

DESIGN AND EVALUATION ISSUES FOR USER-CENTRIC ONLINE PRODUCT SEARCH

THÈSE N° 4045 (2008)

PRÉSENTÉE LE 11 AVRIL 2008

À LA FACULTÉ INFORMATIQUE ET COMMUNICATIONS

GROUPE PU

PROGRAMME DOCTORAL EN INFORMATIQUE, COMMUNICATIONS ET INFORMATION

ÉCOLE POLYTECHNIQUE FÉDÉRALE DE LAUSANNE

POUR L'OBTENTION DU GRADE DE DOCTEUR ÈS SCIENCES

PAR

Jiyong ZHANG

Master of engineering, Tsinghua University, Chine
et de nationalité chinoise

acceptée sur proposition du jury:

Prof. C. Petitpierre, président du jury
Dr P. Pu Faltings, directrice de thèse
Prof. J. Huang, rapporteur
Dr B. O'Sullivan, rapporteur
Prof. F. Ricci, rapporteur



ÉCOLE POLYTECHNIQUE
FÉDÉRALE DE LAUSANNE

Suisse
2008

To my wife, Zhuqing GU

Acknowledgments

First of all, I am deeply grateful to my thesis advisor, *Dr. Pearl Pu*, for her solid support and excellent guidance during my PhD study. I enjoy the freedom of discussing many research ideas with her, and she is always able to provide insightful feedback and prompt help so that I can move forward on the right track.

I would like to thank Prof. Boi Faltings for kindly providing constructive suggestions and encouragement whenever I had some difficulties in research.

I thank the members of the jury, Prof. Jeffrey Huang, Prof. Francesco Ricci and Dr. Barry O’Sullivan, for spending their time assessing my work and providing worthy comments for this dissertation.

Many thanks go to colleagues from the HCI Group and the AI Lab, especially Nicolas Jones, Li Chen, Paolo Viappiani and Vincent Schickel-Zuber, for working together closely, sharing various thoughts, and providing valuable comments to my research. A special thank goes to Nicolas for helping me design the visual interface in Chapter 6 and translating the thesis abstract into French. I also thank James Reilly from UCD for the collaboration work and his contribution on the *CritiqueShop* system.

During the past years it’s my great pleasure to have many friends in my life, both from EPFL and outside. Without their friendships this PhD research would have been unbearable. It is too long to list their names here, but I would like to take this opportunity to thank them all.

Last but not least, I would like to thank my parents and my family, who hold unshakable beliefs that I can accomplish this dissertation in time. My deepest thank goes to my wife, Zhuqing Gu, for her devoted love, understanding, patience and encouragement all the time.

Nowadays more and more people are looking for products online, and a massive amount of products are being sold through e-commerce systems. It is crucial to develop effective online product search tools to assist users to find their desired products and to make sound purchase decisions. Currently, most existing online product search tools are not very effective in helping users because they ignore the fact that users only have limited knowledge and computational capacity to process the product information. For example, a search tool may ask users to fill in a form with too many detailed questions, and the search results may either be too minimal or too vast to consider. Such *system-centric* designs of online product search tools may cause some serious problems to end-users. Most of the time users are unable to state all their preferences at one time, so the search results may not be very accurate. In addition, users can either be impatient to view too much product information, or feel lost when no product appears in the search results during the interaction process.

User-centric online product search tools can be developed to solve these problems and to help users make buying decisions effectively. The search tool should have the ability to recommend suitable products to meet the user's various preferences. In addition, it should help the user navigate the product space and reach the final target product without too much effort. Furthermore, according to behavior decision theory, users are likely to construct their preferences during the decision process, so the tool should be designed in an interactive way to elicit users' preferences gradually. Moreover, it should be decision supportive for users to make accurate purchasing decisions even if they don't have detail domain knowledge of the specific products.

To develop effective user-centric online product search tools, one important task is to evaluate their performance so that system designers can obtain prompt feedback. Another crucial task is to design new algorithms and new user interfaces of

the tools so that they can help users find the desired products more efficiently.

In this thesis, we first consider the evaluation issue by developing a simulation environment to analyze the performance of generic product search tools. Compared to earlier evaluation methods that are mainly based on real-user studies, this simulation environment is faster and less expensive. Then we implement the *CritiqueShop* system, an online product search tool based on the well-known critiquing technique with two aspects of novelties: a user-centric compound critiquing generation algorithm which generates search results efficiently, and a visual user interface for enhancing user's satisfaction degree. Both the algorithm and the user interface are validated by large-scale comparative real-user studies. Moreover, the collaborative filtering approach is widely used to help people find low-risk products in domains such as movies or books. Here we further propose a *recursive* collaborative filtering approach that is able to generate search results more accurately without requiring additional effort from the users.

Keywords: online product search, critiquing, preference, performance evaluation, real-user study.

Aujourd'hui de plus en plus de gens recherchent des produits en-ligne et une quantité importante de ceux-ci sont vendus quotidiennement à travers des systèmes de e-commerce. Il est crucial de développer de bons outils de recherche en-ligne de produits afin d'assister les utilisateurs pour qu'ils trouvent leurs produits désirés, et leur faciliter les décisions d'achats. Actuellement, la majorité des outils de recherche de produits ne sont pas très efficaces pour aider les utilisateurs car ils ne prennent pas en compte le fait que les utilisateurs ont des connaissances limitées sur le sujet et une capacité d'analyse limitée pour absorber l'ensemble des informations fournies sur les produits. Par exemple, un outil de recherche peut demander aux utilisateurs de remplir un formulaire contenant un trop grand nombre de questions détaillées. Dans certains cas, le système peut aussi retourner soit une liste vide, soit une liste trop vaste de produits répondant aux critères de recherche. De tels conceptions d'outils de recherche, qui dépendent en premier des possibilités offertes par le système ("centré-système"), peuvent être la source de problèmes sérieux pour les utilisateurs finaux. La plupart du temps les utilisateurs sont incapables d'indiquer au système toutes leurs préférences en même temps, rendant les résultats imprécis. De plus, les utilisateurs sont souvent impatients de voir toutes les informations sur trop de produits, ou peuvent se sentir perdus lorsqu'il n'y a aucun résultat qui correspond à leur recherche.

Pour résoudre ces problèmes, des systèmes orientés autour des besoins des utilisateurs ("centré-utilisateurs") peuvent être développés, et parviennent à aider les utilisateurs à prendre des décisions d'achats efficaces. L'outil de recherche doit avoir la capacité de recommander des produits adéquats qui satisfont les différentes préférences d'un utilisateur. De plus, l'outil doit pouvoir aider l'utilisateur à naviguer au travers de l'espace de produits à disposition dans le système pour finalement atteindre le meilleur produit cible, et cela sans trop d'efforts. Il se trouve également que d'après la théorie comportementale de la décision, il est fort prob-

able que les utilisateurs construisent leur préférence en partie durant ce processus décisionnel, indiquant que les outils de recherche devraient être conçus d'une manière interactive améliorant ainsi la découverte progressive des préférences de l'utilisateur. Finalement, un tel outil doit faciliter la prise de décision pour permettre aux utilisateurs de prendre une bonne décision d'achat même lorsqu'ils n'ont pas de connaissances détaillées sur le domaine des produits concernés.

Afin de développer de tels systèmes de recherche en-ligne é tant tout à la fois efficaces et orientés utilisateurs, une tâche importante est d'évaluer leur performance afin de fournir un prompt feed-back aux développeurs. Un autre point crucial est l'amélioration des algorithmes de tri et interfaces de présentation pour aider les utilisateurs à trouver leur produits désirés.

Dans cette thèse, nous étudions tout d'abord les méthodes permettant d'analyser un outil de recherche générique. A cette fin, nous avons développé un environnement de simulation. Cet environnement de simulation est plus rapide et moins coûteux comparés aux méthodes précédentes d'évaluation qui sont principalement basées sur des cas d'études réelles, Par la suite, nous avons implémenté un système d'achat nommé "CritiqueShop", un outil de recherche en-ligne basé sur les techniques reconnues de "critiquing", avec deux aspects innovants: un algorithme de "compound critiques" centré-utilisateur qui génère des résultats de recherches optimaux, et une interface de visualisation de ces critiques qui augmente le degré de satisfaction des utilisateurs. Aussi bien l'algorithme que la nouvelle interface sont testés et validés par des études comparatives avec de vrais utilisateurs. De plus, l'approche par Collaborative filtering est utilisée de manière exhaustive pour aider les utilisateurs à trouver des produits à bas-risque financier dans des domaines tels que les films, livres, etc. Ici, nous proposons une approche récursive de ces algorithmes collaboratifs qui permet de générer des résultats plus précis, sans demander d'efforts supplémentaires de la part des utilisateurs.

Mots-clefs: recherche en-ligne de produits, critiquer, préférence, évaluation de performance, étude d'utilisateurs.

1	Introduction	1
1.1	Motivations	1
1.1.1	User-Centric Online Product Search	4
1.1.2	The Performance Evaluation Issue	6
1.1.3	The System Design Issue	8
1.2	Contributions	11
1.2.1	Performance Evaluation Framework	11
1.2.2	Critique-based Product Search	12
1.2.3	Recursive Collaborative Filtering	13
1.3	Thesis Outline	14
2	Background and Related Work	17
2.1	Introduction	17
2.2	Multi-Attribute Decision Problem	18
2.3	User's Preferences	20
2.3.1	Principles of Preference Elicitation	21
2.3.2	The Interaction Paradigm	21
2.3.3	Preference Elicitation Styles	23
2.4	Multi-Attribute Utility Theory	25
2.5	Critique-based Search Tools	28
2.5.1	The FindMe Systems	28
2.5.2	Dynamic Critiquing	31
2.5.3	SmartClient	33
2.5.4	FlatFinder	34

CONTENTS

2.5.5	MobyRek	35
2.5.6	Apt Decision	37
2.5.7	Expertclerk	39
2.6	Recommendation Techniques	40
2.6.1	Collaborative Filtering Recommendation	40
2.6.2	Content-based Recommendation	42
2.7	Other Decision Making Approaches	43
2.7.1	Framework of Constraint Satisfaction Problems (CSPs)	43
2.7.2	CP-network	45
2.7.3	Analytic Hierarchy Process (AHP)	47
2.7.4	Heuristic Decision Making Strategies	48
3	Simulation Environment For Performance Evaluation	51
3.1	Introduction	51
3.2	Related Work	52
3.3	Decision Strategies	53
3.4	The Extended Effort–Accuracy Framework	55
3.4.1	Measuring Cognitive Effort	56
3.4.2	Measuring Decision Accuracy	56
3.4.3	Measuring Elicitation Effort	61
3.4.4	Analysis of Cognitive and Elicitation Effort	62
3.5	Simulation Environment	63
3.6	Simulation Results	65
3.7	Discussion	72
3.8	Summary	73
4	User-Centric Algorithm for Compound Critiquing Generation	75
4.1	Introduction	75
4.2	Related Work	77
4.2.1	Unit Critique and Compound Critique	77
4.2.2	Generating Compound Critiques based on Apriori	79
4.2.3	Other Critiquing Systems	80
4.3	Generating Compound Critiques based on MAUT	81
4.4	An Illustrative Example	85
4.5	Experiments and Results	87
4.6	Discussions	89
4.7	Summary	91
5	Real-User Evaluations of Critiquing-based Search Tools	93

5.1	Introduction	93
5.2	The CritiqueShop Evaluation Platform	94
5.3	Real-User Evaluation Trial 1	95
5.4	Real-User Evaluation Trial 2	96
5.5	Evaluation Results	98
5.5.1	Interaction Efficiency	98
5.5.2	Recommendation Accuracy	100
5.5.3	User Experience	102
5.6	Summary	104
6	Visual Interface for Compound Critiquing	113
6.1	Introduction	113
6.2	Related Work	114
6.3	Interface Design	115
6.3.1	Textual Interface	116
6.3.2	Visual Interface	117
6.4	Real-User Evaluation Trial 3	119
6.4.1	Evaluation Criteria	119
6.4.2	Evaluation Setup	120
6.4.3	Datasets and Participants	122
6.5	Evaluation Results	123
6.5.1	Recommendation Efficiency	123
6.5.2	Recommendation Accuracy	124
6.5.3	User Experience	125
6.6	Discussion	127
6.7	Summary	130
7	Recursive Collaborative Filtering	133
7.1	Introduction	133
7.2	Related Work	134
7.3	Nearest-Neighbor based Collaborative Filtering	136
7.3.1	User Similarity	136
7.3.2	Selecting Neighbors	136
7.3.3	Prediction Computation	137
7.4	The Recursive Prediction Algorithm	137
7.4.1	An Illustrative Example	137
7.4.2	The Strategies for Selecting Neighbors	139
7.4.3	The Recursive Prediction Algorithm	140
7.5	Evaluation	143

CONTENTS

7.5.1	Setup	143
7.5.2	Evaluation Metrics	143
7.5.3	Experimental Results	144
7.6	Discussion	150
7.7	Summary	151
8	Conclusions	153
8.1	Contributions	154
8.1.1	Methodology for Performance Evaluation	154
8.1.2	Algorithm for Generating Compound Critiques	154
8.1.3	Visual Representation of Compound Critiques	155
8.1.4	Improvement on Collaborative Filtering Approach	156
8.2	Limitations	156
8.3	Future Research Directions	157
8.3.1	Generating Diverse Compound Critiques	157
8.3.2	Collaborative Critiquing	158
8.3.3	Adaptive Interfaces for Preference Elicitation	158
8.3.4	Handling Implicit Preferences	159
8.4	Summary	159
	Bibliography	161
	A Publication List	171
	B Curriculum Vitae	175

List of Tables

1.1	Comparison of the two performance evaluation methods: Simulation vs. Real-User Study	8
2.1	Some sample apartments in the example.	19
2.2	Comparing the styles of preference elicitation	23
2.3	The importance relationship table for the AHP approach.	48
3.1	Interaction effort analysis of decision strategies	63
4.1	The example laptop dataset	85
4.2	Critique patterns for the products.	86
4.3	The utility values of the products in the example laptop dataset	87
5.1	Design of Trial 1 (Sept. 2006)	95
5.2	The datasets used in the online evaluation of the dynamic critiquing product search tools.	96
5.3	Design of Trial 2 (Nov. 2006)	97
5.4	Demographic characteristics of participants	97
5.5	Evaluation Questionnaire	103
6.1	Post-Stage Assessment Questionnaire	121
6.2	Final Preference Questionnaire	122
6.3	Demographic characteristics of participants (Trial 3)	123
6.4	Design of the real-user evaluation for Trial 3	124

LIST OF TABLES

- 7.1 The nearest-neighbor users for the active user $x = 1$ to predict the rating value of the item $i = 3$ 138
- 7.2 The top 20 nearest-neighbors that will be selected in the conventional user-based CF approach for the active user $x = 1$ to predict the rating value of the item $i = 3$ 139

List of Figures

1.1	An example of the form-filling style of online product search for flight tickets.	2
1.2	The screen-shot of the keyword-based online product search.	3
1.3	The general architecture of an online product search tool.	9
1.4	The thesis structure.	14
2.1	Example critiquing interaction diagram	22
2.2	Screen-shot of the Entree system (The system entry interface).	28
2.3	Tweaking in the Entree system.	29
2.4	Screen-shot of the QwikShop system that adopts the dynamic critiquing approach. It enables users to apply both unit critiques and compound critiques.	32
2.5	ISY-travel allows users to add preferences by posting soft constraints on any attribute or attribute combination in the display. Preferences in the current model are shown in the preference display at the bottom, and can be given different weights or deleted.	34
2.6	When it is not possible to satisfy all preferences completely, ISY-travel looks for solutions that satisfy as many of them as possible and acknowledges the attributes that violate preferences in red.	35
2.7	Screen-shot of the FlatFinder tool	36
2.8	Screen-shot of the MobyRek tool	37
2.9	Screen-shot of the main interface of the Apt Decision system.	38
2.10	Screen-shot of the ExpertClerck, an agent system that imitates a human salesclerk (Shimazu, 2001).	39

LIST OF FIGURES

2.11	An example of the CP-network.	46
3.1	The <i>C4</i> decision strategy	54
3.2	The architecture of the simulation environment for evaluating the performance of a given product search tool.	64
3.3	Screen-shot of the decision strategy simulation program.	66
3.4	The relative accuracy of various decision strategies when solving MADPs with different number of attributes, where m (number of alternatives) = 1,000	67
3.5	The relative accuracy of various decision strategies when solving MADPs with different number of alternatives, where n (number of attributes) = 10	69
3.6	The elicitation effort of various decision strategies when solving MADPs with different number of attributes, where m (number of alternatives) = 1,000	70
3.7	The elicitation effort of various decision strategies when solving MADPs with different number of alternatives, where n (number of attributes) = 10	71
3.8	Elicitation effort/relative accuracy tradeoffs of various decision strategies	72
4.1	Generating a critique pattern.	79
4.2	The algorithm of critiquing based on MAUT (Part I).	82
4.3	The algorithm of critiquing based on MAUT (Part II).	83
4.4	Screen-shot of the prototype system that we designed to support both unit and compound critiques.	84
4.5	The results of the simulation experiments with the PC data set and the apartment data set. (1)The average interaction cycles for the apartment data set; (2)The average interaction cycles for the PC data set; (3) the accuracy of finding the target choice within given number of interaction cycles for the apartment data set; (4) the accuracy of finding the target choice within given number of interaction cycles for the PC data set.	88
4.6	Application frequency of compound critiques generated by MAUT and the Apriori algorithm	90
5.1	Average session lengths for both approaches on the laptop dataset (Trial 1).	99

5.2	Average session lengths for both approaches on the laptop dataset (Trial 2).	100
5.3	Average session lengths for both approaches on the camera dataset (Trial 2).	101
5.4	Average search accuracy of both approaches on both datasets (Trial 2).	102
5.5	A comparison of the post-stage questionnaires from <i>Trial 1</i> and <i>Trial 2</i> .	105
5.6	The final questionnaire results.	106
5.7	Sample screen-shot of the evaluation platform (with detailed interface).	107
5.8	Screen-shot of the initial preferences (digital cameras).	108
5.9	Screen-shot of the simplified compound critiquing interface (laptop). .	108
5.10	Screen-shot of the detailed compound critiquing interface (laptop). . .	108
5.11	Screen-shot of the CritiqueShop evaluation platform: the first welcome web page at the beginning.	109
5.12	Screen-shot of the CritiqueShop evaluation platform: the web page of asking user’s personal information.	109
5.13	Screen-shot of the CritiqueShop evaluation platform: the questionnaire web page of asking users to evaluate the system that they have just tried (post-questionnaire).	110
5.14	Screen-shot of the CritiqueShop evaluation platform: the questionnaire web page of asking users to compare two systems that they have tried (final-questionnaire).	110
5.15	Screen-shot of the CritiqueShop evaluation platform: the web page of asking user to find out the product that he or she really wants from a list of all products in the dataset.	111
6.1	An illustrative example of the textual interface (above) and the visual interface (below).	116
6.2	The icons that we designed for different features of the two datasets: laptops (left) and digital cameras (right).	119
6.3	Average session lengths for both user interfaces	125
6.4	Average application frequency of the compound critiques for both user interfaces.	126
6.5	Average recommendation accuracy for both user interfaces	127
6.6	Results from the post-stage assessment questionnaire.	128
6.7	Results from the final preference questionnaire.	129
6.8	Screenshot of the interface for initial preferences (with digital camera dataset). Icons are added on the left side of features so that users could get familiar with the icon meanings.	131

LIST OF FIGURES

6.9	Screenshot of the interface for visual compound critiquing (with laptop dataset).	131
6.10	Screenshot of the visual interface for the online shopping system (with laptop dataset).	132
7.1	The recursive prediction algorithm.	141
7.2	Performance results with various neighbor sizes.	144
7.3	Performance results with various recursive levels.	146
7.4	Performance results of the combination strategy(CS) with various combination weight thresholds.	147
7.5	Performance results of the CS+ strategy with various overlap thresholds.	148
7.6	Overall performance comparison of the recursive prediction algorithm with various strategies	149

1.1 Motivations

The rapid growth of web technologies has dramatically changed, and will continue to change, our daily lives in many aspects. Currently, people are able to connect to most online websites, at any time from anywhere (such as home, office, etc), to buy cameras, organize trips, or plan vacations without the need of visiting some shops or travel agencies in person. The e-commerce services provided by these online websites allow people to carry out businesses without the barriers of time or distance. Because of these advantages, e-commerce services have grown into a huge business market and many e-commerce websites — such as Amazon.com,¹ and ebay.com,² — have become very successful. According to the Census Bureau of the United States, the U.S. retail e-commerce sales for the third quarter of the year 2007 was estimated to reach \$32.2 billion, with an annual increase of 18.9%.³

In traditional commerce, the activities are carried out directly between human individuals or organizations. For example, the buyer can enter a shop to look at the products on the shelves or ask a shop assistant for help. By comparison, in e-commerce environment, the buyer interacts with a pre-designed computer system

¹See <http://www.amazon.com/>

²See <http://www.ebay.com/>

³Data source from: <http://www.census.gov/mrts/www/data/html/07Q3.html>.

CHAPTER 1. INTRODUCTION

The screenshot shows the Swiss International Air Lines flight search interface. At the top, there is a navigation menu with 'OFFERS', 'BOOK FLIGHT', 'SERVICES', 'MILES & MORE', and 'ABOUT SV'. Below this is a progress bar with six steps: 1 Travel dates, 2 Outbound, 3 Return flight, 4 Price, 5 Personal data, and 6 Confirmation. The main form area is divided into several sections:

- From/To:** Two input fields for departure and arrival locations, each with a help icon.
- Flight Type:** Radio buttons for 'One way' and 'Direct Flights only' (checked).
- Flights:** Two calendar pickers for 'Outbound flight' and 'Return flight'. The outbound date is 20.11.2007 and the return date is 27.11.2007. Both calendars show the month of November 2007, with the selected dates highlighted in red.
- Passengers:** Three dropdown menus for 'Adults' (1), 'Children (2-11 y.)' (0), and 'Infants (0-2 y.)' (0).
- Class:** A dropdown menu set to 'Economy'.
- Airline:** A dropdown menu set to 'SWISS'.
- Buttons:** 'Show prices', 'Show flights', and a prominent red 'SEARCH FLIGHTS' button.

Figure 1.1: An example of the form-filling style of online flight search. The user is asked to input travel information such as departure/arrival locations, flight type, time, class, airline, etc. (screenshot from <http://www.swiss.com>).

to get information about the product he or she wants to buy. Normally the product information provided by the e-commerce system is far beyond any individual's effort to process without any help from the system. For example, in Amazon.com, there are 3.7 million books for sale (Anderson, 2006). There is little chance for the buyer to navigate through all the items by hand to find a specific book in which he or she is interested. According to Jacob Nielsen, the first usability principle of e-commerce is that if users cannot find the product, they cannot buy it either.⁴ As a result, *online product search* is becoming increasingly critical for helping consumers find their most preferred items in the e-commerce environment.

