	Masters Students	15	4
3	7	PhD Students	10	13
0	6	Researchers	6	8
5	5	Professors	10	5

(a)

(b)

Cross-Language Analysis

To further test potential differences between users in different countries, we created a Portuguese version of TechLens+. In this version, all instructions and questions were in Portuguese but all paper recommendations, such as titles and abstracts, remained in English. The Portuguese version of TechLens+ started 6 days after the English version. During this time, 12 Brazilian users participated in the English Version of the

experiment. After the launch of the Portuguese version, Brazilian users preferred to participate in this version. We then divided the Brazilians into two groups: those that participated in the English and those that participated in the Portuguese version. This population distribution is shown in Table 5-7.

Language had a large impact on all results, including both overall recommendation quality and familiarity. When looking at overall recommendation quality, Brazilians were satisfied with 42% and dissatisfied with 33% of the recommendations in the English version, while in Portuguese, they were satisfied with 81% and dissatisfied with only 3%. The satisfaction rates doubled when Brazilian users used a system in their native language.

These differences also carried over into familiarity. In English, Brazilians were 11% very familiar, 32% familiar, and 57% unfamiliar with the recommendations they received. While in Portuguese, they were 29% very familiar, 31% familiar, and 40% unfamiliar with the recommendations. While not as dramatic as the recommendation quality results, Brazilians were more familiar with recommendations when the system was in their own language. Remember, all recommendations were still in English, only the instructions and survey questions were translated.

Online Experiment Discussion

We can break down the users in our experiment into two categories: Masters and Ph.D. students (as students), and researchers and professors (as professionals). Although it is worth noting that professors were more experienced on average than researchers were. This population breakdown provided highly valuable information. Our analysis showed that the less experienced the user was, the more the user liked the recommendations. In addition, professors were more familiar with the papers (as expected), which might make them “less happy”.

Our cross-country analysis showed that there are no strong cultural differences in receiving research paper recommendations. In addition, our analysis reinforced the results that level of experience influences user satisfaction. Brazilian users had a higher percentage of Masters’ students than American users had. Consequently, Brazilian users

were less familiar with the papers than American users. Therefore, the recommendations given to Brazilian users made them more satisfied. These results suggest that research paper recommender systems should be tailored to the experience level of each user.

Our analysis showed strong language differences. Brazilians in the Portuguese experiment were more satisfied with the recommendations they received. We hypothesize that because most of what is in the Internet is written in English, Brazilian users might be more satisfied being invited to participate in a Portuguese experiment. The idea of a recommender in Portuguese may have biased their results. This suggests that research paper recommender systems interface should be localized to the user's native language, reducing the users' burden of finding good research papers. This is independent of the language of the papers, because most of them were written in English, and Brazilians were happy either way.

Even though the recommendations were the same in both cases, changing the language of the system had a strong effect on the users' perception of the recommender. This has profound implications. Looking back at information seeking theory, Kuhlthau's Information Search Process model suggests that the emotional state of the user is as important if not more important than the information the user finds [70]. Few studies have ever shown that this effect carries over to the Internet. Our result provides evidence suggesting this is so. We will return to this point when we formally introduce Human-Recommender Interaction theory.

Finally, in order for a recommender system to add value to a digital library, it has to generate high quality recommendations consistently. Not every single recommendation has to be good, however. Users want a recommendation set that is of high quality. As we found both sets of experiments from this chapter, users are happy even if they receive only one or two good recommendations out of five.

For example, one user commented: "I was looking for papers that would help me writing a compiler without writing code generators for many different processors". This user considered only one recommendation as relevant. Although the user was looking for a very specific topic, the system was voted 'very useful' and the user considered the

whole set of recommendations as ‘good’. Overall, we found that, 85% of the users said they received at least one good recommendation. This encourages us that our recommender algorithms can be used in digital libraries. This point cannot be stressed enough; only the recommendation list *as a whole* needs to provide value, not every item on that list. By providing one good recommendation in a list of five, a recommender can be very useful in this domain.

Hybrid Algorithm Discussion

Returning to our hypotheses, we found that many of our CF-CBF hybrid recommender algorithms could generate research paper recommendations that users were very happy to receive. In addition, because 85% of our users received at least one good recommendation, we believe that our algorithms can aid digital libraries.

Some of the feature augmentation algorithms we tested, however, did not perform well. We believe this is due to the sequential nature of these hybrid algorithms: the second module is only able to make recommendations seeded by the results of the first module. There are many problems with this approach, including blocking good recommendations, and diluting input baskets. In general, we believe sequential hybrid recommendation algorithms will not perform well because pure recommender algorithms are not designed to receive input from another recommender algorithm.

Our algorithms were tested using a dataset of computer science research papers. However, the algorithms can be used in any domain, as long as the text and citations of the papers are available in digital format. Thus, we believe that most existing and emergent digital libraries, such as [1], [26], or [148], can successfully incorporate our hybrid algorithms. Of particular note is our Fusion algorithm, where any enhancement to each component technique (CF or CBF) can be promptly incorporated into the overall algorithm.

Our online results showed that different algorithms should be used for recommending different kinds of papers, reinforcing results from earlier in this chapter. In addition, our results showed that users with different levels of experience perceive

recommendations differently. For example, professionals were not as “happy” as students were.

We have a vision for the future of a completely personalized or ‘tailored’ digital library. Such a digital library might tailor recommender algorithms for particular user tasks using Table 5-6 as a guide. For example, suppose that the task of “finding related work” could be solved by recommending novel and authoritative papers. Then a system that wanted to support this task should use Pure-CF and Fusion to generate paper recommendations. Second, the digital library might tailor itself to the user’s native language, independent of the language of the papers. Finally, the digital library might tailor the recommendations it displays based on the level of experience a user has. In the following chapters in the dissertation, we will explore the first possibility mentioned here: the ability of a recommender to alter recommendations based on the user’s current information need.

Our results were based on a user profile with explicit input of preferences from the users and for short-term interests. We believe that other user profiles should be tested in order to track evolving reading habits over time. Further studies with users of multiple nationalities would also be desirable and to determine why our feature augmentation algorithms did not perform well online.

Our algorithms were based on Burke’s taxonomy of hybrid recommender algorithms [16]. In this work, we only implemented algorithms in two of his seven categories. It would be interesting to implement algorithms in all of his categories to compare them against each other. Moreover, Burke only classified hybrid algorithms that could be built from standalone recommender algorithms. Further studies comparing Burke’s taxonomy with more tightly integrated hybrid algorithms would be worthwhile to perform.

It would also be interesting to investigate algorithm differences in recommending recent compared to older research papers. We believe this leads to the possibility of recommending “research paths” to users. Given a query of a research area and knowledge of what a user has already read, a recommender could generate a display of

how this area has evolved over time and produce an ordered list of “must-read” papers in that field. We believe this is an important area to look into, not only for educational purposes, but because over 69% of our subjects said they would like recommender systems to help them find papers that built off of known research.

Conclusion (A Funny Thing Happened...)

In this chapter, we proposed a mapping for the ratings matrix to use recommender algorithms in the domain of research papers. This mapping mined the network of citations between papers, and as such, provided the idea for abstracting the rating matrix beyond users and items. By doing so, recommenders in this domain do not suffer from the cold start problem nor the new user problem. Then again, the recommenders are not personalized to a person, per se, but only personalized to a list of papers.

Our experiments in this chapter showed that in this domain, recommenders generate high quality recommendation lists; recommenders can be successful in non-taste-centric item domains. In order for users to be happy, a recommender only needs to generate one useful recommendation in a list of five. This provides further evidence to our notion that users care about recommendation lists as a whole, not just items on the list. Moreover, we expanded our research to show that not only do pure recommender algorithms work in this domain, but hybrid algorithms also generate high quality recommendations.

A funny thing happened when performing this research. Our users started telling us things we did not expect. They told us that our recommender algorithms generated qualitatively different lists from each other. They said our algorithms were different. Not only that, users assigned meaning to these differences. Users told us that collaborative filtering algorithms were better for novel and authoritative papers whereas content-based algorithms were better for survey and introductory papers.

These results are new and different from any other results in the recommender systems literature; no one has ever run experiments on non-accuracy-based user-perceived differences between recommender algorithms before. Historically, algorithms

were considered as functional replacements for each other. The current set of predictive accuracy and decision support metrics perpetuated this belief by showing small, yet meaningful differences between algorithms' performances. Because of these 'facts', designers and developers selected algorithms for systems depending on factors important to them, such as time to build and test the algorithm, size of dataset, and server capacity, and not on factors important to users.

We chose to focus this dissertation on this *funny thing*. After all, if recommender algorithms are different from each other, and if users can tell, then shouldn't we, as recommender systems designers and researcher, be selecting the appropriate algorithm for our recommender systems? Taking this one step further, shouldn't we have a suite of algorithms at our disposal and dynamically select the most appropriate algorithm for a user at a particular point in time?

Until now, all published reports on recommenders said that these systems were 'one algorithm fits all'. Yes, the system was personalized, but personalized in the same way for all users. We argue that recommender systems need to tailor their personalization scheme depending on the user and on the user's current information need. In order to make this vision a reality, we need to look at all of this from the end user's perspective. We need to rethink recommender systems.

Human-Recommender Interaction is a new theory we have developed which does just that.

CHAPTER 6

HUMAN-RECOMMENDER INTERACTION THEORY

In 2002, Jeffery Zaslow wrote an article in the Wall Street Journal about recommender systems entitled, “If TiVo Thinks You Are Gay, Here's How to Set It Straight”. He talked of TiVo, Amazon.com, and NetFlix, among others generating recommendations that made no sense to the users of these systems. The article stated that users felt the recommenders did not understand them, and, more importantly, the recommenders had *their own opinions*. It ends with a telling quotation, “Mr. Leon believes the box was giving them a message: ‘You're definitely gay. And you're watching too much TV.’” [160].

This article generated a large amount of press for personalization and recommenders. The phrase, “TiVo thinks I’m gay”, spread throughout pop culture, even ending up as a plotline for an episode of the CBS comedy, “The King of Queens”.⁹ Unfortunately, recommenders were not presented in the most positive light. This leads us, as researchers, to reflect on recommenders and the recommendation process.

We start with the question, “Why do users come to recommenders?” We postulate that users have specific information needs and come to a recommender as part of an information seeking process. Assuming this, the question then becomes, “How can we, as recommender system designers and researchers, create recommenders that can generate *useful* recommendation lists for users with differing information needs?”

In this dissertation, we have made a few key observations: While most well known recommender algorithms score similarly on accuracy metrics, these algorithms generate qualitatively different recommendation *lists*. Users, unlike most researchers, were not judging the quality of individual recommendations, but were instead experiencing the list as a whole, considering properties such as how well the breadth or

⁹ It was a subplot of the episode "Mammary Lane", which originally aired on October 14, 2002.

depth of the list met their goals. Most important, end users noticed these qualitative differences, and in some cases preferred lists of recommendations that were less accurate (by traditional measures) but more suitable for their needs.

These observations suggest we pause and reflect on the recommendation process. Users look for more than accurate recommendations. Users judge recommendations in the context of the list in which they are presented. Users of recommenders, such as TiVo, generate opinions of not just of recommendations they receive, but also of the recommender itself. What does this mean? How can these reflections help us generate useful recommendations?

Historically, research has focused on making recommendations more accurate, with the implicit assumption that ‘more accurate’ equates to ‘better liked and more useful’. Information retrieval refers to this as the concept of relevance, where recommender systems researchers have equated ‘more accurate’ with ‘more relevant’. Lately, researchers have been questioning this assumption [51]. As a community, perhaps, we have gotten it backwards. First, we need to determine what kinds of recommendations are good and useful to users. We should put the users first. Next, we should devise the appropriate metrics and test algorithm performance. In short, we need to re-think recommenders from a user-centric perspective.

The Gulf of Intention

Recommender systems research has historically focused on giving good, accurate recommendations to users—generating the best recommendations possible. Instead, let us look at the situation from the user’s point of view. What would it be like to use a recommender? You are looking for some piece of information, and the recommender is making suggestions to you. If the recommender starts making wildly off-base recommendations, recommendations that are not helpful to you at all, how can you tell the recommender it is wrong? The recommender may know you as a person, but that does not mean it knows what you are looking for. It is not good enough to generate recommendations for a user; users have many reasons for coming to a recommender.

Even worse, two users may appear to be the same to a recommender. They could have very similar profiles to a recommender—perhaps even identical profiles!—but that does not mean they want identical recommendation lists.

How is this possible? From a recommender’s point of view, the user profile reflects the user’s current state of knowledge. Given the profile, the recommender generates the best recommendations it can. If two users have identical profiles, then *by definition* they will receive identical recommendation lists. Profiles, unfortunately, are not users. Just as the gulf of execution separates user interfaces from users, a gulf of intention between users and their models keeps recommenders from achieving their full potential. This gulf appears because of the inability of users to formulate their information need to a computer, and the inability of computers to formulate questions to understand that information need.

As best described by the last of Taylor’s four stages of information need, users will *compromise* their information need when communicating that need to another individual [142]. In other words, users translate their needs when describing them to an assistant. When this assistant is another human, the richness of human language as well as shared social context will achieve a high quality translation. Moreover, librarians are trained to use ‘reference interviews’ to better understand user information needs [153].

When the assistant is a computer system, what can we say about the quality of the translation? In information filtering systems, user models contain keywords that describe the user’s information need. Recommender systems add preference information for items in a domain to the user model. From this additional information, we can understand a fair amount about a user. Yet we, as recommender system researchers and designers, still do not know why a user has come to a recommender, and we do not know how the user views the recommender. These elements are critical to reduce the gulf of intention.

Human Recommender Interaction: A User-Centric Perspective

Human-Recommender Interaction theory (HRI) was developed to provide an insight into how users perceive recommenders. As such, it is a *descriptive* theory, much like the

mental models from cognitive psychology, design rationale theories from HCI [19], or Clark's Theory of Language Use [24]. These theories describe a view of the world; they force us to perceive the world in a new and different way. From these theories, people derive predictive and constructive models to test and refine these theories.

HRI is a view of the recommendation process from a different angle. For example, the captain and a passenger have different views of an airline flight. Both views are correct, but it may be difficult to explain the passenger's view to the captain or vice-versa. One of the problems is language; the language pilots use to describe airline flights may be foreign to passengers. This same split appears between the designers of a recommender and the users of a recommender.

Based on what we learned in Chapter 5, users of recommender systems were able to use terms to describe the recommendation lists they received. HRI is built on this foundation. It is a common language for all parties in a recommender to use when communicating about the kinds of recommendation lists that recommender algorithms generate. The words in this language are called 'Aspects'. This language itself is organized into three pillars, based on the following premises:

1. Users have an information need and come to a recommender as a part of their information seeking behavior to meet this need.
2. Users perceive the quality of a recommendation in its immediate context; users evaluate lists of recommendations.
3. Users develop opinions of a recommender itself based on the interactions they have with the recommender over time.

These ideas form the core of HRI; as such, each one is the seed of an *HRI Pillar*. There are three pillars in HRI: the *Recommendation Dialog*, the *Recommender Personality*, and the *User Information Seeking Task*. Each pillar has a specific meaning in the recommendation process. Further, each pillar also contains a number of *aspects* that provide a deeper insight into how the pillar can be used to improve

recommendations. We will first discuss the meaning of the three pillars, including theoretical grounding. Afterwards, we will delve into the aspects of each pillar.

The Pillars of HRI

The Recommendation Dialog

When users seek information, they usually have a large amount of uncertainty about what they want and where they will find it [154]. Users will spend time performing multiple searches, interacting with the system, refining their queries, before finding what they want. It is through this interaction that users not only solve their need, but gain confidence in the validity in their results [138]. We believe this carries over to recommenders; there is a ‘relationship’ between the user and the recommender based on the assumption that as the user provides more accurate information the recommender will generate more meaningful recommendations. At the point of interaction, then, we need to be aware of the user’s perceptions of recommender and how it is helping him with his task. To generate good recommendations, we need to know how the dialog is going.

The first pillar, the *Recommendation Dialog*, is a ‘give-and-take’ between the user and the recommender where the recommender provides recommendations and the user provides information, such as ratings. This is focused solely on the immediate interactions between the user and the recommender for one recommendation or prediction session. This term is purposefully defined in vague terms in order to include the different kinds of reactions that a user can have to the recommendations including whether or not we care about individual recommendations or a recommendation list, and whether or not a user has consumed the item.

Theoretical grounding and support for this pillar comes from several areas. The user interface for recommenders has been extensively studied; the most notable research performed in the case-based reasoning literature, where specific interactions with a recommender have been termed “recommendation cycles” [82]. The information gathered in each cycle improves recommender performance. As such, each cycle needs to be a high quality interaction. Moving to the information seeking literature, we find

several models and theories which highlight the importance of the point-of-interaction (i.e. the reference interview [153], and the cyclic, iterative nature of information seeking models as described in Chapter 2 and in [20]).

The Recommender Personality

The Dialog refers to a time independent view of the recommendation process. We now add one more concept: memory. Users have many interactions with the same system over time. As shown by Reeves and Nash, users will often personify aspects of technology [118]; users embed personality characteristics to technology and media as a part of the interactions with such systems and media. We believe a similar phenomenon happens as users interact with recommenders. In order to integrate a recommender into a user's worldview, people assign human-like characteristics and traits to recommenders as a part of their continued interactions with such systems—the recommender stops being a tool for the user and becomes a “teammate”. We are not saying people think recommenders are alive, but rather people think recommenders have personalities (e.g. “My TiVo thinks I’m gay”). Because the goal of recommenders is to tailor and customize information for users, we believe users will develop these opinions earlier and develop stronger opinions on recommenders than on other forms of technology—recommenders have some ‘insight’ into users and people interpret these actions as social queues.

The *Recommender Personality*, the second pillar, is the overall impression a user constructs of the recommender over a period of time. This requires repeated, independent interactions (a.k.a. Recommendation Dialogs) between the user and the system so that the user can create a ‘mental model’ of the recommender and assign it personality characteristics or traits (e.g. “Another love story? The recommender really is a hopeless romantic.”). Whereas the Recommendation Dialog was seen as the immediate interaction of the recommender with the user, the Recommender Personality is the long-term conversation, complete with history, context, and expectations to this interaction.

Beyond the discussions provided by Reeves and Nash, several information seeking theories also discuss the importance of the long term interactions with an

information system, such as a recommender. In Sense-Making theory, the mental and emotional state of the user is as important to determining when the user has completed the information seeking task as the quality of the information itself. If a user is frustrated with information system, it will affect the quality of the information, the user's opinion of the quality of the information gathered, and the user's opinion of the search process itself (including the tools the user used) [20, 70]. Media Use as Social Action (MASA) theory discusses how users interact with media, making the media an active participant in the consumption process (e.g. Many sports fans will yell at their television sets when watching a game; the fans have made the TV itself an social actor) [20]. By viewing the recommender as having personality characteristics, the users have made the recommender a social actor in the user's information seeking process.

The User Information Seeking Task

One of the largest assumptions we make is that users come to a recommender as part of an information seeking task; that recommenders are a part of the users' information seeking behavior. Because of this, there are several implications in how users interact with recommenders. For example, this pillar includes preconceived thoughts or opinions users will have about using the recommender in the context of their information need—what does the user believe about the recommender? Chapter 2 discusses these implications in detail.

One of the largest concerns with this pillar is the difference between users looking for information and users looking for products. Historically, recommenders recommend items to users, not bits of knowledge. How are the two related? We view items as knowledge containers, each item a representation of the information it contains. Users coming to a recommender are seeking the information contained within an item or knowledge about an item. The specifics of the information seeking task may differ, however. A user could know which kinds of items contain the information she is looking for, this a concrete and specific seeking task. On the other hand, a user could only have a vague understanding of the information need and be looking for any item that could help her.

In general, however, we believe it returns to the argument we made in Chapter 1, that users' external criteria has a strong influence on the importance of the information seeking task. In entertainment-based domains, such as music, there are few external influences, thus taste becomes a dominate factor in selecting items for consumption. But, in other domains, such as research papers, there may be several additional criteria an item must meet beyond taste before an item can be consumed. These additional criteria, we argue, form the basis of a user's information seeking task. Thus, depending on the number and importance of a user's external criteria, this pillar will vary in importance.

The Aspects of HRI

Figure 6-1 shows the aspects for each pillar in Human-Recommender Interaction. If we viewed each aspect as a 'word', then the set of aspects would be a language we could use to describe the important parts of each pillar. For example, it is not enough to say that recommenders have personalities. We need to go a step further and provide a set of terms for a user to use when describing a recommender's personality. When examining how an end user would interact with a recommender system, it is worth reviewing which aspects are important to which users and their associated tasks. Not all aspects will be important to all users.

Unlike the Pillars, not all HRI Aspects have a strong theoretical grounding. Several are the product of our long experience with recommenders, and, more importantly, user reaction to recommenders. Because of this, the Aspects presented in this dissertation are archetypes—ones we believe transcend application domains. For each Aspect, we will discuss the meaning of the Aspect and the reasons leading to the creation of this Aspect. For many Aspects, there are two views of the Aspect to consider: a system-centric view and the user's perception. Many of the Aspects imply a measure or metric a system designer could use to gauge a specific recommender with respect to that Aspect. This objective view of the Aspect is important, but the user's subjective view of the Aspect is just as important. For example, it is possible to measure the Boldness of a

recommender, as discussed below, but just because a metric says a recommender is bold does not mean that users will view the recommender as being bold.

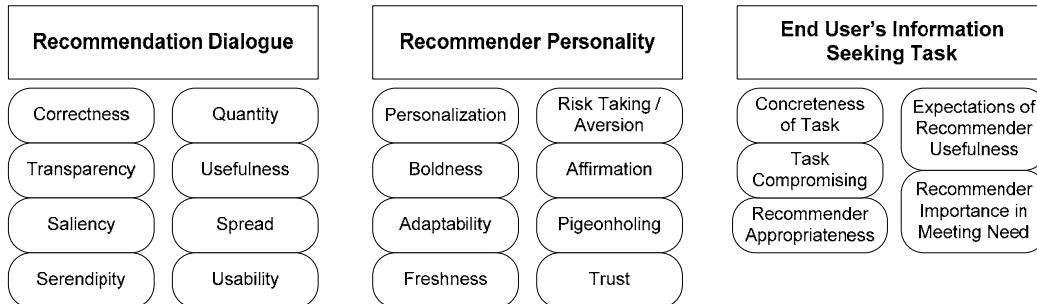


Figure 6-1: The Pillars and Aspects of HRI

Aspects of the Recommendation Dialog

There are eight aspects to the Recommendation Dialog: Correctness, Transparency, Saliency, Serendipity, Quantity, Usefulness, Spread, and Usability.

Correctness

Correctness is judgment of the user that a recommendation provided by the recommender is a good and high quality recommendation for his information need. Relating this to information retrieval, a correct recommendation is a relevant one. Implicit in correctness is the knowledge that the user knows that he is receiving a recommendation and that the user understands the recommendation. If the user feels he is being deceived in any way, then the recommendation is not considered correct.

Discussion: Correctness is the HRI Aspect representing the results returned from predictive accuracy and decision support metrics. This is one of the main aspects representing the current view of recommender algorithms. As we stated before, correctness is important to recommenders, but there are a wide variety of other aspects to consider.

Transparency

A transparent recommendation is one that the user understands why it was recommended for her particular information task. At a more general level, recommendation transparency relates to how well the user understands (at a high level) the algorithms used to generate or select specific recommendations. Moreover, this aspect is directly affected by system design (user interface design, algorithm selection, information gathered etc.). As such, while this is an important aspect of the dialog, much of it is controlled before the first recommendation is generated.

Discussion: Transparency as a term used in recommender systems was first proposed by Sinha et al. [137]. In their paper, they argue for the importance of transparency for building trust in recommenders. While we also believe that transparency is good for recommenders, it may not equally important for all users. For example, it could be very important for new users, but not for users who have already developed trust through experience. There have been several attempts to bring transparency into the recommendation process. Herlocker et al. provided users with a explanation interface for collaborative filtering recommendations [54] and we provided users with a confidence display for recommendation [88].

Saliency

Saliency means to “stand out”. It is vivid, unexpected, notable, conspicuous, and prominent. For some reason a salient item catches your eye. Something salient is not necessarily relevant; it just stands out. A salient recommendation, thus, stands out—it generates an emotional response from the target user, either positive or negative.

Discussion: We argue that saliency is one of the most important aspects that differentiate recommender systems from other machine learning systems. In machine learning, the goal is always correctness. We argue that a goal of a recommender is also to generate salient recommendations. A user may not notice a recommendation with low saliency. If it does not cause a reaction, it may be overlooked. Direct questions about saliency

change the user's experience with a recommender by bring the concept to the conscious level. Saliency is difficult to capture in a metric and suffers from domain-specific problems: different domains may elicit more or less of an emotional response from users (e.g. movies compared to research papers).

Serendipity

Serendipity is defined as “The faculty of making fortunate discoveries by accident; the fact or occurrence of such discoveries; an instance of making such a discovery” [111]. A serendipitous recommendation is an unexpected recommendation. Serendipity, then, is dependent on current recommendation list. Because of this, serendipity and saliency have an interesting relationship.

Discussion: Most likely, a serendipitous recommendation is also a salient one—most users will react to something they didn't expect to see! But a list with several salient recommendations could 'interfere' with a serendipitous recommendation. Finally, there could be serendipitous recommendations that are not salient. A user may not have an immediate reaction to the recommendation, but consume the item anyway because of her trust in the system and its recommendations. This tends to happen for “riskier” items: users who trust a recommender will consume riskier items in the hopes that the item will be a serendipitous recommendation [88].

Quantity

For any recommendation dialog, the system has a choice as to how many recommendations to provide to the user. Based on the user's current information need and other factors (such as coverage, etc) the system will have to decide how many recommendations to display, and the user will make a judgment whether or not she is receiving enough choices or if she is receiving an overload of possibilities.

Discussion: Determining the correct number of items to display on a recommendation list depends on several factors. The first factor is the number of items returned from the

recommender algorithm itself. If the recommender only returns ten items, it is impossible to have a top-20 list. Second, the user interface can restrict the number of items displayed. If a recommender is incorporated into a larger application, such as a digital library or an online retailer, there may only be a small portion of the screen devoted to displaying recommendation lists. Third comes from the user's preferences. Does the user want a variety of options or does the user want just a single recommendation? In the MovieLens movie recommender, for example, the first two factors are not concerns, as MovieLens can usually generate recommendations and the recommendations are the core component of the interface. But the third factor is quite important; because of many requests by users, MovieLens implemented a feature where users could change the number items to display on screen at once, between 5 and 50, to best suit their needs.

Usefulness

Is a given recommendation going to be consumed by a user and/or help the user with his information seeking task? If so, then we would consider the item *useful* to the user. This is independent of whether or not the user eventually rates the item highly; it is a measure of whether or not the items affected the user's perception of his task.