One common implementation of online product search is based on the *form-filling* style: the system acquires the preferences from the user by asking him or her to fill out a form. Usually the form is in a fixed style and the user must input at

⁴Source from: <http://www.useit.com/alertbox/20030825.html>

1.1. MOTIVATIONS

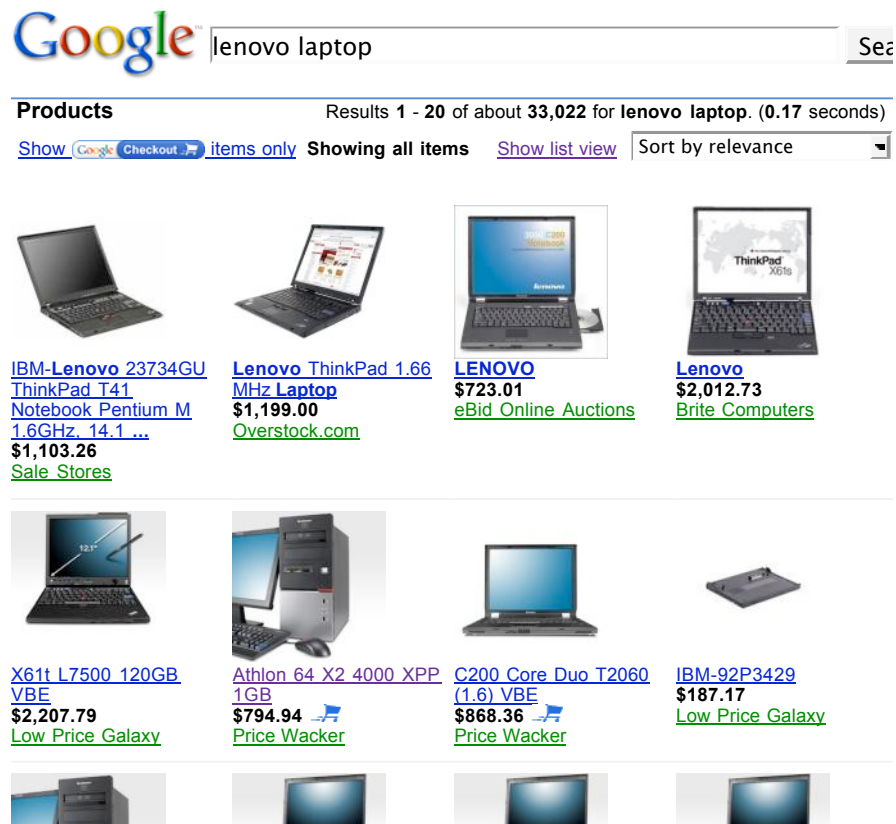


Figure 1.2: The keyword-based online product search (screen-shot from <http://www.google.com/products>).

least a certain amount of information correctly. Based on the input of the user, the system is able to generate a list of products that satisfy the conditions specified by the user. Figure 1.1 shows a detail example about this type of online product search. However, in many cases users are unable to state all their preference into the form at one time, so the search results may not be very accurate. In addition, research results have shown that individuals only have limited knowledge and computational capacity (Simon, 1955) to process the product information. On one hand, if a system provides too much information to the user, it is unlikely that the user can be patient enough to view all the product information and make the right choice. On the other hand, if the product search results contain no product, a user may feel lost during the interaction process. According to (Viappiani, Faltings, & Pu, 2006a), only 25% of users could find their most preferred products by this form-filling approach.

Another implementation of online product search is based on keywords that the

user inputs. Each time the user inputs one or a few keyword(s), and the system will return a list of products that match the keywords. For example, Google has implemented such a tool since 2002.⁵ This system, which is similar to the Google web search engine, can provide a simple way for users to input preferences and returns a huge amount of products for users to choose. However, the limitation is also obvious: it is not easy to use keywords to describe the user's preferences precisely. And because of that, search results are often inaccurate, resulting in many pages of possibilities for the end-users to examine. For example, Figure 1.2 shows a user wants to buy a *Lenovo* laptop, and he or she inputs the keywords "lenovo laptop" into the system. However, in this case the system returns some products which are not laptops. In addition, this system returns too much product information to the user without any decision assistance: the user has to spend a long time viewing the products one by one to decide which one to buy.

The main problem for the above two online product search tools is that they are *system-centric*: they implicitly assume that users have a pre-existing and stable set of preferences. They demand users to input preferences in the format as required by the system, without considering the nature of user's preferences. However, this assumption has been challenged by the studies of actual decision makers' behavior from behavioral decision theory (Payne, Bettman, & Johnson, 1993; Carenini & Poole, 2002). Many years of studies have pointed out the adaptive and constructive nature of human decision making. In particular, a user's preferences are likely to change as a result of the elicitation process itself, so that the answers given to subsequent questions are often inconsistent. Additionally, they only provide the ability for users to access all the product information that the system may have, but do not consider the user's actual effort to process all the information to make buying decisions. Studies from economics and psychology have shown that the individual only has bounded rationality when making decisions due to his/her limited knowledge and computational capacity (Simon, 1955). As a result, such system-centric design of online product search tools could only provide limited help for users to find their desired products.

1.1.1 User-Centric Online Product Search

In this thesis we propose *user-centric* online product search to overcome the above limitations. A user-centric online product search tool should be able to help users find what they want, buy what is recommended to them, and return because of

⁵This product search service was called *Froogle*, but recently Google renamed this service as *Google Product Search*. Website address: <http://www.google.com/products>

the positive interaction experience. In general, it needs to have the following key features:

1. The tool needs to let users find the desired products without too much effort. It has been shown that the users' preferences are constructive, so the tool is required to support multiple interactions between the user and the system. It should allow users to reach the desired products within a small number of interaction cycles.
2. The tool needs to provide accurate search results to end-users. Sometimes users may not have the whole domain knowledge of the products in mind at the beginning of the search process. The tool needs to provide useful product information to guide them to reach the target products gradually.
3. The tool needs to let users be confident to make purchase decisions. Most of the time, the user's preferences cannot be fully satisfied and some trade-offs must be made. The search tool is required to support trade-offs among products so that users feel confident about what they purchased.

In practice, a user-centric online product search tool plays a critical role to the success of online e-commerce websites because they could provide better usability to end-users. As Nielsen has estimated, with better usability, "an average site could increase its sales by 79%".⁶

Our goal is to build user-centric online product search tools to help users find their desired products effectively. This is challenging because 1) users' preference models are incomplete and it is hard to elicit preferences that do not exist; 2) users' beliefs about desirability are ephemeral, uncertain and context dependent; and 3) users have cognitive and emotional limitations for decision making. To that end, we need to understand the process by which humans make tradeoff decisions, how information affects this process, and how to construct effective user interfaces to augment performance. This is a multidisciplinary research that involves psychology, economics, human-computer interactions, artificial intelligence, and information retrieval.

In recent years, many types of online product search tools or systems have been proposed by researchers from different backgrounds. When the products are in low-risk domains such as books, DVDs, or news articles, some recommendation techniques have been applied to effectively generate search results to end-users (Goldberg, Nichols, Oki, & Terry, 1992; Resnick, Iacovou, Suchak, Bergstorm, & Riedl,

⁶Source from: <http://www.useit.com/alertbox/20010819.html>

1994). For example, GroupLens was developed to help users find interesting articles through increasing number of newsgroup messages based on the collaborative filtering approach (Resnick et al., 1994). In this context product search tools can be called *recommender systems*. When searching more expensive and complex products such as laptops, cars, or apartments, some decision supportive approaches (Stolze, 2000; Pu & Faltings, 2000; Torrens, Faltings, & Pu, 2002; Shearin & Lieberman, 2001) are applied to build product search tools. For example, Linden et al. (Linden, Jacobi, & Benson, 2001) described a product search tool called ATA (automated travel assistant) for finding flights. This system uses a constraint solver to obtain several optimal solutions. Each time three optimal solutions in addition to two extreme ones (least expensive and shortest flying time) are shown to the user. ATA uses a candidate critiquing agent to constantly observe the user's modifications to the expressed preferences and refine the preference model in order to improve solution accuracy. As we can see in this case, each product has its own features and a price value, and users are expected to possess a reasonable amount of willingness to interact with the system and expend a certain amount of effort to make a choice. If users' preferences cannot be fully satisfied, the system has to be decision supportive so that users can make trade-offs among them. In this context an online product search tool can also be called a *decision support system* (DSS) (Payne et al., 1993) or a *consumer decision support system* (CDSS) (Yager & Pasi, 2002). Both recommender systems and the decision support approaches will be reviewed in detail in Chapter 2.

Two issues are important for developing effective user-centric online product search tools. One issue is how to *evaluate* the performance of a given product search tool. We need to take into account the fact that users only have limited cognitive resources. The performance evaluation results are able to help system designers compare different system designs and discover potential improvement opportunities. Another issue is how to *design* a new product search tool to help end-users find their desired product efficiently and accurately. Below we will further elaborate the importance of these two issues.

1.1.2 The Performance Evaluation Issue

One method that has been widely used to evaluate the performance of product search tools is the *real-user study* method. For example in (Pu & Kumar, 2004), a real-user study was conducted to evaluate the performance of example-based search tools. Before the real-user study started, the entire platform had to be implemented. During the real-user study, a group of users was hired to complete some specific

search tasks. The interaction process of each subject was recorded in log files. After the search tasks were finished, each end-user was asked to fill a post-study questionnaire to specify whether he or she was satisfied with the search tool. Finally the performance results were obtained by analyzing the log files. This whole user-study procedure lasted more than two months with 16 users in total, and each subject has been paid with a certain amount of incentive.

There are several advantages of the real-user study method to evaluate performance. We are able to closely observe the users' actual behaviors and get their subjective feedback on criteria such as the degree of satisfaction, the willingness of purchase, etc. Also, if these users are representative and are unbiased in their cultural or educational background, the evaluation results converge onto the actual real-world performance of the system.

However, the real-user study method has some limitations as well. First of all, it takes a long time to generate evaluation feedback to system designers. The system designers have to complete and deploy the search tool, and hire a group of users to try it. The evaluation results can only be obtained after these users finish the evaluation process. Next, it is not easy to hire enough users with different backgrounds to participate in the user study. Additionally, a certain amount of incentive must be proposed to attract them. Furthermore, the evaluation results are dependent on the current real-user study conditions. If we change the scale of the underlying product information by adding or deleting some products, the performance results may also be changed and new real-user studies are required.

An alternative method is to evaluate the performance of a given product search tool through *simulation*. We can create one or several artificial user(s) with a certain kind of behaviors, and mimic the interaction procedure between the artificial user(s) and the given product search tool. By analyzing the *artificial* interaction log files, we are able to largely estimate the performance of the given product search tool.

The simulation method enjoys some benefits compared to the real-user study method. It is much faster to carry out the simulation experiments to generate the performance results; system designers don't need to wait for real users to complete the search tasks. Also, there is no cost for hiring real users. Additionally, it is easy to simulate the process of the product search tool with different scales of datasets. The drawback of the simulation method is that the performance results may not be very convincing because of the gap between the artificial user and the real ones. Also, it is impossible to obtain the user's subjective feedback with the simulation method. Table 1.1 shows a comparison of these two evaluation methods.

The simulation method has been applied in different situations in past years.

Table 1.1: Comparison of the two performance evaluation methods: Simulation vs. Real-User Study

Criteria	Simulation	Real-User Study
Effort to obtain evaluation results	low	high
Cost to obtain evaluation results	low	high
Scalability	easy	difficult
Subjective feedback	no	yes
Quality of the evaluation results	low	high

In (Payne et al., 1993), a simulation experiment is introduced to measure the performance of various decision strategies in offline situations. Boutilier et al. (Boutilier, Brafman, Domshlak, Hoos, & Poole, 2004) carried out experiments by simulating a number of randomly generated synthetic problems, as well as user responses to evaluate the performance of various query strategies for eliciting bounds of the parameters of utility functions. In (Reilly, McCarthy, McGinty, & Smyth, 2005), various users' queries were generated artificially from a set of offline data to analyze the recommendation performance of the incremental critiquing approach. These works generally suggest that simulation is a useful methodology for performance evaluation. However, these simulation experiments are not generic in nature because they are limited on the specific task. Some of them didn't measure the decision accuracy, which is an important criterion for calibrating the performance of product search tools. A more general simulation environment is required to be adopted universally for measuring the performance of any given product search tools efficiently.

Considering both the pros and cons of these two evaluation methods, we could combine them together to evaluate the performance of a given product search tool in sequence. Ideally we can first use the simulation method to estimate its performance after the prototype is implemented. Then if we find that this tool is quite efficient according to the simulation results, we can complete the implementation of the system and launch a real-user study to verify its performance.

1.1.3 The System Design Issue

The system design issue is important for us to develop new online product search tools to help users find the desired products efficiently. Essentially an online product search tool is an information system with a client-server model, and the 3-tier layer architecture is widely accepted as the system structure. The benefit of this

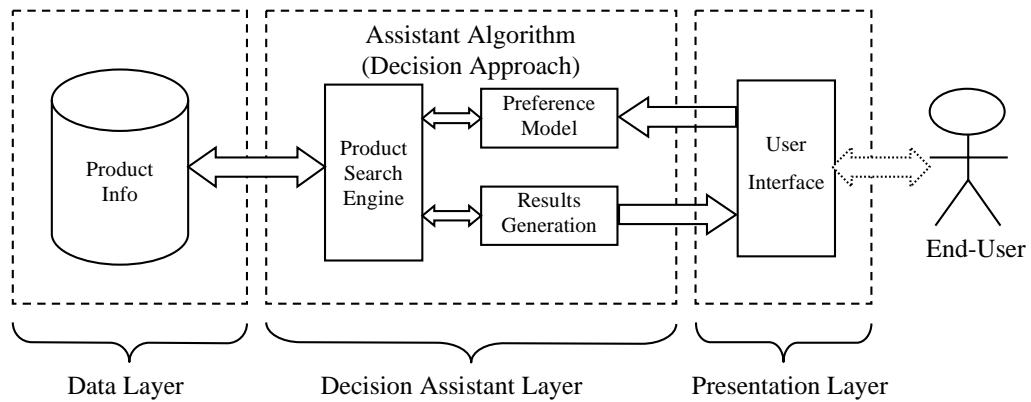


Figure 1.3: The general architecture of an online product search tool.

architecture is that the presentation, the logic and the data storage of the tool are independent to each other. Figure 1.3 shows the general architecture of an online product search tool with the following three specific layers.

- **The data layer.** This layer is responsible for storing and accessing product information required by the decision assistant layer. This layer is dependent on the particular product domain.
- **The decision assistant layer.** This is the logic layer of the system. In each interaction cycle, it accepts the user's various preferences into a preference model, and then the product search engine will determine a list of products that best match the user's preferences according to a certain criteria, and finally the results generation model sends the product information to the user interface model for display. Typically this procedure can be described by a decision assistant algorithm (or decision approach). This layer can also be called *algorithm* layer.
- **The presentation layer.** This layer can also be called the *user interface* layer. It is responsible for displaying the search results generated from the decision assistant layer properly to end-users. For online product search tools, a typical example of this layer is some web pages displayed on a web browser such as Internet Explorer or FireFox.

Under this architecture, the algorithm layer and the user interface layer are not directly dependent on the specific product domain, so they can be designed in a

CHAPTER 1. INTRODUCTION

general way and can be applied to different product domains. In other words, we are able to apply different algorithms and/or user interfaces to build different product search tools on a given product domain.

In past years many different algorithms or decision approaches have been proposed for developing various online product search tools. The SmartClient (Pu & Faltings, 2000) is a tool based on the example-critiquing approach for users to make a travel plan. It is able to refine the user's preference model interactively by showing a set of 30 possible solutions in different visualizations to the user. In (Stolze, 2000), the scoring tree method is proposed for building interactive product search tools based on the multi-attribute utility model (MAUT) (Keeney & Raiffa, 1976). A detailed review of these approaches is given in Chapter 2.

Particularly, the critiquing technique has been applied on many systems and has been proven to be a successful approach for online product search because it can help users express their preferences and feedbacks easily over one or several aspects of the available product space (Burke, Hammond, & Young, 1997; Reilly, McCarthy, McGinty, & Smyth, 2004a; Reilly et al., 2005; Faltings, Pu, Torrens, & Viappiani, 2004a; Ricci & Nguyen, 2005). In Chapter 2 we also review these critique-based tools in detail. However, these critique-based product search tools are implemented in different ways, generally lacking direct performance comparison. Some improvements can be made to help users find desired products more effectively.

For the user interface layer, one important task is to elicit user's preferences as requested by the algorithm layer. For example, if the unit critiquing technique is applied on an online product search tool, the user interface layer needs to provide the function for users to critique on different values. In addition, it is important to present the search results properly to enhance user's overall satisfaction degree.

Overall, in this thesis we tackle both the *design* and *evaluation* issues for user-centric online product search. It is worth mentioning that both the design and evaluation issues are tightly-coupled: on one hand, evaluation results can help system designers discover new possibilities to refine the system design; on the other hand, the efficiency of new design approaches are required to be validated by evaluation results. More specifically, our main work is to design a user-centric online product search based on the critiquing technique. This tool has two novelties: the user-centric algorithm based on the MAUT approach to generate compound critiques, and the visual interface for presenting compound critiques to users. We also evaluate the product search tool with both simulation experiments and real-user studies.

1.2 Contributions

The contributions of this thesis lie in three aspects. First, we propose a general performance evaluation framework for online product search tools. We identify three criteria from users' point of view: cognitive effort, interaction effort and choice accuracy. We specify the method of evaluating the performance of given online product search tools in a simulation environment. Second, we implement the *CritiqueShop* system, an online product search tool based on the well-known critiquing technique with two novelties: a user-centric compound critiquing generation algorithm which generates search results efficiently and a visual user interface for enhancing user's satisfaction degree. Both the algorithm and the user interface are validated by large-scale comparative real-user studies. Finally, collaborative filtering is an approach that has been widely used to recommend products based on user's rating profiles. Here we present a *recursive* collaborative filtering approach to improve the recommendation accuracy without requiring additional effort from end-users.

1.2.1 Performance Evaluation Framework

In this part of work, our main objective is to develop a simulation environment in which various search tools are evaluated in terms of interaction behaviors: what users' effort would be to use these tools and what kind of benefits they are likely to receive from these tools. We propose an extended effort–accuracy framework for quantitatively measuring the performance of different decision strategies in decision support environments. The method of measuring the effort of preference elicitation was given and a variety of decision strategies were then evaluated through simulation experiments.

We base our work on some earlier research (Payne et al., 1993) about the design of the simulation environment in off-line situations. However, we have added important elements to adapt such environments to online e-commerce and consumer decision support scenarios. With this simulation environment, we are able to forecast the acceptance of online product search tools in the real world and curtail the evaluation of each tool's performance from months of user study to rapid simulation process. This allows us to evaluate new tools efficiently and, more importantly, discover design opportunities of new search tools.

1.2.2 Critique-based Product Search

Critiquing is an interactive technique allowing users to construct their preferences through cycles of feedbacks collected based on users' critiques on search results. It is a popular preference elicitation mechanism that has been used in various online product search tools. Currently several different kinds of critiquing methods have been proposed and implemented to let users state their preferences. The simplest form of critiquing is *unit critique*, which allows users to give feedback on a single attribute or feature of the products at a time (Burke et al., 1997). For example, [*CPU Speed: faster*] is a unit critique over the *CPU Speed* attribute of the PC products. If a user wants to express preferences on two or more attributes, multiple interaction cycles between the user and the system are required.

To make the critiquing process more efficient, an alternative strategy is to adopt *compound critiques*, which are collections of unit critiques and allow users to indicate a richer form of feedback. Reilly et al. (Reilly et al., 2004a) have developed an approach called dynamic critiquing to generate compound critiques through the Apriori algorithm. The Apriori algorithm is a data mining approach used in the *market-basket analysis* method (Agrawal & Srikant, 1994). It treats each critique pattern as the shopping basket for a single customer, and the compound critiques are the popular shopping combinations that consumers often purchase together.

The Apriori algorithm is efficient in discovering compound critiques from a given data set. However, selecting compound critiques according to their frequency in the data set may lead to some problems. This approach can reveal "what the system would provide", but does not tell "what the user likes". For example, in a PC data domain if 90 percent of the products have a faster CPU and larger memory than the current reference product, it is still unknown whether the current user likes a PC with a faster CPU and larger memory. If the users find that the compound critiques cannot help them find better products within several interaction cycles, they may be frustrated and give up the interaction process.

In this thesis we propose a new algorithm to generate compound critiques for online product search with a preference model based on the multi-attribute utility theory (MAUT) (Keeney & Raiffa, 1976). In each interaction cycle our approach first determines a list of products via the user's preference model, and then generates compound critiques by comparing them with the current reference product. In our approach, the user's preference model is maintained adaptively based on user's critique actions during the interaction process, and the compound critiques are determined according to the utilities they gain instead of the frequency of their occurrences in the data set.

We further extend the user interface design for critique-based product search tools. Traditionally the user interface for compound critiques are presented in a simple style with plain text. We propose a visual interface to represent compound critiques with various meaningful icons.

We build an online evaluation platform called *CritiqueShop* so that real users can evaluate alternative versions of algorithms/interfaces of online product search. The results from real-user studies validate the efficiency of both the algorithm and the new designs of user interfaces.

1.2.3 Recursive Collaborative Filtering

One of the most popular and successful techniques that has been used in generating recommendation set for low-risk product domains is known as *collaborative filtering* (Herlocker, Konstan, Borchers, & Riedl, 1999; Resnick et al., 1994). The key idea of this approach is to infer the preference of an active user towards a given item based on the opinions of some similar-minded users in the system (Breese, Heckerman, & Kadie, 1998; Herlocker, Konstan, & Riedl, 2002).

The conventional prediction process of the user-based collaborative filtering approach selects neighbor users using two criteria: 1) They must have rated the given item; 2) They must be quite close to the active user (for instance, only the top K nearest-neighbor users are selected). However, in reality most users in recommender systems are unlikely to have rated many items before starting the recommendation process, making the training data very sparse. As a result, the first criterion may cause a large proportion of users being filtered out from the prediction process even if they are very close to the active user. This in turn may aggravate the data sparseness problem.

To overcome the data sparseness problem and enable more users to contribute in the prediction process, we propose a recursive prediction algorithm which relaxes the first criterion mentioned above. The key idea is the following: if a nearest-neighbor user hasn't rated the given item yet, we will first estimate the rating value for him or her *recursively* based on his or her own nearest-neighbors, and then we use the estimated rating value to join the prediction process for the final active user. In this way we have more information to contribute to the prediction process and it should be able to improve the prediction accuracy for collaborative filtering recommender systems.

The main contribution of this part of the work is that we relax the constraint that all nearest-neighbor users must also have rated the given item. The recursive

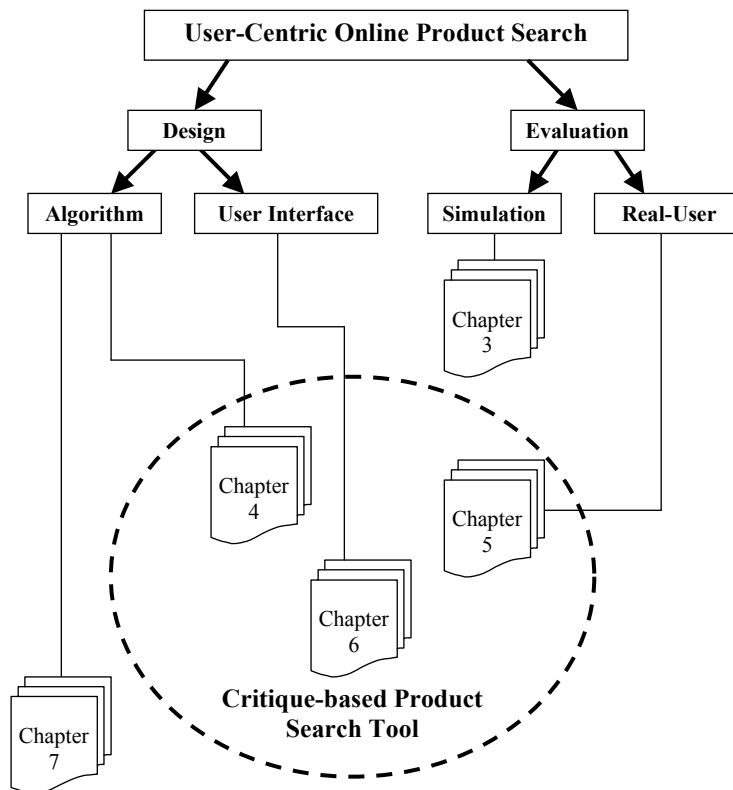


Figure 1.4: The thesis structure.

prediction algorithm enables more flexibility in the prediction process of finding the useful neighbor users. This algorithm is able to improve the recommendation accuracy without requiring any additional effort from users.

1.3 Thesis Outline

The rest of this thesis is organized as follows. Very briefly, Chapter 2 provides a review of the background and related work. Chapter 3 introduces a general performance evaluation framework with simulation method. Chapter 4 – 6 are our main work on the design and evaluation issues for a user-centric online product search tool. In Chapter 7, we introduce an improvement of the collaborative filtering algorithm. Chapter 8 is a summary of the thesis. The structure of the main work of this thesis is shown in Figure 1.4.

Chapter 2 reviews the state of the art background research work in the field of

online product search. We define the multi-attribute decision problem and introduce the example-critiquing interaction paradigm to solve this problem. Different approaches of generating product search results are reviewed, from the decision theoretic approaches to the recommendation techniques. We also review the critiquing techniques comprehensively for online product search.

In Chapter 3 we develop a general evaluation framework for online product search tools. We identify three criteria for assessing the quality of a given product search tool: users' cognitive effort, interaction effort, and decision accuracy. Based on this framework, we propose a simulation environment which enables rapid evaluation of decision strategies for online product search tools.

In Chapter 4 we introduce a novel algorithm of generating compound critiques dynamically. This algorithm is based on the MAUT approach and can generate compound critiques that are close to the user's preference model. Here we also report simulation experiment results to show its efficiency.

Chapter 5 reports the online real-user studies for evaluating the performance of the algorithm proposed in Chapter 4. We validate that the algorithm reaches a good performance from the user's perspective.

In Chapter 6 we develop a visual user interface to represent compound critiques so to enhance user's satisfaction degree during the product search process. A user study is carried out to validate this visual design.

Chapter 7 presents our work on improving the classical collaborative filtering recommendation algorithm. We propose the recursive collaborative filtering algorithm to generate recommendation results more accurately.

Finally, we summarize the thesis and discuss future research directions in Chapter 8.

Background and Related Work

2.1 Introduction

Generally speaking, an online product search tool is an information system that helps buyers find their desired products from a collection of product descriptions that an organization wants to offer. Usually the products offered in an e-commerce system are far more than what the individual decision maker requires. Studies from economics and cognitive psychology have shown that an individual has only a bounded rationality when making decisions due to limited knowledge and computational capacity (Simon, 1955). Therefore, the product search results should be highly selective, only containing products that best match users' preferences. Ideally an online product search tool should be *user-centric*: it should be able to help users find their desired products accurately with little effort.

In this thesis we focus on a specific category of e-commerce systems called electronic product catalog (EPC) (Palmer, 1997; Torrens, 2002), which provides a list of products for the buyer to select. Each product is represented by a number of attributes. The buyer needs to choose the product that most closely satisfies his or her preferences. In most cases these preferences cannot be fully satisfied and some tradeoffs have to be made between different attributes (Pu & Faltings, 2004). The user-centric online product search tools are required to be applied in this context to assist end-users.

In this chapter, we first give a formal definition of the decision problem that we are aiming to solve. Then we explore the nature of users' preferences and the preference elicitation process. Particularly, we highlight that critiquing is the style of acquiring user's preferences that balances the quality of generating search results and the effort required from the user. Next, we introduce the multi-attribute decision theory, which is the main theoretic approach for modeling user's preferences in this thesis. In section 2.5 we review the existing critique-based product search tools. Section 2.6 introduces recommendation techniques, which are closely related to the research topic of online product search. Finally in Section 2.7 some other decision making approaches are also reviewed.

2.2 Multi-Attribute Decision Problem

The process of choosing the most preferred product from a given EPC can be formally described as solving a Multi-Attribute Decision Problem (MADP) defined as below.

Definition A Multi-Attribute Decision Problem (MADP) is a tuple $\Psi = \langle \mathbf{X}, \mathbf{D}, \mathbf{O}, \mathbf{P} \rangle$, where

- $\mathbf{X} = \{X_1, \dots, X_n\}$ is a finite set of attributes the product catalog has,
- $\mathbf{D} = D_1 \times \dots \times D_n$ indicates the space of all possible products in the catalog (each $D_i (1 \leq i \leq n)$ is a set of possible domain values for attribute X_i),
- $\mathbf{O} = \{O_1, \dots, O_m\}$ is a finite set of available products (also called alternatives or outcomes) that the EPC offers, and
- $\mathbf{P} = \{P_1, \dots, P_t\}$ denotes a set of preferences that the decision maker may have. Each preference P_i may be identified in any form as required by the solution methods.

The solution of a MADP is an alternative O most satisfying the decision maker's preferences. Two types of problems are raised when trying to solve a MADP. One is to find one *optimal* solution among the outcome set which best matches the decision maker's preferences. We call this problem the "optimal" problem. In this case the search result will be the specific solution. The other one is to find a list of candidates with ranking order, which can be called the "ranking" problem. In this case the search result will be the candidate list.

2.2. MULTI-ATTRIBUTE DECISION PROBLEM

To illustrate the MADP, here we describe a concrete example in the apartment-renting domain. Suppose we are designing an e-commerce system which provides the service of apartment renting, and to simplify our discussion, we assume that the apartments provided by this system only have 5 distinct attributes: *Type*, *Kitchen*, *Bathroom*, *Size*, and *Price*, and each attribute may take a certain set (or range) of values as listed below:

$$D_{Type} = \{room\ in\ a\ house, apartment, studio\}$$

$$D_{Kitchen} = \{private, share, none\}$$

$$D_{Bathroom} = \{private, share, none\}$$

$$D_{Size} = [20, 200]m^2$$

$$D_{Price} = [300, 4000]CHF$$

In this example, the set of available outcomes O is the list of the apartments provided by the system. It is natural to notice that O is only a subset of the total possible outcome space D . For instance, the apartment with both the biggest area size and the lowest price is a possible outcome in D , but most likely it cannot be offered by the e-commerce system. Table 2.1 gives some sample apartments that the MADP may contain.

Table 2.1: Some sample apartments in the example.

ID	Type	Kitchen	Bathroom	Size(m^2)	Price (CHF)
O_1	apartment	private	private	45	1000
O_2	studio	public	private	18	600
O_3	room in a house	public	public	20	450

There are two important requirements for solving a given MADP. One is to obtain the decision maker's preferences accurately. Some preferences can be generated by the commonsense held by most individuals, for example: "*Other things being equal, the cheaper the better*", or "*if everything else be equal, I prefer the apartment with bigger area*". But the system still needs to acquire the user's personalized preferences through an elicitation process.

The second requirement is to adopt a decision approach to generate the product search results according to the preferences acquired from the user. During past years various approaches have been proposed for this task. These approaches will be reviewed shortly after.

2.3 User's Preferences

As we have pointed out earlier, currently some online product search tools has followed an algorithm-centric approach; they often make the assumption that users can readily articulate their preferences accurately and consistently. Therefore, accurate algorithms are sufficient to help users identify their truly preferred product. However, psychological studies have shown that most people are unable to express preferences directly and their decision behavior are very adaptive to the environment (Payne et al., 1993).

Tversky et al. (Tversky & Simonson, 1993) reported a user study about asking subjects to buy a microwave oven. Participants were divided into 2 groups with 60 users each. In the first group, each user was asked to choose between two products: an Emerson priced at \$100 and a Panasonic priced at \$180. Both products were on sale, taking a third off the regular price. In this case 43% users chose the Panasonic at 180\$. A second group was presented with the same two products, along with a third product which is also Panasonic, but with price \$200 at a 10% discount. In this context, 60% of the users chose the Panasonic priced at \$180. This finding suggests that users' preferences are context-dependent and are constructed gradually as a user is exposed to more information regarding his or her desired product.

In online decision making environments, the way to obtain users' preferences during the interaction process is a fundamental issue for the system design. Most existing systems *elicit* preferences through a series of questions whose answers precisely define the user's preferences. For example, a travel planning tool such as Travelocity¹ asks each user several questions about the itinerary and time and airline preferences, and then returns a set of possible choices based on the resulting preference model. Certain e-commerce sites go further and guide the user through a fixed sequence of questions that determine the final choice. Elicitation through questions is the method proposed in classical decision theory (Keeney & Raiffa, 1976) and research continues on improving its performance (Boutilier et al., 2004). Such elicitation processes implicitly assume that users have a pre-existing and stable set of preferences.

However, this assumption has been challenged by the studies of actual decision makers' behavior from behavioral decision theory (Payne et al., 1993; Carenini & Poole, 2002). Many years of studies have pointed out the adaptive and constructive nature of human decision making. In particular, a user's preferences are likely to change as a result of the elicitation process itself, so that the answers given to

¹Website: <http://www.travelocity.com>

subsequent questions are often inconsistent. The product search tools should be carefully designed to avoid conflicting with these known theories.

2.3.1 Principles of Preference Elicitation

Pu et al. (Pu, Faltings, & Torrens, 2004) pointed out the following principles of the preference elicitation based on the study of the decision behavior theory (Payne et al., 1993):

- Users are not aware of all preferences until they see them violated. For example, a user does not think of stating a preference for intermediate airport until a solution includes a change of airplane in a place that he dislikes. This cannot be supported by the decision tool that requires preferences to be stated in a predefined order.
- Elicitation questions that do not concern the user's true objective can force him to formulate means objectives corresponding to the question. For example, in a travel planning system suppose that the user's objective is to be at his destination at 15:00, but that the tool asks him about the desired departure time. The user might believe that the trip necessarily involves a plane change and take about 5 hours, and thus forms a means objective to depart at 10:00 to answer the question. However, the best option might be a new direct flight that leaves at 12:30 and gets there at 14:30. This solution would not be found using the elicited preference model. This phenomenon has been studied by Keeney (Keeney, 1992) in his work on value-focused thinking.
- Preferences are often in contradiction and require users to make tradeoffs, which require users to add, remove or change preferences initiatives in any order at any time.

To support these properties of human decision making, product search tools are required to have a preference model to support incremental construction and revision of preferences by users.

2.3.2 The Interaction Paradigm

Preference construction must be supported by feedback indicating the influence of the current model on the outcomes. A good way to implement such feedback is to

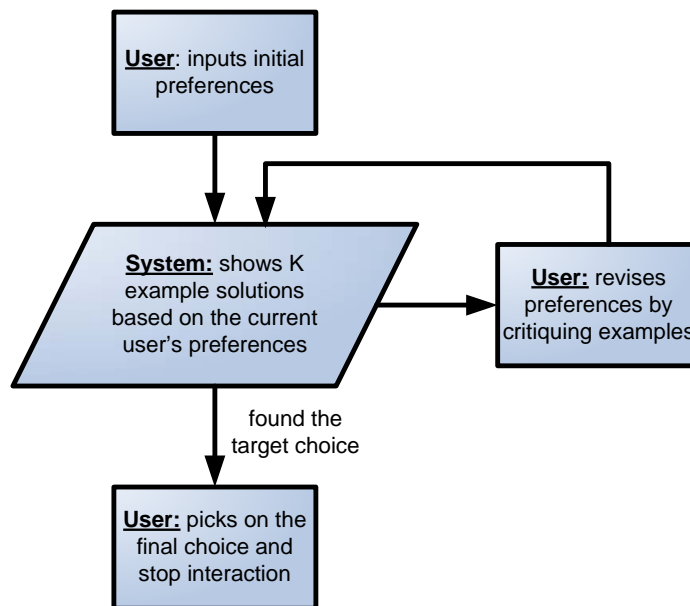


Figure 2.1: Example critiquing interaction diagram. The blue(dark) box is the computer's action, the other boxes show actions of the user.

structure user interaction as mixed-initiative systems (MISs). MISs are interactive problem solvers where human and machine intelligence are combined for their respective superiority (Allen, Schubert, Ferguson, Heeman, Hwang, Kato, Light, Martin, Miller, Poesio, , & Traum, 1994; Horvitz, 1999). MISs are therefore good candidates for such incremental decision systems.

A good way to implement a mixed-initiative decision support system is *example critiquing* interaction (see Figure 2.1). It shows examples of complete solutions and invites users to state their critiques of these examples. Example critiquing allows users to better understand the impact of their preferences. Moreover, it provides an easy way for the user to add or revise her preferences at any time in arbitrary order during the decision making process. Example-critiquing as a preference elicitation method has been proposed by a variety of researchers (Burke et al., 1997; Linden, Hanks, & Lesh, 1997; Shearin & Lieberman, 2001; Faltings et al., 2004a), and its performance has been evaluated in (Pu & Kumar, 2004; Pu & Faltings, 2004). In this thesis we regard the example-critiquing paradigm as the principal interaction style for user-centric online product search tools.

Table 2.2: Comparing the styles of preference elicitation (Smyth & McGinty, 2003)

Style	Cost	Ambiguity	Expertise	Interface
Value Elicitation	***	*	***	***
Item-based	*	***	*	*
Ratings-based	**	***	**	*
Critiquing	**	**	**	*

2.3.3 Preference Elicitation Styles

Users' preferences can be elicited in various styles. For example, the system can either ask the user to input some value to indicate his or her likeness to a given product, or can ask users to compare a given product with another one. It is important for the system to provide flexible preference elicitation styles because most of the time users are not fully aware of what their preferences and unable to fully articulate them to the system.

Smyth and McGinty (Smyth & McGinty, 2003) have compared four preference elicitation styles in four evaluation dimensions: *cost*, *ambiguity*, *expertise*, and *interface*. They highlight their relative pros and cons and indicate the conditions under which each is most appropriate. Table 2.2 gives a brief summary of these four styles across these dimensions. These styles of preference elicitation are introduced briefly as below.

Value Elicitation

Value elicitation is perhaps the most common form of preference acquirement. With this form, users specify preferred feature values e.g. "I want a digital camera with 5M Pixels of resolution". From an implementation perspective, this is perhaps the easiest form of preference elicitation. The system can just provide a form-filling style of user interface to elicit these values. The system can directly convert such preference value into a SQL query and execute it to generate search results. However, this style of preference elicitation requires users to clearly express their requirements in terms of specific feature values and conflicts with the nature of user's preferences. As it has been pointed out earlier, the search tool based on this style of preference elicitation can only gain 25% accuracy (Viappiani et al., 2006a).

Item-based

On every interaction cycle, this style of preference elicitation asks the user to select the most preferable item among a number of candidates provided by the system. Unlike the value elicitation style, the item-based style² is low cost and not requires domain expertise. It is relatively easy to produce an interface for this style of preference elicitation – it simply needs the space to display some recommended products and requires the user to select the preferred one. However, one drawback of the item-based style of preference elicitation is that it is ambiguous; the preferred products selections do not convey much preference information, leaving the system to figure out what the user’s actual preferences are.

Ratings-based

The MovieLens system (Herlocker, Konstan, & Riedl, 2000) gathers users’ preferences with a rating-based style. The system asks users to specify a *rating* to each given item, where a rating of 1 means users dislike the movie and a rating of 5 means the movie is liked. The intermediate ratings allow the user to specify the degree of like or dislike to the movie. This form of preferences is common in collaborative recommender systems where ratings are used to compute similarities in user tastes. Ratings can be considered as a low-cost style of preference elicitation. Users are not necessarily required to know the feature details when assigning a rating to an item. However, the user needs to think about the correct rating to apply to a product and to be consistent in how they rate items. Commonly, in collaborative recommender systems, users have to rate many items before they receive suitable recommendations.

Critiquing

The FindMe family of product search tools (Burke, Hammond, & Young, 1996; Burke et al., 1997; Burke, 2002) introduced the form of preference elicitation called *tweaking* or *critiquing*. More recently, the ExpertClerk system (Shimazu, Shibata, & Nihei, 2001; Shimazu, 2001) also incorporates critiquing as a form of preferences. Put simply, a critique allows a user to express a directional preference on a feature value. For example, when shopping for a PC, a user might select a critique for a *faster processor* – a critique on the *processor speed* feature.

²This style is also called *preference-based* user feedback in (Smyth & McGinty, 2003).

As we can see from Table 2.2, the critiquing style provides a good balance among these criteria. Critiquing only requires users to have minimal familiarity with the product domain, and could provide guidance to users to reach the final target gradually. The critiquing technique has been recognized as an effective approach for building online product search tools. The critique-based product search tools are reviewed in detail in Section 2.5.

2.4 Multi-Attribute Utility Theory

A theoretic model is required to model users' preferences and generate search results properly. In literature the multi-attribute utility theory (MAUT) has been applied to solving decision problems effectively (Keeney & Raiffa, 1976). In this thesis we consider it as the main theoretic model for building our product search tool. In this section, we review the MAUT approach in detail.

The origination of utility theory can be dated back to 1738 when Bernoulli proposed his explanation to the *St. Petersburg paradox* by the terms of utility of monetary value (Bernoulli, 1954). Very briefly, the St. Petersburg paradox is a game of asking people how much they would pay for playing it with the following rules: A fair coin (with two sides *head* and *tail*) will be tossed repeatedly until a *tail* first appears to end the game. If a head comes out of the first toss, the player receives two dollars and stay in the game; if a head comes out again in the second toss, the player receives four dollars and stay in the game; and so on until the game stops (e.g. a *tail* appears). In short, the player wins 2^{k-1} dollars if the coin is tossed k times until the first tail appears. The expected monetary value of this game is infinite: $\sum_{k=1}^{\infty} 2^{k-1} \times \frac{1}{2^k} = \infty$, but most people only want to pay a small amount of money for this game. Bernoulli argued that people estimate the gains of playing this game by utility value, not monetary value. He suggested to use the logarithmic function $u(x) = \ln(x)$ as the utility function, and the expected utility of this game is finite ($\sum_{k=1}^{\infty} u(2^{k-1}) \times \frac{1}{2^k} = \sum_{k=1}^{\infty} \ln(2^{k-1}) \times \frac{1}{2^k} < \infty$).

Two centuries later in 1944 it was von Neumann and Morgenstern who revived this method to solve problems they encountered in economics (von Neumann & Morgenstern, 1944). They proved that the preference relation over a finite set of states could be written as an expected utility. Later in the early 1950s, Marschak (Marschak, 1950) and Herstein and Milnor (Herstein, I. N. & Milnor, John, 1953) established the Expected Utility Theory based on the von Neumann Morgenstern theorem.

CHAPTER 2. BACKGROUND AND RELATED WORK

In 1970s Keeney and Raiffa (Keeney & Raiffa, 1976) extended the utility theory to the case of multi-attributes. The main idea of multi-attribute utility theory (MAUT) is that the user's preferences over some items or outcomes with multi-attributes can be represented as a utility function.

Let the symbol \succsim denote the user's preference order, e.g. $A \succsim B$ means "A is preferred or indifferent to B". According to MAUT, for a given MADP, there exists a utility function $U : O \rightarrow \mathfrak{R}$, that for any two possible products O and $\bar{O} \in O$,

$$O \succsim \bar{O} \iff U(O) \geq U(\bar{O}) \quad (2.1)$$

More specifically, a product O can be represented by a set of attribute values $\langle X_1 = x_1, \dots, X_n = x_n \rangle$ (in short as $\langle x_1, \dots, x_n \rangle$), thus the above formula can be rewritten as

$$\langle x_1, \dots, x_n \rangle \succsim \langle \bar{x}_1, \dots, \bar{x}_n \rangle \iff U(\langle x_1, \dots, x_n \rangle) \geq U(\langle \bar{x}_1, \dots, \bar{x}_n \rangle) \quad (2.2)$$

Usually the utility function U is scaled from zero to one. If the utility function is given, the likeness of each product will be calculated and the preference order of all products can be determined according to the utility values they gain.

Finding the proper utility function U to represent users' preferences precisely is a challenging task. Theoretically it could be in any form such as linear, exponential, logarithmic, or their combinations, etc. In practice a special case of the utility function is commonly used to reduce computational effort.

Before we further introduce the utility function, we give definitions of two important concepts below.

Definition Suppose $\mathbf{Y} = \{Y_1, \dots, Y_k\}$ is a subset of the attribute set \mathbf{X} in a MADP, and $\mathbf{Z} = \{Z_{k+1}, \dots, Z_n\}$ is the complementary set of \mathbf{Y} (e.g. $\mathbf{X} = \mathbf{Y} \cup \mathbf{Z}$). The set of attributes \mathbf{Y} is **preferentially independent (PI)** of its complementary set \mathbf{Z} if and only if for some given value set $\mathbf{z}' = \{z'_{k+1}, \dots, z'_n\}$ and any given two value sets $\mathbf{y}' = \{y'_1, \dots, y'_k\}$ and $\mathbf{y}'' = \{y''_1, \dots, y''_k\}$,

$$\langle \mathbf{y}', \mathbf{z}' \rangle \succsim \langle \mathbf{y}'', \mathbf{z}' \rangle \implies \langle \mathbf{y}', \mathbf{z} \rangle \succsim \langle \mathbf{y}'', \mathbf{z} \rangle, (\text{ for all } \mathbf{z}). \quad (2.3)$$

Definition The attributes X_1, \dots, X_n are **mutually preferentially independent (MPI)** if every subset \mathbf{Y} of these attributes is preferentially independent of its complementary set.

MPI is a very strong condition among the attributes; basically it says that the preference order on the values of each attribute will not be influenced by values of other attributes. Once the MPI is hold, it can be proven that the utility function can be decomposed into a simplified form according to the following theorem (Keeney & Raiffa, 1976).

Theorem 2.4.1 *Given attributes X_1, \dots, X_n , an additive utility function*

$$U(\langle x_1, \dots, x_n \rangle) = \sum_{i=1}^n w_i v_i(x_i) \quad (2.4)$$

exists if and only if the attributes are mutually preferentially independent ((where v_i is a value function over X_i ranged in $[0, 1]$, and w_i is the weight value of attribute X_i satisfying $\sum_{i=1}^n w_i = 1$) (Keeney & Raiffa, 1976).

In this case the utility function is able to be determined based on user's preferences. The weight value for each attribute can be given as $1/n$ by default, and we can allow the user to specify the weight values for some attributes. The value function v_i for each attribute can be determined to satisfy user's preferences related to the attribute X_i . Usually a linear function is enough to represent user's preference on each attribute.

Once the utility function is determined, we are able to rank all the items based on their overall utilities and select the top K products with the highest utility as the search results. In practice, we assume that the attributes in the decision problem are mutually preferentially independent, so the additive form of utility function can always be applied.

The MAUT approach can enable users to make tradeoff among different attributes of the product space and has been used in previous product search tools. For example, Stolze has proposed the scoring tree method for building interactive e-commerce systems based on MAUT (Stolze, 2000). In this system a users is able to express his or her preferences by modifying the values on an existing tree, and the system will translate user's preferences into a MAUT additive utility function, and then calculate the utility value of each product in the system. Finally the search results are determined by the utility values.



Figure 2.2: Screen-shot of the Entree system (The system entry interface).

2.5 Critique-based Search Tools

Critiquing technique provides an easy way for users to reveal their preferences over one or several attributes of the products in a electronic product catalog. It is intuitive for users to convey a sufficient amount of preference information. In this section we review those well-known critiquing-based online product search tools.

2.5.1 The FindMe Systems

The FindMe systems were the first to employ critiquing technique for assisting product browsing through a give electronic catalog (Burke et al., 1996, 1997; Burke,



Figure 2.3: Tweaking in the Entree system.

2002). The user is able to navigate through some candidate products and tweak on the different criteria until the desired product is found. In fact, FindMe represents a series of systems that have applied critiquing technique to various domains. *Car Navigator* is a system for searching automobiles. *RentMe* is a system for users to find apartments. *PickAFlick* let users discover movies similar to the ones that they have already seen. The *Entree* system allows users to search restaurants based on factors such as cuisine, price, style, etc.

The *Entree* system provides service for users to find a desired restaurant in Chicago area. It was in operation as a Web-based application in August 1996. Figure 2.2 shows the entry point of the *Entree* system. There are two possibilities for users to start the navigation process. One way is to specify a particular restaurant that may exist in the restaurant database. Alternatively, the user can select a set of high-level features that he or she would like to have. For instance, the user can specify his or her preference as a casual seafood restaurant for a larger group.

CHAPTER 2. BACKGROUND AND RELATED WORK

Figure 2.3 shows the suggested restaurants in the Chicago area that are similar to the user's choice (the restaurant *Legal Sea Foods*). The user is able to navigate restaurants by using any of the seven fixed tweaks listed in the interface. The user can ask for a restaurant that is nicer, less expensive, more traditional or more creative. He or she and can also look for a similar restaurant but with a different cuisine. Each time the user is able to critique any of those features and the system will show some other restaurants to the user. This interactive process continues until the user finds the desired choice.

Critiquing technique has many advantages. From a user-interface perspective it is relatively easy to incorporate into even the most limited of interfaces. For example, the typical “*more*” and “*less*” critiques can be readily presented as simple icons or links alongside an associated product feature value and can be chosen by the user with a simple selection action. In contrast, value elicitation approaches must accommodate text entry for a specific feature value from a potentially large set of possibilities, via drop-down list, for example. In addition, critiquing can be used by users who have only limited understanding of the product domain. For example, a digital camera buyer may understand that greater resolution is preferable but may not be able to specify a concrete target resolution.

While critiquing enjoys a number of significant usability benefits as indicated above, it can suffer from the fact that the feedback provided by the user is rarely sufficiently detailed to sharply focus the next recommendation cycle. For example, by specifying that they are interested in a digital camera with a *greater resolution* than the current suggestion, the user is helping the recommender narrow its search but this may still lead to a large number of available products to chose from. Contrast this with the scenario where the user indicates that they are interested in a *5 mega pixels* camera, which is likely to reduce the number of product options much more effectively. The result is that critiquing-based recommenders can suffer from protracted recommendation sessions, when compared to value elicitation approaches.