Discussion: If a given recommendation causes the user to change his seeking task or view his task in a different way, then item is considered useful. There is an interesting relationship between correctness and usefulness. A useful item could be rated lowly (e.g. "Now I know exactly what I am not looking for, and I can refine my search"), or not rated at all (e.g. "My searches are leading me towards these items, I'm getting closer to what I want.") Because of this property, usefulness is a more important measure in a conversational recommender, as useful items help a user refine his search. Usefulness is also very dependent on context. In their MetaLens work, Schafer et al. showed that external criteria had a large impact the usefulness of the recommended items [132]. For example, if you were taking your 9-year-old niece to a movie, recommendations would be more useful if you could filter based on the movies' MPAA rating. Thus, items that

are ‘correct’ recommendations may not be useful in a specific situation. Finally, usefulness and saliency are related, as salient items, items more likely to be noticed by the user, are more likely to be useful.

Spread

Recommender spread is the measure of how broadly the user perceives a recommender system using the item space when generating a recommendation list. A greater spread means that the recommender uses a greater percentage of all items to generate this particular list. For example, a user might wonder if his interest in science fiction affected his current recommendation list of children’s books.

Discussion: Spread can be deceptive. For example, in our work with new users, we found that Item-based CF could trap users in a ‘similarity well’ where all of the recommendations were for very similar items [117]. Thus, in this case, a recommendation list from a ‘similarity well’ would have low spread. Yet the user’s opinion may vary widely from the actual system implementation. Because of this, controlling for this aspect is as much an interface issue as it is a recommender algorithm issue.

Usability

When the user understands how to use the recommender to accomplish their information task and that the system responds as the user expects it should, we say that the recommender system is usable. Usability includes all aspects of the user interface, the interactivity of system, and the consistency of how the system behaves, especially when there might be cultural, language, or other issues to consider.

Discussion: For the purposes of this dissertation, we hold this aspect constant, but we still acknowledge the usability of the system as it affects the recommendation process. This is not to say that usability is not important, on the contrary, it is extremely important when designing an effective recommender. If a user cannot understand the interface,

how can the user make use of the recommendations she receives? Changing the interface for new users in a recommender had a large impact on the usage and perception of the recommender [89]. This is not limited to new users, however. Poor usability will negatively influence the user's perception of a recommender. In this dissertation, however, we are interested in the differences between recommender algorithms given the same interface, and as such, we hold this aspect constant.

Aspects of the Recommender Personality

The Pillar of Recommender Personality also has eight aspects. They are Personalization, Boldness, Adaptability, Freshness, Risk taking/aversion, Affirmation, Pigeonholing, and Trust.

Personalization

Personalization is a measure of how unique recommendations lists are for different users. Thus, a personalized system will generate custom-tailored recommendations for each user. There are two parts to this aspect: the measure of the true personalization of an algorithm and an end user's perception of how personalized her recommendation lists are. Much as with spread, knowing both bits of information along with a user's preference on personalization can help a system best match a user's expectations.

Discussion: One way to measure personalization is uniqueness analysis.

Recommendation lists can be compared to each other to determine how many items overlap across lists, or lists can be compared to some external authority, such as popularity. Depending on the algorithm, personalization could correlate with differences in user models. Thinking deeper, similarity between user models should correlate with similarity of recommendation lists. The nature of this correlation could be a strong measure of the personalization of a recommender. The more naïve version is to compare lists, independent of user models.

We expect that user opinion of personalization will be polarized: either a user will want personalization or he will not. We do not expect a user to say, "I'd like it to be a

little less personalized’. Instead, we believe a user may want to ‘turn off’ personalization in order to see what a community opinion is for some seeking task. For example, in the Krakatoa Chronicle personalized newspaper there had to be two different views into the newspaper, a completely personalized version and a non-personalized version, and users would go back and forth between them [66].

Boldness

Boldness is a measure of how strongly the recommender recommends particular items to users. A user would say a recommender is bold if many of the recommendations are at the extremes of the prediction scale (i.e. The system definitively states “you will love this” or “you will hate this” instead of stating “you might like it”). As with saliency and serendipity, this measure is relative—it depends on the other recommendations lists the user has received before.

Discussion: A user may feel that the system is bold even if it does not generate many extreme recommendations as long as those few extreme ones were salient. This measure only has relevance in a system with an explicit prediction scale or confidence scale. Otherwise, users cannot compare the relative strengths of two recommendations.

This measure has roots in the MovieLens movie recommender. In 2003, GroupLens Research released a new version of MovieLens, complete with an updated UI, experimental platform, and recommender engine. After release, we received many emails from users complaining about the recommendations from the new system. They stated that the new MovieLens was “conservative”, “dull”, “boring”, and that the old MovieLens was much “bolder” in its recommendations. Our users demanded more boldness from MovieLens.

Adaptability

Adaptability is the change in recommendations that occurs in response to changes in a user’s profile. Thus, a highly adaptable system will change recommendations as the user provides or changes information in her profile. While dependent on the size of the user’s

current model and the severity of the change, a more adaptable system would be more responsive across all users and situations than a non-adaptable system. It is worth noting that a system could still be personalized and not very adaptable (i.e. once a user has created a profile, it takes a significant change in the model to have a noticeable change in recommendations).

Discussion: This aspect reflects the system's ability to 'grow' with the user. In many recommenders, a user provides a large amount of information when first joining the system, and only incremental information afterwards. Adaptability is the measure of how the system changes as the user provides incremental changes to their model. Users may express this as a measure of how well the recommender "knows them" (e.g. If a user said, "Now that I am married, the recommender doesn't seem to get me, so I had to retrain it a lot", then the system may not be adaptable). In MovieLens, for example, many users have told us they felt that the first 50 or so ratings greatly influenced the recommendations they received, but there was a 'diminishing return' on additional ratings. These extra ratings seem to have very little change on the recommendations the users receive. We would say these users feel that MovieLens is not adapting to the growth in their user models.

Freshness

Freshness refers to the system's ability to provide new ("fresh") recommendations that the user has not seen before for this or other similar information tasks. While related to the influx of new items and changes to the system, the system could keep recommendations 'fresh' by showing items that might not be the best recommendations for a particular task, but that have not been seen before by the user.

Discussion: While freshness is a part of any recommender's personality, it is as much part of the system design as it is a part of the recommender algorithm. Systems focused on traditional filtering tasks will highly value freshness, filtering new streams of content for users. Traditional retrieval systems may not value freshness, choosing to return the best

items it can from the entire dataset. Recommenders can cross both kinds of information systems, and thus freshness may vary from system to system. Moreover, freshness can be used to keep users returning to a system even if the item space has changed little, but perturbing recommendations may have unexpected side effects, as discussed in Chapter 4.

Risk

While related to boldness and personalization, risk is the recommender's preference to generate recommendations for obscure items, under-represented items, or items that might seem not related to the user's information task. A risk-taking recommender would recommend more of these items, while a risk-averting recommender would recommend fewer. Risk differs from boldness through the following example: a recommender could be risk-averting and yet be bold by strongly recommending a rather popular item for a particular information task. Again, the actual risk of a particular algorithm may differ from the user's opinion of risk, especially if the few risky recommendations are highly salient.

Discussion: While there are several ways risk could be measured in a recommender, we explored one version on our previous work. In that paper, we defined risky items are items that the system does not have a lot of information about [88]. Some items may be recommended based on a small amount of data; and it is possible for these items to have extreme recommendation values. Risk determines if a recommender uses unknown items in its calculations, how frequently such items appear on recommendation lists. We connected risk to a confidence measure for recommender systems, and found that the two concepts may confound each other. Specifically while low confidence items may not be risky, risky items had low confidence [88]. Further, in that paper, we found that users had different opinions on the usefulness of risky items depending on their user task.

Affirmation

An affirming recommender is concerned as much with establishing rapport with a user as it is with generating recommendations. Such a recommender would recommend items the user is most likely to know and have an opinion about—‘safe’ recommendations. They can be viewed as a ‘sandbox’ recommender environment and may not generate the most personal or useful recommendations. Once again, this is an issue of system design as well as a property of a recommender algorithm, but is viewed either as a limitation or tuning placed on an algorithm, a filter placed on the dataset, or some combination.

Discussion: Such setups are commonly used for new users, for example, to build trust. New users are an especially interesting part of the recommendation process: the system has to make the best impression it can on the least amount of information. This problem has been studied extensively in [89, 117, 133]. But new users are not the only group to benefit from an affirming recommender. Any user who needs recommendations in an unfamiliar area may want reassurance that the recommender can deliver high quality recommendations.

Pigeonholing

A user would feel pigeonholed if the user received many similar recommendation lists over time. For example, a user of a movie recommender might feel pigeonholed if all she received were recommendations for documentaries no matter what she asked for. While related to spread, this term describes the breadth of returned results viewed over an extended period. Again, like spread, the system’s actual pigeonholing may differ from the user’s view of the system.

Discussion: While pigeonholing is usually viewed as a pejorative term, in this context it is value-neutral. Pigeonholing is related to recommendation spread. Whereas spread is the user opinion on one recommendation list, pigeonholing is the user opinion of the recommender over time. For example, in [117] we found that Item-based CF could push users into “similarity well” where all recommendations were for very similar items. If

these recommendations continued over time, the user could feel pigeonholed. There are reasons why a user may want to be pigeonholed as well. For example, a user of MovieLens may only want recommendations for movies with MPAA ratings of G, PG or PG-13, such requests may be common [132]. Given this, we would argue that if many users felt pigeonholed, the system might have to worry about balkanization, especially if users felt pigeonholed into a few specific areas.

Trust

Users will trust the system on several different levels. They trust that the system will provide them with reasonable and accurate recommendations (not shills or ‘sponsored’ recommendations); they trust they are not being deceived; they trust that the system will behave in a similar way for future interactions. They trust that the system will get better (make better recommendations) the more information they provide; and they trust that the system will keep their data reasonably private or inform them of any changes otherwise.

Establishing trust is also extremely important. New users have different needs and expectations about a recommender system than experienced users have. Among other things, the new user needs to determine whether to continue to use the system. As such, the recommender should make a good first impression towards new users without sacrificing recommendation quality.

Discussion: As discussed in affirmation, the first impressions of a recommender are extremely important to establish a relationship with a new user. Trust, however, runs much deeper. Several have argued on the importance of trust in recommender, the most vocal being [141]. As in many other domains, trust can be slow to establish, but lost easily. Cosley et al. performed a study in MovieLens in which they manipulated recommendations to see if users could be influenced [27]. They found that users could be manipulated into rating items differently than they had previously, but this manipulation came a cost: users were less satisfied with the system overall after participating in the experiment. Users lost trust in the recommendations. We further saw this when we added a confidence metric into MovieLens [88]. Existing users in the experiment were

not pleased to learn that MovieLens generated low confidence recommendations. It did not fit with their worldview, and users lost trust in MovieLens.

Aspects of the User Information Seeking Task

Finally, there is the User Information Seeking Task pillar. It is comprised of Concreteness of Task, Task Compromising, Expectations of Recommender Usefulness, Recommender Importance in Meeting Need, and Appropriateness.

The aspects for this pillar are different in nature from the aspects of the other two tasks. Previously, the aspects served as a language for users to use when describing recommendation lists. Here, aspects run over a different space. Aspects for this pillar refer to specific elements of an information seeking task that a recommender may choose to support. Not all information seeking tasks are expressed in equal clarity nor are all limited to the same scope. The aspects in this pillar refer to these differences and act as a reference for designers when creating recommender systems.

Concreteness of Task

The concreteness of an information seeking task is the user's ability to express their task to themselves or to other people; literally, it is the ability to put the task into words. As first discussed in Taylor's Mechanisms and Motivations model [142], users can express their information need at one of several different levels. It is quite possible that a user can be seeking something yet not be able to succinctly describe what he wants. One classic example is taxes. Suppose a user wants to know how buying a house would affect his taxes. He could know that his task is about "how buying a house affects his taxes" or it could be "the effects of deducting mortgage interest payments and property taxes when taking out a loan", or it could be, "Why is the IRS so eager to talk to me?"

Discussion: Taylor's model includes a phase called 'a conscious need'. In this phase a user knows he needs some information and he has an idea of what it is and where to look for it, but he cannot articulate this need, even to himself. For example, he could see that

his dog is not behaving as she used to, but he does not know if she is sick, hurt, or going through some changes. Unfortunately, he can only phrase his need as ‘my dog is acting strangely’ and know he needs more information, but does not know how to find it or how to express his need.

We expect that users familiar with a recommender may come to the system habitually, even when they do not have a concrete need. These users may start browsing recommendation lists, solidifying their need as they go. A recommender supporting this kind of behavior may need different user interface element or recommendation lists from systems supporting well-defined information needs.

Compromising the Task

Many users modify, or compromise, how they view their task as they search the information space. A user may not know the correct vocabulary to express her need; the task is concrete but the user may not know how to describe her need. This may be because the user is a novice, accustomed to a different vocabulary, or trying to express her need in a different language. As the description of a user’s task changes, we say that the description is ‘compromised’. For example, a user may be interested in technical information on the chemicals in flea collars and the impact on a dog’s health. Because the user does not know how to phrase this task to a search assistant, she may compromise her task and start with “flea collars”. Allowing users to change the description of their need quickly and easily based on recommended items helps support these kinds of users.

Discussion: This aspect also comes from Taylor’s Mechanisms and Motivations model [142]. The final state of this model is the ‘compromised need’, as we discussed in Chapter 2. The language the user uses to describe her need may be very different from the language used to organize and index information related to that need. One of a librarian’s primary tasks is to help translate between the two worlds. Recommenders also need to perform this translation.

Expectations of Usefulness

When a user comes to a recommender system, she will have certain pre-existing expectations about the system and how it will help with her information seeking task. Because the task exists before the user interacts with the recommender, it is impossible to state that there are no expectations. The recommender has a responsibility to make sure that users understand what information the system can and cannot provide so that expectations can be reasonably managed.

Discussion: Expectations come at two levels: an experienced user has expectations based on deep interactions with the systems. A novice or new users generate expectations based on surface opinions, system advertising, or other shallow sources of the recommender. Thus, not only does a recommender need to do a good job helping users with their needs, it also needs to advertise its services faithfully. For example, MovieLens makes it very clear that it only generates movie recommendations, thus users looking for book recommendations would correctly assume that MovieLens could not help them.

Recommender Role in Meeting Need

A user can use many different sources of information to complete their seeking task. The recommender could be used either as a central information source to find information of interest or as a secondary information source where the user validates and verifies information they have previously gathered from other sources. For example, in a movie recommender, a user may look up additional information about recommended movies from a database such as IMDb, or watch the trailer of the movie before deciding whether to follow the recommendation.

Discussion: Users can use a recommender from within any stage in their information seeking behavior. External information sources could be accessed before, after, or in parallel with a recommender to provide additional streams of information to the user. By recognizing this limitation, recommenders can decide how much information to provide

about any given item. For example, does the MovieLens recommender need to provide movie previews and links to purchase DVDs? MovieLens has chosen to limit the amount of information it contains on each item in order to concentrate on generating high quality recommendations knowing that there several other sources of movie information. In fact, MovieLens provides a direct link to IMDb for users to gather more information; MovieLens knows its role in the information seeking process.

Appropriateness

A recommender will be judged by the user not just by providing correct recommendation lists, but also lists appropriate to the user's information need. While particular recommendations may be of high quality, the recommender as a whole may not help the user with their recommendation need, thus it is not appropriate. While related to correctness and usability, this is a judgment about whether or not the recommender can help with a particular information seeking task.

Discussion: This aspect is related to 'expectations of usefulness'. In particular, this aspect comes into play after a user has chosen to use a recommender for a given information seeking task. For example, an experienced researcher may find a research paper recommender designed for novices and students not appropriate for her research paper needs. Other aspects are also closely related to this aspect, most notably correctness and usefulness. The difference is that this aspect relates to recommender as an entity; correctness and usefulness make judgments regarding an individual recommendation list.

The HRI Analytic Process Model

By itself, Human-Recommender Interaction theory is a descriptive model, a way to describe and understand recommenders from a different point of view. When added to a larger process model however, it becomes constructive—a way to analyze and redesign recommenders to better meet user information needs. Figure 6-2 shows an overview of the HRI Analytic Process Model.



Figure 6-2: The HRI Analytic Process Model

There are four parts to this model: the user’s information seeking task, HRI itself, recommender metrics, and recommender algorithms. The model flows in both directions, but we will explain it moving from left to right. It works as follows:

1. We postulate a user interacts with a recommender as part of an information seeking process. As such, a key part of this model is to understand the user information seeking tasks in the recommendation domain. For example, why would a user come to a book recommender? To find a book for a gift? For professional development? We need to express the user’s need both in a descriptive statement and as a description of the kinds of items that would best meet the user’s information seeking task.
2. The language we use to describe the kinds of items a user wants is HRI. For example, would a particular task require risky recommendations or perhaps recommendations from a more affirming recommender? Only some aspects will be important to any given task.
3. By looking at which HRI aspects are important to different user tasks, we can design metrics to categorize the relevant and important differences between tasks. A variety of metrics is needed to cover the space of HRI aspects. In Chapter 8, we present a selection of metrics based on HRI Aspects. For example, spread and pigeonholing suggest the importance of a metric to study item similarity in a recommendation list, such as the Intra-List Similarity Metric from Chapter 4.

4. These metrics can, in turn, benchmark a wide variety of known recommender algorithms—creating a catalog of algorithm behaviors. The most appropriate algorithms can be chosen for any given set of HRI Aspects.

As shown in the figure, HRI and the recommender metrics form the bridge between tasks and recommender algorithms. This bridge is key in tailoring recommender systems to information seeking tasks; we can translate information seeking tasks into a formalized language with known properties. Moreover, we can select recommender algorithms whose performance best matches these properties. This has several implications for the design of recommender systems.

Applying HRI and the Process Model to Recommender Design

HRI and the Process Model can be applied in several different ways to the design, construction, and testing of a recommender system. First, HRI provides a framework for understanding users and their information seeking tasks. Any task is broken down into the Aspects important to a user wanting to accomplish that task. For example, a task focused on finding very similar items to a list may want high usefulness, low risk, and be pigeonholed.

Second, HRI provides a toolset for evaluating the usefulness of a recommender system design to see if it meets user needs. Aspects important to user tasks in a domain lead to questions system designers can ask about their recommender. For example, how will users view recommendations? Will they return to the system often? How transparent are the recommendations to users? How bold or fresh are the recommendations? What level of risk is appropriate? What other tools besides this recommender might a user use to solve their information need?

Third, recommender algorithms are linked to aspects through the Process Model. Algorithms are benchmarked based on the recommendation lists they generate. These lists can be described in terms of the aspects which best portray the composition of items on that list. Once analyzed, we can map recommender algorithms to user information

seeking tasks via HRI Aspects. By selecting the appropriate algorithm(s) for a given task, we can personalize a recommender not just to a user, but also to their current information need.

Finally, HRI gives us a language to describe the different components of a recommender. A review of this space would allow system designers to see what ‘holes’ where user tasks and algorithms are not meeting each other or evaluate the potential use of a new recommender algorithm in a existing system.

Limitations of HRI and the Process Model

While HRI provides many benefits in understanding the recommendation process from the end user’s perspective, it also has limitations. In this section, we will review a few limitations and discuss possibilities for overcoming them.

1. HRI assumes users have information seeking tasks

In this dissertation, we assume users of recommenders have information seeking tasks; it is not proved. It is the third Pillar of HRI. As discussed in Chapter 1, the importance of this task will vary by domain. One of the strengths of HRI is that the first two pillars are independent of the information seeking task; they are the language pillars. This language can be used by a user regardless of the importance of an information seeking task. If a user says he wants bolder recommendations, HRI can accommodate him, if even the system does not know why the user wanted bolder recommendations.

2. HRI focuses on a single user interacting with a single recommender

There many interesting group/team recommender applications that HRI cannot support. To do so, several parts of HRI would have to be re-evaluated to account for the differences between good recommendations for one user compared to good ones for the group. For example, this could involve the creation of a fourth pillar to represent the group interactions in the

recommendation process. As HRI is applied to more domains, we hope to expand and augment it as needed.

3. HRI itself compromises user information seeking tasks

This limitation claims that in order to use HRI, a user's information seeking task will be compromised when mapped to HRI terminology. This is a concern, but not unique to HRI. The design of the recommender plays an important part. For example, HRI can be used to narrow the scope of a recommender, to limit the user tasks it chooses to support. For users with a supported task, there is very little compromise. Further, for other recommenders, an HRI-based interface may do a better job interpreting a user's information seeking task than other interface designs. In general, HRI can only compromise a user task if the user performs the translation to Aspects herself. A solution to this problem is to create a 'reference interview' interface in the recommender that asks users questions about their need; the answers can be used to select HRI aspects. This approach is similar to current library practices [142, 153].

4. HRI is incomplete.

HRI is not meant to be complete, but rather provide a rich framework to discuss what it means for a user to receive a recommendation and how a user might judge a given recommendation as 'good'. While the three pillars have theoretical grounding, the aspects were born out of experience. They were chosen after a careful review of the recommender systems literature and discussion with experts and end users. We believe these aspects are important, but they are not exclusive. As recommenders evolve, so will the aspects on each of the three pillars. Not only do we expect this, we encourage it. We encourage any discussion and debate on Aspects, as it helps bring focus back towards end users and their needs in recommender systems.

Conclusions

In this chapter we presented Human-Recommender Interaction theory, a descriptive theory expressing the recommendation process from an end user's perspective. HRI is composed of three Pillars, each of which contains a set of Aspects. These Aspects are language elements with which users and recommenders can communicate about the kinds of recommendation lists the recommender is generating.

In addition, we presented the HRI Process Model, a constructive model detailing how HRI links a user's information seeking task to a particular recommender algorithm. This is done by using HRI Aspects as a translation between the user's task and the recommender. On the recommender's end, HRI Aspects are connected to a series of metrics. These metrics, in turn, have categorized a variety of recommender algorithms, and depending on the HRI Aspect sent to the recommender and the algorithm the user is currently using, the recommender can choose to alter or replace the algorithm with a different one better suited for the given Aspect.

In order for the Process Model to work, we need three things. First, we needed HRI, as was presented in this chapter. We also need a set of new recommender metrics and a benchmarking of recommender algorithms against these metrics. We will present both in Chapter 8.

Before we do so, we will demonstrate the effectiveness of HRI by returning to the domain of digital libraries. In Chapter 7, we use this domain to present a series of example users and user tasks from which we will demonstrate how HRI and the Process Model can be applied to users and their information seeking tasks.

CHAPTER 7

RECOMMENDER TASKS IN A DIGITAL LIBRARY

For any given domain, it is important to understand the end users and their tasks. As such, we present an example list of users, tasks, and personas for the domain of recommending research papers in a digital library. We used an iterative approach to create this listing. Over an extended period, we organized small and large interactions with domain experts, librarians, and potential end users. We compared their suggestions against information theories, our prior research in this area [90, 146], the generic recommender task listings provided by Herlocker et al. [51], and current online digital library practices. Repeating this process, we converged on the set of tasks and users listed here. Please note: this is not exhaustive, but rather a representative sample that, in theory, could be supported by a recommender in a digital library. We present this example list to demonstrate how HRI may be applied to user tasks in a domain. Such a task analysis will help system designers bring a recommender system into a new domain. We first present a listing of user types, followed by example tasks in this domain, and end with three personas that we will use as examples in future chapters of this dissertation.

Four Kinds of Users

There are four kinds of users in a digital library. For each user type, we will discuss our assumptions about this kind of user and provide a typical example.

Novice Users

In this context, ‘inexperienced’ means not being familiar with the research area. Thus, users in this category are not paper authors and might not know particular authors, papers, or even keywords to express their information need. These users have a vague and ill-defined information need, or have needs defined in a language that does not map to the language used in the digital library. As such, it may be easy to satisfy these users’

information needs incrementally, since even a rough result will help them narrow their information need.

These users are further separated into two sub-categories: those that know how to use the digital library and those who do not. For example, a new graduate student might have training on how to use the DL and know a few professors' names, but not know research areas, whereas an experienced programmer might know what keywords to look for, but not know that the digital library exists.

Experienced Users in This Field

We define an experienced user as someone who has at least one paper published in the digital library. While this definition might not directly parallel other meanings of 'experienced', it will serve us well. An experienced user will know how to use the digital library and will know about authors, papers, and keywords describing his/her research area. The second half of the definition of this category is "in this field". The user is looking for information in his area of expertise. As such, this user will have specific information needs and may be difficult to satisfy because of the exact nature of these information needs. An example of a user in this category would be a machine learning professor or senior graduate student who is looking for papers from that specialty.

Experienced Users in a Related Field

This category is closely related to the previous category. The only difference is the user is looking for information in an area that is not their field of expertise. While they know how to use the digital library, they do not know specific authors, papers, or keywords to meet their information need. Because of their familiarity, they expect the DL will help them find the appropriate information. As such, these users have vague information needs but high expectations and demands. An example user would be a machine learning researcher interested in bio-informatics looking for papers about protein folding.

Experienced Users in a Non-Related Field

The distinction between related and non-related fields is arbitrary, but we have chosen a concrete way to distinguish 'related-ness' of fields. We assume that a digital library

contains information on one topic, for example, the ACM Digital Library holds computer science papers and PubMed [102] holds medical research papers. Thus, an experienced user in a non-related field has at least one paper published, but not in the current digital library.

It is assumed the user's familiarity with digital libraries carries over to this DL. This user could even be cited in this DL and not be aware of it. As opposed to novice users, these users have exact information needs, but not know enough of the authors, papers, or terminology to describe their need. Their expectations are dependent on previous experiences with other digital libraries and their opinion of the field represented by this DL. An example user would be a social psychologist curious about theories of information sharing in distributed systems.

User Tasks

We present eight user tasks. They are a sample of possible tasks a recommender could support in this domain, and are not meant as an exhaustive list. Rather they inform recommender design and allow us to apply HRI without having real users in this domain. Each task is defined in terms of the papers sent into the recommender, papers received from the recommender, and the interpretation of recommended list.

Fill Out Reference Lists

In: a small set of citations (references for the paper in progress)

Out: a small set of citations (recommended additions to reference list)

In this task, the user is looking to complete a reference list for a given or proposed paper. As such, the user is interested in a small set of papers to review and potentially include as additional related work. The generic version of this task is considered a core tasks for this domain, similar to Herlocker's "Find Good Items" task [51]. There are many variations, including: find authoritative references, find novel references, find similar references, find references to complete a subfield, etc. A user does not need to specify

extra contextual information, but if he does, the recommender should tune the results accordingly. The implications of this are discussed below.

Maintain Awareness

In: a large set of citations (user's current state of knowledge)

Out: a small set of citations (papers the user should review)

In this task, the user wants to maintain or grow their existing state of knowledge, employing the recommender to make suggestions. This task could be run on a regular basis, perhaps as an email reminder or set up as an RSS feed. Variations on this task could be accomplished with filters or merging of lists, or example, "only show papers that appeared in these conferences or journals", or "only show papers from this set of authors". The principle behind this task is same, however: to maintain an awareness of the published work in their research area.