The critiques described so far are all examples of, what we refer to as, *unit* critiques. That is, they express preferences over a single feature; Entrée's *cheaper* critiques a *price* feature, and *more formal* critiques a *style* feature, for example. This too ultimately limits the ability of the recommender to narrow its focus, because it is guided by only single-feature preferences from cycle to cycle. Moreover it encourages the user to focus on individual features as if they were independent and can result in the user following false-leads. For example, a price-conscious digital camera buyer might be inclined to critique the price feature until such time as an acceptable price has been achieved only to find that cameras in this region of the

product space do not satisfy their other requirements (e.g., high resolution). The user will have no choice but to roll-back some of these price critiques, and will have wasted considerable effort.

An alternative strategy is to consider the use of what we call *compound* critiques (McCarthy, Reilly, McGinty, & Smyth, 2004; Reilly et al., 2004a; Reilly, McCarthy, McGinty, & Smyth, 2004b; Smyth, McGinty, Reilly, & McCarthy, 2004). These are critiques that operate over multiple features. This idea of compound critiques is not novel. In fact the seminal work of Burke et al. (Burke et al., 1996) refers to critiques for manipulating multiple features. For instance, in the Car Navigator system, an automobile recommender, users are given the option to select a *sportier* critique. By clicking on this, a user can increase the *horsepower* and *acceleration* features, while allowing for a greater *price*. Similarly we might use a *high performance* compound critique in a PC recommender to simultaneously increase *processor speed*, *RAM*, *hard-disk capacity* and *price* features.

Obviously compound critiques have the potential to improve recommendation efficiency because they allow the recommender system to focus on multiple feature constraints within a single cycle. However, until recently, the usefulness of compound critiques has been limited by their static nature. The compound critiques have been hard-coded by the system designer so that the user is presented with a fixed set of compound critiques in each recommendation cycle. These compound critiques may, or may not, be relevant depending on the products that remain at a given point in time. For instance, in the example above the *sportier* critique would continue to be presented as an option to the user despite the fact that the user may have already seen and declined all the relevant car options.

2.5.2 Dynamic Critiquing

McCarthy et al. (McCarthy et al., 2004) proposed a method of discovering the compound critiques dynamically through the *Apriori* algorithm (Agrawal, Imielinski, & Swami, 1993; Agrawal & Srikant, 1994). It treats each critique pattern as the shopping basket for a single customer, and the compound critiques are the popular shopping combinations that the consumers would like to purchase together. Based on this idea, Reilly et al. (Reilly et al., 2004a, 2004b; Smyth et al., 2004) have developed an approach called dynamic critiquing to generate compound critiques. As an improved version, the incremental critiquing (Reilly et al., 2005) approach has also been proposed to determine the new reference product based on the user's critique history. Figure 2.4 shows the prototype system based on this approach.

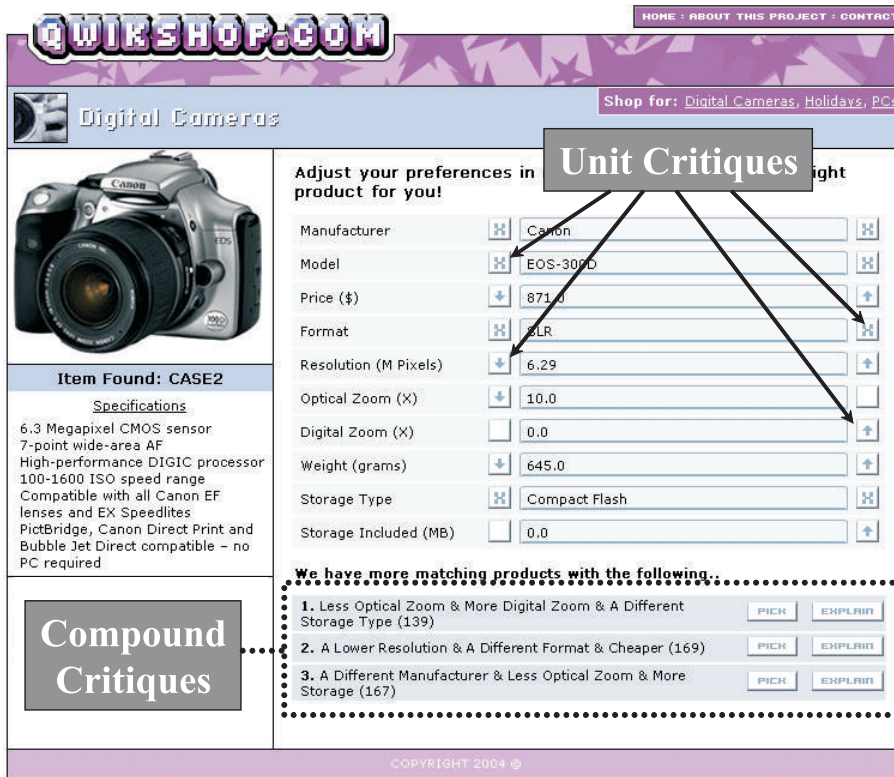


Figure 2.4: Screen-shot of the QwikShop system that adopts the dynamic critiquing approach. It enables users to apply both unit critiques and compound critiques.

Essentially, each compound critique describes a set of products in terms of the feature characteristics they have in common. For example in the PC domain, a typical compound critique might be for *Faster CPU* and a *Larger Hard-Disk*. By clicking on this the user narrows the focus of the recommender to only those products that satisfy these feature preferences. The Apriori data-mining algorithm (Agrawal & Srikant, 1994) is used to quickly discover these patterns and convert them into compound critiques on each recommendation cycle.

The first step involves *generating critique patterns* for each of the remaining product options in relation to the currently presented example. For example, the critique pattern $[Price <]$ will be used to present that the comparison laptop is cheaper than the current recommendation. The next step involves *mining compound critiques* by using the Apriori algorithm to identify groups of recurring unit critiques; we might expect to find the co-occurrence of some unit critiques like $[ProcessorSpeed >]$ infers $[Price >]$. We can combine these unit critiques into a

compound critique. The Apriori algorithm returns a list of compound critiques of the form $\{[ProcessorSpeed >], [Price >]\}$ along with their *support* values (i.e., the % of critique patterns for which the compound critique holds). The final step is to select some compound critiques from the list and to present them to the end-user.

It is not practical to present large numbers of different compound critiques as user-feedback options in each cycle. For this reason, a filtering strategy is used to select the k most useful critiques for presentation based on their support values. Importantly, compound critiques with low support values eliminate many more products from consideration if chosen. More recent work in the area considers compound critique diversity during the filtering stage, reducing compound critique repetition and better coverage of the product space (McCarthy, Reilly, Smyth, & McGinty, 2005).

2.5.3 SmartClient

SmartClient (Torrens, 2002; Pu & Faltings, 2000; Faltings, Torrens, & Pu, 2004b) is an example-based critiquing system architecture for searching products from a given product catalog with constraint-based preferences models. ISY-travel is a tool for travel planning based on the SmartClient architecture. ISY-travel was commercialized by Iconomic Systems and later by i:FAO known as *reality* (Pu & Faltings, 2000; Torrens et al., 2002; Pu & Faltings, 2002). In ISY-travel, the user starts by giving dates and destinations of travel. The tool then gathers all available airline schedules that may be relevant to the trip, and generates 30 examples of solutions that are good according to the current preference model. The preference model is initially preset with a number of common-sense preferences, such as short travel time, few connections, low price, etc. Seeing the examples, the user incrementally builds his preference model by adding preferences as shown in Figure 2.5.

Preferences can be added on any attribute or pair of attributes in any order. Preferences on pairs of attributes arise when a user conditions a preference for one attribute on another one. For example, one can select a different preferred departure time for each possible departure airport. Preferences can also be removed or given lower or higher weight by operations in the preference panel. When the preference model has been sufficiently modified, the user can ask the system to re-compute the 30 best solutions according to these preferences again.

When there are too many preferences, it can happen that there is no longer a single solution that satisfies them all. In this case, the system shows solutions that satisfy the preferences to the largest degree possible. For example, in Figure 2.6,

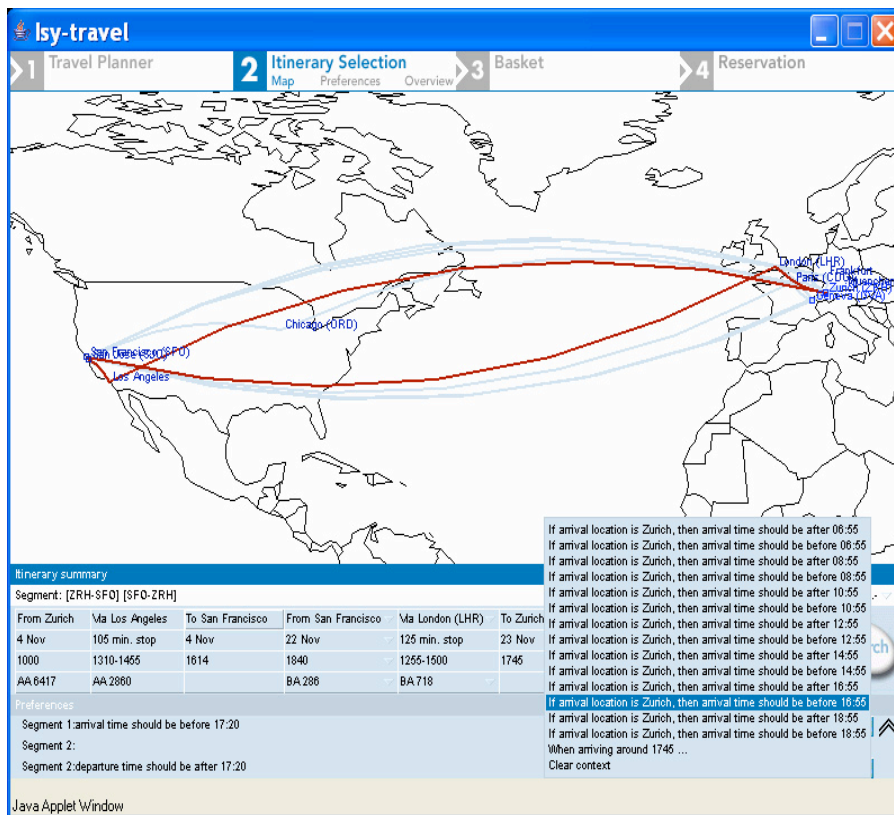


Figure 2.5: ISY-travel allows users to add preferences by posting soft constraints on any attribute or attribute combination in the display. Preferences in the current model are shown in the preference display at the bottom, and can be given different weights or deleted.

the user has posted constraints on the departure and arrival times that cannot both be satisfied. Thus, the system proposes solutions that satisfy only one of the preferences, and acknowledges the violation by showing the attribute in question on a red background.

2.5.4 FlatFinder

Recently Viappiani developed a product search tool based on the example-critiquing interaction paradigm, with additional examples as suggestions (Viappiani, Faltings, Schickel-Zuber, & Pu, 2005; Viappiani, Faltings, & Pu, 2006b; Viappiani, 2007). In this tool, suggestions are generated based on the following *lookahead principle*: suggestions should not be optimal under the current preference model, but should

Itinerary summary				
Segment: [GVA-HAM] [HAM-GVA]				
From Geneva	Via Muenchen	To Hamburg	From Hamburg	To Geneva
12 Oct	35 min. stop	12 Oct	12 Oct	12 Oct
0855	1025-1100	1215	1730	1910
LH 3769	LH 817		LH 5474	
Preferences				
Segment 1: arrival time should be before 11:00				
Segment 1: departure time should be after 08:45				
Segment 2:				

Figure 2.6: When it is not possible to satisfy all preferences completely, ISY-travel looks for solutions that satisfy as many of them as possible and acknowledges the attributes that violate preferences in red.

provide a high likelihood of optimality when an additional preference is added. In other words, suggestions are evaluated according to their probability of becoming Pareto-optimal. To become Pareto-optimal, the new preference has to make the current solution to escape the dominance with better solutions.

Based on this idea, the tool FlatFinder was implemented for finding student accommodations. It contains around 200 items of accommodation information available from the faculty housing program. Figure 2.7 shows an example of an interaction with FlatFinder. Each time the tool shows 3 options that best match the user's current preferences, plus 3 suggestions that could give the user some new hints so that he or she could specific more preferences in the future. Online user study results show that such suggestions are attractive to users and can stimulate them to express more preferences to improve the chance of identifying their most preferred item by up to 78% (Viappiani et al., 2006b).

2.5.5 MobyRek

MobyRek is a mobile recommender system that helps users search for travel products with critique techniques (Ricci & Nguyen, 2005, 2006, 2007). MobyRek elicits users' preference by asking questions with the style of critiquing. This tool considers both long-term and session-specific preferences. User's long-term preferences are those keep stable during the interaction session, such as "a non-smoking restaurant". Session-specific preferences are dependent to the specific search scenario, such as "a restaurant open on the day of the request", or "a low-cost restaurant".

The general recommendation process with MobyRek is as follows. The inter-

CHAPTER 2. BACKGROUND AND RELATED WORK

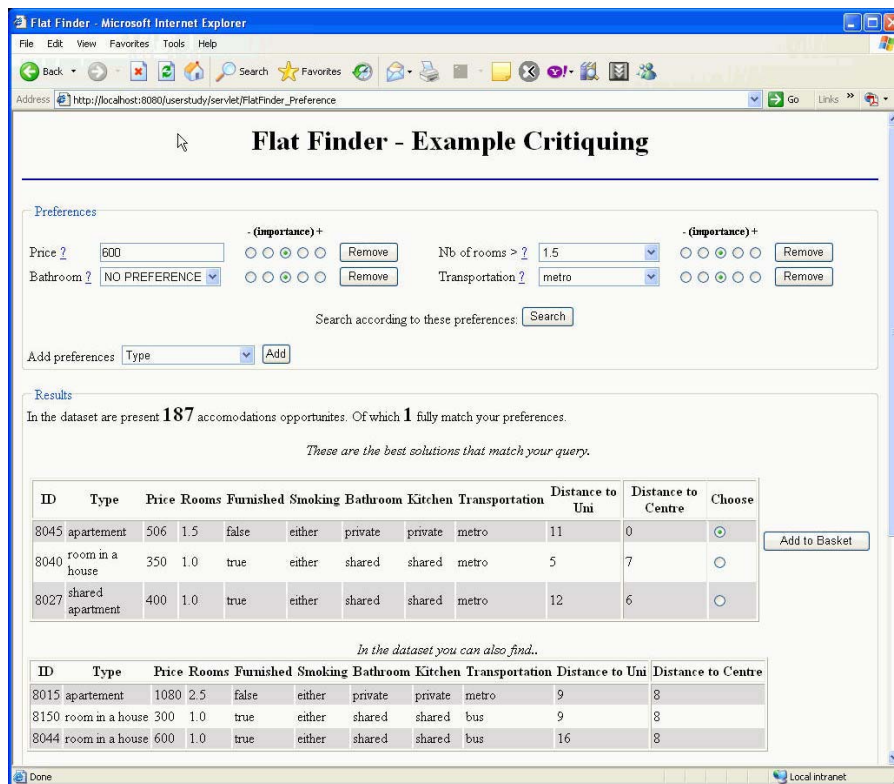


Figure 2.7: Screen-shot of the FlatFinder tool

action begins when a mobile user asks the system for a product recommendation. The user is able to specify his or her preferences with three options (as shown in Figure 2.8d): (1) *No, use my profile*: lets the system automatically construct the initial search query by utilizing the user's long-term preferences; (2) *Let me specify*: lets the user specify the initial preferences; or (3) *Similar to*: lets the user specify a known product as the start point. At each interaction cycle, the system shows recommended products (as shown in Figure 2.8a) that the user can browse (as shown in Figure 2.8b) and critique (as shown in Figure 2.8c). The interaction process ends when the user selects a product or terminates the session without making a selection.

The MobyRek tool was evaluated with real users with respect to usability, recommendation quality and overall satisfaction, and the results showed that this tool is quite effective in supporting on-the-go users in making product choice decisions.

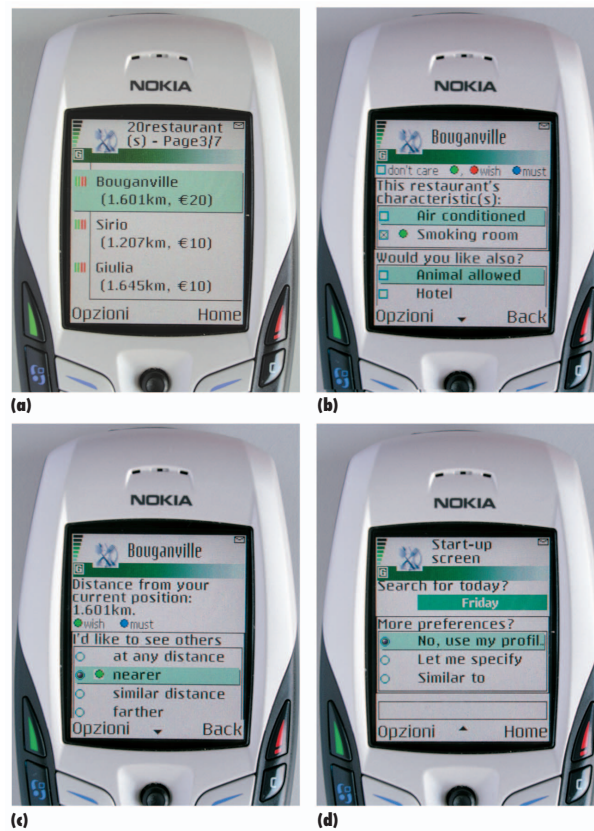


Figure 2.8: Screen-shot of the MobyRek tool

2.5.6 Apt Decision

Apt Decision is a tool that employs the critiquing technique for users to search apartments (Shearin & Lieberman, 2001). Users provide a small number of criteria in the initial interaction, receive a display of sample apartments, and then react to any feature of any apartment independently, in any order. Users are able to learn which features are important to them as they discover the details of specific apartments. Meanwhile the Apt Decision agent learns user preferences in the domain of rental real estate by observing the user's critique of apartment features. The agent uses interactive learning techniques to build a profile of user preferences, which can then be saved and used in further interaction process. As shown in Figure 2.9, the user can browse through the retrieved sample apartments in the left-hand list box, and the features of the selected apartment are shown on the right side of the window.

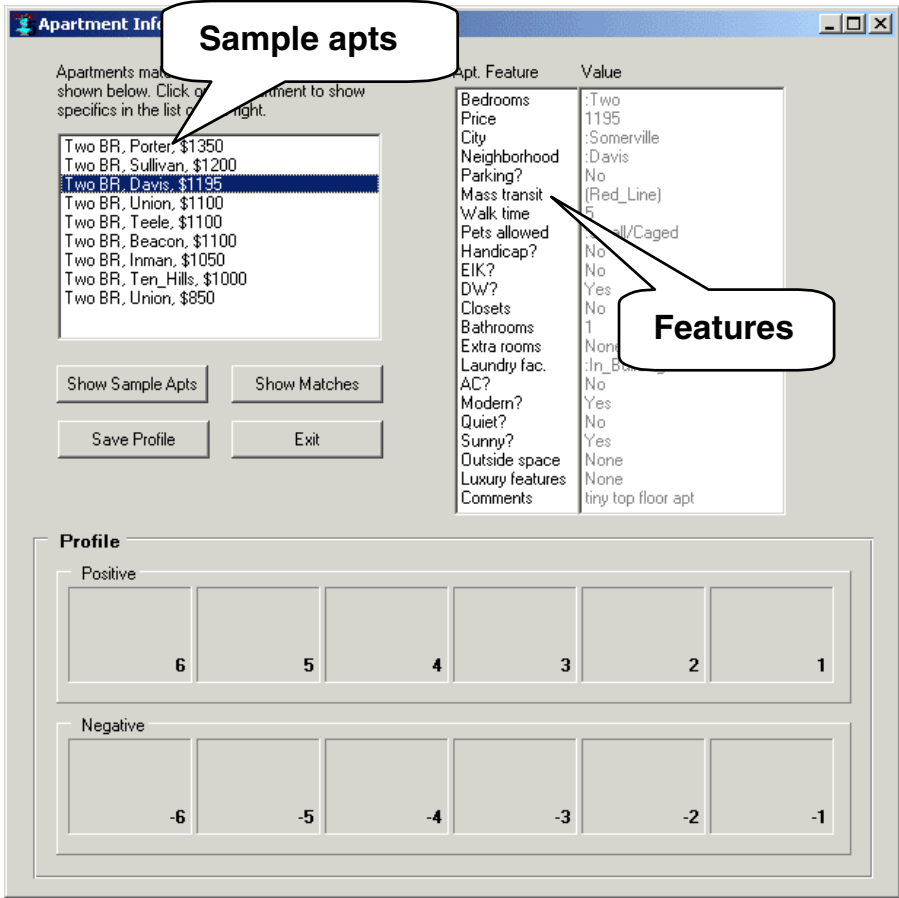


Figure 2.9: Screen-shot of the main interface of the Apt Decision system.

The user’s preferences are represented by a weighted vector, which shows the importance of each of the possible features. Critiquing takes place by giving a new weight to features. The user profile is represented graphically by a series of slots, each of them assigned to a difference level of positive or negative importance. The user can manually “move” features in one of the slots to change weights on individual features.

The application allows the user to directly compare two options and express his preference for one of the two so that the system can autonomously infer the features that are important to the user and update the profile (profile expansion): the items which are unique in the chosen apartment and not present in the profile, would be added to the right side of the profile.

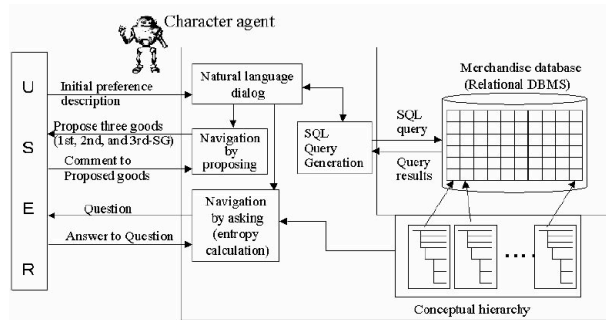
2.5. CRITIQUE-BASED SEARCH TOOLS

The limitation of this interface is that usually users are not good at providing some numeric weights to those criteria, even if they are facilitated by the graphical display of the slots.

2.5.7 Expertclerk



(a) The user interface of ExpertClerk.



(b) The system design of ExpertClerk.

Figure 2.10: Screen-shot of the ExpertClerck, an agent system that imitates a human salesclerk (Shimazu, 2001).

ExpertClerk is an agent system that imitates a human salesclerk (Figure 2.10a) in an e-commerce setting, generating a richer conversation than a question-answering dialogue (Shimazu, 2001). The system has two modes for interacting with users: *navigation-by-asking* and *navigation-by-proposing* (Figure 2.10b). In the first mode,

navigation-by-asking, the agent tries to narrow down the possibilities by asking questions to construct a preference model. The questions are selected according to an entropy measurement. In the second mode, *navigation-by-proposing*, the agent proposes three contrasting sample products, one in the central and two in the opposite extreme region of the available product space, to highlight their individual selling points. Users are then given the opportunity to critique the recommendation features to update the preference model. If the desired product is still not found, the user may switch back to the first mode. The dialogue terminates when the user accepts one of the three recommendation products.

2.6 Recommendation Techniques

Recommender systems are designed for customers to overcome information overload problem in e-commerce environments (Schafer, Konstan, & Riedl, 2001). As we mentioned earlier, the goal of a recommender system and a product search tool is the same: generate one product (or a list of products) to satisfy user's requirement. A variety of different recommendation techniques have been proposed, driven by the need for personalization in the presence of increasing amounts of information and product options. These techniques can be placed into two general categories: *collaborative filtering* and *content-based*. Collaborative filtering recommenders compute recommendations by identifying users with similar tastes and making recommendations based on their selections. For example, a collaborative filtering recommender will make a recommendation for a user by identifying other users who have liked the same movies and selecting one the target user has liked but not yet seen. On the other hand, content-based recommenders compute recommendations based on the content, or the descriptions of the recommendation items and how these align with user preferences. For example, a content-based recommender may make recommendations for movies by analyzing the genres of movies the user has liked in the past and comparing them to those available. In this section we review these two categories of recommendation techniques.

2.6.1 Collaborative Filtering Recommendation

One of the earliest collaborative filtering recommender systems was implemented as an email filtering system called *Tapestry* (Goldberg et al., 1992). Later on this technique was extended in several directions and was applied in various domains such as music recommendation (Shardanand & Maes, 1995) and video recommendation (Hill, Stead, Rosenstein, & Furnas, 1995).

The main idea behind the collaborative filtering approach is “*similar users like similar things*”. For example, if we know that user A and user B are very similar in their preferences, and we also know that user A likes a new item O , then we can guess that user B will also like the given item O . In this approach, users are required to state preferences by rating a set of items, which are then stored in a user-item rating matrix $R = \{r_{i,j}\}$, where $r_{i,j}$ represents the rating value given by user i to item j . The similarity between users are determined by their rating values.

Generally speaking, collaborative filtering algorithms can be classified into 2 categories: One is *memory-based*, which predicts the vote of a given item for the active user based on the votes from some other neighbor users. Memory based algorithms operate over the entire user voting database to make predictions on the fly. The most frequently used approach in this category is nearest-neighbor CF: the prediction is calculated based on the set of nearest-neighbor users for the active user (user-based CF approach) or, nearest-neighbor items of the given item (item-based CF approach). The second category of CF algorithms is *model-based*. It uses the users’ voting database to estimate or learn a probabilistic model (such as cluster models, or Bayesian network models, etc), and then uses the model for prediction. The detail of these methods and their respective performance have been reported in (Breese et al., 1998).

The user-based CF approach (Resnick et al., 1994) works as follows. The general prediction process is to select a set of nearest-neighbor users for the active user based on a certain similarity criterion (such as the Pearson correlation), and then aggregate their rating information to generate the prediction for the given item. More recently, an item-based CF approach has been proposed to improve the system scalability (Linden et al., 2001; Sarwar, Karypis, Konstan, & Reidl, 2001). The item-based CF approach explores the correlations or similarities between items. Since the relationships between items are relatively static, the item-based CF approach may be able to decrease the online computational cost without reducing the recommendation quality. The user-based and the item-based CF approaches are broadly similar, and it is not difficult to convert an implementation of the user-based CF approach into the item-based CF approach and vice versa.

When sufficient preferences (i.e. item ratings) from the users are available, studies have shown that the collaborative filtering approach could produce good prediction accuracy. Also, the user’s preferences could be accumulated along the time so that the system could perform better when more rating values are obtained. However, researchers have found that the collaborative filtering approach suffers from a number of problems. One is the *data sparsity* problem. When there are too many items for users to rate, the user-item rating matrix is very empty and only a small

number of ratings can be used during the prediction process. Another one is the problem called *cold-start* for new users. When a new user comes to the system with no (or few) rating value, the system may don't have enough preference information about the user and can't recommend item precisely.

Despite the above mentioned problems, collaborative filtering technique is still regarded as an very efficient approach in recommending items and it has been applied in many systems to help people find desired product easily. One of the most popular collaborative filtering systems is MovieLens³ that recommends movies based on ratings scaled from 1 to 5. The item-based CF approach has been applied on the popular website amazon.com (Linden et al., 2001).

2.6.2 Content-based Recommendation

Content-based recommendation has its origins in information retrieval research and typically operates on textual information. Recommendations are delivered by analyzing the descriptions of items and comparing them to a user profile. For example, when making a recommendation, a content-based movie recommender will try to recognize what aspects of movies a user has liked (or disliked) in the past (e.g. genre, director, actors) and recommend movies that best match those aspects.

Generally speaking, content-based recommenders must address two challenges: (1) how to represent items and (2) how to construct a profile that accurately represents user preferences. Depending on the domain, item descriptions can be *structured*, *unstructured* or *semi-structured*. Structured items are usually stored in a database where each item is described in terms of a finite number of features (also called attributes) and there is a known set of values that each feature may have. Machine learning algorithms can be employed to learn a user profile from item selections by analyzing which features and values the user prefers. To the opposite, unstructured items are described by plain textual information. Typically in this case the unstructured items are converted into structured ones before the recommendation process. For example, an item could have a list of Boolean features indicating whether some particular *keywords* are included or not. Semi-structured items are in between structured and unstructured data. For example, a mp3 music file is a semi-structured item: it has some header fields containing the basic information such as the title or singer, and some unstructured music data. In this case most likely we still need to convert the unstructured data into some kinds of structured features before recommendation process.

³MovieLens website: <http://www.movielens.org>.

2.7. OTHER DECISION MAKING APPROACHES

Content-based recommender systems have been successfully developed to recommend items in various domains such as news articles (Billsus & Pazzani, 2000), restaurants (Burke et al., 1997) and television programs (Smyth & Cotter, 2000). It is important to point out here that the critique-based tools that we have discussed above belong to the category of content-based recommendation.

Both collaborative filtering and content-based systems have their respective pros and cons, and may not be equally suitable for every domain or recommendation scenario. Often the limitations of one recommendation technique can be offset by another. Hybrid recommender systems attempt to leverage the power of multiple recommendation systems in order to improve the overall accuracy and precision of recommendations made to users. There are some other recommendation algorithms which have attempted to address some of the deficiencies of the content and collaborative approaches. For example, demographic recommenders (Montaner, López, & Rosa, 2003) attempt to avoid the problem of making recommendations to new users by assuming a set of preferences based on demographic data.

2.7 Other Decision Making Approaches

2.7.1 Framework of Constraint Satisfaction Problems (CSPs)

Constraint satisfaction problems (CSPs) (Mackworth, 1988; Tsang, 1993) have been widely used in AI research area for many years to solve different real-life problems ranging from map coloring, vision, robotics, VLSI design, etc. It provides a natural way of representing problems for the user needs only to state the constraints of the problem to be modeled. The formal definition of a CSP is given below.

Definition A constraint satisfaction problem (CSP) is defined by a triple $\langle \mathbf{X}, \mathbf{D}, \mathbf{C} \rangle$:

- a set of variables $\mathbf{X} = \{X_1, \dots, X_n\}$;
- a set of domain values $\mathbf{D} = \{D_1, \dots, D_n\}$, where each $D_i (1 \leq i \leq n)$ is a set of possible values for the variable x_i ;
- a set of constraints $\mathbf{C} = \{C_1, \dots, C_p\}$, where each $D_j (1 \leq j \leq p)$ is a constraint function on a subset of variables \mathbf{X} to restrict the values they can take.

A solution for a CSP is a set of value assignment $\{X_1 = x_1, \dots, X_n = x_n\}$ (in short as $\{x_1, \dots, x_n\}$) satisfying all constraints in \mathbf{C} . If a CSP has a solution, we say that it can be satisfied. Once the constraints are specified, some effective searching algorithms can be adopted to find the optimal solution.

CHAPTER 2. BACKGROUND AND RELATED WORK

If a CSP only has constraints on unary or binary variables, it is called *binary* CSP. It is possible to convert a CSP with n-ary constraints to another equivalent binary CSP (Rossi, Petrie, & Dhar, 1990). So usually the binary CSP is used without losing generality.

Besides hard constraints (also known as feasibility constraints) that can never be violated, a CSP may also include soft constraints. These are functions that map any potential value assignment to a variable or combination of variables into a numerical value that indicates the preference that this value or value-combination carries. Solving a CSP with soft constraints also involves finding assignments that are optimally preferred with respect to the soft constraints.

There are various soft constraint formalisms that differ in the way the preference values of individual soft constraints are combined (combination function). For example, in weight-CSPs (WCSPs) (Schiex, Fargier, & Verfaillie, 1995), the optimal solution minimizes the weighted sum of preferences. WCSPs allow one to model optimization problems where the goal is to minimize the total cost (such as time, space, number of resources, etc.) of the proposed solution. In WCSPs, there is a cost function for each constraint, and the total cost of a n-tuple value is defined by summing up the costs of each constraint with the corresponding sub-tuple values. Thus the aim is to find the n-tuples with minimal total cost as the optimal solution. Some other soft CSPs such as Fuzzy-CSPs (Fargier, Hang, & Schiex, 1993; Ruttkay, 1994) and Probabilistic-CSPs (Fargier & Lang, 1993) also have been widely used. Both classical CSPs and soft CSPs can be described under the semiring-based CSP framework (Bistarelli, Montanari, & Rossi, 1997; Bistarelli, 2004). More detail information about the CSP framework can be found in (Torrens, 2002).

A MADP can be looked as a CSP with a set of preferences which can be violated. The soft CSPs are quite suitable for modeling the MADPs since the preference statements in a MADP can be transformed to some soft-constraints of a soft CSP. For a given MADP, we first need to determine which kind of soft CSP is the ideal form for modeling the problem, depending on the features of the preferences set. For example, if the cost of violating each preference statement is easier to obtain, then we may use weighted-CSPs as the framework. Once the specific soft CSP framework is determined, we need to transform the preference statements into soft-constraints as required. Finally the optimal solution can be generated by some search algorithms.

The CSPs and Soft CSPs frameworks have been proposed for designing online product search tools in recent years. The SmartClient (Torrens, 2002; Pu & Faltings, 2000; Faltings et al., 2004b) system that we have introduced earlier is one implementation of a travel planning tool based on soft CSPs and the critiquing

technique. In (Zhang, Pu, & Faltings, 2006a), we analyzed the approach of modeling users' preferences with soft constraints in detail. More recently, O'Sullivan et al. (O'Sullivan, Papadopoulos, Faltings, & Pu, 2007) proposed a new approach to generate a representative set of explanations for solving some over-constrained decision problems. This approach is useful to help users find relaxations of the constraints that they have specified in interactive decision making scenarios.

2.7.2 CP-network

Boutilier et al. (Boutilier et al., 2004; Boutilier, Brafman, Geib, & Poole, 1997; Boutilier, Brafman, Hoos, & Poole, 1999) proposed a graphical representation of preferences that reflects conditional dependence and independence of preference statements under a *ceteris paribus* (all else being equal) interpretation: *CP-network*. The CP-network is based on the concept of conditional preferential independence: Let \mathbf{Y} , \mathbf{Z} , and \mathbf{W} be nonempty sets that partition \mathbf{X} (the set of all attributes), \mathbf{Y} and \mathbf{Z} are conditionally preferentially independent given an assignment w to \mathbf{W} if and only if, for all y_1, y_2, z_1, z_2 (here y_1, y_2 are two values of \mathbf{Y} , z_1, z_2 are two values of \mathbf{Z}), we have $(y_1, z_1, w) \succsim (y_2, z_1, w) \iff (y_1, z_2, w) \succsim (y_2, z_2, w)$ (denoted as $CPI(\mathbf{Y}, w, \mathbf{Z})$). If for all $w \in \mathbf{W}$ we have $CPI(\mathbf{Y}, w, \mathbf{Z})$, then \mathbf{Y} and \mathbf{Z} are CPI given \mathbf{W} (denoted as $CPI(\mathbf{Y}, \mathbf{W}, \mathbf{Z})$).

To construct the CP-network of a multi-attribute decision problem, the decision maker is asked to specify a set of parent attributes $Pa(x)$ that can affect her preferences over the values of each attribute x . That is, given a particular value assignment to $Pa(x)$, the decision maker should be able to determine a preference ordering for the values of x , *all other things being equal*. With this information, we are able to create the graph of the CP-network in which each node x has $Pa(x)$ as its immediate predecessors. Then the decision maker is asked to explicitly specify her preferences over the values of x for each assignment to $Pa(x)$. This conditional preference ranking over the values of x captured by a conditional preference table (CPT) which annotates the node x in the CP-network. Formally, the CP-network is defined as below.

Definition A CP-network over attributes $\mathbf{X} = \{x_1, \dots, x_n\}$ is a directed graph \mathbf{G} over x_1, \dots, x_n whose nodes are annotated with conditional preference tables $CPT(x_i)$ for each $x_i \in \mathbf{X}$. Each conditional preference table $CPT(x_i)$ associates to a total order $\succsim_{\mathbf{u}}^i$ with each instantiation \mathbf{u} of x_i 's parents $Pa(x_i)$.

The following simple example illustrates the form of a CP-network. Suppose a MADP has only two attributes x_1 and x_2 , where x_1 is a parent of x_2 and x_1 has no

CHAPTER 2. BACKGROUND AND RELATED WORK

parents. Attribute x_1 has two values: a and \bar{a} , and attribute x_2 has two values: b and \bar{b} . Assume the following conditional preferences exist:

$$a \succ \bar{a}; \quad a : b \succ \bar{b}; \quad \bar{a} : \bar{b} \succ b$$

With the above information, this CP-network would be constructed as Figure 2.11.

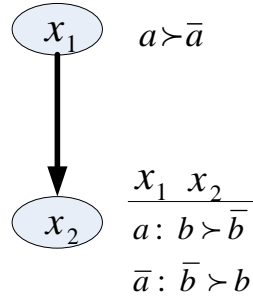


Figure 2.11: An example of the CP-network.

In this example, the conditional preferences information is surprisingly sufficient to totally order the outcomes of the given MADP: $ab \succ a\bar{b} \succ \bar{a}\bar{b} \succ \bar{a}b$.

Given a CP-network structure which specifies the decision maker’s preferences over outcome space, two kinds of useful queries can be answered. One is the outcome comparison query – preferential comparison between a pair of outcomes. We can construct a sequence of increasingly preferred outcomes using only valid conditional independence relations represented in the CP-network by *flipping* values of attributes. If we want to compare a pair of outcomes O_1 and O_2 , we can start from outcome O_1 , changing the value of a “higher priority” attribute (higher in the CP-network) to its preferred value, even if this introduces a new preference violation for some lower priority attribute (a descendent in the CP-network). This *flipping* operation is repeated until either the outcome O_2 is reached or no more attribute in outcome O_1 can be flipped. If outcome O_2 is reached, we say that O_2 is preferred to O_1 . If an outcome O is not preferred by any other outcomes, we say that O is a non-dominated outcome.

Another useful query is the outcome optimization query – determining the set of non-dominated feasible outcomes. Some search algorithms can be helpful for determining the non-dominated outcome set. One possible search method is a straightforward, depth-first, branch-and-bound style algorithm (Lawler & Wood, 1966; Reingold, Nievergelt, & Deo, 1977). The algorithm proceeds by assigning values to attributes in a depth-first fashion, using a variable ordering that is consistent

2.7. OTHER DECISION MAKING APPROACHES

with the ordering constraints imposed by CP-arcs (i.e., no child can be assigned before its parents). Suppose initially x is an attribute without parent nodes in the CP-network with the assigned value a , the set of constraints passed on to the next search node can be reduced: the CP-arcs that emanate from x can be removed in all subsequent search steps. This can result in disconnected fragments of the CP-network, and each of which can be optimized independently given $x = a$. During this procedure there is some pruning information that can take place in the search tree. Suppose that the attribute x has two values a and b , and a is preferred to b , if assignment $x = b$ satisfies an equal or smaller set of constraints than that was satisfied by $x = a$, then we do not continue to search under the branch of $x = b$: any feasible outcome involving $x = b$ is dominated by some feasible outcome involving $x = a$. After the non-dominated set is determined, if it contains only one outcome, then this outcome is the optimal solution for the multi-attribute decision problem. Otherwise the decision maker needs to select the most preferred outcome from the non-dominated set.

The CP-network has the advantage of representing the decision maker's preferences effectively by the conditional preference statements which is natural to be captured. While traditional CP-network is a qualitative method which cannot represent quantitative utility information, Recently it has been further extended to UCP-network by adding quantitative utility information to the conditional preference table of each attribute (Boutilier, Bacchus, & Brafman, 2001).

2.7.3 Analytic Hierarchy Process (AHP)

The Analytic Hierarchy Process (AHP) (Saaty, 1980) is a decision support approach that solves complex decision problems based on a series of pair-wise comparisons among attributes and alternatives.

Let's suppose a given multi-attribute decision problem has n attributes and m alternatives. The first step of the AHP approach is to determine the weight for each attribute. For each pair of attribute X_i and X_j , the user is asked to choose one option from the table below to decide the value of the importance relationship $t_{i,j}$:

Value 2, 4, 6, or 8 will be given if the user thinks the importance is in between. For example, if the user select "weak importance" when comparing X_i to X_j , then $t_{i,j} = 3$. At the same time the value of $t_{j,i}$ can also be determined automatically as $t_{j,i} = 1/t_{i,j}$. Thus for each pair of attributes X_i and X_j the user only needs to answer one question.

Once the matrix $T = \{t_{i,j} | 1 \leq i, j \leq n\}$ is established, we need to weigh the

Table 2.3: The importance relationship table for the AHP approach.

Option	Value
Equal importance	1
Weak importance	3
Strong importance	5
Demonstrated importance	7
Absolute importance	9

attributes according to a vector of priority weights. These are often in practice computed as the geometrical mean of each row, that is a reasonable approximation of the eigenvector associated with the maximal eigenvalue of the matrix T as the weight vector for attributes $W = \{w_1, \dots, w_n\}$.

The next step is to estimate the vector of all alternatives $A_i = \{a_{i,1}, \dots, a_{i,m}\}$ for each given attribute X_i . This procedure is quite similar to the above step. The only difference is that the user is asked to compare different pair of alternatives under each given attribute.

Finally, the importance value of each alternative O_i ($1 \leq i \leq m$) is determined as $\sum_{j=1}^n a_{i,j} \times w_j$. The solution of the given MADP will be the alternative O with the maximal importance value.

The AHP approach is able to solve decision problems in a precise way by decompose a complex decision problem into many small one. However it requires the user to answer a high number of questions. Based on the above procedure, the complexity of the total number of questions would be $O(n^2 + m^2n)$. It is not practical for designing product search tools with this approach when there are a lot of products available.

2.7.4 Heuristic Decision Making Strategies

Behavioral decision theory provides adequate knowledge describing people’s decision behavior and presents typical approaches of solving decision problems in traditional environments where no computer aid is involved (Payne et al., 1993). According to this research, a variety of choice strategies could be adopted to help decision makers find the preferred solution(s) for a given decision problem. Each choice strategy can be thought of as a method (or a sequence of operation) for searching through all available alternatives. Here we review some of the heuristic deci-

2.7. OTHER DECISION MAKING APPROACHES

sion making strategies, and discuss the potential of solving MADPs by using these heuristic strategies.

The weighted additive (WADD) strategy is based on the multi-attribute utility theory (MAUT) (Keeney & Raiffa, 1976) that we mentioned earlier. The WADD strategy evaluates the value of each alternative by formula (2.4), and the alternative with the highest overall utility value is chosen as the optimal solution.

The equal weight (EQW) strategy. This strategy is a simplified version of the WADD strategy which ignores information about the relative importance (weight) of each attribute. An overall value for each alternative is obtained by simply summing the values for all of its attributes, and the alternative with the highest overall value is selected as the final solution.

The elimination-by-aspects (EBA) strategy. This strategy begins by determining the most important attribute. The cutoff value for that attribute is retrieved, and all alternatives with values for that attribute below the cutoff are eliminated. The process continues with the second most important attributes, then the third, and so on, until only one alternative remains. This strategy was first described by Tversky (Tversky, 1972).

The majority of confirming dimensions (MCD) strategy. Described by Russo and Doshier (Russo & Doshier, 1983), this strategy involves processing pairs of alternatives. The values for each of the two alternatives are compared on each attribute, and the alternative with a majority of winning (better) attribute values is selected. The retained alternative is then compared with the next one among the set of alternatives. This comparison process repeats until all alternatives have been evaluated and the final winning alternative has been identified.

The satisficing (SAT) strategy. Satisficing is one of the oldest heuristics identified in decision making literature (Simon, 1955). With this strategy, alternatives are considered one at a time, in the order they occur in the set. Each attribute of an alternative is compared to a predefined cutoff value, which is often known as the *aspiration level*. If any attribute value is below the aspiration level, then that alternative is rejected. The first alternative which passes the cutoffs for all attributes is chosen. A choice can therefore be made before all alternatives have been evaluated. In the case where no alternative passes all the cutoffs, the cutoff can be relaxed and the process repeated, or an alternative can be randomly selected.

The lexicographic (LEX) strategy. For this strategy, the most important attribute is determined, the values of all the alternatives on that attribute are examined, and the alternative with the best value on that attribute is selected. If two alternatives

CHAPTER 2. BACKGROUND AND RELATED WORK

have equal values, the second most important attribute is examined. This continues until the tie is broken.

The frequency of good and bad features (FRQ) strategy. Alba and Marmorstein suggested that decision makers may evaluate or choose alternatives based simply upon counts of the good or bad features the alternatives possess (Alba & Marmorstein, 1987). To implement this strategy, the decision maker needs to develop cutoffs for specifying good and bad features, and then to count the number of such features. This strategy could be viewed as the application of a voting rule to multi-attribute choice, where the attributes can be viewed as voters.

The heuristic strategies are obviously useful for individuals when they are trying to find the optimal solution of the MADP. As mentioned above, the effort of solving MADP with heuristic strategies is relative low while the accuracy is not degraded too much. The optimal solution found by heuristic strategies has the advantage of matching with the decision maker's mental model, which implies that the decision maker is easier to accept the solution.

The computer system can also implement one or several of these strategies to help users find products. For example, the popular ranked list method is an implementation of the LEX strategy.

Two problems require further study before implementing some algorithms to solve MADP based on heuristic strategies. One is the error of the decision that caused by heuristic strategies. Some simulation experiments have shown that none of the heuristic strategies can get 100% accuracy (Payne et al., 1993). For a given decision problem, we need to select the right strategy so to get minimal error, and we also need to study what degree of error is acceptable for the decision maker. The other problem is the adaptive nature of heuristic strategies: people change heuristic strategies implicitly if the context changes. To solve this, we can study this phenomenon and try to make the change of heuristic strategies be predictable, or we can find several solutions by different strategies simultaneously, and then select the optimal solution among them by a certain criteria.

Simulation Environment For Performance Evaluation

3.1 Introduction

With the rising prosperity of the World Wide Web (WWW), consumers are dealing with an increasingly large amount of product and service information, which is far beyond any individual's cognitive effort to process. In early e-commerce practice, on-line intermediaries were created. With the help of these virtual store fronts, users were able to find product information on a single website, which gathers product information from thousands of merchants and service suppliers. Examples include shopping.yahoo.com, shopping.com, cars.com, pricegrabber.com, etc. Due to the increasing popularity of electronic commerce, the amount of online retailers grows rapidly. As a result, there are now easily millions of brand-name products available on a single online intermediary website. Finding something is once again difficult even with the help of various commercially available search tools. Recently, much attention in e-commerce research has focused on designing and developing more advanced search and product recommender tools (Burke et al., 1997; Pu & Faltings, 2000; Reilly et al., 2004a; Shearin & Lieberman, 2001; Shimazu, 2001; Stolze, 1999). However, they have been not employed in large scales in practicing e-commerce websites. Pu and Kumar (Pu & Kumar, 2004) gave some reasons as to why this is the case and when such advanced tools are expected to be adopted. This work was based on empirical studies of how users interact with product search

CHAPTER 3. SIMULATION ENVIRONMENT FOR PERFORMANCE EVALUATION

tools, providing a good direction as to how to establish the true benefits of these advanced tools. However, insights gained from this work are limited. This is mainly due to the lack of a *large* amount of *real* users for the needed user studies and the high cost of user studies even if real users were found. Each of the experiments reported in (Pu & Kumar, 2004) and (Pu & Chen, 2005) took more than 3 months of work, including the design and preparation of the study, the pilot study, and the empirical study itself. After the work was finished, it remains unclear whether a small amount of users recruited in an academic institution can forecast the behavior of the actual user population, which is highly diverse and complex.

In this chapter we introduce a simulation environment to evaluate the performance of a given online product search tool. Here we also call online product search tools as *consumer decision support systems* (CDSSs). Our main objective in this research is to use a simulation environment to evaluate various search tools in terms of interaction behaviors: what users' effort would be to use these tools and what kind of benefits they are likely to receive from these tools. We base our work on some earlier work (Payne et al., 1993) in terms of the design of the simulation environment. However, we have added important elements to adapt such environments to online e-commerce and online product search scenarios. With this simulation environment, we hope to be able to forecast the acceptance of online product search tools in the real world and curtail the evaluation of each tool's performance from months of user study to hours of simulation process. This should allow us to evaluate more tools and, more importantly, discover design opportunities of new tools.

3.2 Related Work

In traditional environments where no computer aid is involved, behavioral decision theory provides adequate knowledge describing people's choice behavior and presents typical approaches of solving decision problems. For example, Payne et al. (Payne et al., 1993) established a well known effort–accuracy framework that describes how people adapt their decision strategies by trading off accuracy and cognitive effort to the demands of the tasks they face. The simulation experiments carried out in that work were able to give a good analysis of various decision strategies that people employ and the decision accuracy they would expect to get in return.