Find Starting Point for Research

In: a small set of citations (current state of knowledge)

Out: a small set of citations (papers that fill out the research area)

Finding papers to read in an unfamiliar research area can be a daunting task. A recommender can recommend papers to act as a "starting point" in that area. There is an implication that returned papers will be either core or authoritative papers of the area, possibly highly cited or otherwise distinguished. In essence, the recommender helps the user feel comfortable as she starts researching this area and recommends papers accordingly. It is possible that papers sent into the recommender could appear in the results set (i.e. Yes, you already know a core paper in this area).

Explore Research Interest

In: a large set of citations (user's current state of knowledge), a small set of citations (user's research interest, i.e. the current query)

Out: a small set of citations (papers the user should review in context of the given research interest)

A user with some knowledge of a research area could use a recommender to find interesting and novel work in that area. By starting with a detailed user model and an interest (as a research direction), a recommender can make personalized recommendations tailored to this interest. This task is unique because a user provides both a known information state and a relevant query; the user provided context. This context will ‘flavor’ the returned result set. Because of this, it is implied that the documents returned from this task are more novel than those returned from “Finding Starting Point” are.

Find Relevant in List

In: a large set of citations (the candidate list), a small set of citations (research interest)

Out: a small set of citations (papers relevant to research interest from the candidate list)

The user is looking for a subset papers most relevant to a described interest from within a specific collection. The collection could be defined either by the user or by the system. It could range, say, from a hand-picked bibliography to a set of conference proceedings, for example. Only items found on that list could be returned as recommendations. For example, Relescope was a system from IBM that analyzed a user’s publication history and the papers at a conference and recommended which paper sessions the user should attend, as well as which paper authors to converse with [36].

Fill in Knowledge Gap

In: a small set of citations (current state of knowledge)

Out: a small set of citations (citations to expand state of knowledge)

In this task, a user is familiar with a research area but worried about gaps in his knowledge. The recommender expands their current state of knowledge, attempting to “fill in” the implied research area. The goal is to expand outward using the given list as a base. Thus, the returned results may not be directly similar to the given list but are citations that are somehow related in the literature. For example, searches of collaborative filtering algorithms might return machine learning or data mining results.

Explore Novelty of a Given Paper

In: one paper

Out: a small set of citations (citations that are similar to given paper)

For this task, a user wants to determine how novel a paper is in the literature, for example, to determine if the paper is worthy of publication. Thus, the user is looking for previous and related work that the paper failed to cite or other papers closely related to this work. To find such publications, a recommender may have to cast its ‘searching net’ far and wide.

Find More like This

In: a small set of citations

Out: a small set of citations (recommended additions)

This task answers the question, “What papers should be added next this list given that the list is meaningful?” An expert in the field implicitly performs this task when he says, “Oh, you read papers x and y ? Then have you read paper z ?” This task becomes more interesting when the number of results is constrained. For example, this task could be viewed as, “I’m flying to a workshop and I need to read a few more papers in this area. Here is what I know; I’ll only have time to read two papers.”

Relationships between Tasks

There are varieties of relationships between the tasks. Here, we will explore a few such relationships in an attempt to discern meaningful patterns.

The Large Set Phenomenon

Three of the tasks, “Maintain Awareness”, “Explore Research Area”, and “Find Relevant in List”, make use of a large set of citations, such as a user’s personal bibliography, or a set of papers gathered from journals or conference proceedings. It is expected that such sets would be treated as a reference store—created or updated infrequently, yet accessed often. A large set is used in two different ways: first, it is the ‘information context’ within which a recommendation list is generated. Second, the set can be an externally defined universe of citations from which the recommendations have to come, as in “Find Relevant in List”. In general, any of the other recommendations tasks can be augmented to include either of these two large sets; only a sampling of the possible tasks is included in this discussion.

Large sets change the nature of the affected user tasks. It is likely these tasks will be re-occurring, where the user is interested in continued streams of information based. For example, the “Explore Interest” task could become a “generate reading list” task, where the recommender provides one paper at a time for the user to read, say, per week. In a similar fashion, “Find Relevant in List” could run each month to review new papers added to the digital library a returning a list of important papers.

The Fluidity of “Find More References”

While appearing different on the outside, four of the tasks above are variations of a generic “find more references” task. These tasks are “Fill out References”, “Find More like This”, “Fill Knowledge Gaps”, and “Find Starting Point.” All tasks generate recommendations based on a small set of entered citations (the citation basket). Yet, to the user, these tasks have a different feel from each other. A user could start with one task, but as they become more knowledgeable or comfortable, the task could morph into a different variation. The difference, however, is the user’s initial interaction with the recommender; to accommodate this, we created different user tasks as “starting points” for this one generic task of “Find More References”. This implied fluidity between tasks can be expressed as differing points on a two-dimensional grid, as shown in Table 7-1.

Table 7-1: The Fluidity of 'Find More References'

	Similar-to-Interest	Expand-from-Interest
Novel / Risky	Find More Like This	Fill out Reference List
Authoritative / Conservative	Find Starting Point	Fill in Knowledge Gaps

The Interest dimension defines the extent to which the recommender looks for content outside the scope of the citation basket. Measures of similarity, usually content-based, can be a first approximation measuring ‘closeness’. Think of this as an angler deciding where to cast the net: close to the boat or (relatively) far away. The net is the same size, but the locations are different. Larger fish do not approach the boat, but many small fish do. Does the angler want many small fish, or the possibility of a few large fish?

On the other hand, the other dimension has to do with the quality and size of the angler’s net. Instead of looking for things with either the same or expanding levels of content, this dimension explores the perceived ‘quality’ of the returned items... ranging from items that are known to be good, but could be considered boring, versus returning more exciting results that could potentially be bad. These differences are not set in stone—they are only a starting point from which the user can customize the task to fit their particular information need.

Finally, all of the tasks presented take papers as input and generate a small list of papers as output. There are many other possible tasks in this domain; we have chosen a paper-centric subset to provide a coherent discussion of the differences between tasks.

Personas in this Domain

While we can gain quite a bit of information from understanding users and tasks, personas allow us to visit the recommendation process from an alternative point of view. We present the following personas for this domain.

Jill

Jill is a scientist at a corporate research laboratory. Her lab director just assigned her to start a program on ubiquitous computing. She has read a bit on the subject, but she is not overly familiar with it. She conducts a search for overview material on ubiquitous computing and selects a few articles that seem relevant, but does not know how authoritative, comprehensive, or complete her selection is. She has little knowledge about this particular area and she is looking for an easy way to get an overview of the area.

Chris

Chris is a well-known and respected researcher. As a dedicated professional, she forgets sometimes to take regular vacations. Eventually, she is convinced to take a semester-long leave to relax and reconnect with her family; she takes no research with her. She returns relaxed, refreshed, and wide-awake only to remember that she agreed to attend a symposium to talk about the latest advances in her area. She feels nearly a year behind, because of both the semester off and the crunch beforehand during which she neglected her reading. She needs to get up-to-speed on the latest on her field, but she only has the time on the airplane to read these papers. What should she take with her?

Max

Max is a new graduate student writing one of his first conference papers. He has performed great research and did an exhaustive search for related work to cite. He still feels however, that he could be missing an important citation or perhaps may be citing the 'wrong paper' for an area. He is looking for validation of the related work he has already found and suggestions on what areas he might have missed.

As these personas have differing information needs, so they will want differing recommendation lists. For example, while Jill and Max may both be interested in authoritative papers, Chris would much rather have fringe papers, possibly cross-disciplinary work. Max is only looking for a few specific papers to add to his existing list, so he will need tightly relevant recommendations whereas Jill is starting with very

little and wants to grow her list as widely as possible. These differences imply that these users have a different internal definition of what a useful or ‘good’ recommendation list would be for them. A good list for Max may not be good for Chris. We explore these differences when we apply HRI to this domain.

Applying HRI to the Domain of Digital Libraries

There is a complex relationship between users and tasks. Not all users will be interested in all possible tasks. Rather, only specific user/tasks pairs are possible. For example, Table 7-2 states our beliefs on which users would use which in this environment. The specifics of this table are open for debate; we do not intend this table to be complete, rather it allows us to use HRI on the user/task pairs from this domain.

Table 7-2: Example User Type/User Task Matrix

	Novice, no DL exp.	Novice, has DL exp.	Expert in Field	Expert in Related	Expert Non- Related
Fill out References	X	X		X	X
Awareness in Area			X		
Explore Interest			X	X	
Fill Gaps in Knowledge	X	X		X	X
More Like This	X	X			X
Relevant in List	X	X	X	X	
Novelty of Paper	X	X	X	X	X
Starting Point	X	X			X

Once we understand user/task pairs, we can use HRI to characterize the important aspects of each pairing. To ground our discussion, we will turn to our personas, review each one’s task, and use HRI to understand the differences in good recommendations for these users. We do this by assigning HRI Aspects to each user type and each user task. These aspects describe papers each user wants for her current task—what is in a good recommendation list. This assignment does not have to be exact or balanced in any way;

multiple aspects will fit a given user/task pair. Some Aspects are important to a given task, while others are not relevant.

We are not determining what values should be associated with a given Aspect (i.e. Expert Users want low Affirmation and high Boldness). Instead, we are stating which Aspects are the most important—the Aspects a system designer needs to be concerned with when creating a system to support this user/task combination. Perhaps there need to be specific user interface controls, or an analysis of the user’s navigation pattern in response to an Aspect. Thus, if a designer finds that the user tasks he wants to support all have the same three Aspects in common then those Aspects should drive the design of his system.

Jill

Jill is a corporate research scientist interested in finding information on ubiquitous computing. Therefore, she is an *Expert in a Related Field* who wants to *Find a Starting Point for Research*.

Her Task HRI Aspects are Correctness, Transparency, Serendipity, Quantity, Usefulness, Spread, Personalization, Boldness, Pigeonholing, Trust, Expectations of Usefulness, and Recommender Role in Meeting Need.

There are many Aspects important to Jill. Jill has high expectations and wants to be in control of the interaction. A recommender wanting to support her will have to be flexible and have a rich set of controls and options for her to provide an exacting tuning to meet her needs.

Chris

Chris is a professor looking for the latest and most interesting advances in her area. She is an *Expert in her Field* who wants to *Maintain Awareness*.

Her HRI Aspects are Serendipity, Personalization, Saliency, Quantity, Spread, and Freshness.

Chris has a relatively few number of aspects she cares about. But those that she does care about can dramatically alter the recommendation lists she receives. For

example, one week she may only have time for one new paper, another week she may want five. Thus a recommender system designed to support her should allow her to quickly and easily make (and revert!) changes to her recommendation lists.

Max

Max is a new graduate student wanting citations for a paper he is writing. He is a *Novice in his Field with DL Experience* who wants to *Fill out a Reference List*.

His HRI Aspects are Transparency, Saliency, Serendipity, Quantity, Spread, Affirmation, Adaptability, Trust, Task Concreteness, and Appropriateness.

A recommender system designed to support Max needs to provide him with a “helping hand”. It needs to build trust and make all actions clear, moreover it needs to be prepared to change focuses rapidly as Max becomes more familiar with the space and his tasks solidify.

The complete mappings for user types and user tasks are shown in Table 7-3 and Table 7-4 respectively. Aspect values between the two lists are additive; if a user type and a user task both feel an Aspect is important, then the Aspect is ‘doubly’ important to that user/task pairing. These tables were based on our knowledge of HRI, our review of the digital library domain, and our training in Human-Computer Interaction. Even so, without having a functional recommender system to base these user tasks upon, these mappings are only representative sampling of how to use HRI to analyze user information seeking tasks. They should be treated as examples in this domain.

Table 7-3: Example HRI Aspect Mappings for User Types

	Novice Users, no DL experience	Novice Users, with DL experience	Expert Users in this Field	Expert Users in a Related Field	Expert Users in a Non-Related Field
Transparency	X	X			X
Serendipity			X	X	
Spread				X	X
Usability	X				X
Personalization			X	X	
Adaptability				X	
Risk					
Affirmation	X	X			
Trust	X	X			X
Task Concreteness	X	X			
Compromising Task	X				X
Expectations of Usefulness			X	X	
Role in Meeting Need			X	X	
Appropriateness		X			X

Table 7-4: Example HRI Aspect Mapping for User Tasks

	Fill out References	Fill Knowledge Gap	Find More List This	Start Research	Novelty of Paper	Awareness	Explore Interest	Find Relevant
Correctness				X	X			X
Transparency		X		X				X
Saliency	X		X			X	X	
Serendipity	X		X		X	X	X	
Quantity	X			X		X		
Usefulness			X	X	X			X
Spread	X	X			X	X	X	
Usability		X		X	X			
Personalization						X	X	
Boldness		X		X	X			X
Adaptability	X	X					X	X
Freshness						X	X	
Risk	X		X				X	
Affirmation		X		X				X
Pigeonholing			X	X			X	
Trust		X		X				

Conclusions

In this chapter, we presented a set of user types, user tasks, and personas in the domain of digital libraries. These users and tasks were created through an extensive literature review and consultation with domain experts. Even so, we present them here as an example analysis of this domain. In particular, we limited our tasks to those that used papers as both inputs and outputs, and only returned a small set of papers as output. Even with this limitation, we found interesting discussion points on the fluidity of different tasks.

We used these examples to ground our discussion of HRI. When applying HRI to users and information seeking tasks in a domain, one approach is to create a matrix mapping users and their tasks to the different HRI Aspects. This approach requires a detailed analysis in consultation with domain experts or with users themselves. Once created, these matrices allow us to translate a user and her information seeking task into a set of meaningful HRI Aspects which can be sent to a recommender. Continuing with our examples from this domain, we created both a User-to-Aspect matrix and a Task-to-Mapping matrix and demonstrated this approach using a set of personas we created for this domain.

CHAPTER 8

UNDERSTANDING RECOMMENDER ALGORITHMS, PART I: DESIGNING METRICS, RUNNING BENCHMARKS

In this chapter, we return to one of the main tenets of this dissertation: each recommender algorithm has specific strengths and weaknesses, different from other algorithms. Given this information, a recommender system should select and tune the appropriate recommender algorithm (or algorithms) for a given user/information seeking task combination. To accomplish this, we present a family of new algorithm metrics by which we can quantify the differences between recommender algorithms and run a series of offline experiments demonstrating these differences in well-known algorithms.

There are many ideas to keep in mind as we discuss recommender metrics. As discussed in Chapter 3, predictive accuracy and decision support metrics only provide a narrow band of information about recommender algorithms: they evaluate the *correctness* of individual recommended items. While correctness is very important when judging recommenders, it is not the only important aspect. In Chapter 4, we argued for recommendation *lists* as the important unit of measure for judging the user experience in a recommender. Since users receive recommendations in a list context, recommendations should be judged as such. We also proposed a new list-based metric for recommender algorithms, the Intra-List Similarity metric, and our experimental results showed list usefulness depended on more factors than only the usefulness of the items on the list. Depending on the user need, users were more satisfied with a diverse recommendation list, even at the expense of recommendation accuracy. Encouraged by these results, we developed and tested more recommender metrics.

New Recommender Metrics

The following metrics were designed based on HRI, results from our previous work, and our understanding of recommender systems. It is a necessarily incomplete list; it is an

attempt to understand the relationship between HRI Aspects and recommender algorithms. To ground our discussion of these metrics, they are discussed in terms of generating recommendations for research papers in a digital library environment. We expect the translation to other domains to be straightforward.

Popularity

The popularity of a given item is a measure of how well known the item is. Popularity is at the same time both very simple and quite complicated. Its simplicity stems from our intuitions about popularity in the real world—there are differences in popularity between people, items, locations, etc. Popularity is complicated as there are different ways an item can be popular, just as an item or person can be popular for specific crowds or in specific locations. Popularity does not transfer across similar items, nor does it always cross between items with shared popular properties, for example, not all of Brad Pitt’s movies are well known.

Following the definition of popularity from previous work [117], papers that are frequently referenced (papers with high numbers of in-links) will be considered popular, and papers that are not referenced would be considered obscure. A measure of the popularity of a recommendation list is the mean popularity score of all items on that list. Raw citation counts should not be used for this measure as it unfairly biases the metric towards older papers.

For citation j , let $citation_count(j)$ be the number of citation to j that exist in the database. Let $years(j)$ be the number of years since citation j was published, rounding up. So, “1” means “less than and up to one year ago”, “2” means one year and one day up to two years ago, etc. Thus,

$$popularity(j) = \frac{citation_count(j)}{years(j)}$$

Popularity is a non-personalized measure, with a large variance in measured values (e.g. in the ACM Digital Library dataset, values ranged from just under 30 to approaching 0).

Discussion Because of our history relating the term ‘popularity’ to the number of ratings an item has (i.e. [117], among others), we will continue to define popularity this way. But there are other possibilities. For example, how frequently was this item download from a digital library? How often was it read in reading groups or used in examinations? What is the popularity of the authors in various research communities? What is the popularity of the venues where the item was published? Or, we could look at other measures that make use of other properties of our data, such as measuring authors’ citation histories. Our definition of popularity is an internal measure; it only uses information from within the dataset itself to form the metric. We have moved external measures to a different metric, Authority.

Ratability

The ratability of a given item is the probability of a user consuming and then providing an opinion on an item given the current user model (the list of previously consumed items by that user). Items with high ratability come from two sources: items the user has consumed but has not added to the user model, and items the user has not consumed yet but is likely to in the near future. Restated, given what I know about user x , which item will user x most likely rate next? The higher the ratability score, the higher the probability that the given item is one that the user will rate. This is different from which item user x will like the best or which item would be the best for user x to rate next (for whatever reason), but simply which item user x will rate next. A measure of the ratability of a recommendation list is the mean ratability score of all items on that list.

There are many different ways to implement ratability; we choose a simple approach. If we assume independence of rating events and we define a conditional event in terms of co-citations between two papers, then we define the ratability of a target item as the posterior probability generated by a naïve Bayes classifier. The independence assumption is a strong one to make and it is not clear if that holds in the world of citation graphs. However, assuming it did, the classifier would be training on two cases: to include the target item or not. All training examples are positive. Running the classifier

over all potential target items for a given input basket would return a ratability distribution.

The ratability scores of all items returned on a recommendation list are averaged to generate the ratability score for that list. The higher the score, the more ‘ratable’ that this list. The lower the score, the more novel the list is. For a detailed discussion of a naïve Bayes classifier, see Chapter 3.

Discussion Ratability is a measure of the ‘obviousness’ of a user rating a given recommendation. Highly ratable items are items which, statistically speaking, are the next ones that will be added to the user’s user model. Items with low ratability would be considered novel with respect to the user model if the user rates them and would be more interesting to know about than the rating of items with high ratability. While the definition of Ratability does not necessarily match up directly with a naïve Bayes classifier, they share the same base concept: classifying unknown items in to either “user has consumed/will consume” or “user has not/will not consume”. As such, it is expected that any statistical classifier used as a recommender algorithm will have exceptionally high (perhaps even perfect) ratability.

Authority

The authority of an item is a measure of its centrality and importance as compared to all other items in the dataset. An authoritative item is one that has had a high level of influence on other items or users in the dataset. Unfortunately, influence is difficult to measure. Much like popularity, authority can either be defined internally as a measure in the dataset, or externally, through a variety of factors. We view authority as an ordering imposed on a domain by an external judging entity. For example, ISI’s Impact Factor [47] could be used to provide an authority score to all articles in those journals. Many external measures, however, are imperfect and may cause problems as an authority measure. For example, ISI’s Impact Factor measures the impact at the journal level, not at the publication level. Finding an external measure that maps directly to all items in a dataset may be difficult to find. A measure of the authority of a recommendation list is

the mean authority score of all items on that list. There was no implementation of an authority metric for this dissertation.

Discussion As authority is a ranking of items based on a judging entity, the choice of entity is paramount to the usefulness of this metric. As mentioned, an impact factor of place of publication is a good measure for authority. A Google-inspired PageRank measure [106] might also work well (as authoritative items tend to have higher quality references). There are several graph-theoretic measures that could also be used as a proxy for authority [134], assuming the measures are qualitatively different from popularity. Authority is meant to be a reflection of independent, subjective opinions of those who have consumed the item in the past.

Personalization

Personalization is a measure of how different recommendations are for each user in a recommender system. Thus, a personalized system will generate custom-tailored recommendations for each user. The implication is that users with different user models will receive different recommendation lists. Recommendation lists for different users are compared against each other to determine how many items overlap between the lists. The lists can be further compared against the popularity of items in the system to determine if a subset of the items is being recommended a disproportionately large percentage of the time. A measure of the personalization of an algorithm is the mean personalization score from all user comparisons.

A popular and mathematically rigorous metric for comparing two lists of items is Spearman's Footrule for Top- k Lists. This metric is explained in detail in [35]. Only a brief summary is included here.

Spearman's Footrule metric is the L_1 distance between two permutations of the same n element list. For two lists σ_1 and σ_2 of size n , the Footrule is

$$F(\sigma_1, \sigma_2) = \sum_{i=1}^n |\sigma_1(i) - \sigma_2(i)|$$

The version for two Top- k lists is denoted as F^l where missing elements are given effective rank l . For our measurements, l is set at $k + 1$. The metric is run as above, summing over the union of the two sets of elements. Any element not found in a given list is given rank l .

We measure two implementations of personalization in this thesis:

1. Comparing each recommendation list to all other recommendation lists using Spearman's Footrule. The mean score over all pair wise comparisons is the comparison personalization score for the algorithm.
2. Comparing each recommendation list against an external, non-personalized ordering of the items (top 200 results from popularity, in this case), again using the Footrule. Basket items appearing on the popularity list are filtered from the popularity list before comparing it to the generated recommendation list. The mean score over all recommendation lists is the popularity personalization score for the algorithm.

Discussion Since recommenders are supposed to go through 'the long tail' and pick out the interesting items that each individual user will like, each person should get a custom list, and each list should contain more than just the most popular items. Because of this, personalization may be affected by the distribution of ratings in a dataset. If the dataset has a large difference between popular and obscure items, algorithms may have a difficult time generating personalized lists. This is especially true for longer recommendation lists.

Boldness

Boldness is a measure of how "extreme" a recommender algorithm is. It measures how strongly the recommender recommends particular items to users by analyzing the recommendation score of the generated recommendations. It does so by measuring how frequently the recommender makes recommendations at the extremes of the rating scale

compared to how frequently such recommendations are expected. Because of this, boldness is only defined for recommenders using an explicit ratings scale.

To calculate boldness, we first need to do two things. First, we must define ‘extreme’ ratings. We suggest this definition be an equal number of choices from both ends of the rating/prediction scale. For example, on a 1 to 10 scale the values ‘1’ and ‘10’ could be considered ‘extreme’. Next, we need the expected percentage of the time the recommender will generate extreme recommendations. We suggest using a flat probability distribution, but any distribution will do. Returning to our example, since 2 of 10 choices are ‘extreme’, we would expect 20% of the generated recommendations to be extreme.

Given the above definitions, we define boldness as

$$\text{Boldness} = \frac{\% \text{ of actual extreme predictions}}{\% \text{ of expected extreme predictions}}$$

Discussion Boldness may have a powerful psychological effect on users. Tuning a recommender based on boldness should be done with care. Cosley et al. showed that a recommender could influence user rating behavior with modified recommendations [27]. While the users were fooled, their overall satisfaction with the recommender dropped considerably. Similarly, bold recommendations made with low confidence may create polarized user opinions of the recommender.

Boldness is also noticed by users. In 2003, GroupLens Research redesigned the MovieLens movie recommender. As a part of this redesign, we switched to a new recommendation engine, moving from a User-based CF algorithm to an Item-based CF algorithm. After launching the system, we received many emails from users complaining that the system was “more conservative” than before.

Adaptability

Adaptability measures how a recommendation list changes in response to changes in a user’s profile. This could happen if, for example, a user goes through a life-altering event, such a getting married, or if a user changes some aspect of their professional life,

such as a researcher moving into a new field. As a user alters or adds new information to her profile, she may expect the recommender to ‘keep up’, to alter recommendation lists accordingly. Adaptability is a measure of how well an algorithm does just that.

An overview of our adaptability metric is shown in Figure 8-1. There are two users: the target user, labeled ‘user A’; and a randomly selected user, labeled ‘user B’. To simulate a change in user A’s profile, we randomly replace half of user A’s ratings with ratings from user B. In doing so, we create a new pseudo-user labeled ‘user pA’. We then compare recommendation lists generated by user A and user pA; we look for items appearing in user B’s ratings. Specifically, we are interested in the subset of user B’s ratings not used to create user pA. The intuition is if we alter user A to be more like user B, then a highly adaptable algorithm should generate recommendations for items user B has rated. Thus, pA should *behave* more like B than A would.

We used a leave-*n*-out methodology and computed rank scores for user pA. The adaptability score for an algorithm is the mean score from all pairs of users.

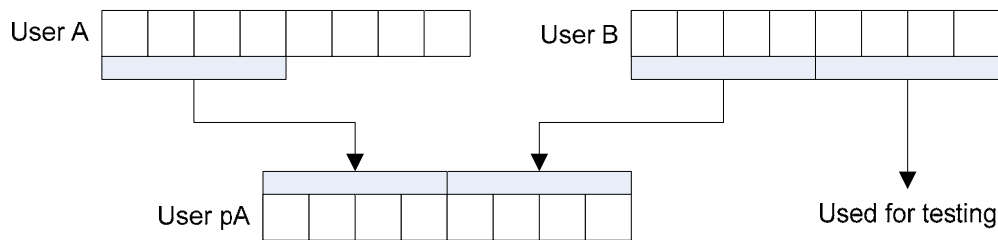


Figure 8-1: Overview of the Adaptability Metric

To compute adaptability for the ACM Digital Library dataset, we chose a random subset of 20 papers from our test set, each of which had more than 16 references. Each paper was limited to 16 references, reordered randomly. Each reference list was split in half. The front half from each paper was used to create ‘pseudo-papers.’ For each pair of papers A and B, we created three pseudo-papers, in which we varied the ratio of items from A and B. “Even Split” is based on the first 8 references from each of A and B, “Adapt Heavy” is based on the first 4 from A and first 8 from B, and “Adapt Light” is based on the first 8 from A and first 4 from B.

Discussion We expect external factors will affect adaptability, such as new users or items entered into the system. We assume a static database for the purposes of this metric. Yet, many internal factors can affect this metric. Not all users are equal in a recommender system; some users can influence the recommendations of other users [116]. An algorithm may have a harder time adapting these users, or adapting users who are under their influence. In a similar light, not all items are equal either. Some items may cause a greater adaptation than others. Analyzing the properties of these users and items would further elucidate our understanding of the adaptability of a particular recommender algorithm. Finally, we measure adaptability by comparing recommendations to the existing ratings of user B. A more extensive measure would be to measure against B's ratings and B's current recommendation list.

Previously Existing Recommender Metrics

As a brief review, we will summarize previously defined recommender metrics used in the offline simulation experiments in this chapter.

Intra-List Similarity

As presented in Chapter 4 and in [162], this metric measures the diversity of a recommendation list. The intra-list similarity of a given recommendation list is based on the pair-wise similarity of all items on the list. This metric does not define a specific measure to calculate the pair-wise similarity, instead it requires the use of an external similarity measure. In this thesis, we use the taxonomy-based similarity metric from [163].