In the online electronic environment where the support of computer systems is pervasive, we are interested in analyzing users' choice behaviors when tools are integrated into their information processing environments. That is, we are interested

in analyzing when given a computer tool with its system logic, how much effort a user has to expend and how much decision accuracy he or she is to obtain from that tool. On one hand, though the decision maker's cognitive effort is still required, it can be significantly decreased by having computer programs carry out most of the calculation work automatically; on the other hand, the decision makers must expend some effort to explicitly state their preferences to the computer according to the requirements of the underlying decision support approach employed in that system. We would like to call this extra user effort (in addition to the cognitive effort) preference elicitation effort. We believe that elicitation effort plays an important role in the new effort–accuracy model of users' behavior in online choice environments.

Many other researchers have carried out simulation experiments in evaluating the performance of their systems or approaches. Payne et al. (Payne et al., 1993) introduced a simulation experiment to measure the performance of various decision strategies in offline situations. Recently, Boutilier et al. (Boutilier et al., 2004) carried out their experiments by simulating a number of randomly generated synthetic problems, as well as user responses to evaluate the performance of various query strategies for eliciting bounds of the parameters of utility functions. In (Reilly et al., 2005), various users' queries were generated artificially from a set of offline data to analyze the recommendation performance of the incremental critiquing approach. More recently, Nguyen and Ricci (Nguyen & Ricci, 2007) presents an simulation methodology by replaying live-user interactions to compare different user-query representation approaches. These work generally suggest that simulating the interaction between users and the system is a promising methodology for performance evaluation. But so far it is lack of systematic approach of leading a simulation process. In addition, these approaches can only give simulate results about the interaction cycles, lack of criteria on the measurement of decision accuracy. In our work, we go further in this direction and propose the general simulation environment which can be adopted to evaluate the performance of various CDSSs systematically within the extended effort–accuracy framework.

3.3 Decision Strategies

In this work, we focus on the following decision strategies and study the performance of CDSSs based on these decision strategies. They have been introduced in Chapter 2. More detail information can also be found in (Payne et al., 1993).

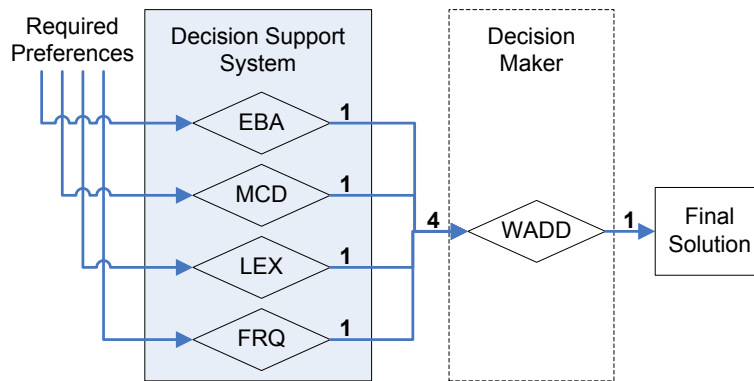


Figure 3.1: The *C4* decision strategy

1. *The Weighted Additive (WADD) Strategy.* It is a normative approach based on Multi-Attribute Utility Theory (MAUT) (Keeney & Raiffa, 1976). In fact, the WADD strategy adopts the additive form of the utility function (see Equation 2.4) to calculate utility value for each product, and it selects the product with the highest utility value. In our simulation experiment, we use it as the baseline strategy.
2. *Basic heuristic strategies.* They are the equal weight (EQW) strategy, the elimination-by-aspects (EBA) strategy, the majority of confirming dimensions (MCD) strategy, the satisficing (SAT) strategy, the lexicographic (LEX) strategy and the frequency of good and bad features (FRQ) strategy. Their detailed definitions are introduced in Chapter 2 and can be found in (Payne et al., 1993).
3. *Hybrid decision strategies.* Besides the basic heuristic strategies, people may also use a combination of several of them to make a decision to try to get a more precise decision result. These kinds of strategies are called hybrid decision strategies.

As a concrete example of hybrid decision strategies, here we propose a specific hybrid strategy called *C4*, which is a combination of four basic heuristic strategies: EBA, MCD, LEX, and FRQ. The decision procedure is illustrated in Figure 3.1. First the decision maker inputs his/her preferences to the system according to the requirements of the four strategies. Then the decision support system executes the four basic strategies simultaneously and produces up to 4 different alternatives for the decision maker. Finally the decision maker spends a certain amount of

3.4. THE EXTENDED EFFORT–ACCURACY FRAMEWORK

cognitive effort to select the final alternative using the WADD strategy. As the WADD strategy is completed by the decision maker, it requires no elicitation effort. The elicitation effort for *C4* would be counted by the total parameters that the four heuristic strategies require. We expect that the *C4* strategy could gain much higher decision accuracy than using the underlying basic strategies individually.

In a CDSS, the user interface component is used to obtain the consumers' preferences. However, such preferences are largely determined by the underlying decision support approach that has been adopted in the system. For example, the popular ranked list interface is in fact an interface implementing the lexicographical (LEX) strategy. Also, if we adopt the Weight Additive Strategy in a consumer decision support system, the user interface will be designed in the manner of asking the user to input corresponding weight and middle values for each attribute. In our current work, we assume the existence of a very simple user interface. Thus, we regard the underlying decision support approach as the main factor of the consumer decision support system.

3.4 The Extended Effort–Accuracy Framework

The performance of the system can be evaluated by various criteria such as the degree of a user's satisfaction with the recommended item, the amount of time a user spends to reach a decision, and the decision errors that the consumer may have committed. Without real users' participation, the satisfaction of a consumer with a CDSS is hard to measure. However, the other two criteria can be measured.

The amount of time a user spends to reach a decision is equivalent to the amount of time he or she uses to express preferences and process the list of recommended items in order to reach a decision. The classical effort–accuracy framework mainly investigated the relationship of decision accuracy and cognitive effort of processing information by different decision strategies in the offline situation (Payne et al., 1993). In the online decision support situation, however, the effort of eliciting preferences during the interaction process must be considered as well.

Furthermore, most products carry a fair amount of financial and emotional risks. Thus the accuracy of users' choices is quite important. That is, there is a posterior process where users evaluate the search tools in terms of whether the products they have found via the search tool is really what they want and whether they had enough decision support. This is what we mean by decision accuracy.

We therefore propose an extended effort–accuracy framework by explicitly measuring three factors of a given consumer decision support system – cognitive effort,

interaction effort (it is also called elicitation effort), and decision accuracy. In the remainder of this section, we first recall the measurement of cognitive effort in the classical framework, we give various definitions of decision accuracy, and then we detail the method of measuring elicitation effort. Finally the cognitive and elicitation effort of these decision strategies are analyzed in online situation.

3.4.1 Measuring Cognitive Effort

Based upon the work of Newell and Simon (Newell & Simon, 1972), a decision approach can be seen as a sequence of *elementary information processes* (EIPs), such as reading the values of two alternatives on an attribute, comparing them, and so forth. Assuming that each EIP takes equal cognitive effort, the decision maker's cognitive effort is then measured in terms of the total number of EIPs. Conformed with the classical framework, a set of EIPs for the decision strategies is defined as the following:

- (1) **READ**: read an alternative's value on an attribute into short-term memory (STM),
- (2) **COMPARE**: compare two alternatives on an attribute,
- (3) **ADD**: add the values of two attributes in STM,
- (4) **DIFFERENCE**: calculate the size of the difference of two alternatives for an attribute,
- (5) **PRODUCT**: weight one value by another,
- (6) **ELIMINATE**: eliminate an alternative from consideration,
- (7) **MOVE**: move to next element of the external environment, and
- (8) **CHOOSE**: choose the preferred alternative and end the process.

3.4.2 Measuring Decision Accuracy

Accuracy and effort form an important performance measure for the evaluation of consumer decision support systems. On one hand consumers expect to get highly accurate decisions. On the other hand, they may not be inclined (or able) to expend a high level of cognitive and elicitation effort to reach a decision. Three important factors influence the decision accuracy of a consumer decision support systems: the

3.4. THE EXTENDED EFFORT-ACCURACY FRAMEWORK

underlying decision approach used; the number of interactions required from the end user (if a longer interaction is required, a user may give up before he finds the best option); the number of options displayed to the end user in each interaction cycle (a single item is likely to miss the target choice compared to a list of items; however, a longer list of items requires more cognitive effort to process information). In our current framework, we investigate the combined result of these three factors (i.e., decision approach as well interface design components) of a given consumer decision support system.

In the following, we start with classical definitions of decision accuracy, analyze their features and describe their weaknesses for the online environments, and then we propose two definitions that we have developed which are likely to be more adequate for measuring decision accuracy in e-commerce environments. To eliminate the effect of a specific set of alternatives on the decision accuracy results, in the following definitions we assume that there is a set of N different MADPs to be solved by a given consumer decision support system which implements a particular decision strategy S . The accuracy will be measured as the average among all those MADPs.

Accuracy Measured by Selection of Non-Dominated Alternatives

This definition comes from Grether et al. (Grether & Wilde, 1983). After adapting it to decision making with the help of a computer system, this definition says that a solution given by CDSS is correct if and only if it is non-dominated by other alternatives. So the decision accuracy can be measured by the numbers of solutions which are Pareto Optimal (i.e., not dominated by other alternatives, see also (Viappiani et al., 2005, 2006b)). We use O_S^i to represent the optimal solution given by the CDSS with strategy S when solving $MADP_i$ ($1 \leq i \leq N$). The accuracy of selection of non-dominated alternatives $ACC_{NDA}(S)$ is defined as the following:

$$ACC_{NDA}(S) = \frac{N - \sum_{i=1}^N Dominated(O_S^i)}{N} \quad (3.1)$$

where $Dominated(O_S^i)$ equals to 1 if O_S^i is a dominated alternative in the given $MADP_i$, otherwise $Dominated(O_S^i)$ equals to 0.

According to this definition, it is easy to see that a system employing the WADD strategy has 100% accuracy because all the solutions given by WADD are *Pareto Optimal*. Also, this definition of accuracy measurement is effective only when the system contains some dominated alternatives, otherwise the accuracy of the system is always 100%.

This definition of accuracy can distinguish the errors caused by choosing dominated alternatives of the decision problems. However, measuring decision accuracy using this method is limited in e-commerce environments. In an efficient market, it is unlikely that the consumer products or business deals are dominated or dominating. That is, it is unlikely an apartment would be both spacious and less expensive compared to other available ones. We believe that although this definition is useful, it is not helpful to distinguish various CDSSs in terms of how good they are for supporting users to select the best choice (not just the non-dominated ones).

Accuracy Measured by Utility Values

This definition of measuring accuracy was used in the classical effort–accuracy framework (Payne et al., 1993). Since no risk or uncertainty is involved in the MADPs, the expected value of an alternative is equivalent to the utility value of each alternative. The utility value of each alternative is assumed to be in the weight additive form. Formally this accuracy definition can be represented as:

$$ACC_{UV}(S) = \frac{\sum_{i=1}^N \frac{U(O_S^i)}{U(O_{target}^i)}}{N} \quad (3.2)$$

where U is the utility function, which is assumed to be in the additive form. In this case the target product O_{target} determined by the maximal utility value is exactly the one determined by the WADD strategy (i.e. $O_{target}^i = O_{WADD}^i$) for the given $MADP_i$. Thus a system employing the WADD strategy is 100% accurate because it always gives out the solution with the maximal utility value.

One advantage of this measure of accuracy is that it can indicate not only that an error has occurred but also the severity of the error of the decision making. For instance, a system achieving 90% accuracy indicates that an average consumer is expected to choose an item which is 10% less valuable from the best possible option. While this definition is useful for choosing a set of courses to take for achieving a particular career objective, it is not most suitable in e-commerce environments. Imagine that someone has chosen and purchased a digital camera. Two months later, she discovers that the camera that her colleague has bought was really the one she wanted. She did not see the desired camera, not because the online store did not have it, but because it was difficult to find and compare items on the particular e-commerce website. Even though the camera that she bought satisfied some of her needs, she is still likely to feel a great sense of regret if not outright disappointment. Her likelihood of returning to that website is in question. Given that

3.4. THE EXTENDED EFFORT-ACCURACY FRAMEWORK

bad choices can cause great emotional burdens (Luce, Payne, & Bettman, 1999), we develop the following definition of decision accuracy.

Accuracy Measured by Selection of Target Choice

In the earlier work (Pu & Chen, 2005), the decision accuracy is measured by the percentage of users who have chosen the right option using a particular decision support system. We call that option the target choice. In empirical studies with real users, we first asked users to choose a product with the consumer decision support system, and then we revealed the entire database to them in order to determine the target choice. If the product recommended by the consumer decision support system was consistent with the target choice, we say that the user had made an accurate decision.

In simulation environment, we take the WADD strategy as the baseline. That is, we assume the solution given by WADD is the user's final most preferred choice. For another given strategy S , if the solution is the same as the one determined by WADD, then we count it as one Hit (this definition is called the hit ratio). The accuracy is measured statistically by the ratio of hit numbers to the total number of decision problems:

$$ACC_{HR}(S) = \frac{\sum_{i=1}^N Hit(O_S^i)}{N} \quad (3.3)$$

where

$$Hit(O_S^i) = \begin{cases} 1 & \text{if } O_S^i = O_{target}^i, \text{ for a given } MADP_i \\ 0 & \text{else} \end{cases} \quad (3.4)$$

The WADD strategy is used as the baseline to determine the target solution (i.e., $O_{target}^i = O_{WADD}^i$), thus it will achieves 100% accuracy. This measure of decision accuracy is ideally consistent with the consumers' attitude towards the decision results. However, by this definition, it is assumed that the consumer decision support system only recommends one product to the consumer each time. In reality the system may show a list of possible products to the consumer, and the order of the product list is also important to the consumer: the products displayed at the top of the list are more likely to be selected by the consumer. Therefore, we have developed the following definition to take into account that a list of products is displayed, rather than a single product.

Accuracy Measured by Selection of Target Choice among K-best Items

Here we propose measuring the accuracy of the system according to the ranking orders of the K-best products it displays. This is an extension of the previous definition of accuracy. For a given $MADP_i$, instead of using strategy S to choose a single optimal solution, we can use it to generate a list of solutions with ranking order $L_S^i = \{O_{S,1}^i, O_{S,2}^i, \dots, O_{S,K}^i\}$, where $O_{S,1}^i$ is the most preferred solution according to the strategy S , and $O_{S,2}^i$ is the second preferred solution, and so on. The first K -best solutions consist of the solution list. If the user's target choice is in the list, we assign a *Rank Value* to the list according to the position of O_{target}^i in the list. Formally, we define this accuracy of choosing K-best items as

$$ACC_{HR_i n_K best}(S) = \frac{\sum_{i=1}^N RankValue(L_S^i)}{N} \quad (3.5)$$

where

$$RankValue(L_S^i) = \begin{cases} 1 - \frac{k-1}{K} & \text{if } O_{S,k}^i = O_{target}^i, \text{ for a given } MADP_i \\ 0 & \text{else} \end{cases} \quad (3.6)$$

The WADD strategy is used as the baseline to determine the target solution thus it will achieves 100% accuracy. A special case of this accuracy definition is that when $K = 1$, it degenerates to the previous definition of *Hit Ratio*. In the simulation experimental results that we will show shortly, we have set K to 5.

Relative Accuracy

In practice, it is required to eliminate the effect of random decision, and we expect that the strategy of random choice (selecting an alternative randomly from the alternative set, denoted as RAND strategy) could only produce zero accuracy. By doing so we define the relative accuracy of the consumer decision support system with strategy S according to different definitions as

$$RA_Z(S) = \frac{ACC_Z(S) - ACC_Z(RAND)}{1 - ACC_Z(RAND)} \quad (3.7)$$

where $Z = NDA, UV, HR, \text{ or } HR_in_Kbest$.

For example, $RA_{HR}(LEX)$ denotes the relative accuracy of the LEX strategy under the accuracy measure definitions of *Hit Ratio*.

3.4. THE EXTENDED EFFORT-ACCURACY FRAMEWORK

From the above definitions, we can see that each definition represents one aspect of the accuracy of the decision strategies. We think that the definitions of *Hit Ratio* among *K-best Items* are more suitable among them to measure the accuracy of various consumer decision support systems.

Other Accuracy Measure Metrics

Besides the metrics that we have discussed, there are some other metrics for measuring the accuracy of a system. *Mean Absolute Error (MAE)* (Sarwar et al., 2001) can be used to evaluate the accuracy of some rating-based recommender systems. It measures the accuracy according to the mean average deviation between the expected ratings and the true ratings. However, this metric is not suitable for measuring the accuracy of a product search approach without having rating information. Another set of metrics, *precision* and *recall* has been applied in some information retrieval systems to measure the accuracy (Baeza-Yates & Ribeiro-Neto, 1999). Precision measures the proportion of relevant items are in the recommendation set, while recall is defined as the number of relevant items retrieved over all relevant items in the database. In our simulation work, however, each time we assume that there is single one target item in the dataset, so this set of metrics is not suitable to measure accuracy in this situation. Moreover, some kind of rank correlation coefficient, such as the Kendall tau coefficient (Kendall, 1948) or Spearman's coefficient (Spearman, 1906), can be used to measure the degree of agreement between two ranking list and assessing their significance of difference. These methods can be used as criteria to measure the accuracy of two ranking lists, but in our simulation work these criteria are not appropriate because each time there is only one single item determined as the target product at the end of the decision process.

3.4.3 Measuring Elicitation Effort

In computer-aided decision environments, a considerable amount of decision effort goes into preference elicitation since people need to “tell” their preferences explicitly to the computer system. So far, no formal method has been given to measure the preference elicitation effort. An elicitation process can be decomposed into a series of basic interactions between the user and the computer, such as selecting an option from a list, filling in a blank, answering a question, etc. We call these basic interaction actions elementary elicitation processes (EEPs). In our analysis, we define the set of EEPs as follows:

- (1) **SELECT**: select an item from a menu or a dropdown list,

- (2) **FILLIN**: fill in a value to an edit box,
- (3) **ANSWER**: answer a basic question,
- (4) **CLICK**: click a button to execute an action.

It is obvious that different EEPs require different elicitation effort (for instance, the EEP of one CLICK would be much easier than an EEP of FILLIN a weight value for a given attribute). For the sake of simplification, we currently assume that each EEP requires an equal amount of effort from the user. Therefore, given a specific decision approach, elicitation effort is measured by the total amount of EEPs it may require.

This elicitation effort is a new factor for the online environment. The main difference between cognitive effort and elicitation effort lies in the fact that cognitive effort is a description of the mental activities in processing information, while the elicitation effort is about the interaction effort between the decision maker and the computer system through pre-designed user interfaces. Even though the decision makers already have clear preferences in their mind, they must still state their preferences in a way that the computer can “understand”. With the help of computer systems, the decision maker is able to reduce the cognitive effort by compensating with a certain degree of elicitation effort.

Let’s consider a simple decision problem with 3 attributes and 4 alternatives. When computer support is not provided, the cognitive effort of solving this problem by the WADD strategy will be 24 READS, 8 ADDS, 12 PRODUCTS, 3 COMPARES, and 1 CHOOSE. The total number of EIPs is therefore 48. However, with the aid of a computer system, the decision maker could get the same result by spending 2 units of elicitation effort (FILLIN the weight value of first 2 attributes) and 1 unit of cognitive effort (CHOOSE the final result).

3.4.4 Analysis of Cognitive and Elicitation Effort

With the support of computer systems, the cognitive effort for WADD, as well as the basic heuristic strategies, is quite low. The decision maker inputs his or her preferences, and the decision support system executes that strategy and shows the proposed product. Then the decision maker chooses this product and the decision process is ended. Thus the cognitive effort is equal to 1 EIP: CHOOSE the final alternative and exit the process. For the $C4$ strategy, the cognitive effort of solving a MADP with n attributes and m alternatives in the online situation is equal to

Table 3.1: Interaction effort analysis of decision strategies

Strategy	Parameters required to be elicited
WADD	weights, component value functions
EQW	component value functions
EBA	importance order, cutoff values
MCD	<i>none</i>
SAT	cutoff values
LEX	importance order
FRQ	cutoff values for good and bad features
<i>C4</i>	cutoff values, importance order

that of solving a problem with n attributes and 4 alternatives in the traditional situation, the cognitive effort of which has been studied in (Payne et al., 1993).

According to their definitions, various decision strategies require that preferences with different parameters be elicited. For example, in the WADD strategy, the component value function and the weight for each attribute must be obtained. While for the EBA strategy, the importance order and cutoff value for each attribute are required. The required parameters for each strategy are shown in table 3.1.

For each parameter in the aforementioned strategies, a certain amount of elicitation effort is required. This elicitation effort may vary with different implementations of the user interface. For example, to elicit the weight value of an attribute, the user can just FILLIN the value to an edit box, or the user can ANSWER several questions to approximate the weight value. In our analysis and the following simulation experiments, we follow the *at least* rule: the elicitation effort is determined by the least number of EEP(s). In the above example, the elicitation effort for obtaining a weight value is measured as 1 EEP.

3.5 Simulation Environment

Our simulation environment is concerned with the evaluation of how users interact with online product search tools, how decision results are produced, and the quality of these decision results.

The consumer first interacts with the system by inputting his or her preferences through the user interface. With the help of decision support, the system generates a set of recommendations to the consumer. This interactive process can be executed in multiple times until the consumer is satisfied with the recommended results (i.e., a product to purchase) or gives up due to losing patience.

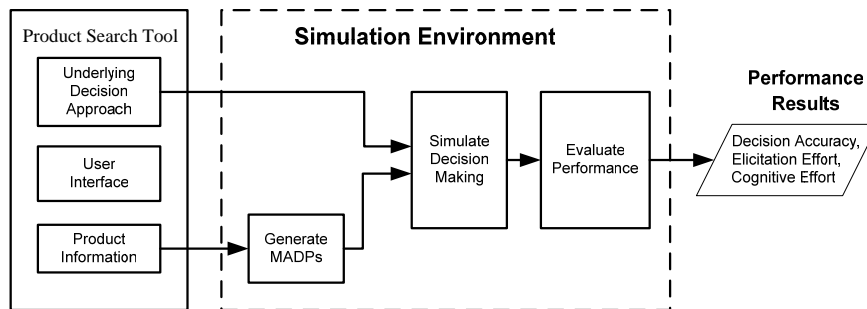


Figure 3.2: The architecture of the simulation environment for evaluating the performance of a given product search tool.

As shown in Figure 3.2, for a given CDSS, we evaluate its performance in a simulated environment by the following procedures:

1. we generate a set of MADPs randomly to simulate the presence of an electronic catalog up to any scale and structure characteristics using;
2. we generate a set of consumer preferences randomly as well, taking into account user diversity and scale;
3. we carry out the simulation of the underlying decision approach of the CDSS to solve these MADPs;
4. we obtain associated decision results for the given CDSS (which product has been chosen given the consumer's preferences);
5. we evaluate the performance of these decision results in terms of cognitive effort, preference elicitation effort and decision accuracy under the extended accuracy–effort framework.

The simulation environment can be used in many ways to provide different performance measures of a given CDSS. For instance, if both the detail product information of CDSS and the consumer's preferences are unknown, we can simulate both the alternatives and the consumer's preferences, and the simulation results would be the performance of the CDSS independently of users and the set of alternatives; if the detail product information of the CDSS is provided, we then only need to simulate the consumer's preferences, and the alternatives of the MADPs can be copied from the CDSS instead of being randomly generated. The simulation results would be the performance of the CDSS under the specified product set.

As a concrete example to demonstrate the usage of such a simulation environment, we will show a procedure in evaluating the performance of various CDSSs in terms of the scale of the MADPs, which is determined by two factors: the number of attributes n and the number of alternatives m . Since we are trying to study the performance of different CDSSs (currently built on heuristic decision strategies) in different scales of MADPs, we assume that users and alternatives are both unknown and they are simulated to give results independently of the user and the system. More specifically, we classify the decision problems into 20 categories according to the scales of n (the number of attributes) and m (the number of alternatives): n has five values (5, 10, 15, 20, and 25), and m has four (10, 100, 1000 and 10000). To make the performance evaluation result more accurate, each time we randomly generate 500 different MADPs in the same scale and use their average performance as the final result. The detail simulation result will be reported in the experimental result section.

3.6 Simulation Results

In this section we report our experimental results of the performance of various consumer decision support systems under the simulation environment which were introduced earlier. To simplify the experiments, we only evaluate those CDSSs built on the decision strategies listed in Table 3.1. Without loss of generality, we will also use the term decision strategy to represent the CDSS built on that decision strategy.

We have developed a simulation program to generate simulation results (see Figure 3.3). This program simulates the process of people making decisions when they face decision problems. More specifically, it is able to simulate these decision strategies and to give the decision results in terms of decision accuracy and required effort. The attributes of the decision problem can be customized with specific domain values. The product list can either be generated randomly, or be loaded from an external xml file.

For each CDSS, we first simulate a large variety of MADPs, and then run the corresponding decision strategy on the computer to generate the decision results. Then the elicitation effort and decision accuracy are calculated according to the extended effort–accuracy framework. For each MADP, its domain values for a given attribute are determined randomly: the lower bound of each attribute is set to 0, and the upper bound is determined randomly from the range of 2 to 100. Formally speaking, for each attribute X_i , we define $D_i = [0, z_i]$, where $z_i \in [2, 100]$.

As shown in table 3.1, each decision strategy (except MCD) requires the elic-

CHAPTER 3. SIMULATION ENVIRONMENT FOR PERFORMANCE EVALUATION

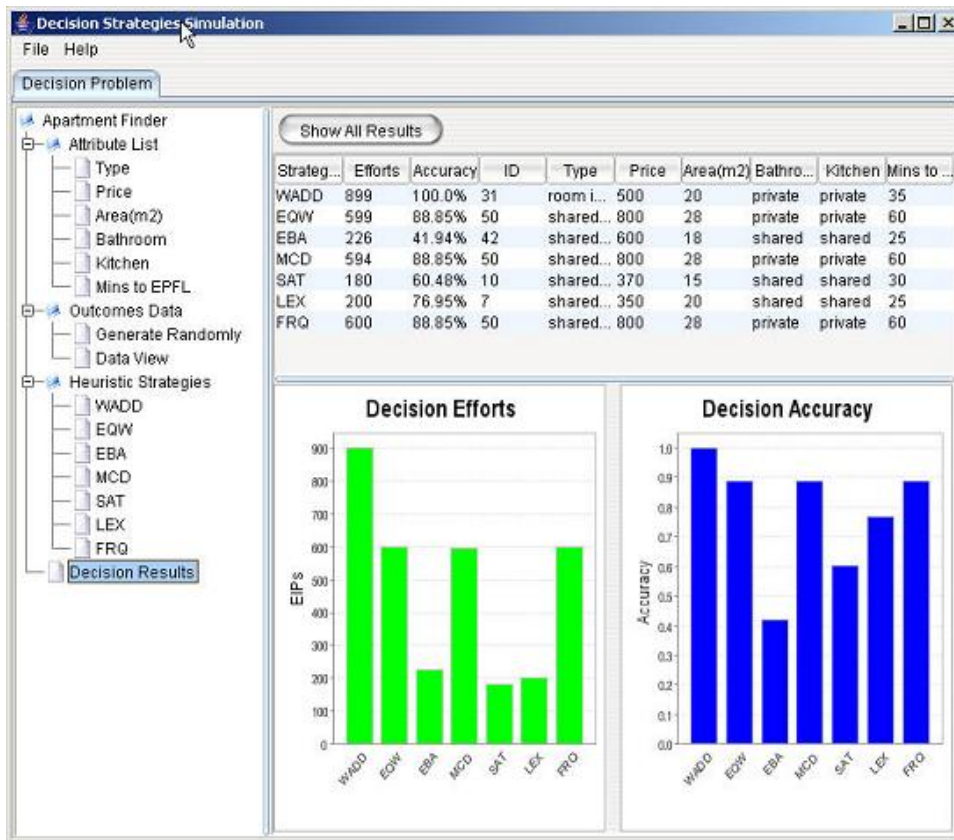


Figure 3.3: Screen-shot of the decision strategy simulation program. It shows the overall decision results of a given decision problem.

itation of some specific parameters such as attribute weights or cutoff values to represent the user's preferences. To simulate the component value functions that required by the WADD strategy, we assume that the component value function for each attribute is approximated by 3 mid-value points that are randomly generated. Thus each component value function requires 3 units of EEPs. Other required parameters such as the weight and cutoff value (each requires 1 unit of EEP) for each attribute are also simulated by the random generation process. The order of importance is determined by the weight order of the attributes for consistency.

In our simulation experiments, the WADD strategy is appointed as the baseline strategy, and the relative accuracy of a strategy is calculated according to equation (3.7). The elicitation effort is measured in terms of the total number of EEPs required by the specific strategy, and the cognitive effort is measured by the required units of EIPs. Since the relationship between accuracy and cognitive effort has al-

3.6. SIMULATION RESULTS

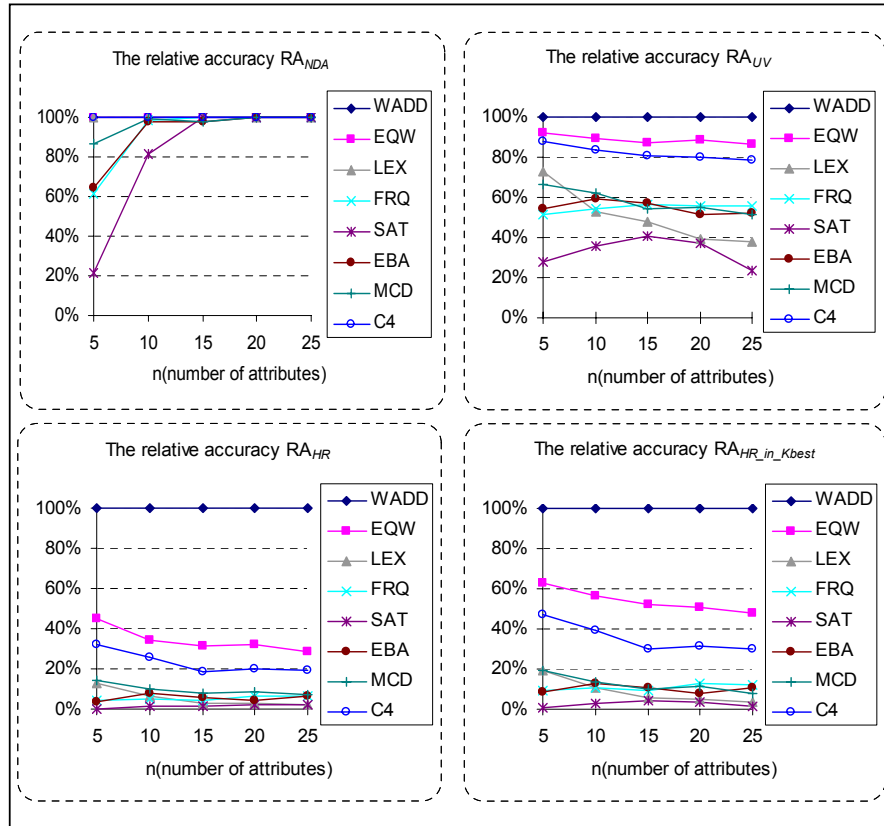


Figure 3.4: The relative accuracy of various decision strategies when solving MADPs with different number of attributes, where $m(\text{number of alternatives}) = 1,000$

ready been studied and analyzed by Payne et al. (Payne et al., 1993), in this section we only focus on the performance of each strategy in terms of decision accuracy and elicitation effort.

Figure 3.4 shows the changes in relative accuracy with 4 different accuracy measure definitions for the listed decision strategies as the number of attributes increases in the case that each MADP has 1,000 alternatives. In all cases the WADD is the baseline strategy thus it achieves 100% accuracy. When measured by the selection of non-dominated alternatives (RA_{NDA}), the relative accuracy of each heuristic strategy increases as the number of alternatives increases. This is mainly because the alternatives are more likely to be Pareto Optimal when more attributes are involved. Furthermore, the RA_{NDA} of all strategies could achieve 100% accuracy when the attributes number is 20 or 25. This shows that the RA_{NDA} is not

CHAPTER 3. SIMULATION ENVIRONMENT FOR PERFORMANCE EVALUATION

able to distinguish the decision errors occurred with the heuristic strategies in the simulated environment. When the accuracy is measured under the definitions of RA_{UV} , RA_{HR} and $RA_{HR_in_Kbest}$, the EQW strategy achieves the highest accuracy besides the baseline WADD strategy, and the SAT strategy has the lowest relative accuracy. The four basic heuristic strategies EBA, MCD, LEX and FRQ are in the middle-level range. The LEX strategy, which has been widely adopted in many consumer decision support systems, is the least accurate strategy among the EBA, FRQ and MCD strategies when there are over 10 attributes. When the accuracy is measured by RA_{UV} , the EQW strategy could gain over 90% relative accuracy, while it could only achieve less than 50% relative accuracy when measured by RA_{HR} . This comparison generally suggest that most of the decisions result given by EQW strategy may be very close to a user's target choice (which is determined by the WADD strategy), but are not identical. Also, in all cases, the accuracy measured by $RA_{HR_in_Kbest}$ (where $K = 5$ in the experiment) is always higher than that measured by RA_{HR} (which is a special case of $RA_{HR_in_Kbest}$ when $K = 1$). This shows that under this definition, the possibility of containing the final target choice in a K-item list is higher when K is larger. Of particular interest is that the proposed C4 strategy, which is a combination the above four basic strategies, could achieve a much higher accuracy than any of them alone. For instance, when there are 10 attributes and 1000 alternatives in the MADPs, the relative accuracy of C4 strategy could exceed the average accuracy of the four basic strategies by over 27% when the definition of $RA_{HR_in_Kbest}$ is adopted.

Figure 3.5 shows the relationship between relative accuracy and the number of alternatives (or the number of available products in a catalog) for the listed decision strategies. When the accuracy is measured by the selection of non-dominated alternatives (RA_{NDA}), all strategies except SAT could gain nearly 100% of relative accuracy without a significant difference. This generally shows that the RA_{NDA} is not a good definition of accuracy measurement in the simulated environment. When the accuracy is measured by the utility values (RA_{UV}), the accuracy of the heuristic strategies remains stable as the number of alternatives increases. With the definitions of Hit Ratio (RA_{HR}) and Hit ratio in K-best items ($RA_{HR_in_Kbest}$), however, the heuristic strategies strongly descend into a lower range of accuracies as the size of a catalog increases. This corresponds to the fact that consumers have increasing difficulties finding the best product as the number of alternatives in the catalog increases. The C4 strategy, though its accuracy decreases when the number of alternatives increases, could still maintain a considerably higher relative accuracy than that of the EBA, MCD, LEX, and FRQ strategies when using the accuracy definition of RA_{HR} and $RA_{HR_in_Kbest}$.

3.6. SIMULATION RESULTS

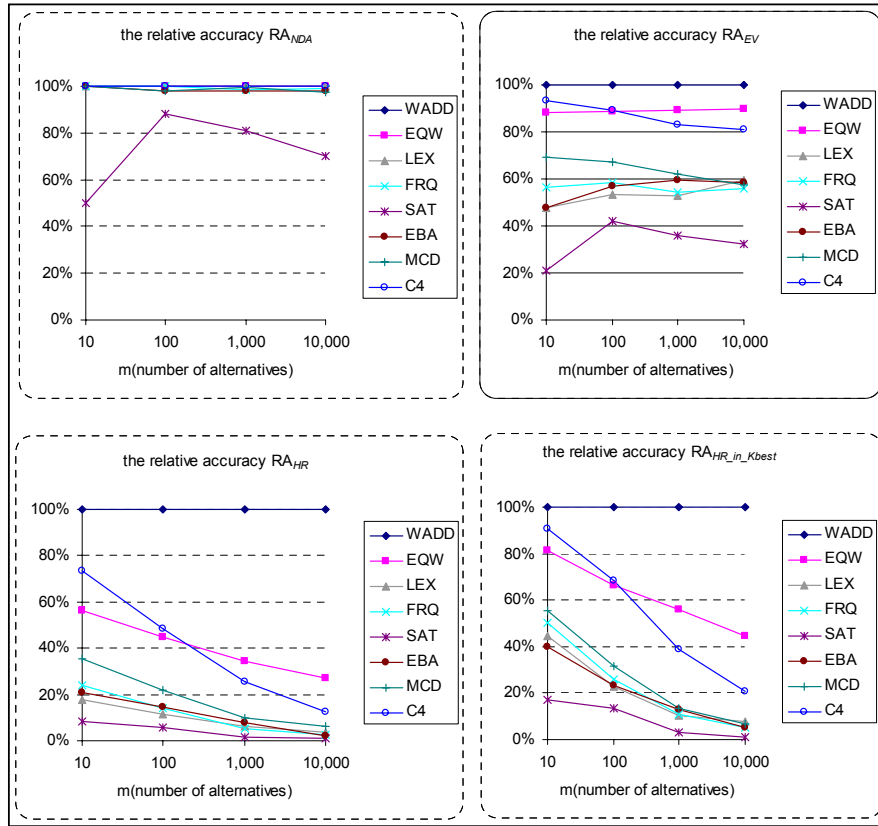


Figure 3.5: The relative accuracy of various decision strategies when solving MADPs with different number of alternatives, where $n(\text{number of attributes}) = 10$

The effect of the number of attributes on elicitation effort for each strategy is shown in figure 3.6. As we can see, the elicitation effort of the heuristic strategies increases much slower than that of the WADD strategy as the number of attributes increases. For instance, when the number of attributes is 20, the elicitation effort of the FRQ strategy is only about 25% of that of WADD strategy. The FRQ and SAT strategies require the same level of elicitation effort since both of them requires the decision maker to input a cutoff value for each attribute. Except the MCD strategy, which requires no elicitation effort in the simulation environment, the LEX strategy is the one that requires the least elicitation effort in all cases among the listed strategies. The combined C4 strategy, which could share preferences among its 4 underlying basic strategies, requires only a slightly higher elicitation effort than the EBA strategy.

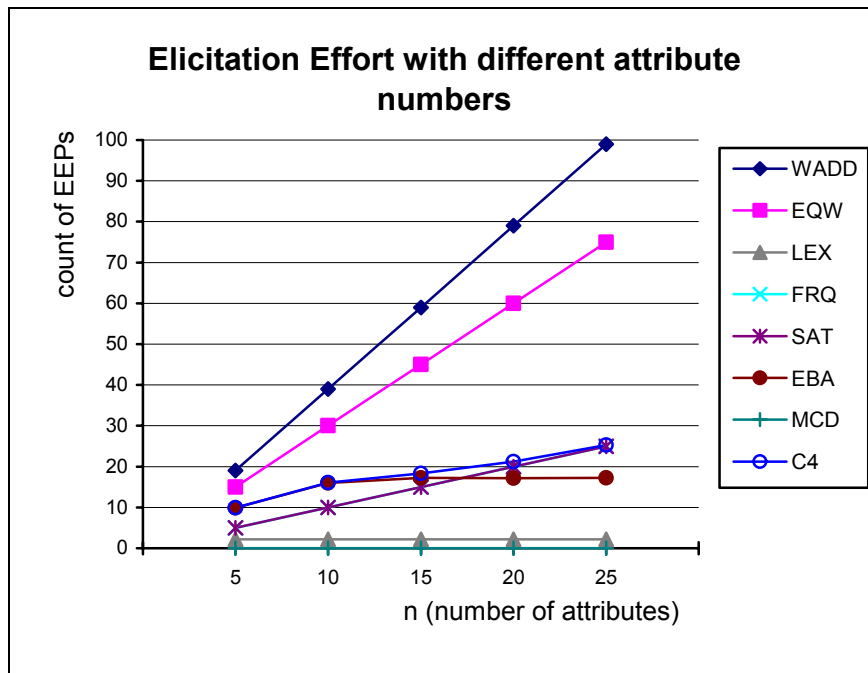


Figure 3.6: The elicitation effort of various decision strategies when solving MADPs with different number of attributes, where $m(\text{number of alternatives}) = 1,000$

Figure 3.7 shows the relationship between elicitation effort and the number of alternatives for each strategy. As the number of alternatives increases exponentially, the level of elicitation effort for WADD, EQW, MCD, SAT, and FRQ strategies remains unchanged. This shows that the elicitation effort of these strategies is unrelated to the number of alternatives that a decision problem may have. For the LEX, EBA and C4 strategies, the elicitation effort increases slowly as the number of alternatives increases. As a whole, Figure 5 shows that the elicitation effort of the studied decision strategies is quite robust to the number of alternatives that a decision problem has.

A combined study from figure 3.4 to figure 3.7 can lead to some interesting conclusions. For each category of MADPs, some decision strategies, such as WADD and EQW, could gain relatively high decision accuracy with proportionally high elicitation effort. Other decision strategies, especially C4, MCD, EBA, FRQ, and LEX, could achieve a reasonable level of accuracy with much lower elicitation effort compared to the baseline WADD strategy. Figure 3.8 illustrates the relationship between elicitation effort and $RA_{HR_in_Kbest}$ for various strategies when solving different scales of decision problems. For the MADPs with 5 attributes and 100 al-

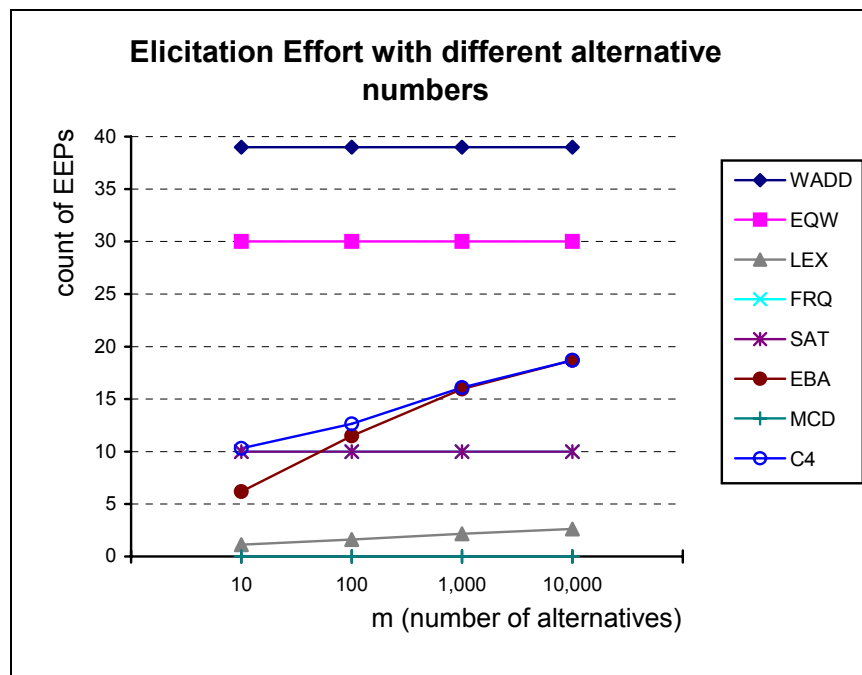


Figure 3.7: The elicitation effort of various decision strategies when solving MADPs with different number of alternatives, where n (number of attributes) = 10

ternatives, the MCD strategy could achieve around 35% relative accuracy without any elicitation effort. The C4 strategy in particular could achieve over 70% relative accuracy while only requiring about 45% elicitation effort compared to the WADD strategy.

For all the decision strategies we have studied here, we say that a decision strategy S is dominated if and only if there is another strategy S' which has higher relative accuracy and lower cognitive and elicitation effort than S in the decision problem. Figure 3.8 shows that when the MADPs have 10 attributes and 1,000 alternatives, the WADD, EQW, C4, and MCD are non-dominated approaches. However, for a smaller scale of MADPs (5 attributes and 100 alternatives), only the WADD, C4 and MCD strategies have the possibility of being the optimal strategy. This figure also shows that if the user's goal is to make decisions as accurately as possible, WADD is the best strategy among the listed strategies; while if the decision maker's goal is to have reasonable accuracy with a certain elicitation effort, then the C4 strategy may be the best option.

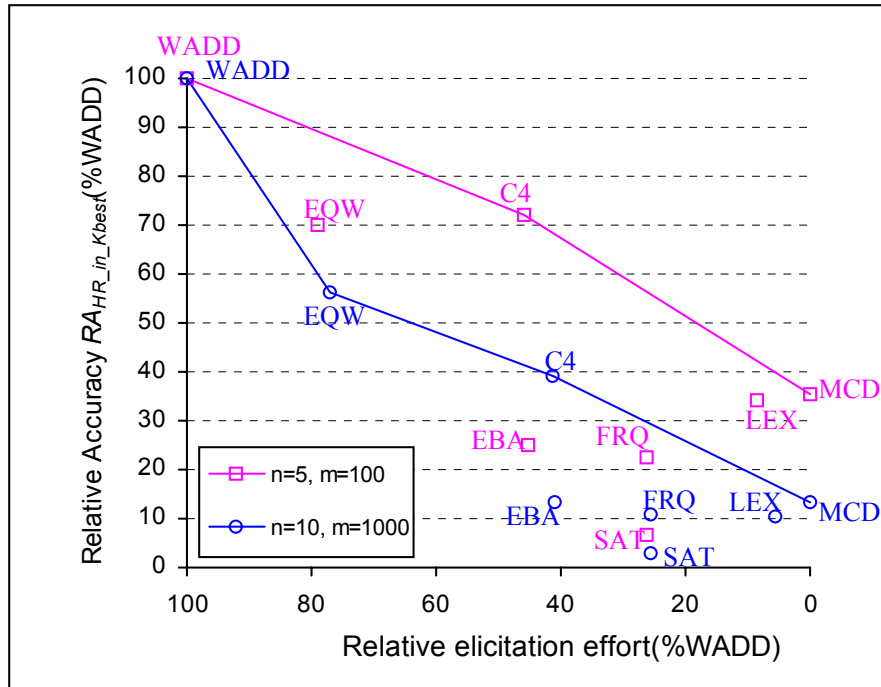


Figure 3.8: Elicitation effort/relative accuracy tradeoffs of various decision strategies

3.7 Discussion

The simulation results suggest that the tradeoff between decision accuracy and elicitation effort is the most important design consideration for inventing high performance CDSSs. That is, while advanced tools are desirable, we must not ignore the effort that users are required to make when stating their preferences.

To show how this framework can provide insights to improve user interfaces for the existing CDSSs, we have demonstrated the evaluation of the simplest decision strategies: WADD, EQW, LEX, EBA, FRQ, MCD and SAT (Payne et al., 1993). The performance of these strategies was measured quantitatively in the proposed simulation environment within the extended effort–accuracy framework. Since the underlying decision strategy determines how a user interacts with a CDSS system (preference elicitation and result processing), the performance data allowed us to discover better decision strategies and eliminate sub-optimal ones. In this sense our work provides a new design method for developing user interfaces for consumer

decision support systems.

For example, LEX is the underlying decision strategy used in the ranked list interface that many e-commerce websites employ (Pu & Kumar, 2004). However, our simulation results show that LEX produces relatively low decision accuracy, especially as products become more complex. On the other hand, a hybrid decision strategy, C4, based on the combination of LEX, EBA, MCD and FRQ, was found to be more effective. Combining LEX and EBA together for example, we can derive an interface which looks like SmartClient. EBA (elimination by aspect) corresponds to eliciting constraints from users and this feature was implemented as a constraint problem solving engine in SmartClient (Torrens et al., 2002). After users have eliminated the product space by preference constraints, they can use the LEX strategy (*ranked list*) to further examine the remaining items. Even though this hybrid strategy does not include any interface features to perform tradeoff navigation, the simulation results are still consistent with our earlier empirical work on evaluating CDSSs with real users (Pu & Chen, 2005; Pu & Kumar, 2004). That is, advanced tools such as SmartClient can achieve a higher accuracy while requiring users to expend slightly extra cognitive and elicitation effort than the basic strategies it contains.

Finally, we do emphasize that the simulation results need to be interpreted with some caution. First of all, the elicitation effort is measured by approximation. As mentioned earlier, we assumed that each EEP requires an equal amount of effort from the users. Currently, it is unknown whether this approximation would affect the simulation results largely. In addition, when measuring the decision accuracy, the WADD strategy is chosen as the baseline, assuming that it contains no error. However, this is not the case in reality. Moreover, as the MADPs in the simulation experiments are generated randomly, there is a potential gap between the simulated MADPs and the product catalog in real applications.