Rank, Coverage, and Adjusted Rank

These accuracy metrics were first discussed in Chapter 5. Following a leave-one-out methodology, Rank measures the position on a recommendation list of the removed item. Coverage is the percentage of the time that a recommendation list contains the removed item. Finally, the Adjusted Rank metric is a combination of rank and coverage, similar to the F1 metric, but is calculated as Rank scaled by Coverage.

Metric Discussion

Popularity and Authority both drive towards the same concept: an ordering over all items in the domain. Popularity looks internally to calculate this ordering while Authority looks externally. While these names are meaningful for this domain, they may not hold across other domains.

We have taken a slightly different approach to Coverage, Rank, and Adjusted Rank than in the recommender systems literature. Other papers have defined coverage as the percentage of the time that the recommender could generate any recommendations for a given input basket. Previous works shows that most recommender algorithms can achieve high coverage by this definition [55]. Our definition is more restrictive. We not only want an algorithm to return results, but to return the withheld items in order to count as a ‘hit’. It is worth noting that an algorithm that scores poorly on one definition will also score poorly on the other.

Previous work suggests that there will be tradeoffs between metric scores. For example, we have shown that modifying recommendation lists to be more or less similar affects the accuracy of the recommendations. Yet, even when making these changes, the recommendations were still viewed as useful to users [162]. We theorize that similar kinds of tradeoffs will appear as metrics interact with each other.

Benchmarking Recommender Algorithms

In this set of offline simulation experiments, the metrics defined in this chapter are used to benchmark a selection of the recommender algorithms discussed in Chapter 3.

Research Questions

These are the research questions we will explore:

1. Can these new metrics distinguish among different recommender algorithms based on their suitability for various tasks and contexts?
2. There are different ways that collaborative filtering can be applied in this domain. How do these differences affect the generated recommendations?

Experimental Design and Setup

We chose a grid experimental design where each algorithm is run against each metric—there is no definition of ‘good’ or ‘bad’ for any given measure, rather we are interested in comparing and categorizing algorithm performance.

We used a snapshot of the ACM Digital Library taken in October 2004 as our dataset. We limited the snapshot so that all papers contained at least two citations and were cited at least twice. This pruning was done to remove loosely connected papers that would have introduced noise into the calculations. For each paper, we have access to its citation data, title, authors, keywords, abstract, and classification in the ACM Computing Classification Scheme [2]. In the end, there were over 24,000 papers in this dataset.

Table 8-1: Algorithm and Metric Listings

Recommender Algorithms		Metrics	
User-User CF (x6)	Naïve Bayesian Classifier	Popularity	Personalization, Within-Comparison
Item-Item CF (x6)	Content-based Filtering	Ratability	Personalization, Compare-to-Popular
Symmetric CF User-User (x6)	Localized Graph Search	Intra-List Similarity	Adaptability, even-split
Symmetric CF Item-Item (x6)	Co-citation Matching	Coverage, Rank, and Adjusted Rank	Adaptability, non-even-splits (x2)
Fusion: CBF and User-User CF (x6)	Fusion: CBF and Item-Item CF (x6)		
CBF Separated - User-User CF (x6)	CBF Separated - Item-Item CF (x6)		
User-User CF - CBF Separated (x6)	Item-Item CF - CBF Separated (x6)		

We divided our dataset into training and test datasets at a 90% to 10% ratio as follows: 90% of the papers directly became the training data. For the remaining 10% of the papers, we randomly removed one citation to create a test case for the test dataset. We included the incomplete citation lists from the test cases in the training dataset. We used a 10-fold cross validation for all experiments in this analysis, where we randomly generated the test and training datasets for each fold.

As shown in Table 8-1, we ran 54 algorithm variants against 9 metrics. The four variants of pure collaborative filtering and six hybrid algorithms were each run at six different neighborhood sizes: 10, 30, 50, 100, 200, and 300. Due to performance limitations with our TF/IDF engine, we were only able to use CBF Separated for our feature augmentation hybrid algorithms. Instead, we ran all hybrids using both User-based CF and Item-based CF.

There are two versions of the Personalization metric: one in which a generated recommendation list is compared against the most popular items in the dataset, and a second where each generated recommendation list from a test set is compared against all other recommendation list also generated from that test set. The Adaptability metric was run with three different variations: even-split where an equal number of citations were merged, adaptation-light split where twice as many citations were selected from the ‘original’ profile (user A) as from the randomly selected profile (user B), and adaptation-heavy, where twice as many were selected from user B as from user A.

Experiment Results

Each recommender algorithm generated recommendations lists for all profiles in the test sets. Each metric was run with scores generated at five different levels: top-10, top-20, top-30, top-40, and top-all (limited at 200) recommendations. Thus, if some algorithm received a score of 4.5 for the top-10 measure of the popularity metric, then the mean popularity of the top ten recommendations for that algorithm was 4.5. The metrics scores are cumulative; a score for top-40 includes the results of top-10 in it.

All shown results are statistically significant ($P < 0.001$), but in keeping with our user-centric model, we will focus on differences we believe users could notice and appreciate. For the collaborative filtering and hybrid algorithms, we will only show one representative neighborhood size, 50, when comparing against other algorithms. So, all results below are based on 14 algorithms. A detailed neighborhood size analysis follows.

Results by Metric

Figure 8-2 shows a summary of results from Popularity. There are differences among all 14 algorithms; Co-citation and both User-based CF algorithms generated more popular results than the other algorithms. Content-based filtering and the two CF-CBF hybrids generated the least popular results. Fusion hybrid algorithms were tempered compared to their respective pure CF counterparts. It is also worth noting the differences top- n sizes have. The highest-popularity algorithms were also the ones to vary the most based on top- n size, with top-40 and top-all measures similar to other algorithms. The popularity results are even more striking when comparing different neighborhood sizes, see below.

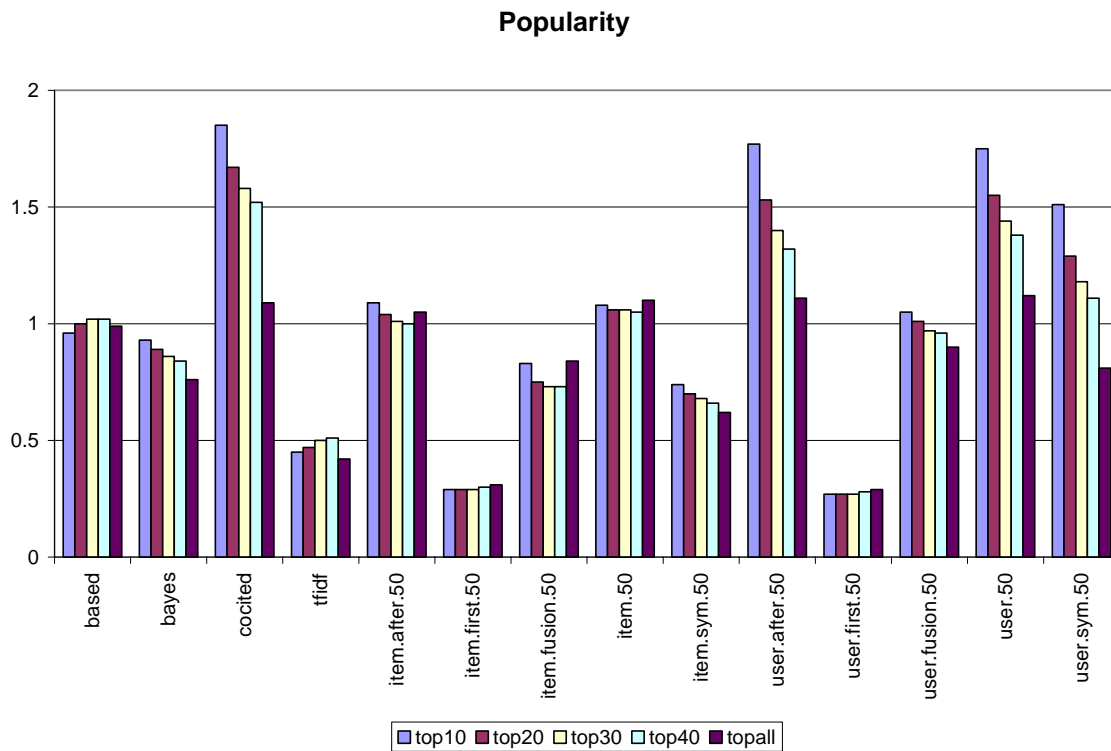


Figure 8-2: Popularity Results

The Ratability results are shown in Figure 8-3. These results are displayed on a logarithmic scale, so the spread is much wider than it appears. As expected, the Naïve Bayes Classifier generated the most ratable results, while Content-based filtering, the Hybrids ending with CBF, and Localized Graph Search generated the least ratable results.

It is also worth noting that the Symmetric versions and the Fusion versions of User- and Item-based CF generated less ratable results than their original counterparts, but that comparing between User- and Item-based CF shows no discernable differences. These results mostly hold for differences in CF neighborhood size.

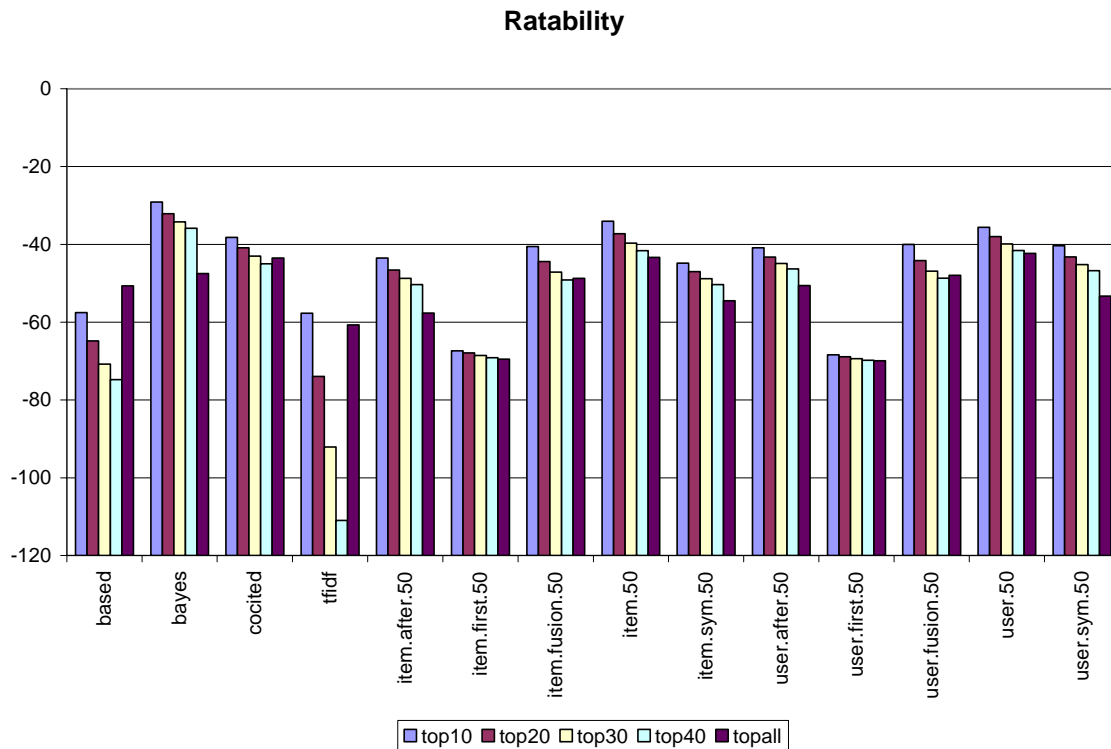


Figure 8-3: Ratability Results

The Intra-List Similarity results reveal that almost all of the algorithms we tested generated dissimilar recommendation lists. All algorithms with the exception of the Symmetric CF algorithms did not score higher than 3% similarity, with the score dropping to 1% as top- n increased. The Symmetric CF algorithms peaked at 6% and dropped to between 3% and 2% as top- n increased. Symmetric Item-based CF appeared to generate the most similar while Bayes and User-based CF appeared to generate the least. All hybrid variants held constant between 2% and 3%. While these results are statistically significant, we doubt a user would notice. These results hold true across all neighborhood sizes.

The Personalization, Compare-to-Popularity metric revealed that all algorithms return different results from most the popular, with the least personalized being the two fusion algorithms topping out at 90% and Co-citation Matching at top-all with a score of 96%. Content-based filtering scored a perfect 100% across all top- n 's, User-based variants and Bayes averaging at 98%, and Item-based variants at 99%. Scores dropped slightly as top- n increased. The “feature augmentation” hybrid algorithms averaged between 97% and 99%. We believe none of these differences would be notable to an end user, however. As neighborhood sizes increase, the CF algorithms, especially User-based, become less personalized, falling to 95% at worse case.

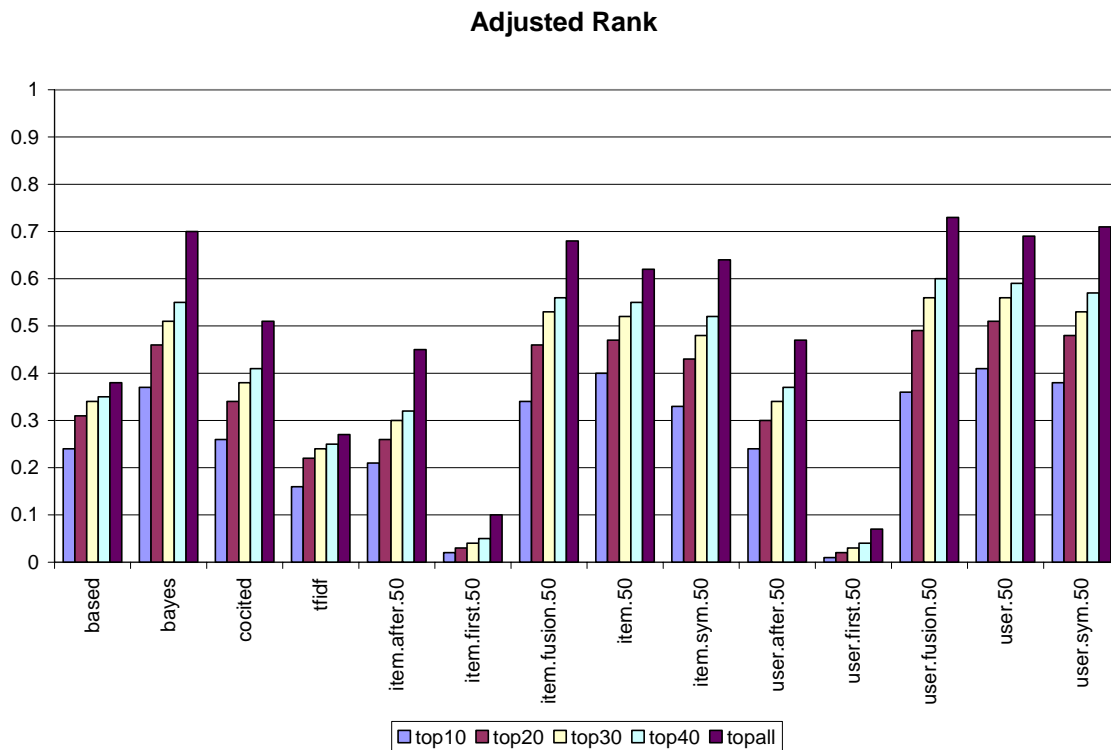


Figure 8-4: Adjusted Rank Results

Personalization, Within-Comparison shows more variation. Localized Graph Search and Content-based filtering were the least personalized, with scores ranging from around 60% to 20% decreasing as top- n increased. The Bayesian classifier was the most personalized with scores around 99%. Both User-based and Item-based CF performed

similarly, ranging from approximately 96% to the mid 80% range, decreasing as top- n increased. Moreover, the Symmetric variants and “feature augmentation” hybrids were more personalized, with scores ranging from 98% to 93%. Co-citation Matching ranged from 91% to 72%, dropping as top- n increased. The Fusion hybrids demonstrated an interesting drop off from 99% to between 85% and 89%. There are interesting effects for the CF algorithms at 10 neighbors: personalization drops sharply, especially for the non-Symmetric algorithms. See below for details.

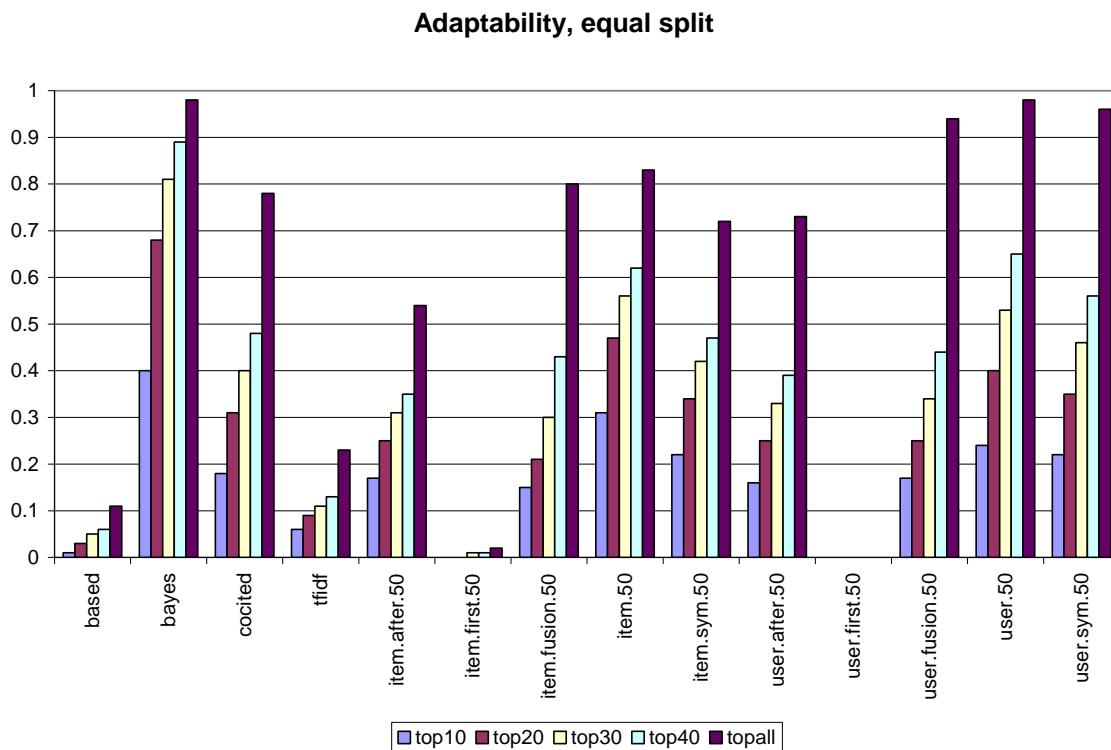


Figure 8-5: Adaptability, Equal-Split Results

The Adjusted Rank results are shown in Figure 8-4. As expected, all algorithms improved on Adjusted Rank as top- n 's grew, with the striking exception of the CF-CBF hybrids; the hybrids performed miserably on this metric. The Naïve Bayes Classifier and the User-based CF variants performed best on this accuracy metric along with the two Fusion hybrids algorithms. Whereas Co-citation, Localized Graph Search, and Content-

based Filtering performed as the worst three algorithms. These results are similar to, but not quite the same, as our previous results from Chapter 5, and we will discuss that below. Rank scores for the CF algorithms remained relative constant across changes in neighborhood sizes, but grew slightly as their coverage score increased.

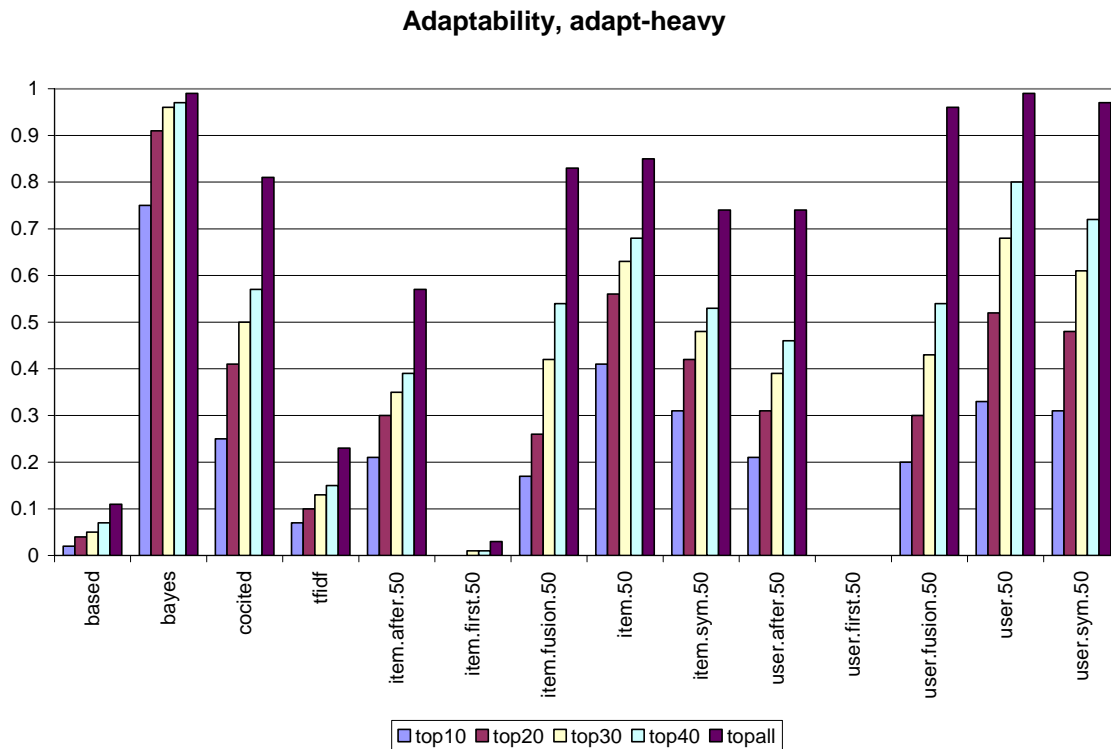


Figure 8-6: Adaptability, Adapt-Heavy Results

Figure 8-5 shows the Adaptability, Even-split results. The two CF-CBF hybrid algorithms scored close to 0 on this metric. Both Localized Graph Search and Content-based Filtering are not very adaptable. On the other hand, the Naïve Bayes Classifier was very adaptable. While there was no difference between User-based CF and Symmetric User-based CF, there is a difference between the Item-based variants, with traditional Item-item being more adaptable. The Fusion counterparts performed slightly worse than their respective CF counterparts. Finally, there are great differences between neighborhood sizes, revealing differences between User-based and Item-based CF, see below for details.

Table 8-2: Summary of Results by Algorithm

	Popularity	Ratability	Intra-list Similarity	Personalization, Compare-to-Pop	Personalization, Within-comparison	Coverage, Rank, and Adjusted Rank	Adaptability, Equal-split	Adaptability, Adapt-heavy	Adaptability, Adapt-light
User-based CF	H	H	LL	HH*	HH	M	M	M	L
Item-based CF	M	H	LL	HH	HH	M	M	M	M
Symmetric User-based CF	M	M	LL*	HH*	HH	M	M	M	L
Symmetric Item-based CF	L	M	LL*	HH	HH	M	M	M	L
Naïve Bayes Classifier	M	HH	LL	HH	HH	M	H	HH	M
Content-based Filtering	L	L	LL	HH	M	LL	LL	L	LL
Localized Graph Search	M	M	LL	HH	L	L	LL	LL	LL
Co-citation Matching	HH	H	LL	HH*	L	L	M	M	L
Fusion: CBF and User-User	M	H	LL	HH*	HH	M	L	M	LL
Fusion: CBF and Item-Item	L	H	LL	HH*	HH	M	L	M	LL
CBF Separated - User-User	H	H	LL	HH	HH	L	L	L	LL
CBF Separated - Item-Item	M	H	LL	HH	HH	L	L	L	LL
User-User - CBF Separated	LL	M	LL	HH	HH	LL	LL	LL	LL
Item-Item - CBF Separated	LL	M	LL	HH	HH	LL	LL	LL	LL

Comparing Figure 8-5 to Figure 8-6 shows a strong reaction in adaptability to varying the number of items from each profile. On average, all results are roughly 10% greater than Even-Split Adaptability. The adaptation-light variant had an even stronger affect, with values reduced by 15% or more. The exceptions being, of course, the CF-CBF hybrid variants.

We summarize our results in Table 8-2. Results range from ‘HH’, for ‘very high’, to ‘LL’ for ‘very low’, with ‘M’ meaning “in the middle”. Results with different letters are statistically significant from each other. Items marked with an (*) denote statistically significant differences from the other items with the same letter, but we do not believe that difference would be notable to users.

Results for Different Neighborhood Sizes

All CF-based and hybrid algorithms were run at six neighborhood sizes: 10, 30, 50, 100, 200, and 300.

Popularity results, as shown in Figure 8-7 and Figure 8-8, reveal striking differences between all ten CF variations across all neighborhood sizes. Item-based algorithms remain relatively flat as top- n increases. In a similar fashion, User-based algorithms follow a similar pattern, but are highly affected by top- n size, generating less popular results as top- n increases. The CF-CBF hybrid algorithms changed very little across neighborhood sizes. All four algorithms display consistently different values on this metric across all neighborhood sizes.

Ratability scores remained constant across all neighborhood sizes except for one interesting phenomenon highlighted in Figure 8-9 and Figure 8-10. The Ratability scores for both Item-based CF algorithms dropped significantly at 10 neighbors. Interestingly, ratability was at its highest at 10 neighbors for the CF-CBF hybrid algorithms. The User-based algorithms were not nearly as affected at low numbers of neighbors. This pattern of interesting results at low neighborhood sizes continues for most of the metrics we tested.

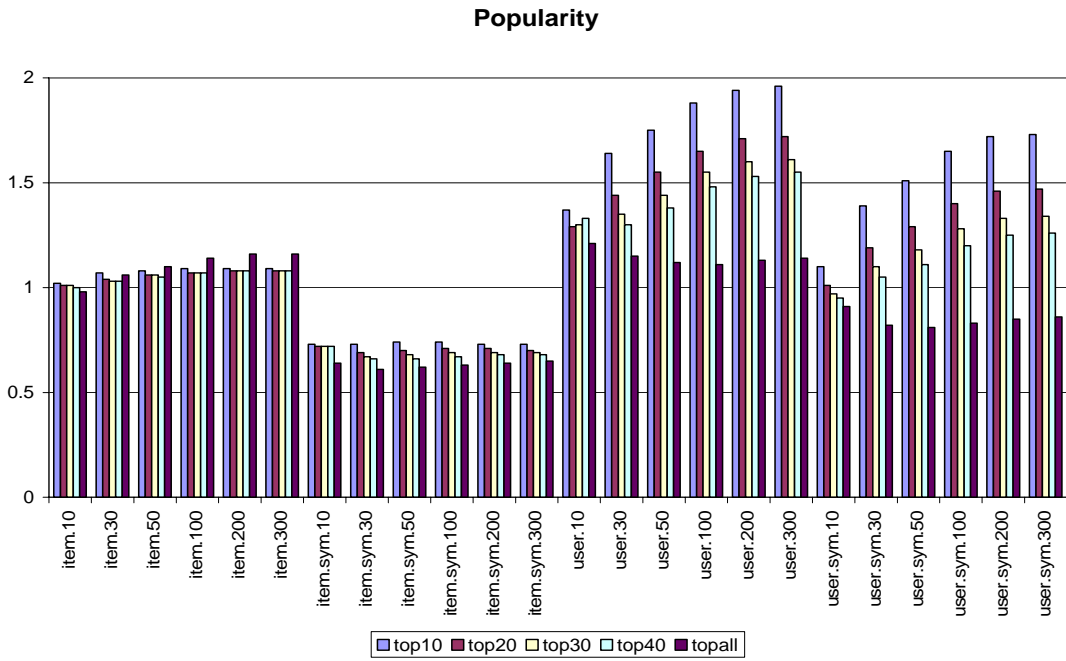


Figure 8-7: Popularity Neighborhood Results

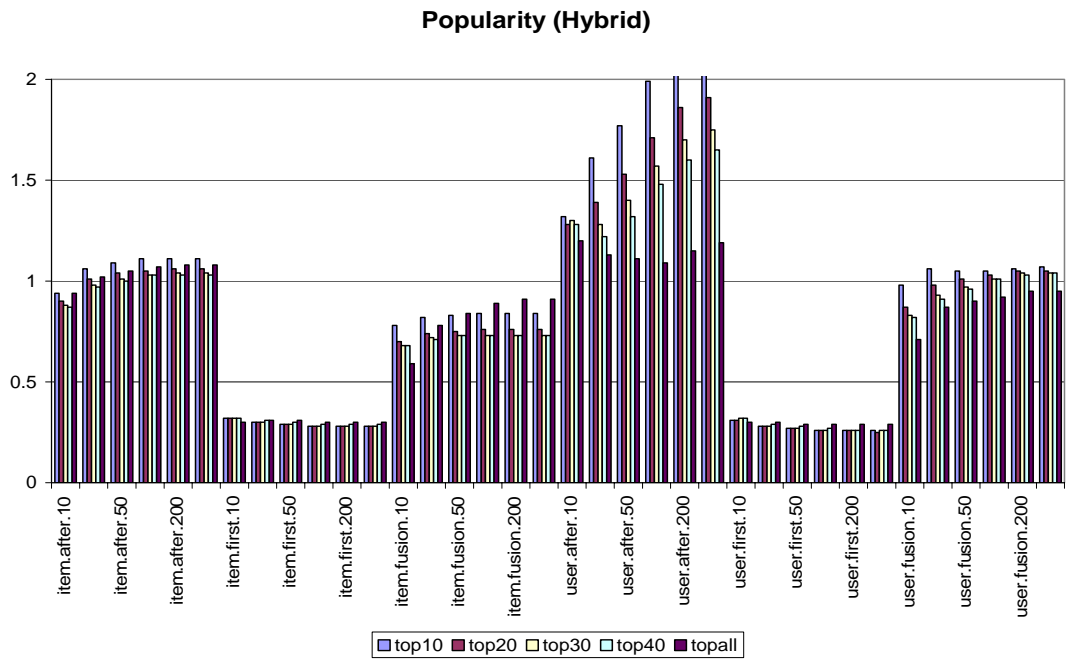


Figure 8-8: Hybrid Popularity Neighborhood Results

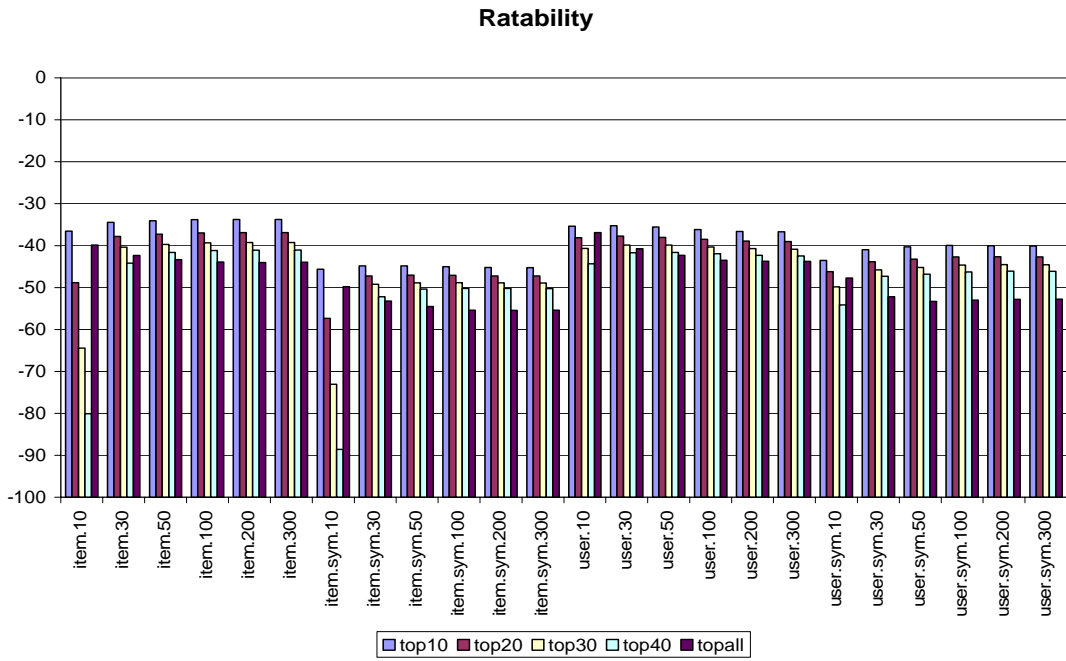


Figure 8-9: Ratability Neighborhood Results

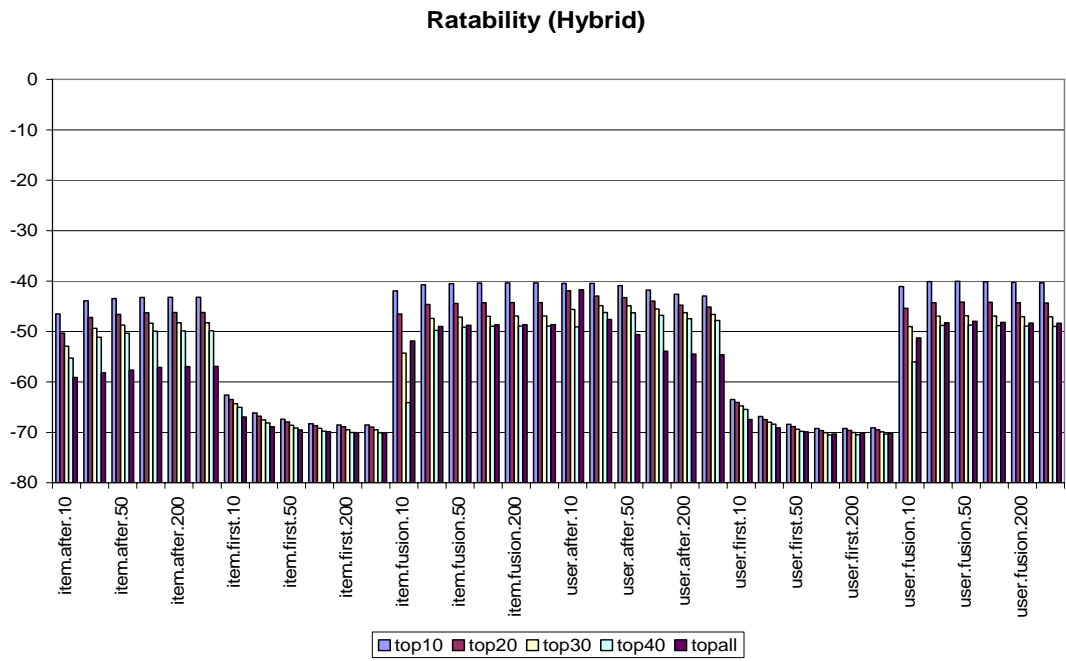


Figure 8-10: Hybrid Ratability Neighborhood Results

There is very little change in Intra-list Similarity due to changes in neighborhood sizes. Both Symmetric CF algorithms remain more similar than all other algorithms. The hybrid algorithms showed slighter higher similarity at 10 neighbors, topping out at 3%. At 10 neighbors, however, all four pure variants score slightly less similar (1%) to the other neighborhood sizes, across all top- n 's.

Following our previous trend, there is a drop in within-comparison personalization for all four pure CF variants at 10 neighbors. These drops are exceptionally significant, falling to below 20% personalized in some cases. These results are on par with the results seen by Content-based Filtering and Localized graph search. All four pure algorithms remain consistent with previous results for all other neighborhood sizes at all top- n 's. Not only do the Fusion hybrid algorithms demonstrate the strange behavior at 10 neighbors, they show levels dropping from a peak at 98% for 30 neighbors down to 85% or worse as neighborhood size increases. The “feature augmentation” algorithms remained relatively constant across all neighborhood sizes.

For Personalization, compare-to-popular, there is strong trending for the User-based algorithms to become more popular as neighborhood size increases. User-based CF at 10 neighbors is around 99%, but at 300 neighbors drops to 95%. This affect is stronger for larger top- n 's. The trend exists for the Item-based algorithms but is not nearly as strong, only reaching 97% at 300 neighbors. While this trend is interesting, it worth remembering the scale of these values—it is highly unlikely users would notice this difference. The Fusion and CBF-CF algorithms all show interesting behavior at 10 neighbors, reverting to much more popular items, especially for User-based CBF-CF and Item-based Fusion.

Figure 8-11 and Figure 8-12 show the Adaptability, Even-Split results. As with Popularity, this metric reveals a significant difference between User-based and Item-based CF: as neighborhoods increase, User-based CF becomes less adaptable, while Item-based CF becomes more adaptable. Eight of the ten algorithms vary in predictable patterns based on top- n size, with User-based algorithms showing a greater sensitivity to

top- n changes. The two CF-CBF algorithms show no adaptability at any neighborhood size. These results persist for both non-even split versions of the Adaptability metric, varying as mentioned above with Adapt-Heavy being more adaptable and Adapt-Light being less adaptable for all algorithms when compared to Equal-Split.

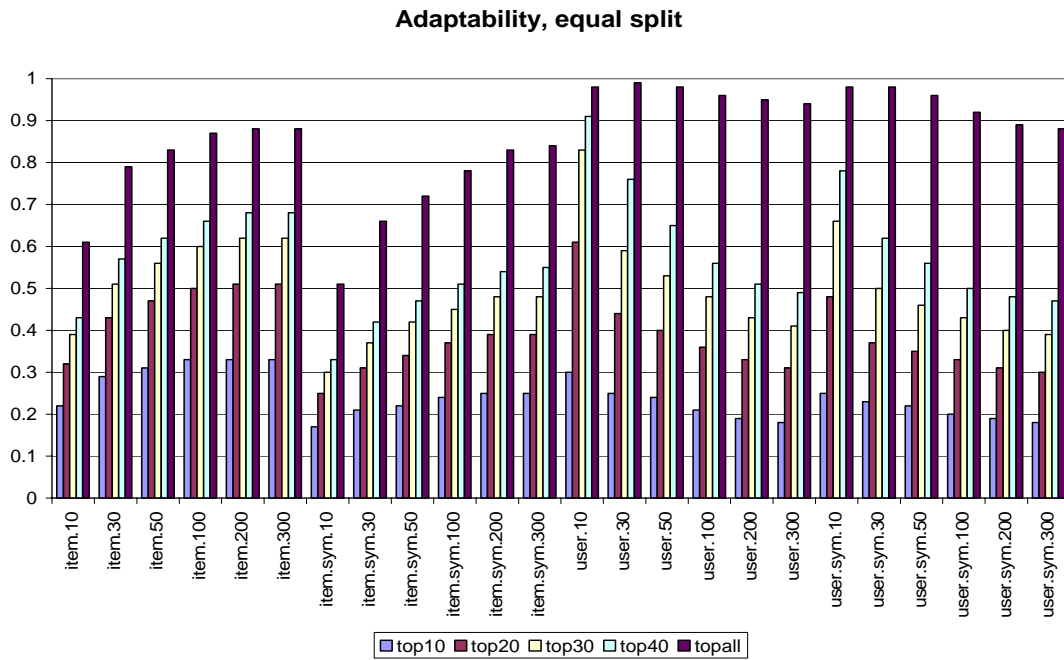


Figure 8-11: Adaptability, Equal-Split Neighborhood Results

Rank scores revealed an interesting trend. At 10 neighbors, all algorithms had the highest Raw Rank score, and as neighborhood size increased, Raw Rank score decreased. At the same time, Coverage at 10 neighbors was at its lowest and increased as neighborhood size increased. The net result of these changes is a small (1-2%) increase in the Adjusted Rank score as neighborhood size increases. This pattern had slight variations between the pure and hybrid algorithms, with the Fusion algorithms both showing a divergent pattern, with rank dropping off at low top- n 's as neighbors increased, but rank increasing for high top- n 's as neighbors increased.

Adaptability, equal split (Hybrid)

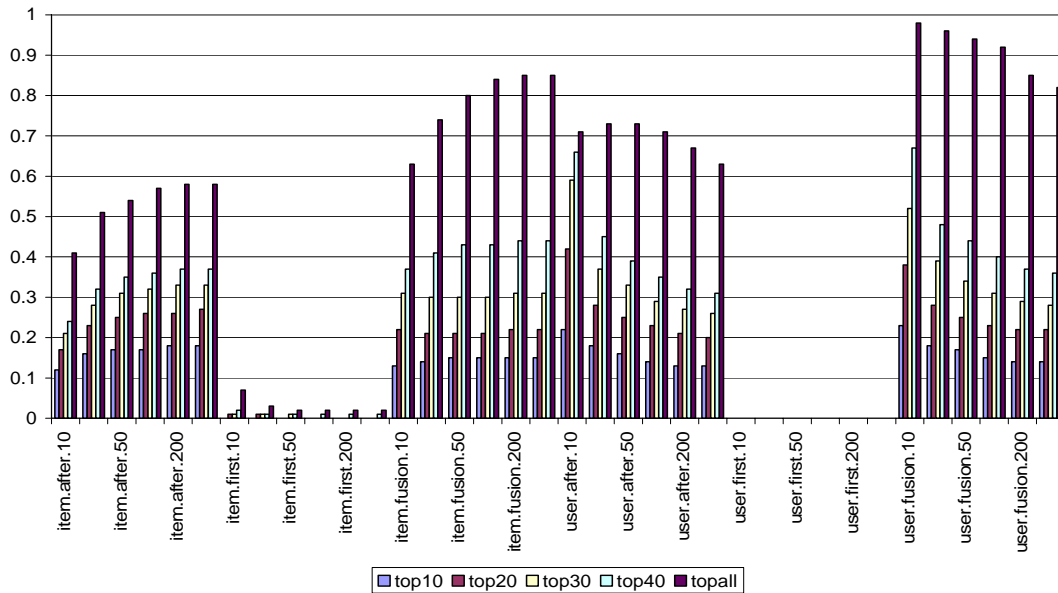


Figure 8-12: Hybrid Adaptability, Equal-Split Neighborhood Results

Experiment Discussion

One of the interesting splits is the division between citation-based recommenders and content-based recommenders. Content-based filtering and Localized Graph Search demonstrated very different behaviors from the other algorithms. This result underscores the limitations of current digital libraries and their content-based search systems. Adding collaborative recommenders will provide a fundamentally different kind of service to digital libraries that currently does not exist.

For example, the content-based recommenders performed quite well on the Raw Rank metric, scoring well above 80% for all top-*n*'s. But their poor performance on the Coverage metric greatly reduced their Adjusted Rank. This differs from collaborative recommenders, especially the Naïve Bayes Classifier, which did not reach 100% coverage only because we limited recommendation lists to 200 citations. Allowed to grow to 100% coverage, the Bayesian classifier scores 72%, User-based CF scores 81%,

and Item-based CF scores 83% Adjusted Rank score at top-30.¹⁰ This ‘cherry-picking’ behavior of content-based recommendations has been noted before [90], but continues to be an important, driving distinction between content and collaborative approaches in this domain.

The hybrid approaches revealed striking differences. The CF-CBF variants generated recommendations that were exceptionally different from all other algorithms. On a closer inspection of this hybrid algorithm, we can understand the reasons for the variation. As a review, in this algorithm an input basket is sent to a CF recommender, the output of which is sent to the CBF algorithm for final recommendations. The problems stem from the nature of CF algorithm recommendations. Their collaborative nature generates interesting and unexpected recommendations. Without the input basket to anchor the recommendation list, a content-based approach on a CF-generated list is worst of both worlds: it first casts the net far and then looks for items closely related to the results. The CBF-CF variant does not suffer from this problem as CBF is allowed to run on the original list first, growing it slightly, before calling the CF recommender. The Fusion algorithm showed interesting results, acting in many instances to ‘temper’ the results of its pure CF variants. This is to be expected as items that both CBF and CF agree on are moved to the top of the recommendation list.

When taking a closer look across the top- n scores over all metrics, an interesting pattern emerges. As top- n grows, metric scores across all algorithms decrease except for Rank and Adaptability, where metric scores increased across all algorithms. Only Item-based CF on the Popularity metric does not follow this trend. This behavior perfectly mirrors the design of the metrics. Both Rank and Adaptability make use of a leave- n -out methodology, thus large list sizes improve the chances of scoring well. The other metrics all measure a property of items on a list. As such, a larger list size leads to a greater potential variation in score.

¹⁰ Collaborative filtering algorithms rarely reach 100% coverage as we have defined it, but they consistently score two to three times higher than content-based filtering scores.

In Chapter 1, we discussed problems with a leave- n -out methodology. As such, this metric could reward algorithms that performed a ‘conservative’ adaptability, recommending items that we know to be a part of the adaptee’s ratings list. A variation would be to compare the recommendation list generated by adapted profile against the recommendation lists for other users in the system. We would expect that as two profiles became similar, so would their recommendation lists, but the correlation function describing this interaction would provide us with an interesting view of the adaptability of the recommender algorithm. This we leave as potential future work.

We are not surprised by the results of our Personalization metrics. All algorithms demonstrated that they generate different recommendation lists for each user and do not return the most popular items. This result could be an artifact of our dataset as well. A possibly more interesting measure would be to calculate the changes in recommendation lists over time or over various parts of the space.

Symmetric Collaborative Filtering is different from traditional CF in this domain: it is less popular, less ratable, and more similar (statistically, not noticeably). While the data suggests the differences are not great, there are other implications of this result. This difference in variants is due to the denser ratings matrix Symmetric CF used. This lends additional evidence to a long held belief: collaborative filtering algorithms behave differently in denser data environments than in sparse environments. Gaining a deeper understanding of this issue will help system designers when selecting algorithms for a particular domain.

The most striking result from the analysis of neighborhood sizes is what we call “The Power of 10”. Collaborative filtering algorithms in small-neighborhood situations seem to be erratic when compared to larger-neighborhood variants. These differences confirm that neighborhood size is an important tuning parameter for collaborative filtering algorithms. It suggests that a recommender should have two versions of a CF algorithm, one with a low neighborhood size and one with a larger neighborhood. This could be used to increase serendipity or provide riskier recommendations to users who want such lists.

Differences between User-based and Item-based CF become apparent when we analyze them at multiple neighborhood sizes. User-based is more volatile as neighborhood sizes change. Moreover, as neighborhoods grow, User-based becomes more popular and less adaptable; Item-based is the opposite, becoming more adaptable as neighborhood sizes increase. Yet both algorithms appear similar in terms of Ratability, Personalization, and Intra-List similarity. The importance of these differences cannot be stressed enough; Item-based CF has been presented as replacement for User-based CF in sparse data environments [130]. These results state that the two algorithms have diverging behavior, there is more than meets the (accuracy metric-trained) eye.

In general, we are confident our results demonstrate these algorithms have unique behaviors even though they all score similarly on accuracy metrics. Armed with this information, we know where to focus our efforts to continue to classify these metrics, but more importantly, we can generate recommendation lists tuned to a user's information seeking task.

HRI and Recommender Algorithms

Much in the same way that the user types and information tasks were mapped to HRI Aspects, these recommender algorithms also contain mappings. Table 8-3 shows the metric mappings. Metrics are linked to Aspects expected score for that Aspect. A plus sign (+) means a high score is linked to the Aspect, where as a minus sign (-), means a low score is linked. For example, Correctness is linked with high Adaptability scores and high Coverage, Rank, and Adjusted Rank scores, while it is linked to low Intra-list Similarity scores.

Table 8-3: HRI Mappings for Recommender Metrics in the Domain of Research Papers

	Popularity	Ratability	Intra-list Similarity	Authority	Personalization	Coverage, Rank, and Adjusted Rank	Boldness	Adaptability
Correctness			-			+		+
Transparency	+			+	-			
Saliency		-			+		+	
Serendipity	-		-	-	+	-		+
Quantity						+		
Usefulness		+				+		+
Spread			-		+	+		
Usability								
Personalization	-			-	+			+
Boldness	-			-			+	
Adaptability			-		+			+
Freshness								
Risk	-			-		-	+	
Affirmation	+	+		+				
Pigeonholing			+		-			-
Trust		+				+	-	

Applying HRI to the Domain of Digital Libraries, Revisited

Returning to Figure 1-2, we have demonstrated how a recommender system designer can user Human-Recommender Interaction theory to analyze users and their information tasks, benchmark recommender algorithms against various metrics, and create mapping to bridge users and their tasks to metrics. We can now user HRI to ‘close the loop’ and choose the most appropriate recommender algorithm(s) for each of our users and their information seeking tasks.

Jill

Jill is a corporate research scientist interested in finding information on ubiquitous computing. Therefore, she is an *Expert in a Related Field* who wants to *Find a Starting Point for Research*.

Her Task HRI aspects are Correctness, Transparency, Serendipity, Quantity, Usefulness, Spread, Personalization, Boldness, Pigeonholing, Trust, Expectations of Usefulness, and Recommender Role in Meeting Need.

The correct algorithm for her would score high on Coverage, high on Adaptability, low on Popularity, high on Ratability, low on Similarity, and high Personalization. We believe the best algorithms for Jill would be either the Naïve Bayes Classifier or Item-based CF at lower neighborhood sizes.

Chris

Chris is a professor looking for the latest and most interesting advances in her area. She is an *Expert in her Field* who wants to *Maintain Awareness*.

Her HRI aspects are Serendipity, Personalization, Saliency, Quantity, Spread, and Freshness.

The correct algorithm for her would score high on Personalization, high on Adaptability, high on Coverage, low on Popularity, low on Similarity, and low on Ratability. We believe the best algorithms for her would be either Symmetric User-based CF at lower neighborhood sizes or Symmetric Item-based CF at higher neighborhood sizes.

Max

Max is a new graduate student wanting citations for a paper he is writing. He is a *Novice in his Field with DL Experience* who wants to *Fill out a Reference List*.

His HRI aspects are Saliency, Serendipity, Quantity, Spread, Adaptability, Trust, Task Concreteness, and Appropriateness.

The correct algorithm for him would score high on Personalization, high on Coverage, low on Similarity, high on Adaptability, and low on Popularity. We believe the best algorithms for him would be either Symmetric Item-based CF at a medium

neighborhood size or Localized Graph Search, depending on how important adaptability is to him.

Discussion and Limitations

The above examples illustrate how the HRI framework can map user information seeking tasks to recommender algorithms through HRI aspects. These mappings come with the same caveats as Chapter 7: they are an example of the possibilities afforded by HRI.

Without performing a detailed task analysis with domain experts and a deep user study, these mappings are a theoretical ideal and not to be treated otherwise. Yet through these mappings, we can see the power of HRI as an intermediate language to bridge user needs and recommender metrics. Finally, these examples show only a snapshot of usage; user tasks will vary over time making sustained system usage an interesting challenge.

Conclusion

Recommender algorithms differ from each other across many dimensions in terms of the recommendation lists they generate. In this chapter, we presented a set of algorithm metrics to benchmark these differences and a series of offline experiments comparing metric results across a variety of pure and hybrid recommender algorithms. Only by defining a set of metrics, each of which explores a different property, can we better understand subtle differences in recommendation lists. Specifically, we found that differences between collaborative filtering and content-based algorithms suggest that the information seeking interfaces to current digital libraries can be enhanced by adding a collaborative filtering-based search interface. Hybrid recommender algorithms can produce a variety of possible results depending on how the component algorithms are combined.

In support of our thesis, we have now presented HRI and the HRI Process Model, presented a new set of recommender metrics, and categorized the differences of several recommender algorithms. In Chapter 9, we take the last step in this process: we run a user study to validate our theories.

CHAPTER 9

UNDERSTANDING RECOMMENDER ALGORITHMS, PART II: A USER EVALUATION

If recommenders are to help people be more productive, they need to support a wide variety of real-world information seeking tasks, such as those found when seeking research papers in a digital library. There are many potential pitfalls, including not knowing what tasks to support, generating recommendations for the wrong task, or even failing to generate any meaningful recommendations whatsoever. Our previous chapters have shown recommender algorithms generate qualitatively different recommendation lists, according to several specific metrics we developed. Building on the offline simulation studies from Chapter 8, we report on a user study of over 130 users on research paper recommendations we generated from the ACM Digital Library using several different recommender algorithms. We asked users about the suitability of these algorithms for different information seeking tasks, as well as for their rating of the recommendation lists across multiple dimensions.¹¹

Our Recommender Algorithms

We focused on four recommender algorithms to cover this domain: three collaborative algorithms and one content-based. We selected User-Based Collaborative Filtering (CF), a Naïve Bayes Classifier, a version of Probabilistic Latent Semantic Analysis, and a textual TF/IDF-based algorithm. A full discussion of these algorithms is available in Chapter 3. We will discuss the relevance of each algorithm to our offline experiments from Chapter 8, and briefly summarize a few important points about each algorithm here. Table 9-1 has a summary of the four algorithms.

¹¹ All research described in this chapter is published in [85].

Table 9-1: Summary of Recommender Algorithm Properties

	Run-time Speed	Pre-Processing Time	Expected Rec. Type	Offline Metric Results
User-User CF	Slow	None	High Serendipity	High quality, our standard
Naïve Bayes Classifier	Fast	Slow	High Ratability	Scored 'higher' than standard
PLSI	Very Fast	Very Slow	"Local" Serendipity	N/A
TF/IDF	Very Slow	Fast	High Similarity	Scored 'lower' than standard

User-based Collaborative Filtering

Collaborative filtering (CF) has been widely used in recommender systems for over 12 years. It relies on opinions to generate recommendations: it assumes people similar to you also have similar opinions on items as you do, thus their opinions are recommendations to you. The most well known algorithm is User-based collaborative filtering (a.k.a. User-User CF, or Resnick’s algorithm), a k -nearest neighbor recommendation algorithm [55, 120]. To generate recommendations for a target user, a neighborhood of k similar users is calculated, usually using Pearson correlation, and the highest weighted-average opinions of the neighborhood are returned as a recommendation list.

User-based collaborative filtering has many positives and negatives as a recommender algorithm. It is a well-known and studied algorithm, and it has typically been among the most accurate predictors of user ratings. Because of this, it can be considered the “gold standard” of recommender algorithms. Conceptually, it is easy to understand and easy to implement. It has a fast and efficient startup, but can be slow during run-time, especially over sparse datasets. Choice of neighborhood sizes and similarity metrics can affect coverage and recommendation quality. It is felt in the community that User-User can generate ‘serendipitous recommendations’ because of the mixing of users across the dataset, but the structure of the dataset greatly affects the algorithm—it is possible that for some users, a User-based CF algorithm will never generate high quality recommendations.

In Chapter 8, we found that User-based CF performed quite consistently across all of our metrics, with strong changes in adaptability and excellent personalization scores. Just as User-based CF is considered a ‘gold standard’ in the recommender community, we also consider it a gold standard in our benchmarking results. We hope to compare the results of our three other algorithms against this algorithm.

Based on results of previous work, we expect User-based CF to perform very well generating research paper recommendations. Since we are guaranteed each item in a digital library rates other items (all papers cite other papers!), we expect good level of ‘cross-pollination’ for serendipitous recommendations as well as high coverage. While the size of ACM Digital Library is not overwhelmingly large, issues of scale and sparsity could become relevant if User CF is applied to larger databases, such as PubMed medical paper database from NIH [102].

Naïve Bayes Classifier

If we assume independence of co-citation events in a research paper, then a Naïve Bayes Classifier [13] can be used to generate citation recommendations. All co-citations pairs are positive training examples. Items (citations) that are strongly classified as belonging to the target paper class are returned as recommendations (usually a top- n list).

A Bayes Classifier also has many pros and cons. It is a well-known and established algorithm in the machine learning literature. The independence assumption between co-citation pairs is a large one to make, but even in domains where this assumption fails, this classifier still performs quite well [34]. The classifier requires processing time to create a model before generating recommendations, and this model must be re-built when new data is entered. But run-time is quick.

Because at heart, a Naïve Bayesian Classifier is a classification algorithm, we have specific expectations for it. Classifiers calculate ‘most likely events’; they determine what classes items belong to. From a user’s point of view in a recommender, a classifier returns the next most likely item the user will rate. This ‘ratability’ property of classifiers may not match a user’s expectations in a recommender. As such, we believe

that the recommendations would be more “mainstream”, and not as serendipitous as User-based CF. Finally, full coverage could make a difference in low-data situations.

Our results from the offline benchmarking show that Naïve Bayes has extremely high ratability, as could be expected given our operational, but also is very adaptable and does not generate overly popular recommendation lists; in general is a very good recommendation algorithm with small changes from User-based CF. We are interested in to seeing how well users perceive the differences between these algorithms.

Probabilistic Latent Semantic Indexing

Probabilistic Latent Semantic Indexing (PLSI) is a relatively new algorithm to be applied to recommender systems [59]. PLSI is a probabilistic dimensional reduction algorithm with a strong mathematical foundation. In PLSI, the ratings space (citation space) is modeled using a set of independent latent classes. A user (i.e. paper) can have probabilistic memberships in multiple classes. PLSI uses a variant of the EM algorithm to optimize the class conditional parameters through a variational probability distribution for each rating (citing) instance based on previous model parameters. Items (citations) with the highest probabilities relative to the latent classes are recommended.

Similar in spirit to the Bayesian classifier, PLSI also has similar benefits and drawbacks. It is mathematically rigorous, using latent classes to find probabilistic relationships between items. Model creation takes a very long time, and requiring exceptional computing resources, but runtime is very efficient. It also will have 100% coverage. The latent classes are the key feature of this algorithm, thus performance and quality are highly dependent on the number of latent classes used.

This is a relatively new algorithm in this domain. We believe the latent aspect of this algorithm makes it more like User-based CF being able to generate interesting and serendipitous recommendations. By performing local maximizations, the EM nature of the algorithm could reinforce more “popular” connections between items at the expense of “further reaching” (and potentially more interesting) connections, thus recommendations could be interesting locally, finding unexpected papers closely related to the input, but not interesting for finding a broad set of items from across the dataset.

PLSI was not studied in our offline experiments in Chapter 8. While PLSI shows great potential as a recommender algorithm, due its extremely slow startup processing time we did not perform the full set of offline benchmarking experiments. Our limited experimentation shows that PLSI is similar to User-based CF and Naïve Bayes Classifier, thus we expect user responses to follow according. We include it in these online experiments to see how users respond to this, one of the newest recommender algorithms proposed. Depending on the results here, we hope to benchmark this algorithm with our metrics in the future.

Content-based Filtering with TF/IDF

Content-based filtering for papers uses the full text to generate recommendations. One of the most popular and well-used content filtering algorithms is Porter-stemmed Term Frequency/Inverse Document Frequency (TF/IDF) [114]. By using the frequency that stemmed words appear in a document compared to the entire corpus, this algorithm recommends items based on how well they match on important keywords.

TF/IDF is well known in the information retrieval community, and is considered as the standard vector-space model (a.k.a. “bag of words”) approach to information processing. Since all papers contain content, they can be immediately processed, and thus recommended. This compares to the collaborative methods where the strength of the citations can unduly influence recommendations towards older, more cited works. Yet, “bag of words” content analysis is limited by semantic differences (e.g. “car” is not equated to “automobile”). It also may return many irrelevant results, recommending papers who mention the search terms only in passing. These issues are not as important in scientific literature as papers in one area have a generally agreed-upon vocabulary. Finally, TF/IDF does have an associated cost in pre-processing time and storage space, like other model-based algorithms.

Prior results show that when Content-based Filtering works, it performs very well at generating highly similar results. But more often than not, it fails to generate relevant results. This cherry-picking behavior limits its usefulness, especially when searching for novel or obscure work. On the other hand, this algorithm may excel at more conservative

user tasks, especially those that start with a fair amount of information (e.g. “I know many of the conference papers in this research area, but I don’t know about journal papers or workshops”). In general, we expect it to generate a narrow, yet predictable kind of recommendation list.

In Chapter 8, we found Content-based filtering to generate very different results from the collaborative filtering approaches, scoring quite low on both popularity and ratability, as well as indicating that it is not a very adaptable algorithm either. Because of these strong differences in the offline metrics, we expect there to be strong, noticeable differences in user opinion of this algorithm compared to the other three. While the differences between the three other algorithms may be subtle, we expect the differences to this algorithm to be as strong as the results from our offline analysis.

User Study

Our previous experiments in this dissertation suggest that recommender algorithms behave differently from each other on a variety of metrics. In this study, we tackle the following research questions:

1. How will real users describe the recommendations generated by different recommender algorithms?
2. How will users rate the ability of recommender algorithms to meet the needs of specific user tasks?
3. How do the differences in user opinion of recommender algorithms compare to the differences between algorithms we cataloged in our offline simulation experiments from Chapter 8?

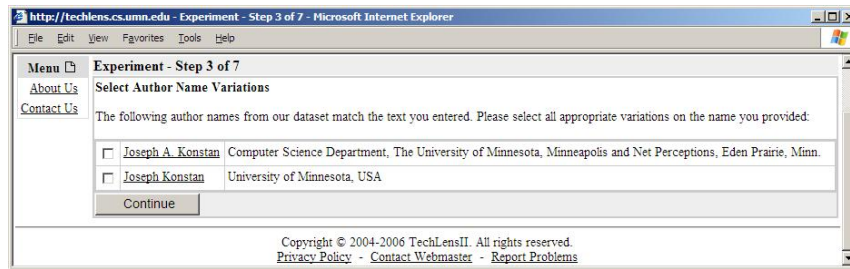


Figure 9-1: The Author Selection Page

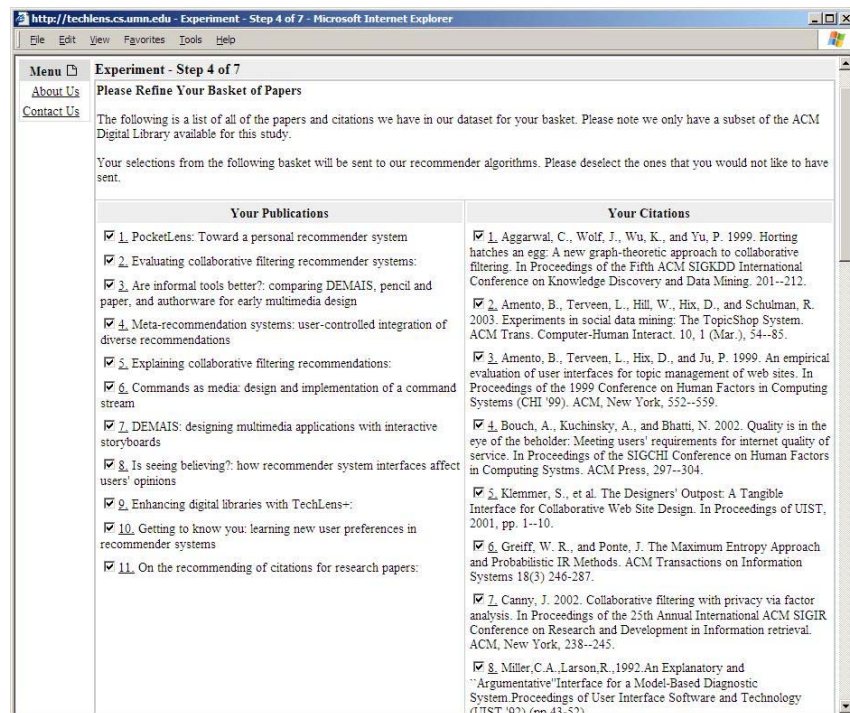


Figure 9-2: The Paper and Citation Selection Page

Experimental Design

In the experiment, we generated two recommendation lists of five items each, called ‘List A’ and ‘List B’. These recommendation lists were based on a ‘basket’ of papers that the user provided to us. Users were randomly assigned two algorithms; the display of the lists was counterbalanced to cancel any possible order effects. Algorithm recommendation lists were pre-screened to make sure there was not strong overlap between result lists. At most, lists had 2 out of 5 recommendations in common. Since order is important in the returned results, if there was overlap, the list that has the shared

result ‘higher up on the list’ retained the item. In the rare event of a tie, List A always won.

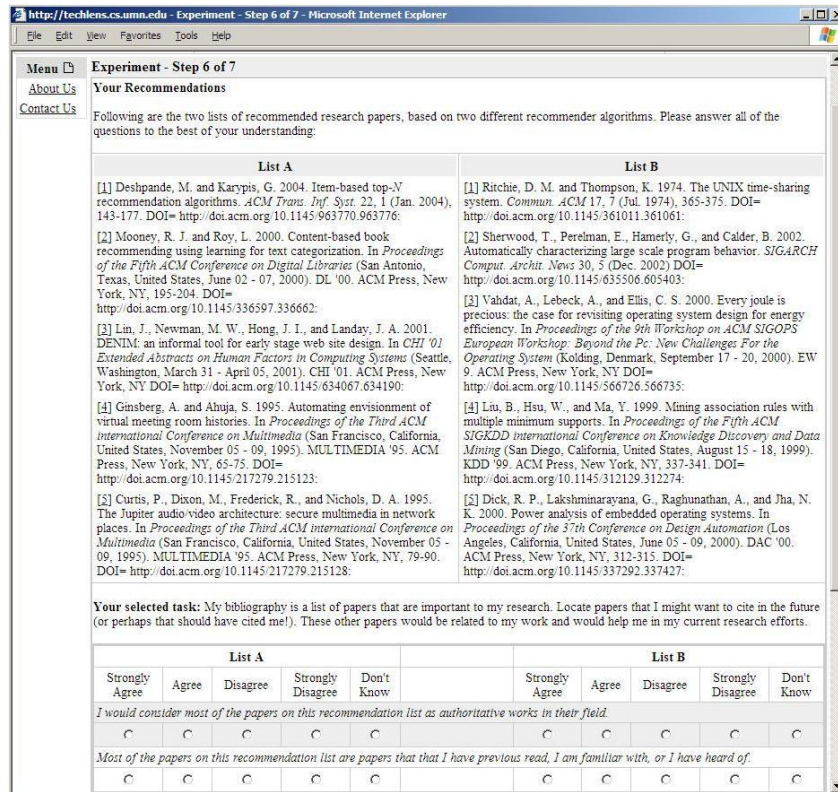


Figure 9-3: The Recommendation List Screen

Our dataset is based on a snapshot of the ACM Digital Library, containing over 24,000 papers. For each paper, we have text of the paper’s abstract; each paper cites at least two other papers in the dataset and is cited by at least two other papers in the dataset. We used the same dataset for our experiments in Chapter 8.

Users were selected from several different pools. A subject either had to be an author of papers appearing in the dataset or be “very familiar” with the literature in a particular research area. Users were mined from the DL itself: a random selection of users who authored more than five papers in the dataset were sent an email invitation to participate. We also sent invitations to several computer science mailing lists asking for participants.

As a summary, the following algorithms were used in the online experiment:

1. User-based collaborative filtering at 50 neighbors, we used the SUGGEST Recommender engine as our implementation [67]
2. Porter-stemmed TF/IDF content-based filtering over paper titles, authors, keywords, and abstracts; we used the Bow Toolkit as our implementation [80]
3. Naïve Bayes Classifier; we used the same custom implementation as used in Chapter 8, which is itself an optimized version of the implementation we used in Chapter 5
4. Probabilistic Latent Semantic Indexing with 1000 classes, we used a custom implementation¹²

Table 9-2: Available Information Seeking Tasks

Group	Available Tasks
Author, with my own work	My bibliography is a list of papers that are important to my research. Locate papers that I might want to cite in the future (or perhaps that should have cited me!). These other papers would be related to my work and would help me in my current research efforts.
	Given my bibliography as a description of what I find interesting in research, locate papers I would find interesting to read. These papers would not necessarily be work in my area, and could suggest potential collaborators or new research questions to explore in the future.
Someone else's publications	You would like to build a list of papers to describe a research area, and you started with this input basket. You are concerned about covering the area and want to make sure you do not miss any important papers. Which papers would next appear on this list?
	You are looking to expand your research interests into new or related areas, hoping to find interesting papers to read. Your input basket represents some papers in your current research area and interests. What papers might expand your knowledge, find new collaborators, or suggest new research areas to explore in the future?

Experiment Walkthrough

After consenting, subjects were asked as to their status as a researcher/academic (novice, expert, or outside the field of computer science) and as to their familiarity with the ACM

¹² Shilad Sen implemented the version of PLSI used in this chapter.

Digital Library. Next, subjects were asked to create their ‘basket’, a list of papers to be sent to the recommender engines. This list is seeded by the name of an author who has published in the ACM Digital Library. There were options for selecting ‘yourself’ as an author or selecting ‘someone else’. Figure 9-1 shows the author selection page.

After confirming the author selection, the subject was presented with a list of papers by that author in our dataset. The subject was allowed to prune any papers he did not want in the basket. If the subject stated he was an author himself, he saw a listing of papers he had written as well as a listing of papers he had cited from our dataset. If the subject selected another author, he was only presented with the listing of papers that author had published in our dataset. This decision had great implications on our results, as we will discuss later.

After pruning was finished, we presented the user with a selection of two possible information-seeking tasks, see Table 9-2. After selecting a task, the subject was presented with two recommendation lists—see Figure 9-2 and Figure 9-3 for an overview of the paper selection and recommendation interfaces. There were two pages of questions associated with the recommendation lists. On the first page, we asked comparative questions: user opinion about the kinds of papers appearing on the lists. On the second page, we asked task-related questions: rating the suitability of each list to meet the subject’s chosen information seeking task.

A summary of all questions is in Table 9-3. On the first page, there were five possible responses to each question: ‘Strongly Agree’, ‘Agree’, ‘Disagree’, ‘Strongly Disagree’, and ‘Not Sure’. Question 2-1, asking which list was better for the chosen task, had two options: ‘List A’ and ‘List B’. We forced users to choose between the lists. In Question 2-2, we ask for the user’s satisfaction with that selection, and responses ranged from ‘Very Satisfied’ to ‘Very Dissatisfied’, with a ‘Not Sure’ option. Question 2-3 was a hypothetical question, asking users to ponder the better list for the alternate task, and it had ‘List A’, ‘List B’, and ‘Not Sure’ as its possible responses. Note that for the hypothetical question we allowed a ‘Not Sure’ response. In Question 2-4, we offered users an option to receive a copy of the citations we generated for them, asking which

citations they would prefer: ‘List A Only’, ‘List B Only’, ‘Both Lists’, or ‘Neither List’. Finally, in question 2-5, we asked them which format they would prefer: ‘the ACM DL BibTex Format’ or the ‘Standard ACM Citation Format’. While at first glance, Questions 2-4 and 2-5 appear to be a thank-you service for users participating in our study, we wanted to study the difference between stating they are satisfied with a recommendation list, and choosing to receive a copy of the list for future use—the difference between action and words. We believe action is a much stronger statement of satisfaction.

Table 9-3: Summary of All Survey Questions

Page 1: Comparative Questions	
1-1	I would consider most of the papers on this recommendation list as authoritative works in their field.
1-2	Most of the papers on this recommendation list are papers that that I have previous read, I am familiar with, or I have heard of.
1-3	This recommendation list is closely tuned-tailored- personalized to my given input basket, and is not a generic list of papers in this research area.
1-4	This list feels like a good recommendation list. I like it. It resonates with me.
1-5	This recommendation list contains papers that I was not expecting to see, but are good recommendations considering my input basket.
1-6	This list contains a good spread of papers from this research area and is not overly specialized, given the input basket.
Page 2: Task-related Questions	
2-1	In your opinion, which recommendation list generated a better set of recommendations for you and your task?
2-2	How satisfied are you with the recommendations in the list you preferred?
2-3	Pretend you were going to perform the alternate task, the one you did not choose (see above). In your opinion, which recommendation list generated a better set of recommendations for this other task?
2-4	We have the ability to send you a text file containing the standard ACM citation format for these recommendation lists. Would you like to keep a copy of these recommendation lists?
2-5	If so, which format would you prefer?
Note: For referencing, questions may be referred by number or by the bolded text in the question. No words were bold in the online survey.	

Results

134 subjects completed the survey over the 3-week experimental period. Each subject provided answers for two algorithms in parallel. The display of the two algorithms was counter-balanced to create twelve experimental conditions. The number of user per algorithm-pair is shown in Table 9-4.

Table 9-4: Number of Users per Experimental Condition

Algorithm Pair	Users
User-User CF and TF/IDF	25
User-User CF and Naïve Bayesian	29
User-User CF and PLSI	19
TF/IDF and Naïve Bayesian	26
TF/IDF and PLSI	18
Naïve Bayesian and PLSI	17

Results by Survey Question

Figure 9-4 and Figure 9-5 summarize the results for the six questions on the first page. Answers from all twelve experimental conditions have been binned into three categories: ‘Agree’, ‘Disagree’, and ‘Not Sure’. As the figures show, there is a dramatic difference in user opinion about the four algorithms. Users tended to like (‘agree’ with) User CF and TF/IDF, where as users tended to dislike (‘disagree’ with) Bayes and PLSI. Results between these pairs are exceptionally significant ($p \ll 0.01$). We have found a pitfall; we discuss these unusual results in a separate in-depth analysis.

Focusing on User CF and TF/IDF, differences are statistically significant Question 1-2 ($p < 0.01$) and almost so for Question 1-1 ($p = 0.11$). User CF is more authoritative and generates more familiar results than TF/IDF. The trends on the other four questions also suggest that User CF generates recommendations that are more ‘interesting’.

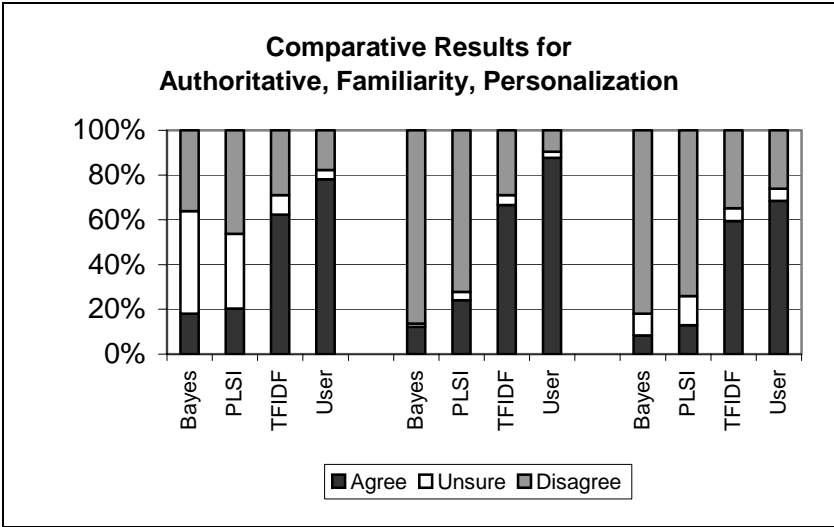


Figure 9-4: Results for First Three Survey Questions

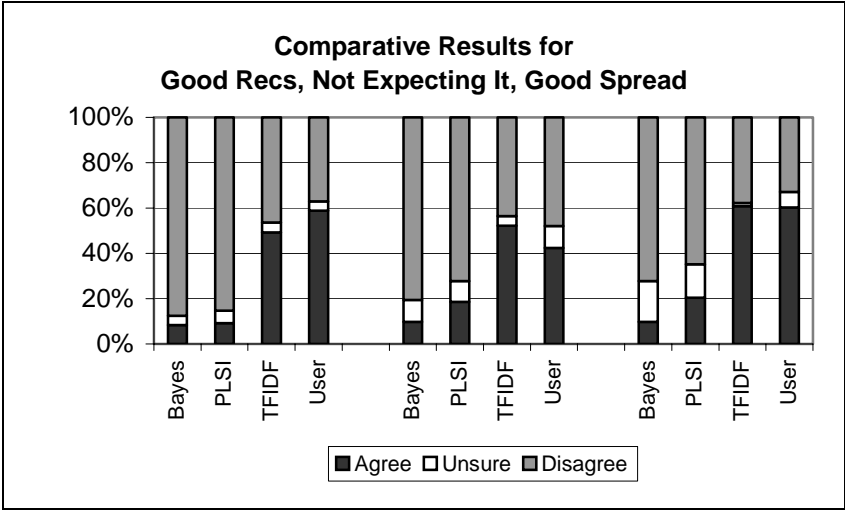


Figure 9-5: Results for Second Three Survey Questions

There were two kinds of information-seeking tasks: “Find closely related papers”, and “find more distant partnerships”. Users were required to select one of these tasks before receiving recommendations. Figure 9-6 shows how users judged the suitability of each of the four algorithms to their chosen task. Please note: users were forced to answer Question 2-1; there was no ‘Not Sure’ option. Continuing the above trend, users chose User CF and TF/IDF over Bayes and PLSI at about a 3:1 ratio ($p < 0.01$). Question 2-2

asked users about how satisfied they were with the algorithm they chose. Users were not pleased with Bayes or PLSI (all results for Bayes were ‘Disagree’!), and were happy (‘agree’) with User CF and TF/IDF just over half of the time.

Question 2-3 asked which algorithm would be best for the task the user did not select. When asked about this alternate task, 56% chose the same algorithm, 16% chose the other algorithm, and 28% were not sure. This trend carried across all four algorithms, except for Bayes, where ‘Not Sure’ was selected 50% of the time.

The final questions asked users if they wanted to keep a copy of the recommendations. While offered as a service, it provides an insight into user satisfaction. 32% of all users elected to ‘keep a copy’. 67% of satisfied users chose to while only one user who was ‘dissatisfied’ chose to. Finally, users chose to keep recommendations generated from User CF and TF/IDF over Bayes and PLSI again at a 3:1 ratio ($p < 0.01$).

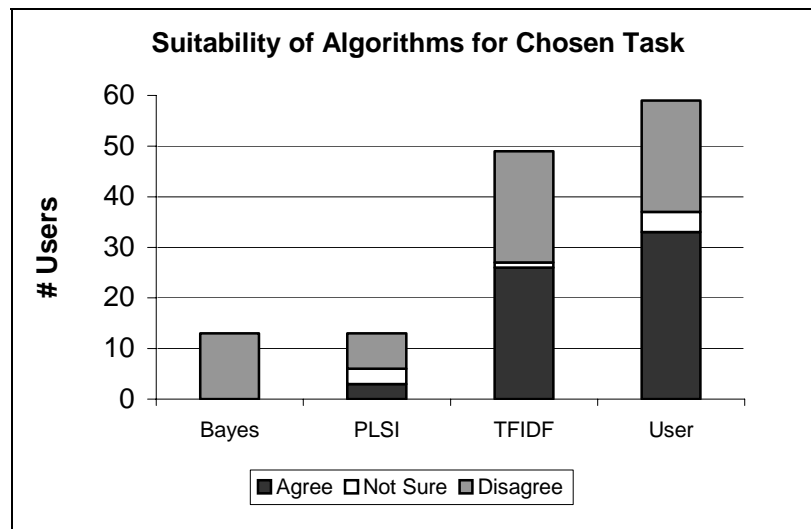


Figure 9-6: User Opinion of Suitability for Chosen Task

Results across Algorithm Pairs

Users answered all questions in the context of a pair of algorithms. The comparative tasks showed minor levels of interaction. When either User CF or TF/IDF was paired with Bayes or PLSI, users tended to score algorithms towards the extremes, significantly for questions 1-2 (“familiarity”), 1-3 (“personalized”), and 1-4 (“good recommendation

list”) . There were no discernable effects when User CF was paired with TF/IDF or when Bayes was paired with PLSI.

When selecting tasks, User CF and TF/IDF dominated over Bayes and PLSI. When shown together, User CF was selected 90% of the time over Bayes and 95% over PLSI. TF/IDF was selected 88% and 94%, respectively. Compared against each other, User CF was selected more frequently (60%) than TF/IDF. Bayes and PLSI were preferred an equal number of times when placed together. More interestingly, when paired against Bayes and PLSI, User CF received higher praise, earning all of its ‘Very Satisfied’ scores in these cases. TF/IDF saw no such increase.

When asked for algorithm preferences for the alternate task, users rarely switched algorithms. Of the switches between algorithms, users switched to PLSI 9% of the time, and switched to Bayes 20% of the time. Users who first chose PLSI or Bayes always switched to either User CF or TF/IDF. Between User CF and TF/IDF, users were twice as likely to switch from User CF to TF/IDF as vice-versa.

Analysis and Discussion

Before discussing the results of this study, we first perform a closer analysis of the atypical results we found and review our results for any potential order bias.

Analysis of Atypical Results

The Bayes and PLSI algorithms did not perform as expected. Prior work suggests that a Naïve Bayes Classifier should be similar to User-based CF in this domain [90], and PLSI has shown to be of high quality in other domains [59]. What happened?

A careful review of our log files reveals that Bayes was generating similar recommendation lists for all users. Looking at top-10 recommendations lists, on average 20% of the recommended papers were identical for all users. Changing the input basket size did not have an effect: see Table 9-5. In the table, we computed an overlap score for our recommender algorithms and binned results into different basket sizes. Our score was the average of the intersection/union for all possible pairs between top-10 lists.

As shown in the table, instead of returning personalized recommendations, Bayes returned the most highly co-cited items in the dataset.

Table 9-5: Overlap of Top-10 Recommendation Lists, by Basket Size

	< 5	5 - 15	15 - 30	30+
Bayes	0.22	0.36	0.20	0.04
PLSI	0.01	0.01	0.002	0.004
TF/IDF	0.002	0	0	0.03
User	0	0	0.001	0.04

PLSI was doing something equally as odd. While it returned personalized recommendations, a random sampling from logs revealed seemingly nonsensical recommendations. For example, a basket composed of CHI papers received recommendations for operating system queuing analysis. Once again, our analysis did not indicate any changes in behavior based on basket size.

In this work, the input basket size varied greatly (average of 26 items, stddev 32.6), with many baskets containing five or fewer items. This variation came from our decision to add citations into the active basket for users who were authors in our dataset but only provide papers authored by a given name for users who said they were not an author. Yet, this variation is not a flaw in our design; it was grounded in real differences between users—novices will not know many papers. It provides us with unexpected, but extremely important results. Much as search engines need to return relevant results with little input [136], recommenders need to adapt to a variety of user models. User CF and TF/IDF we able to do so, Bayes and PLSI were not.

Yet, it is not accurate to say those algorithms ‘failed’—some users did receive high quality recommendations. Rather, Bayes and PLSI were *inconsistent*. While Bayes averaged an overlap score of 0.20, the standard deviation was 0.26. We believe the quality of the recommendations generated by these two algorithms depended on how connected items in the input basket were to the rest of the dataset, especially for Bayes. Performing some additional testing we found that adding one well-cited item to a

questionable Bayesian basket would radically improve the results. Adding such items to PLSI was not always as helpful.

Order Bias

In this experiment, all users were exposed to two algorithms. The experiment was cross-balanced, so each algorithm appeared an equal number of times as List A and List B. With this balance, we were able to detect an order bias, where people preferred List A (always appearing on the left) to List B (always appearing on the right). Specifically, equal numbers of people selected List A and List B as their preferred list (the answer to Question 2-1). But looking at the answer to Question 2-2, those who chose List B were less satisfied with their selection ($p < 0.10$), see Table 9-6.

Table 9-6: Order Effect for Question 2-2

	Satisfied	Dissatisfied	Not Sure
List A	38	27	4
List B	24	37	4

Comparative and User Task Analysis

The atypical results of the two recommender algorithms have skewed the results of the survey, making a detailed comparative analysis difficult. For User CF and TF/IDF, users recognized the recommendations they received and deemed them as authoritative items in their research area, but users felt the lists did not contain unexpected results, and users were not sure if the recommendations came from a wide spread of papers in the dataset. The differences between User CF and all other algorithms for authority and familiarity were significant.

These results are different from our results in Chapter 5 where User CF generated more unexpected recommendations than TF/IDF. We do reinforce previous results that User CF generates authoritative paper recommendations. Further, Chapter 5 suggested that TF/IDF had a higher level of user satisfaction, whereas here, both algorithms

received positive scores. Of course, the interaction effects may have influenced these responses, artificially boosting the scores, especially for User CF.

TF/IDF showed to be equally as useful for all four given user tasks with around a 54% satisfaction rating for all tasks. User CF showed a higher satisfaction rating (60%) for the ‘find closely related papers’ task. These results also reinforce findings from Chapter 5, but have to be taken in context of the possible interaction effects.

Finally, as we compare these results to our metric analysis from Chapter 8, and we reflect on how what these results mean for HRI, we find several interesting points. First, TF/IDF and User-based CF scored quite differently on the Popularity, Ratability, and Adaptability metrics. Here, there were differences in authority and familiarity, but no other strong differences between the two algorithms, at least not as strong as we were expecting based on the simulation results. Second, while Bayes and PLSI were atypical, they were atypical in different ways. Bayes mixed personalized and non-personalized results together, whereas PLSI generate poor personalized recommendations. While both approaches are poor, the trend data suggests that the Bayes approach is worse. Third, there were some interaction effects between poor and good algorithms, with User-based CF benefiting most from this mismatching. There are several possibilities for this in terms of HRI that we discuss below.

Dataset Limitations

While our dataset was of high quality, it contained several limitations. The two striking ones are the scope of the data, and range of the data. Only items for which the ACM holds copyright were contained in this dataset, many relevant papers were not included. Due to the nature of the DL, the bulk of items were published in the last 15 years. Finally, items had to cite and be cited by other items in the dataset. This limitation excluded much of the newest published work. Further, we received several emails from users complaining about these limitations, including requests to add their work to our dataset, statements that they have changed research fields, and concerns for only receiving recommendations for older papers.

Further, these limitations may have affected PLSI's and the Naïve Bayesian Classifier's performance in the study. While all papers were connected to each other, many were only weakly so. Both of these algorithms use strong statistical models as a basis for their recommendations. When calculated over a dataset such as this one, the meaningful statistics could be 'washed out' by the other calculations. In Bayes, a paper with a strong prior probability could influence posterior probabilities, or perhaps the naïve assumption was one we should not make in this domain. In PLSI, the local maximization calculations could reinforce stronger global calculations in place of the weaker, yet meaningful local connections. In other words, PLSI reinforced poor latent classes. Finally, it is worth asking the question if measuring co-citations is the correct mapping for recommenders in this domain. A complete analysis and discussion, however, would be considerable and excellent future work.

Implications and Future Work

As previously mentioned, Bayes and PLSI perform well as recommenders in offline simulation experiments. Specifically, both have scored well on accuracy metrics in using a leave- n -out methodology [13, 59]. As we have argued in this dissertation and elsewhere (i.e. [86, 87]) such offline measures may not translate into useful measures for end users.

In particular, Bayes recommended users a combination of personalized and non-personalized recommendations. In many cases, the personalized recommendations were good suggestions. In fact, in our offline leave- n -out scenario from Chapter 8, this algorithm scored very well. It did not matter that the recommendation lists contained a mixture of highly personalized and non-personalized items, as long as the withheld item appeared on the list, the algorithm scored well on the metric. Users, however, were not satisfied with these recommendation lists. These results suggest that the dependence on offline experiments have created a disconnect between algorithms that score well on accuracy metrics and algorithms that users will find useful.

This argument is even more subtle than stated before. In Chapter 8, we also performed an analysis of the recommendation lists generated by different algorithms. If the Naïve Bayes Classifier were generating a mixture of personalized and non-personalized results, the personalization metrics should have revealed this problem. In our results, Bayes generated very personalized responses. The difference was in the input baskets. The baskets from Chapter 8 were based on the citations from a paper, usually such lists contain a mixture of well and loosely connected papers, lists for which Bayes could generate completely personalized recommendations. The input baskets in our online experiment were very different, revealing not just the importance of the dataset but also the importance of the input basket when analyzing algorithms. The disconnect is even larger than we thought.

Earlier in this dissertation, we argued that showing one good recommendation in a list of five was enough to satisfy users. It is not that simple: showing one horrible recommendation in five is enough for users to lose confidence in the recommender. We call this the *Don't Look Stupid* principle: only show recommendation lists to users when you have some confidence in their usefulness. While this principle applies most dramatically when talking about Bayes and PLSI in our results, we believe it is just as important when dealing with users' information seeking tasks. A recommendation list is bad when it is not useful to the user, independent of why it is bad.

To understand this principle in context, we can use HRI. This experiment was one-time online user survey. These users had no previous experience with the recommender algorithms, and they were given an information seeking task. Because of this, many HRI Aspects are not relevant to our discussion, but a few become very important, such as: Correctness, Saliency, Trust, and Expectations of Usefulness. By being asked to be in an experiment, users had a heightened awareness of the recommendation algorithms; they expected the algorithms to be useful and they expected them to generate correct results. Indeed, we received several emails from users worried about the poor recommendations they received from either Bayes or PLSI. We had no time to build trust with our users, nor did the users gain a sense of the algorithms

personality. Because of this, when users received nonsensical results, we believe they had a strong emotional reaction to these results. The results went against their expectations of being an experiment to receive personalized recommendations. Thus, the users provided the strong negative feedback.

There many threads of possible future work. First, we need a deeper understanding of Bayes and PLSI in this domain. How much of the difficulties experienced in this work are related to properties of the algorithms, properties of the dataset and input baskets, and implications of how these algorithms were applied in this domain. Second, while this study provides evidence to the tenet of HRI that specific recommender algorithms are better suited to certain information needs, more work needs to be done. Yet it does raise one interested question from our HRI analysis, could a recommender be ‘stupid’ in front a user with whom the recommender has already built a relationship? This work must be done with real users; offline analysis is not enough. Finally, the performance of Bayes and PLSI in this domain suggest that dataset properties and input basket selection greatly influences recommendation lists, this implies the need for a study comparing multiple datasets across multiple algorithms.

Conclusion

Recommending research papers in a digital library environment can help researchers become more productive. Human-Recommender Interaction argues that recommenders need to be approached from a user-centric perspective in order to remain relevant, useful, and effective in both this and other domains. HRI suggests tailoring a recommender to a user’s information need is a way to this end. To test these ideas, we ran a study of 134 users using four recommender algorithms over the ACM Digital Library. Instead of validating our research questions, we ran into a large pitfall and discovered a more telling result: *Don’t Look Stupid*. Recommenders that generate nonsensical results were not liked by users, even when the nonsensical recommendations were intermixed with meaningful results. These results suggest that it is critically important to select the correct recommender algorithm for the domain and users’ information seeking tasks.

HRI analysis shows how violating a few Aspects can severely affect a new user's perception of a recommender system. Further, recommender algorithm evaluation must be done with real users, as current accuracy metrics run on datasets with pre-selected input baskets cannot detect these problems.

CHAPTER 10

IMPLICATIONS, FUTURE WORK, AND CONCLUSIONS

In this, the last chapter of this dissertation, we summarize our contributions, discuss the implications of our work, and reflect on possible avenues for future work.

Summary of Contributions and Implications

Here we summarize our key findings from this dissertation and discuss potential implications.

Recommending Research Papers

By mining the citation network between research papers, collaborative filtering algorithms can generate high quality recommendation in this domain, without requiring user opinions or ratings of the items. Users felt recommendation in this domain were useful and of high quality. Moreover, users stated that specific recommender algorithms were better suited to different usage scenarios. Hybrid recommender algorithms combining collaborative filtering and content-based algorithms can generate high quality research paper recommendations. The usefulness of these recommendations varied across user populations with novices and students finding the system more useful than professors and research professionals. Finally, users felt that different algorithms generate qualitatively different recommendation lists.

Researchers and librarians have long sought out better and more efficient ways of finding and organizing information. By applying collaborative filtering and other machine learning algorithms in the domain of research papers, we provide another tool researchers can use to overcome information overload. We do not believe that any of these algorithms will replace existing search and retrieval systems. We prefer instead to see recommenders as providing researchers with options to explore during their information seeking behavior. For example, a recommender can provide serendipitous

results in parallel to a user's current search task, giving the user both a searching and browsing interface. For many tasks, such as finding new collaborators, shifting research areas, or locating authoritative papers in an area, such interfaces could greatly enhance a user's experience.

An even more exciting implication is the generalization of this approach across other digital libraries. By basing recommendations on the citations between items, the recommender does not have to be limited by subject area—these algorithms can help novice users or users new to a field discover important information. We believe an application of this work would be to generate a research overview for a field, even a cross-disciplinary or emerging field. Imagine having a recommender not only provide you with a list of papers to read, but also recommend the best order in which to read them!

Re-examining the Recommendation Process

Users view recommendation lists as entities; results show list usefulness depends on more factors than only the usefulness of the items on the list. Depending on the user need, users were more satisfied with a diverse recommendation list, even at the expense of recommendation accuracy. Users noticed and appreciated small changes to diversity in recommendation lists.

Looking the recommendation process from the end user's perspective highlights the differences between recommender designers and end users. Human-Recommender Interaction theory is an intermediary language that both a recommender and its users can use to help generate more meaningful and useful recommendations. Using the HRI Process Model, user information seeking tasks can be mapped to recommender algorithms via the HRI Aspects and a family of metrics.

The fact that users preferred diverse lists at the expense of accuracy has substantial implications for recommender systems research. As we have discussed throughout this dissertation, a majority of recommender systems research has focused on creating and testing the accuracy of new algorithms. While we encourage the creation of more algorithms, we also encourage the community to seek out better evaluation

methods. Even more important, however, is deeper analyses of users in recommender systems and how they perceive recommendations.

To this end, we feel that HRI can be the basis of a new line of research, one that applies HCI principles to personalized information systems. The information filtering and retrieval communities have always talked about the importance of understanding the user [5]. For historical reasons, these communities have largely focused their attention on professionals (librarians, researchers, analysts, etc.). The Information Age brings these information seeking concepts to all users, and recommender systems may be the first interaction many users have with these systems. As researchers, we need to understand these users and create systems useful for all people. HRI is the first step in that direction.

Understanding Recommender Algorithms

Recommender algorithms differ from each other across many dimensions in terms of the recommendation lists they generate. Only by defining a set of metrics, each of which explores a different property, can we better understand the subtle differences in recommendation lists. The differences between collaborative filtering and content-based algorithms suggest that the information seeking interfaces to current digital libraries can be enhanced by adding a collaborative filtering-based search interface.

When recommending to real users, *don't look stupid*. This is especially true with new users; violating expectations destroys trust and reduces credibility. Recommenders that generate nonsensical results were not liked by users, even when the nonsensical recommendations were intermixed with meaningful results. Earlier, we found that generating one good recommendation in five was enough for users, but now generating one nonsensical recommendation is enough to lose faith in a recommender, even if the other four recommendations are good. Recommender algorithm evaluation must be done with real users, as offline simulation experiments using current predictive accuracy metrics and pre-selected input baskets cannot detect these problems.

Users are easy to please, but even easier to alienate. When dealing with personalized systems, we, as designers and researchers, need to reflect on how users

perceive and interact with our systems. One of the take-away lessons is that even detailed simulation experiments cannot always find algorithm problems that could alienate users. While we performed a series of experiments over many different algorithms, it may be beneficial to select a few algorithms and explore their characteristics on a deeper level. In a personalized system information flows in two directions, it is just as important to understand the algorithm as it is to understand the users, and it is just as important to understand the input basket as it is to understand the generated recommendation list. A failure to understand any section could lead to serious problems.

Future Work

Other than the challenges outlined above, there are several immediate next steps in this research. They include an understanding of the role datasets play in recommender personalities, the design of an interface to support recommendation browsing, applying recommender algorithms to more complex user tasks, and expanding beyond digital libraries.

This dissertation focused on the recommending of research papers, specifically papers from ResearchIndex and from the ACM Digital Library. These two datasets had very different properties: The ResearchIndex dataset was large, containing almost 200,000 papers, but as it was compiled through emergent digital library algorithms; as such, there were errors and omissions in the data. The ACM Digital Library dataset, on the other hand, was of very high editorial quality, but it was smaller, at around 24,000 items. In our user study from Chapter 9, we found the size of the dataset to be limiting in almost all aspects. Users complained about the lack of available papers, and were concerned about the recommended results. We believe the dataset contributed to the atypical results from the two statistical recommender algorithms. Thus, while we are confident in our results that recommender algorithms have distinct personalities, we believe datasets are exceptionally important in shaping the personalities, and this part needs to be explored.

Another interesting next step is to provide users with the ability to tune the recommender algorithms themselves. As described by Marchionini [77], digital interfaces to information must be aware of both searching and browsing interfaces. In the years since that paper was published, search engines have gotten good at the searching interface, but are poor at browsing. The algorithms described in this dissertation can generate lists of potentially serendipitous items for users; with the right interface, these algorithms can allow users to browse. Going further, the browsing aspect of the interface could be further enhanced if the interface provided users with elements to adjust the relative importance of HRI Aspects in real-time. Such an interface would allow users to ‘play’, allowing for a truly engaging experience, potentially leading to high user satisfaction and usefulness.

HRI suggests that users view recommenders in a very different way than system designers have in the past. There are a few assumptions made in this dissertation about HRI that would be interesting to explore. For example, we have made a strong distinction between taste-based domains and use-based domains with external criteria. In reality, users will have different levels of external criteria in the same domain. One important and difficult problem is to detect which users have external criteria from those who do not. Another important research question is determining how many HRI Aspects are important to a user at a given time. While we assumed in this dissertation that users are interested in a set of Aspects at once, it is possible that a few Aspects, perhaps even only one, will dominate a user’s need. Finally, HRI has been applied to a static view of a user and her information need. It would be exciting to track a user interacting with a recommender for over a single task to see which Aspects are important to the user at different times. Moreover, it would be fascinating to determine if some Aspects are important to a user at all times or if the Aspects change with each interaction.

The user types and information seeking tasks described in Chapter 7 were only designed to highlight the potential of HRI for recommending research papers. In reality, users will have much more complex information needs, and not all will want papers, even from digital libraries. Many will want information at other levels of abstraction, such as

authors, journals, or publication dates. For example, a user might want recommendations as to which journals she should submit the paper she is currently writing, or she may want a recommendation of which conference to attend to see people working in a new area of interest to her. The approach of applying collaborative filtering recommender algorithms to a citation network cleanly abstracts from papers to other information in that network: papers can vote for their authors, vote for their publication venue, or, more importantly, make such votes both for themselves and for their citations. This view creates rich recommendation structure to meet a wide variety of user needs.

In fact, this approach should work for any directed graph, citation networks are only a specialized case. Really, this is the converse of the discussion from [4]. In that paper, they translate the ratings matrix into a graph to use graph-centric algorithm to generate recommendations. Here, we propose that any directed graph can become a ratings matrix. Weights can be used as ratings for explicit rating scale algorithms. Thus, this approach would be useful for social networking, network routing tables, traffic and power grids, and legal cases, among others.

Conclusion

One of the strongest messages from this dissertation is that recommender algorithms are not the same. While each one generates high quality recommendations, they all have their own strengths and weaknesses. Our online user studies and experience with recommenders show that users are acutely aware of these differences, even able to describe them in meaningful ways. Through our detailed offline simulations, we can identify these differences between algorithms, gaining knowledge about which algorithm characteristics relate to which user-identified difference. We encourage researchers to continue developing new recommender algorithms, but it is imperative that we understand their personality differences and use them to our advantage. This information gives us a unique opportunity: to build personalized systems in which the user is in complete control, from what information they provide to which algorithms generate recommendations. HRI, the Aspect Model, and our new set of metrics provide a

framework on which we can systematically understand these differences, and a basis from which we can build personalized systems that better meet user information needs.

REFERENCES

- [1] ACM, "ACM Digital Library", <http://www.acm.org/dl>, 2005.
- [2] ACM, "ACM Computing Classification Scheme", <http://www.acm.org/class/1998/>, 1998.
- [3] G. Adomavicius, R. Sankaranarayanan, S. Sen, and A. Tuzhilin, "Incorporating Contextual Information in Recommender Systems using a Multidimensional Approach", *ACM Trans.Inf.Syst.*, vol. 23(1), pp. 103-145, 2005.
- [4] C.C. Aggarwal, J.L. Wolf, K. Wu and P.S. Yu, "Horting Hatches an Egg: A New Graph-Theoretic Approach to Collaborative Filtering", in *Proceedings of the Fifth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 201-212, 1999.
- [5] R. Baeza-Yates and B. Ribeiro-Neto, *Modern Information Retrieval*, New York: Addison Wesley, 1999, pp. 544.
- [6] M. Balabanovic and Y. Shoham, "Fab: Content-Based, Collaborative Recommendation", *Commun ACM*, vol. 40(3), pp. 66-72, 1997.
- [7] N.J. Belkin and W.B. Croft, "Information Filtering and Information Retrieval: Two Sides of the Same Coin?", *Commun ACM*, vol. 35(12), pp. 29-38, 1992.
- [8] B.L.D. Bezerra and de A.T. de Carvalho, Francisco, "A Symbolic Approach for Content-Based Information Filtering", *Information Processing Letters*, vol. 92(1), pp. 45-52, October, 2004.
- [9] K. Bharat, T. Kamba, and M. Albers, "Personalized, Interactive News on the Web", *Multimedia Syst.*, vol. 6(5), pp. 349-358, 1998.
- [10] K.D. Bollacker, S. Lawrence and C.L. Giles, "CiteSeer: An Autonomous Web Agent for Automatic Retrieval and Identification of Interesting Publications", in *Proceedings of the Second International Conference on Autonomous Agents*, pp. 116-123, 1998.
- [11] J. Bollen, S. Vemulapalli and W. Xu, "Digital Library Evaluation by Analysis of User Retrieval Patterns", in *Proceedings of ECDL 2002, the 6th European Conference on Research and Advanced Technology for Digital Libraries*, pp. 432-447, 2002.

- [12] B.R. Boyce, C.T. Meadow and D.H. Kraft, *Measurement in Information Science*, San Diego, CA: Academic Press, 1994, pp. 283.
- [13] J.S. Breese, D. Heckerman and C. Kadie, "Empirical Analysis of Predictive Algorithms for Collaborative Filtering", in *Proceedings of the Fourteenth Conference on Uncertainty in Artificial Intelligence*, pp. 43-52, 1998.
- [14] D. Bridge and A. Ferguson, "Diverse Product Recommendations using an Expressive Language for Case Retrieval", in *Proceedings of ECCBR 2002: 6th European Conference on Advances in Case-Based Reasoning (LNCS 2416)*, pp. 43-57, 2002.
- [15] D.G. Bridge and J. Kelleher, "Experiments in Sparsity Reduction: Using Clustering in Collaborative Recommenders", in *Proceedings of the 13th Irish International Conference on Artificial Intelligence and Cognitive Science*, pp. 144-149, 2002.
- [16] R. Burke, "Hybrid Recommender Systems: Survey and Experiments", *User Modeling and User-Adapted Interaction*, vol. 12(4), pp. 331-370, 2002.
- [17] R. Burke, "Knowledge-based Recommender Systems," in *Encyclopedia of Library and Information Science, Volume 69, Supplement 32 A*. Kent and C.M. Hall eds., New York, NY, USA: Marcel Dekker, 2000, pp. 180-200.
- [18] J. Canny, "Collaborative Filtering with Privacy Via Factor Analysis", in *Proceedings of the 25th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pp. 238-245, 2002.
- [19] J.M. Carroll, *HCI Models, Theories, and Frameworks: Towards a Multidisciplinary Science*, San Francisco, CA: Morgan Kaufmann, 2003, pp. 576.
- [20] D.O. Case, *Looking for Information: A Survey of Research on Information Seeking, Needs, and Behavior*, San Diego: Academic Press, 2002, pp. 350.
- [21] J. Cho and S. Roy, "Impact of Search Engines on Page Popularity", in *WWW '04: Proceedings of the 13th International Conference on World Wide Web*, pp. 20-29, 2004.
- [22] C.W. Choo, B. Detlor and D. Turnbull, *Web Work: Information Seeking and Knowledge Work on the World Wide Web*, Dordrecht, The Netherlands: Kluwer Academic Publishers, 2000, pp. 219.
- [23] G.G. Chowdhury and S. Chowdhury, *Introduction to Digital Libraries*, London: Facet Publishing, 2003, pp. 359.

- [24] H.H. Clark, *Using Language*, Cambridge, UK: Cambridge University Press, 1996, pp. 444.
- [25] M. Claypool, A. Gokhale, T. Miranda, P. Murnikov, D. Netes and M. Sartin, "Combining Content-Based and Collaborative Filters in an Online Newspaper", in *ACM SIGIR '99 Workshop on Recommender Systems*, 1999.
- [26] Cornell University, "ArXiv e-Print Archive", <http://arxiv.org>, 2005.
- [27] D. Cosley, S.K. Lam, I. Albert, J.A. Konstan and J. Riedl, "Is Seeing Believing?: How Recommender System Interfaces Affect Users' Opinions", in *Proceedings of the Conference on Human Factors in Computing Systems*, pp. 585-592, 2003.
- [28] D. Cosley, S. Lawrence and D.M. Pennock, "REFEREE: An Open Framework for Practical Testing of Recommender Systems using ResearchIndex.", in *Proceedings of the 28th International Conference on very Large Databases, VLDB 2002*, pp. 35-46, 2002.
- [29] P. Cotter and B. Smyth, "PTV: Intelligent Personalised TV Guides", in *Proceedings of the Seventeenth National Conference on Artificial Intelligence and Twelfth Conference on Innovative Applications of Artificial Intelligence*, pp. 957-964, 2000.
- [30] S.E. Crudge and F.C. Johnson, "Using the Information Seeker to Elicit Construct Models for Search Engine Evaluation", *J.Am.Soc.Inf.Sci.Technol.*, vol. 55(9), pp. 794-806, 2004.
- [31] S.J. Darmoni, F. Roussel, J. Benichou, B. Thirion, and N. Pinhas, "Reading Factor: A New Bibliometric Criterion for Managing Digital Libraries", *Journal of the Medical Library Association*, vol. 90(3), pp. 323-327, July, 2002.
- [32] P.M. Davis, "Information-Seeking Behavior of Chemists: A Transaction Log Analysis of Referral URLs", *J.Am.Soc.Inf.Sci.Technol.*, vol. 55(4), pp. 326-332, 2004.
- [33] M. Deshpande and G. Karypis, "Item-Based Top-N Recommendation Algorithms", *ACM Trans.Inf.Syst.*, vol. 22(1), pp. 143-177, 2004.
- [34] P. Domingos and M. Pazzani, "Beyond Independence: Conditions for the Optimality of the Simple Bayesian Classifier", in *Proceedings of the 13th International Conference on Machine Learning (ICML 96)*, pp. 105-112, 1996.
- [35] R. Fagin, R. Kumar, and D. Sivakumar, "Comparing Top k Lists", *SIAM J.Discret.Math.*, vol. 17(1), pp. 134-160, 2004.

- [36] S. Farrell, C. Campbell and S. Myagmar, "Relescope: An Experiment in Accelerating Relationships", in *CHI '05: CHI '05 Extended Abstracts on Human Factors in Computing Systems*, pp. 1363-1366, 2005.
- [37] M.A. Fitzgerald and C. Galloway, "Relevance Judging, Evaluation, and Decision Making in Virtual Libraries: A Descriptive Study", *J.Am.Soc.Inf.Sci.Technol.*, vol. 52(12), pp. 989-1010, 2001.
- [38] R.B. Fitzpatrick, "ISI's Journal Citation Reports on the Web", *Med.Ref.Serv.Q.*, vol. 22(4), pp. 45-56, 2003.
- [39] N. Ford, "Modeling Cognitive Processes in Information Seeking: From Popper to Pask", *J.Am.Soc.Inf.Sci.Technol.*, vol. 55(9), pp. 769-782, 2004.
- [40] J. Furner, "On Recommending", *J.Am.Soc.Inf.Sci.Technol.*, vol. 53(9), pp. 747-763, 2002.
- [41] D. Goldberg, D. Nichols, B.M. Oki, and D. Terry, "Using Collaborative Filtering to Weave an Information Tapestry", *Commun ACM*, vol. 35(12), pp. 61-70, 1992.
- [42] K. Goldberg, T. Roeder, D. Gupta, and C. Perkins, "Eigentaste: A Constant Time Collaborative Filtering Algorithm", *Inf.Ret.*, vol. 4(2), pp. 133-151, 2001.
- [43] N. Good, J.B. Schafer, J.A. Konstan, A. Borchers, B. Sarwar, J. Herlocker and J. Riedl, "Combining Collaborative Filtering with Personal Agents for Better Recommendations", in *Proceedings of the Sixteenth National Conference on Artificial Intelligence and the Eleventh Conference on Innovative Applications of Artificial Intelligence*, pp. 439-446, 1999.
- [44] Google Inc., "The Google Search Engine", <http://www.google.com>, 2005.
- [45] Google Inc., "The Google Scholar Search Engine", <http://scholar.google.com>, 2005.
- [46] D.A. Grossman and O. Frieder, *Information Retrieval: Algorithms and Heuristics*, Dordrecht, The Netherlands: Springer, 2004, pp. 332.
- [47] S.P. Harter and T.E. Nisonger, "ISI's Impact Factor as Misnomer: A Proposed New Measure to Assess Journal Impact", *Journal of the American Society for Information Science*, vol. 48(12), pp. 1146-1148, December, 1997.
- [48] C. Hayes and P. Cunningham, "Context Boosting Collaborative Recommendations", *Knowledge-Based Systems*, vol. 17(2-4), pp. 131-138, 2004/5.

- [49] C. Hayes, P. Cunningham and B. Smyth, "A Case-Based Reasoning View of Automated Collaborative Filtering", in *Proceedings of Case-Based Reasoning Research and Development, 4th International Conference on Case-Based Reasoning, ICCBR 2001 (LNCS 2080)*, pp. 234-248, 2001.
- [50] M. Hepworth, "A Framework for Understanding User Requirements for an Information Service: Defining the Needs of Informal Carers", *J.Am.Soc.Inf.Sci.Technol.*, vol. 55(8), pp. 695-708, 2004.
- [51] J.L. Herlocker, J.A. Konstan, L.G. Terveen, and J.T. Riedl, "Evaluating Collaborative Filtering Recommender Systems", *ACM Trans.Inf.Syst.*, vol. 22(1), pp. 5-53, 2004.
- [52] J.L. Herlocker and J.A. Konstan, "Content-Independent Task-Focused Recommendation", *IEEE Internet Computing*, vol. 5(6), pp. 40-47, 2001.
- [53] J.L. Herlocker. *Understanding and Improving Automated Collaborative Filtering*. Ph.D., University of Minnesota. 2000.
- [54] J.L. Herlocker, J.A. Konstan and J. Riedl, "Explaining Collaborative Filtering Recommendations", in *Proceedings of the 2000 ACM Conference on Computer Supported Cooperative Work*, pp. 241-250, 2000.
- [55] J.L. Herlocker, J.A. Konstan, A. Borchers and J. Riedl, "An Algorithmic Framework for Performing Collaborative Filtering", in *Proceedings of the 22nd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pp. 230-237, 1999.
- [56] F. Heylighen, "Change and Information Overload: Negative Effects", *Principia Cybernetica Web*, <http://pespmc1.vub.ac.be/CHINNEG.html>, Feb. 19, 1999.
- [57] T. Hideaki, "Bibliometrics by using Web of Science", *Journal of Information Processing and Management*, vol. 44(1), pp. 2-7, April, 2001.
- [58] W. Hill, L. Stead, M. Rosenstein and G. Furnas, "Recommending and Evaluating Choices in a Virtual Community of use", in *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pp. 194-201, 1995.
- [59] T. Hofmann, "Latent Semantic Models for Collaborative Filtering", *ACM Trans.Inf.Syst.*, vol. 22(1), pp. 89-115, 2004.
- [60] T. Hofmann, "Unsupervised Learning by Probabilistic Latent Semantic Analysis", *Mach.Learning*, vol. 42(1-2), pp. 177-196, January, 2001.

- [61] T. Hofmann, "Probabilistic Latent Semantic Indexing", in *Proceedings of the 22nd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pp. 50-57, 1999.
- [62] T. Hofmann and J. Puzicha, "Latent Class Models for Collaborative Filtering", in *Proceedings of the Sixteenth International Joint Conference on Artificial Intelligence*, pp. 688-693, 1999.
- [63] Z. Huang, H. Chen, and D. Zeng, "Applying Associative Retrieval Techniques to Alleviate the Sparsity Problem in Collaborative Filtering", *ACM Trans.Inf.Syst.*, vol. 22(1), pp. 116-142, 2004.
- [64] Z. Huang, W. Chung, T. Ong and H. Chen, "A Graph-Based Recommender System for Digital Library", in *Proceedings of the Second ACM/IEEE-CS Joint Conference on Digital Libraries*, pp. 65-73, 2002.
- [65] JSTOR, "JSTOR, the Scholarly Journal Archive", <http://www.jstor.org>, 2005.
- [66] T. Kamba, K. Bharat and M.C. Albers, "The Krakatoa Chronicle: An Interactive, Personalized Newspaper on the Web", in *Proceedings of the Fourth International Conference on the World Wide Web*, pp. 159-170, 1995.
- [67] G. Karypis, "SUGGEST Top-N Recommendation Engine", <http://www-users.cs.umn.edu/~karypis/suggest/index.html>, 2000.
- [68] H. Kautz, B. Selman, and M. Shah, "Referral Web: Combining Social Networks and Collaborative Filtering", *Commun ACM*, vol. 40(3), pp. 63-65, 1997.
- [69] K. Kim, "Implications of User Characteristics in Information Seeking on the World Wide Web", *Int.J.Hum.-Comput.Interact.*, vol. 13(3), pp. 323-340, September, 2001.
- [70] C.C. Kuhlthau, *Seeking Meaning: A Process Approach to Library and Information Services*, Westport, CT: Libraries Unlimited, 2004, pp. 247.
- [71] S.K. Lam and J. Riedl, "Shilling Recommender Systems for Fun and Profit", in *Proceedings of the 13th International Conference on World Wide Web*, pp. 393-402, 2004.
- [72] S. Lawrence, "Access to Scientific Literature," in *The Nature Yearbook of Science and Technology*, D. Butler ed., Nature Publishing Group (Palgrave Macmillan), 2001, pp. 86-92.

- [73] S. Lawrence, K. Bollacker and C.L. Giles, "Indexing and Retrieval of Scientific Literature", in *Proceedings of the Eighth International Conference on Information and Knowledge Management*, pp. 139-146, 1999.
- [74] S. Lawrence, C.L. Giles, and K. Bollacker, "Digital Libraries and Autonomous Citation Indexing", *Computer*, vol. 32(6), pp. 67-71, 1999.
- [75] S. Lee, J. Yang, and S. Park, "Discovery of Hidden Similarity on Collaborative Filtering to Overcome Sparsity Problem," in *Proceedings of Discovery Science: 7th International Conference, DS 2004, (LNAI 3245)*, pp. 396-402, 2004,.
- [76] J.D. Mackinlay, R. Rao and S.K. Card, "An Organic User Interface for Searching Citation Links", in *CHI '95: Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pp. 67-73, 1995.
- [77] G. Marchionini, "Resource Search and Discovery," in *Research Agenda for Networked Cultural Heritage*, Getty Art History Information Program ed., Santa Monica, CA: Getty AHIP, 1996, pp. 35-40.
- [78] G. Marchionini, *Information Seeking in Electronic Environments*, Cambridge, UK: Cambridge University Press, 1995, pp. 236.
- [79] C.C. Marshall and S. Bly, "Saving and using Encountered Information: Implications for Electronic Periodicals", in *CHI '05: Proceeding of the SIGCHI Conference on Human Factors in Computing Systems*, pp. 111-120, 2005.
- [80] A.K. McCallum, "Bow: A Toolkit for Statistical Language Modeling, Text Retrieval, Classification and Clustering", <http://www-2.cs.cmu.edu/mccallum/bow/>, 1996.
- [81] D.W. McDonald, "Recommending Collaboration with Social Networks: A Comparative Evaluation", in *Proceedings of the Conference on Human Factors in Computing Systems*, pp. 593-600, 2003.
- [82] L. McGinty and B. Smyth, "On the Role of Diversity in Conversational Recommender Systems", in *Proceedings of Case-Based Reasoning Research and Development: 5th International Conference on Case-Based Reasoning, ICCBR 2003 (LNCS 2689)*, pp. 276-290, 2003.
- [83] L. McGinty and B. Smyth, "Deep Dialogue Vs Casual Conversation in Recommender Systems", in *Proceedings of the Workshop on Personalization in eCommerce at the Second International Conference on Adaptive Hypermedia and Web-Based Systems (AH-02)*, pp. 80-89, 2002.

- [84] M.R. McLaughlin and J.L. Herlocker, "A Collaborative Filtering Algorithm and Evaluation Metric that Accurately Model the User Experience", in *SIGIR '04: Proceedings of the 27th Annual International Conference on Research and Development in Information Retrieval*, pp. 329-336, 2004.
- [85] S.M. McNee, N. Kapoor and J.A. Konstan, "Don't Look Stupid: Avoiding Pitfalls when Recommending Research Papers", in *Proceedings of the 2006 ACM Conference on Computer Supported Cooperative Work (CSCW 2006)*, 2006.
- [86] S.M. McNee, J. Riedl and J.A. Konstan, "Being Accurate is Not enough: How Accuracy Metrics have Hurt Recommender Systems", in *Extended Abstracts of the 2006 ACM Conference on Human Factors in Computing Systems (CHI 2006)*, pp. 997-1001, 2006.
- [87] S.M. McNee, J. Riedl and J.A. Konstan, "Making Recommendations Better: An Analytic Model for Human-Recommender Interaction", in *Extended Abstracts of the 2006 ACM Conference on Human Factors in Computing Systems (CHI 2006)*, pp. 1003-1008, 2006.
- [88] S.M. McNee, S.K. Lam, C. Guetzlaff, J.A. Konstan and J. Riedl, "Confidence Displays and Training in Recommender Systems", in *Proceedings of the INTERACT '03 IFIP TC13 International Conference on Human-Computer Interaction*, pp. 176-183, 2003.
- [89] S.M. McNee, S.K. Lam, J.A. Konstan and J. Riedl, "Interfaces for Eliciting New User Preferences in Recommender Systems", in *Proceedings of the 9th International Conference on User Modeling (UM'2003)*, pp. 178-187, 2003.
- [90] S.M. McNee, I. Albert, D. Cosley, P. Gopalkrishnan, S.K. Lam, A.M. Rashid, J.A. Konstan and J. Riedl, "On the Recommending of Citations for Research Papers", in *Proceedings of the 2002 ACM Conference on Computer Supported Cooperative Work*, pp. 116-125, 2002.
- [91] D. McSherry, "Diversity-Conscious Retrieval", in *Proceedings of the 6th European Conference on Advances in Case-Based Reasoning, ECCBR 2002 (LNCS 2416)*, pp. 219-234, 2002.
- [92] P. Melville, R.J. Mooney and R. Nagarajan, "Content-Boosted Collaborative Filtering for Improved Recommendations", in *Eighteenth National Conference on Artificial Intelligence*, pp. 187-192, 2002.
- [93] S.E. Middleton, N.R. Shadbolt, and D.C.D. Roure, "Ontological User Profiling in Recommender Systems", *ACM Trans.Inf.Syst.*, vol. 22(1), pp. 54-88, 2004.

- [94] N. Mirzadeh, F. Ricci and M. Bansal, "Feature Selection Methods for Conversational Recommender Systems", in *Proceedings of the 2005 IEEE International Conference on e-Technology, e-Commerce and e-Service, 2005 (EEE '05)*, pp. 772-777, 2005.
- [95] N. Mirzadeh, F. Ricci and M. Bansal, "Supporting User Query Relaxation in a Recommender System", in *Proceedings of E-Commerce and Web Technologies: 5th International Conference, EC-Web 2004 (LNCS 3182)*, pp. 31-40, 2004.
- [96] W.H. Mischo, T.G. Habing and T.W. Cole, "Integration of Simultaneous Searching and Reference Linking Across Bibliographic Resources on the Web", in *Proceedings of the Second ACM/IEEE-CS Joint Conference on Digital Libraries*, pp. 119-125, 2002.
- [97] T.M. Mitchell, *Machine Learning*, Boston, MA: McGraw-Hill Science/Engineering/Math, 1997, pp. 432.
- [98] H.B. Mokros and M. Aakhus, "From Information-Seeking Behavior to Meaning Engagement Practice. Implications for Communication Theory and Research", *Human Communication Research*, vol. 28, pp. 298-312, 2002.
- [99] R.J. Mooney and L. Roy, "Content-Based Book Recommending using Learning for Text Categorization", in *Proceedings of the Fifth ACM Conference on Digital Libraries*, pp. 195-204, 2000.
- [100] H. Murray Jr., *Methods for Satisfying the Needs of the Scientist and the Engineer for Scientific and Technical Communication*, Washington D.C.: A Press Release, 1966.
- [101] S. Narayanan, L. Koppaka, N. Edala, D. Loritz, and R. Daley, "Adaptive Interface for Personalizing Information Seeking", *CyberPsychology & Behavior*, vol. 7(6), pp. 683-688, December, 2004.
- [102] National Institutes of Health (NIH), "Entrez PubMed", <http://www.ncbi.nlm.nih.gov/entrez/query.fcgi?DB=pubmed>, 2005.
- [103] M.E.J. Newman, "Scientific Collaboration Networks. I. Network Construction and Fundamental Results", *Phys. Rev. E*, vol. 64(016131), pp. 1-8, 28 June, 2001.
- [104] D. O' Sullivan, D. Wilson and B. Smyth, "Improving Case-Based Recommendation: A Collaborative Filtering Approach", in *Proceedings of the 6th European Conference on Advances in Case-Based Reasoning, ECCBR 2002 (LNCS 2416)*, pp. 278-291, 2002.

- [105] M. O'Mahony, N. Hurley, N. Kushmerick, and G.C.M. Silvestre, "Collaborative Recommendation: A Robustness Analysis", *ACM Trans.Inter.Tech.*, vol. 4(4), pp. 344-377, 2004.
- [106] L. Page, S. Brin, R. Motwani, and T. Winograd, "The PageRank Citation Ranking: Bringing Order to the Web", *Tech Report, Stanford Univ.*, vol. 1999(66), pp. 1-17, 1999.
- [107] D.M. Pennock, E. Horvitz and C.L. Giles, "Social Choice Theory and Recommender Systems: Analysis of the Axiomatic Foundations of Collaborative Filtering", in *Proceedings of the Seventeenth National Conference on Artificial Intelligence and Twelfth Conference on Innovative Applications of Artificial Intelligence*, pp. 729-734, 2000.
- [108] D.M. Pennock, E. Horvitz, S. Lawrence and C.L. Giles, "Collaborative Filtering by Personality Diagnosis: A Hybrid Memory and Model-Based Approach", in *Proceedings of the 16th Conference on Uncertainty in Artificial Intelligence*, pp. 473-480, 2000.
- [109] Pennsylvania State University, "EBizSearch", <http://gunther.smeal.psu.edu/>, 2005.
- [110] Pennsylvania State University (PSU), "CiteSeer.IST", <http://citeseer.ist.psu.edu/>, 2005.
- [111] J.P. Pickett Ed., *The American Heritage Dictionary of the English Language*, Boston: Houghton Mifflin Company, 2000, pp. 2074.
- [112] P. Pirolli, "Computational Models of Information Scent-Following in a very Large Browsable Text Collection", in *CHI '97: Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pp. 3-10, 1997.
- [113] A. Popescul, L.H. Ungar, D.M. Pennock and S. Lawrence, "Probabilistic Models for Unified Collaborative and Content-Based Recommendation in Sparse-Data Environments", in *Proceedings of the 17th Conference in Uncertainty in Artificial Intelligence*, pp. 437-444, 2001.
- [114] M.F. Porter, "An algorithm for suffix stripping," in *Readings in Information Retrieval*, K.S. Jones and P. Willett eds., San Francisco, CA: Morgan Kaufmann Publishers Inc, 1997, ch. 6, pp. 313-316.
- [115] P. Pu, B. Faltings and M. Torrens, "Effective Interaction Principles for Online Product Search Environments", in *Proceedings of the IEEE/WIC/ACM International Conference on Web Intelligence (WI'04)*, pp. 724-727, 2004.

- [116] A.M. Rashid, G. Karypis and J. Riedl, "Influence in Ratings-Based Recommender Systems: An Algorithm-Independent Approach", A Poster Paper in *Proceedings of the 2005 SIAM International Conference on Data Mining*, 2005.
- [117] A.M. Rashid, I. Albert, D. Cosley, S.K. Lam, S.M. McNee, J.A. Konstan and J. Riedl, "Getting to Know You: Learning New User Preferences in Recommender Systems", in *Proceedings of the 7th International Conference on Intelligent User Interfaces*, pp. 127-134, 2002.
- [118] B. Reeves and C. Nass, *The Media Equation: How People Treat Computers, Television, and New Media Like Real People and Places*, Chicago, IL: Center for the Study of Language and Information, University of Chicago Press, 2003, pp. 320.
- [119] P. Resnick and H.R. Varian, "Recommender Systems", *Commun ACM*, vol. 40, pp. 56-58, March, 1997.
- [120] P. Resnick, N. Iacovou, M. Suchak, P. Bergstrom and J. Riedl, "GroupLens: An Open Architecture for Collaborative Filtering of Netnews", in *Proceedings of the 1994 ACM Conference on Computer Supported Cooperative Work*, pp. 175-186, 1994.
- [121] C. Reynolds and R. Picard, "Ethical Evaluation of Displays that Adapt to Affect", *CyberPsychology & Behavior*, vol. 7(6), pp. 662-666, December, 2004.
- [122] F. Ricci, K. Wober and A. Zins, "Recommendations by Collaborative Browsing", in *Proceedings of the International Conference on Information and Communication Technologies in Tourism 2005*, pp. 172-182, 2005.
- [123] F. Ricci and F. Del Missier, "Supporting travel decision making through personalized recommendation," in *Designing Personalized User Experiences in eCommerce*, C. Karat, J.O. Blom and J. Karat eds., Dordrecht, The Netherlands: Kluwer Academic Publishers, 2004, ch. 13, pp. 231-253.
- [124] F. Ricci, A. Venturini, D. Cavada, N. Mirzadeh, D. Blaas and M. Nones, "Product Recommendation with Interactive Query Management and Twofold Similarity", in *Proceedings of Case-Based Reasoning Research and Development: 5th International Conference on Case-Based Reasoning, ICCBR 2003 (LNCS 2689)*, pp. 479-493, 2003.
- [125] S.Y. Rieh, "On the Web at Home: Information Seeking and Web Searching in the Home Environment", *J.Am.Soc.Inf.Sci.Technol.*, vol. 55(8), pp. 743-753, 2004.
- [126] D.E. Rose and D. Levinson, "Understanding User Goals in Web Search", in *WWW '04: Proceedings of the 13th International Conference on World Wide Web*, pp. 13-19, 2004.

- [127] G. Salton and C. Buckley, "Term-Weighting Approaches in Automatic Text Retrieval", *Inf.Process.Manage.*, vol. 24(5), pp. 513-523, 1988.
- [128] G. Salton, *Introduction to Modern Information Retrieval*, New York, NY: McGraw-Hill Companies, 1983, pp. 448.
- [129] B. Sarwar, G. Karypis, J.A. Konstan and J. Riedl, "Incremental SVD-Based Algorithms for Highly Scaleable Recommender Systems", in *Proceedings of the Fifth International Conference on Computer and Information Technology (ICCI 2002)*, 2002.
- [130] B. Sarwar. *Sparsity, Scalability, and Distribution in Recommender Systems*. Ph.D., University of Minnesota. 2001.
- [131] B. Sarwar, G. Karypis, J. Konstan and J. Riedl, "Item-Based Collaborative Filtering Recommendation Algorithms", in *Proceedings of the Tenth International Conference on World Wide Web*, pp. 285-295, 2001.
- [132] J.B. Schafer, J.A. Konstan, and J. Riedl, "View through MetaLens: Usage Patterns for a Meta-Recommendation System", *IEE Proceedings-Software*, vol. 151(6), pp. 267-79, December, 2004.
- [133] A.I. Schein, A. Popescul, L.H. Ungar and D.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*, pp. 253-260, 2002.
- [134] J.P. Scott, *Social Network Analysis: A Handbook*, London, UK: SAGE Publications, 2000, pp. 240.
- [135] U. Shardanand and P. Maes, "Social Information Filtering: Algorithms for Automating "Word of Mouth"", in *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pp. 210-217, 1995.
- [136] C. Silverstein, H. Marais, M. Henzinger, and M. Moricz, "Analysis of a very Large Web Search Engine Query Log", *SIGIR Forum*, vol. 33(1), pp. 6-12, 1999.
- [137] R. Sinha and K. Swearingen, "The Role of Transparency in Recommender Systems", in *CHI '02 Extended Abstracts on Human Factors in Computing Systems*, pp. 830-831, 2002.
- [138] A. Spink, T.D. Wilson, N. Ford, A. Foster, and D. Ellis, "Information Seeking and Mediated Searching Study. Part 3: Successive Searching", *J.Am.Soc.Inf.Sci.Technol.*, vol. 53(9), pp. 716-727, 2002.

- [139] L.T. Su, "A Comprehensive and Systematic Model of User Evaluation of Web Search Engines: II. an Evaluation by Undergraduates", *J.Am.Soc.Inf.Sci.Technol.*, vol. 54(13), pp. 1193-1223, 2003.
- [140] L.T. Su, "A Comprehensive and Systematic Model of User Evaluation of Web Search Engines: I. Theory and Background", *J.Am.Soc.Inf.Sci.Technol.*, vol. 54(13), pp. 1175-1192, 2003.
- [141] K. Swearingen and R. Sinha, "Beyond Algorithms: An HCI Perspective on Recommender Systems", in *Workshop Notes of the 2001 ACM SIGIR Workshop on Recommender Systems*, pp. 19-28, 2001.
- [142] R.S. Taylor, "Question-Negotiation and Information Seeking in Libraries", *College and Research Libraries*, vol. 29pp. 178-194, May, 1968.
- [143] L. Terveen, W. Hill, B. Amento, D. McDonald, and J. Creter, "PHOAKS: A System for Sharing Recommendations", *Commun ACM*, vol. 40(3), pp. 59-62, 1997.
- [144] C.A. Thompson, M.H. Goker, and P. Langley, "A Personalized System for Conversational Recommendations", *Artificial Intelligence Research*, vol. 21pp. 393-428, January, 2004.
- [145] M. Torrens, B. Faltings, and P. Pu, "SmartClients: Constraint Satisfaction as a Paradigm for Scaleable Intelligent Information Systems", *Constraints*, vol. 7(1), pp. 49-69, January, 2002.
- [146] R. Torres, S.M. McNee, M. Abel, J.A. Konstan and J. Riedl, "Enhancing Digital Libraries with TechLens+", in *Proceedings of the 2004 Joint ACM/IEEE Conference on Digital Libraries*, pp. 228-236, 2004.
- [147] L.H. Ungar and D.P. Foster, "Clustering Methods for Collaborative Filtering", in *Proceedings of the AAAI-98 Workshop on Recommender Systems at the 15th National Conference on Artificial Intelligence*, pp. 112-125, 1998.
- [148] University of Lethbridge, "New Zealand Digital Library Project", <http://www.sadl.uleth.ca/nz/cgi-bin/library>, 2005.
- [149] M. van Setten, S. Pokraev and J. Koolwaaij, "Context-Aware Recommendations in the Mobile Tourist Application COMPASS", in *Proceedings of the Third International Conference on Adaptive Hypermedia and Adaptive Web-Based Systems (AH 2004) LNCS 3137*, pp. 235-245, 2004.
- [150] M. van Setten, M. Veenstra, A. Nijholt and B. van Dijk, "Case-Based Reasoning as a Prediction Strategy for Hybrid Recommender Systems", in *Proceedings of*

Advances in Web Intelligence: Second International Atlantic Web Intelligence Conference (AWIC 2004) LNCS 3034, pp. 13-22, 2004.

- [151] M. van Setten, M. Veenstra, A. Nijholt and B. van Dijk, "Prediction Strategies in a TV Recommender System - Method and Experiments", in *Proceedings of IADIS International Conference WWW/Internet 2003*, pp. 203-210, 2003.
- [152] H.D. White, B. Wellman, and N. Nazer, "Does Citation Reflect Social Structure?: Longitudinal Evidence from the "Globenet" Interdisciplinary Research Group", *J.Am.Soc.Inf.Sci.Technol.*, vol. 55(2), pp. 111-126, 2004.
- [153] M.D. White, "Question-Negotiation and Information Seeking in Libraries", *RQ*, vol. 20pp. 373-381, Summer, 1981.
- [154] T.D. Wilson, N. Ford, D. Ellis, A. Foster, and A. Spink, "Information Seeking and Mediated Searching. Part 2: Uncertainty and its Correlates", *J.Am.Soc.Inf.Sci.Technol.*, vol. 53(9), pp. 704-715, 2002.
- [155] T.D. Wilson, "Models in Information Behaviour Research", *J. Documentation*, vol. 55(3), pp. 249-270, June, 1999.
- [156] A. Woodruff, R. Gossweiler, J. Pitkow, E.H. Chi and S.K. Card, "Enhancing a Digital Book with a Reading Recommender", in *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pp. 153-160, 2000.
- [157] Yahoo! Inc., "LAUNCHcast Radio on Yahoo Music", <http://music.yahoo.com/launchcast/default.asp>, 2005.
- [158] K. Yu, V. Tresp and S. Yu, "A Nonparametric Hierarchical Bayesian Framework for Information Filtering", in *Proceedings of the 27th Annual International Conference on Research and Development in Information Retrieval*, pp. 353-360, 2004.
- [159] W. Yue and C.S. Wilson, "The Relationship of Two Derived Measures: Impact Factor and Immediacy Index", in *Proceedings of the 8th International Conference on Scientometrics and Informetrics (ISSI-2001)*, pp. 893-6, 2001.
- [160] J. Zaslow, "If TiVo Thinks You are Gay, here's how to Set it Straight --- Amazon.Com Knows You, Too, Based on what You Buy; Why all the Cartoons?", *The Wall Street Journal*, vol. A, pp. 1, November 26, 2002.
- [161] C.N. Ziegler. *Towards Decentralized Recommender Systems*. Ph.D., Albert-Ludwigs-Universitat Freiburg. 2005.

- [162] C.N. Ziegler, S.M. McNee, J.A. Konstan and G. Lausen, "Improving Recommendation Lists through Topic Diversification", in *Proceedings of the Fourteenth International World Wide Web Conference (WWW 2005)*, pp. 22-32, 2005.
- [163] C. Ziegler, G. Lausen and L. Schmidt-Thieme, "Taxonomy-Driven Computation of Product Recommendations", in *CIKM '04: Proceedings of the Thirteenth ACM Conference on Information and Knowledge Management*, pp. 406-415, 2004.