3.8 Summary

The acceptance of an e-commerce site by consumers strongly depends on the quality of the tools it provides to help consumers reach a decision that makes them confident enough to purchase. Evaluation of these consumer decision support tools on real-user studies makes it difficult to compare their characteristics in a controlled environment, thus slowing down the design process of such tools. In this chapter, we described a simulation environment to evaluate the performance of product search tools more efficiently. In this environment, we can simulate the underly-

CHAPTER 3. SIMULATION ENVIRONMENT FOR PERFORMANCE EVALUATION

ing decision support approach of the system to generate decision results based on the consumers' preferences and the product catalog information that the system may have. The decision results can then be evaluated quantitatively in terms of decision accuracy, interaction effort and cognitive effort described by the extended effort–accuracy framework. To show how this simulation environment can evaluate the performances of product search tools, we carried out a set of experiments to evaluate the performance of some simple decision strategies. Results show that if the decision maker's goal is to reach a reasonable level of accuracy with a moderate amount of elicitation effort, some hybrid heuristic strategies (such as the C4 strategy) may be the best option among these decision strategies.

User-Centric Algorithm for Compound Critiquing Generation

4.1 Introduction

Critiquing techniques have proven to be a popular and successful approach in online product search because it can help users express their preferences and feedbacks easily over one or several aspects of the available product space (Burke et al., 1997; Reilly et al., 2004a, 2005; Faltings et al., 2004a). The simplest form of critiquing is unit critiquing which allows users to give feedback on a single attribute or feature of the products at a time (Burke et al., 1997). For example, *[CPU Speed: faster]* is a unit critique over the *CPU Speed* attribute of the PC products. If a user wants to express preferences on two or more attributes, multiple interaction cycles between the user and the system are required. To make the critiquing process more efficient, a wise treatment is to generate compound critiques dynamically to enable users to critique on several attributes in one interaction cycle (Reilly et al., 2004a, 2005). Typically, for each interaction cycle there are a large number of compound critiques available. However, the system is able to show only a few of them on the user interface. Thus a critical issue for online product search tools based on compound critiques is to dynamically generate a list of high quality compound critiques in each interaction cycle to save the users' interaction effort.

McCarthy et al.(McCarthy et al., 2004) have proposed a method of discovering the compound critiques through the *Apriori* algorithm that has been used as

CHAPTER 4. USER-CENTRIC ALGORITHM FOR COMPOUND CRITIQUING GENERATION

a *market-basket analysis* method (Agrawal & Srikant, 1994). It treats each critique pattern as the shopping basket for a single customer, and the compound critiques are the popular shopping combinations that the consumers would like to purchase together. Based on this idea, Reilly et al. (Reilly et al., 2004a) have developed an approach called dynamic critiquing to generate compound critiques. As an improved version, the incremental critiquing (Reilly et al., 2005) approach has also been proposed to determine the new reference product based on the user's critique history. A typical interaction process of both dynamic critiquing and incremental critiquing approach is as follows. First the system shows a reference product to the user. At the same time the system generates hundreds of compound critiques from the data set via the Apriori algorithm, and then determines several of them according to their support values for the user to critique. After the user's critique is chosen, the system then determines a new reference product and updates the list of critiques for the user to select in the next interaction cycle. This process continues until the target product is found.

The Apriori algorithm is efficient in discovering compound critiques from a given data set. However, selecting compound critiques by their support values may lead to some problems. The Apriori algorithm is a data mining approach, which generates compound critiques purely based on the product space. The critiques determined by the support values can only reveal "what the system would provide," but cannot predict "what the user likes." For example, in a PC data domain if 90 percent of the products have a faster CPU and larger memory than the current reference product, it is unknown whether the current user may like a PC with a faster CPU and larger memory. Even though the system based on the incremental critiquing approach maintains a user preference model to determine which product to be shown in the next interaction cycle, some good compound critiques may still be filtered out before the user could choose because their support values do not satisfy the requirement. If the users find that the compound critiques cannot help them find better products within several interaction cycles, they may be frustrated and give up the interaction process in some situations. As a result, a better approach for generating compound critiques should allow the users to gradually approach the products they preferred and to find the target products with less number of interaction cycles.

In this thesis we believe that determining the compound critiques based on the user's preference model would be more efficient in helping users find their target products. More specifically, here we propose a new approach to generate compound critiques for online product search tools with a preference model based on multi-attribute utility theory (MAUT) (Keeney & Raiffa, 1976). In each interaction cycle our approach first determines a list of products via the user's preference model, and

then generates compound critiques by comparing them with the current reference product. In our approach, the user's preference model is maintained adaptively based on user's critique actions during the interaction process, and the compound critiques are determined according to the utilities they gain instead of the frequency of their occurrences in the data set. In this chapter we also carry out a set of simulation experiments to show that the compound critiques generated by our approach can be more efficient than those generated by the Apriori algorithm.

4.2 Related Work

4.2.1 Unit Critique and Compound Critique

Critiquing was first introduced as a the interaction style for online product search in the FindMe systems (Burke et al., 1996, 1997), and was perhaps best known for the role it played in the Entrée restaurant recommender. During each cycle Entrée presents users with a fixed set of critiques to accompany a suggested restaurant case, allowing users to *tweak* or critique this case in a variety of directions; for example, the user may request another restaurant that is *cheaper* or *more formal*, for instance, by critiquing its *price* and *style* features. A similar interface approach was later adopted by the RentMe and Car Navigator systems from the same research group.

As a form of feedback critiquing has many advantages. From a user-interface perspective it is relatively easy to incorporate into even the most limited of interfaces. For example, the typical “*more*” and “*less*” critiques can be readily presented as simple icons or links alongside an associated product feature value and can be chosen by the user with a simple selection action. Contrast this to value elicitation approaches where the interface must accommodate text entry for a specific feature value from a potentially large set of possibilities, via drop-down list, for example. In addition, critiquing can be used by users who have only limited understanding of the product domain. For example, a digital camera buyer may understand that greater resolution is preferable but may not be able to specify a concrete target resolution.

While critiquing enjoys a number of significant usability benefits, as indicated above, it can suffer from the fact that the feedback provided by the user is rarely sufficiently detailed to sharply focus the next interaction cycle. For example, by specifying that they are interested in a digital camera with a *greater resolution* than the current suggestion the user is helping the system to narrow its search but this may

CHAPTER 4. USER-CENTRIC ALGORITHM FOR COMPOUND CRITIQUING GENERATION

still lead to a large number of available products to choose from. Contrast this with the scenario where the user indicates that they are interested in a *5 megapixel* camera, which is likely to reduce the number of product options much more effectively. The result is that critiquing-based product search tools can suffer from protracted interaction sessions, when compared to value elicitation approaches.

The critiques described so far are all examples of, what we refer to as, *unit* critiques. That is, they express preferences over a single feature; Entrée's *cheaper* critiques a *price* feature, and *more formal* critiques a *style* feature, for example. This too ultimately limits the ability of the search tool to narrow its focus, because it is guided by only single-feature preferences from cycle to cycle. Moreover it encourages the user to focus on individual features as if they were independent and can result in the user following false-leads. For example, a price-conscious digital camera buyer might be inclined to critique the price feature until such time as an acceptable price has been achieved only to find that cameras in this region of the product space do not satisfy their other requirements (e.g., high resolution). The user will have no choice but to roll-back some of these price critiques, and will have wasted considerable effort to little or no avail.

An alternative strategy is to consider the use of what we call *compound* critiques (Reilly et al., 2004a). These are critiques that operate over multiple features. This idea of compound critiques is not novel. In fact the seminal work of Burke *et al.* (Burke et al., 1996) refers to critiques for manipulating multiple features. For instance, in the Car Navigator system, an automobile search tool, users are given the option to select a *sportier* critique. By clicking on this, a user can increase the *horsepower* and *acceleration* features, while allowing for a greater *price*. Similarly we might use a *high performance* compound critique in a PC search system to simultaneously increase *processor speed*, *RAM*, *hard-disk capacity* and *price* features.

Obviously compound critiques have the potential to improve search efficiency because they allow the system to focus on multiple feature constraints within a single cycle. However, until recently, the usefulness of compound critiques has been limited by their static nature. The compound critiques have been hard-coded by the system designer so that the user is presented with a fixed set of compound critiques in each interaction cycle. These compound critiques may, or may not, be relevant depending on the products that remain at a given point in time. For instance, in the example above the *sportier* critique would continue to be presented as an option to the user despite the fact that the user may have already seen and declined all the relevant car options.

4.2.2 Generating Compound Critiques based on Apriori

One strategy for dynamically generating compound critiques, called *dynamic critiquing* (Reilly et al., 2004a), discovers feature patterns that are common to remaining products on every interaction cycle. Essentially, each compound critique describes a set of products in terms of the feature characteristics they have in common. For example in the PC domain, a typical compound critique might be for *Faster CPU* and a *Larger Hard-Disk*. By clicking on this the user narrows the focus of the system to only those products that satisfy these feature preferences. The Apriori data-mining algorithm (Agrawal, Mannila, Srikant, Toivonen, & Verkamo, 1996) is used to quickly discover these patterns and convert them into compound critiques on each interaction cycle.

The first step involves *generating critique patterns* for each of the remaining product options in relation to the currently presented example. Figure 4.1 shows how a critique pattern for a sample product p differs from the current recommendation for its individual feature critiques. For example, the critique pattern shown includes a “<” critique for Price— we will refer to this as $[Price <]$ —because the comparison laptop is cheaper than the current recommendation.

The next step involves *mining compound critiques* by using the Apriori algorithm (Agrawal et al., 1996) to identify groups of recurring unit critiques. The basic idea is to generate candidate critique sets of a particular size and then scan the database to count these to see if they are frequent. By this method it is able to find the co-occurrence of unit critiques like $[ProcessorSpeed >]$ infers $[Price >]$. Apriori returns lists of compound critiques of the form $\{[ProcessorSpeed >], [Price >]\}$ along with their *support* values (i.e., the % of critique patterns for which the compound critique holds). During this process a compound critiques can be selected only if its support value is bigger than a certain threshold (Typically it is given as 0.25).

	Current Product	Product p	Critique Pattern
Manufacturer	Apple	Sony	!=
Price (Euro)	2450	2039	<
Screen-Size (inches)	17	13.3	<
Operating System	Mac OS X	Windows XP Home	!=
RAM (MB)	2048	1024	<
HardDisk (GB)	100	120	>
Processor Type	Intel Core Duo	Intel Core Duo	=
Speed (GHz)	2.16	1.83	<
Weight (Kgs)	2.5	1.9	<
Battery-Life (Hours)	5.6	6	>

Figure 4.1: Generating a critique pattern.

CHAPTER 4. USER-CENTRIC ALGORITHM FOR COMPOUND CRITIQUING GENERATION

The final step is to *grade compound critiques*. It is not practical to present large numbers of different compound critiques as user-feedback options in each cycle. For this reason, a filtering strategy is used to select the k most useful critiques for presentation based on their support values. Two filtering strategies are proposed in (Reilly et al., 2004a): (1) LS – the top 5 critiques with the lowest support are chosen; (2) HS – the top 5 critiques with the highest support are chosen. The HS strategy leads to the frequent application of small compound critiques whereas the LS strategy leads to the frequent of large critiques. The experimental results in (Reilly et al., 2004a) indicated that the LS strategy has the ability to reduce the average session length by 33% compared to the HS strategy.

Additionally, the above dynamic critiquing approach for compound critiquing generation can be extend by constructing a model of user preferences from the critiques specified. It is important to notice that users are not always consistent in the feedback they provide, so the aim of the model is to resolve any preference conflicts that may arise as the session proceeds. Put simply, when making a recommendation, the system computes a compatibility score for every product (informed by their critiquing history), and ranks them accordingly. This *incremental critiquing* approach (Reilly, McCarthy, McGinty, & Smyth, 2004c) has been shown to deliver significant benefits in terms of recommendation quality and efficiency in prior evaluations. The more detail about this approach can be found in (Reilly, 2007).

4.2.3 Other Critiquing Systems

Other than the unit critiquing and compound critiquing approaches that we have mentioned, a number of various critiquing approaches based on examples also have been proposed in recent years. The ATA system (Linden et al., 1997) uses a constraint solver to obtain a set of optimal solutions and shows five of them to the user (three optimal ones and two extreme solutions). The Apt Decision (Shearin & Lieberman, 2001) uses learning techniques to synthesize a user's preference model by critiquing the example apartment features. The SmartClient approach (Pu & Faltings, 2000) gradually refines the user's preference model by showing a set of 30 possible solutions in different visualizations to assist the user making a travel plan. The main advantage of these example-based critiquing approaches is that users' preferences can be stimulated by some concrete examples and users are allowed to reveal preferences both implicitly (choosing a preferred product from a list) and explicitly (stating preferred values on specific attributes). In fact, these example-based critiquing approaches can also "generate" compound critiques easily by comparing the critique examples with the current recommended product. But they are

4.3. GENERATING COMPOUND CRITIQUES BASED ON MAUT

more viewed as tradeoff navigation because users have to state the attribute values that they are willing to compromise against those that they are hoping to improve (Pu & Kumar, 2004). The approach of generating compound critiques that we proposed here can be regarded as an example-based critiquing approach because we determine the compound critiques from a list of critique examples. However, the difference is that our approach concentrates on constructing user's preferences automatically through the choice of the compound critiques, and the user can save some effort in stating the specific preferences values during the interaction process.

4.3 Generating Compound Critiques based on MAUT

MAUT(Keeney & Raiffa, 1976) is a well known and powerful method in decision theory for ranking a list of multi-attribute products according to their utilities and it has been introduced in Chapter 2. Here we use its simplified weighted additive form to calculate the utility of a product $O = \langle x_1, x_2, \dots, x_n \rangle$ as follows:

$$U(\langle x_1, \dots, x_n \rangle) = \sum_{i=1}^n w_i V_i(x_i) \quad (4.1)$$

where n is the number of attributes that the products may have, the weight $w_i (1 \leq i \leq n)$ is the importance of the attribute i , and V_i is a value function of the attribute x_i which can be given according to the domain knowledge during the design time.

The general algorithm of the interaction process with this proposed approach (called Critique_MAUT) is illustrated by Figure 4.2 & 4.3. We use a preference model which contains the weights and the preferred values for the product attributes to represent the user's preferences. At the beginning of the interaction process, the initial weights are equally set to $1/n$ and the initial preferences are stated by the user. In each interaction cycle, the system generates a set of critique strings for the user to select as follows. Instead of mining the critiques directly from the data set based on the Apriori algorithm, the Critique_MAUT approach first determines top K (in practice we set $K = 5$) products with maximal utilities, and then for each of the top K products, the corresponding critique string is determined by comparing it with the current reference product. This "from case to critique pattern" process of producing compound critique strings is straightforward and has been illustrated in (McCarthy et al., 2004).

After the user has selected one of the critique strings, the corresponding critique product is assigned as the new reference product, and the user's preference model is updated based on this critique selection. For each attribute, the attribute value

CHAPTER 4. USER-CENTRIC ALGORITHM FOR COMPOUND CRITIQUING GENERATION

```
PM— user's preference model;  
ref— the current reference product;  
IS— item set;  
CI— critique items;  
CS— critique strings;  
U— utility value;  
 $\beta$ — the weight adaptive factor
```

```
//The main procedure  
1. procedure Critique_MAUT ( )  
2.   PM = GetUserInitialPreferences ( )  
3.   ref = GenInitialItem (PM)  
4.   IS  $\leftarrow$  all available products – ref  
5.   while not UserAccept (ref)  
6.     CI = GenCritiqueItems (pm, IS)  
7.     CS = GenCritiqueStrings (ref, CI)  
8.     ShowCritiqueInterface (CS)  
9.     id = UserSelect (CS)  
10.    ref' = CIid  
11.    ref  $\leftarrow$  ref'  
12.    IS  $\leftarrow$  IS – CI  
13.    PM = UpdateModel (PM, ref)  
14.  end while  
15.  return
```

```
//user select the critique string  
16. function UserSelect (CS)  
17.   cs = the critique string user selects  
18.   id = index of cs in CS  
19.   return id
```

Figure 4.2: The algorithm of critiquing based on MAUT (Part I).

4.3. GENERATING COMPOUND CRITIQUES BASED ON MAUT

```
//select the critique items by utilities
20. function GenCritiqueItems (PM, IS)
21.   CI = {}
22.   for each item  $O_i$  in IS do
23.      $U(O_i) = \text{CalcUtility}(PM, O_i)$ 
24.   end for
25.    $IS' = \text{Sort\_By\_Utility}(IS, U)$ 
26.    $CI = \text{Top\_K}(IS')$ 
27.   return CI

//Update user's preferences model
28. function UpdateModel(PM, ref)
29.   for each attribute  $x_i$  in ref do
30.      $[pv_i, pw_i] \leftarrow PM$  on  $x_i$ 
31.     if  $(V(x_i) \geq pv_i)$ 
32.        $pw'_i = pw_i \times \beta$ 
33.     else
34.        $pw'_i = pw_i / \beta$ 
35.     end if
36.      $PM \leftarrow [V(x_i), pw'_i]$ 
37.   end for
38.   return PM
```

Figure 4.3: The algorithm of critiquing based on MAUT (Part II).

CHAPTER 4. USER-CENTRIC ALGORITHM FOR COMPOUND CRITIQUING GENERATION

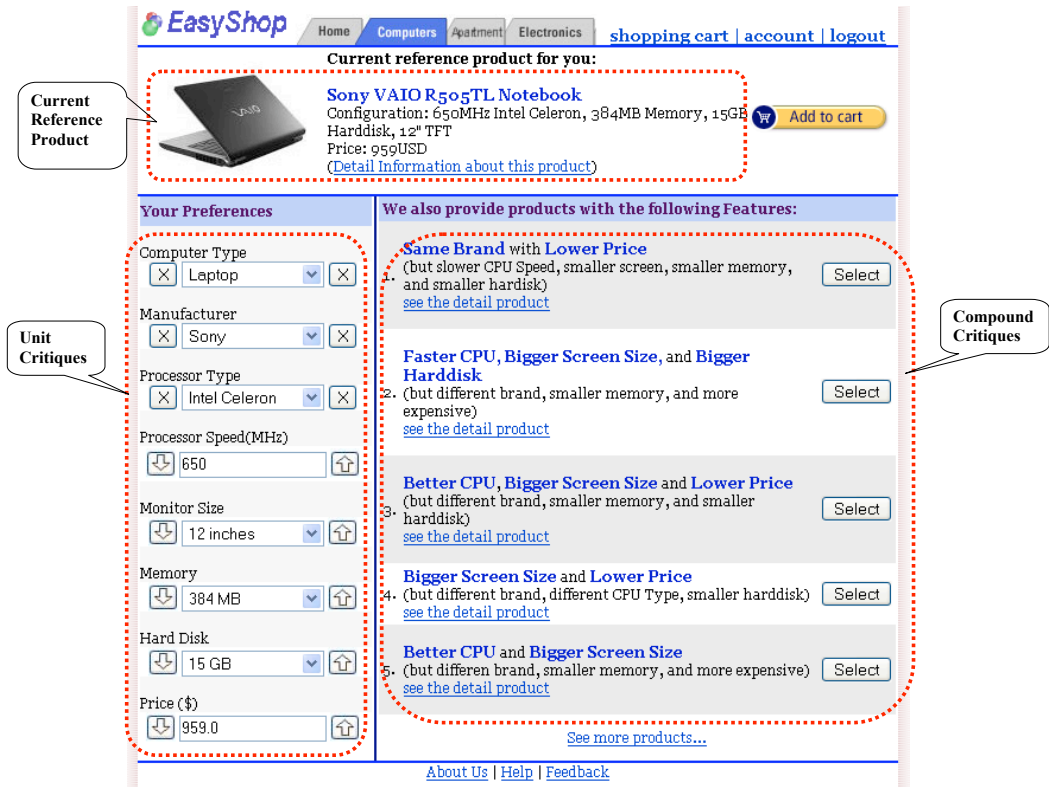


Figure 4.4: Screen-shot of the prototype system that we designed to support both unit and compound critiques.

of the new reference product is assigned as the preference value, and the weight of each attribute is adaptively adjusted according to the difference between the old preference value and the new preference value. If the new preference value is equal or better than the old preference value, the current weight on the given attribute is multiplied by a factor β , otherwise it is divided by β (See line 30-36 on Figure 4.3). In practice we set the factor $\beta = 2.0$. Based on the new reference product and the new user preference model, the system is able to generate another set of critique strings for the user to critique until the user finds the target product or stops the interaction process.

Figure 4.4 shows a screen-shot of a personal computer search tool that we have developed based on the proposed approach. In this interface, the user can see the detail of a reference product, and he or she can either conduct a unit critique or a compound critique to reveal additional preferences. It is very similar to the user interface proposed in (Reilly et al., 2005) except two differences. One difference is

that here we would like to show 5 different compound critiques generated by our approach in each interaction cycle. Another difference is on the way we present the compound critiques. We find it is not very convenient for users to read long sentences describing compound critiques, so here we split a compound string into two parts: the positive part containing features that will be improved if the critique is chosen, and the negative part containing features that will be compromised. The positive part is highlighted because we believe that users will pay more attention on these features.

4.4 An Illustrative Example

While both the above two approaches can generate compound critiques dynamically, in fact they are very different in the way of deciding the set of critiques. Here we make a simple example to illustrate how each of them could generate compound critiques.

Suppose the system is a laptop search tool which only provides 6 products as shown in Table 4.1. Here we also suppose that the product *PC1* is currently selected.

Table 4.1: The example laptop dataset

	Price	Brand
PC1	2000	IBM
PC2	3000	Sony
PC3	2500	Toshiba
PC4	2500	Sony
PC5	1800	Sony
PC6	1800	IBM

The Apriori Approach

For the Apriori approach, The first step is to first discover all critique patterns. Based on the method we have introduced in Section 4.2.2, in this case the critique patterns for products *PC2* to *PC6* can be generated as shown in Table 4.2.

In this situation 3 compound critiques will be considered with their support values: (1) $\{[Price >], [Brand! =]\}$ with support value 0.6; (2) $\{[Price <], [Brand! =]\}$ with support value 0.2; and (3) $\{[Price <], [Brand =]\}$ with support value 0.2. In this case since the first compound critique is bigger than the threshold (0.25), so the

Table 4.2: Critique patterns for the products.

	Pattern(Price)	Pattern(Brand)
PC2	>	!=
PC3	>	!=
PC4	>	!=
PC5	<	!=
PC6	<	=

compound critique $\{[Price >], [Brand! =]\}$ will be presented in the next interaction cycle.

The MAUT Approach

For the MAUT approach, we first need to determine the utility function. Based on the current selection and domain knowledge, we can determine the value function for each attribute. For the value function of the attribute *Price*, suppose that the minimal price is 1000 (assign the utility value as 1.0) and the maximal price is 3000 (assign the utility value as 0.0), As a result the value function V_1 for attribute *Price* could be given as below (here we also assume the value function on price is in linear form):

$$V_1(x) = \frac{3000 - x}{2000}, 1000 \leq x \leq 3000 \quad (4.2)$$

For the attribute *Brand*, since the user currently selects *IBM*, the value function V_2 for the attribute *Brand* can be given as

$$V_2(x) = \begin{cases} 1 & x = IBM \\ 0 & others \end{cases} \quad (4.3)$$

The weights for the utility function will be given as $1/n$ by default and be adjusted during the interactive process. Suppose in the current situation they are $w_1 = 0.8$ and $w_2 = 0.2$ respectively. Consequently, the utility value for each product can be calculated according to the Equation 4.1 and the results are shown in Table 4.3.

According to their utility values, we can rank these products as $PC6 \succ PC5 \succ PC4 = PC3 \succ PC2$.

Finally the compound critiques will be generated by comparing the candidate products with the current product. If only one compound critique is presented in

Table 4.3: The utility values of the products in the example laptop dataset

	Price	Brand	Utility Value
PC2	3000	Sony	0.0
PC3	2500	Toshiba	0.2
PC4	2500	Sony	0.2
PC5	1800	Sony	0.48
PC6	1800	IBM	0.68

the next interaction cycle, then it will be $\{[Price <], [Brand =]\}$ (the corresponding compound critique of product *PC6*).

From this simple example we can see the compound critique $\{[Price >], [Brand! =]\}$ generated by the Apriori algorithm is irrelevant to the user’s current preferences, and might lead the user to a more expensive product. By comparison, the compound critique $\{[Price <], [Brand =]\}$ generated by the MAUT approach is closer to the user’s true preference, and it is able to help the user find a cheaper product.

4.5 Experiments and Results

We carried out a set of simulation experiments to compare the performance of the basic unit critiquing approach (denoted as Critique_Unit), the incremental critiquing approach which generates compound critiques with the Apriori algorithm (denoted as Critique_Apriori) (Reilly et al., 2005), and our approach generating compound critiques based on MAUT (denoted as Critique_MAUT).

The general procedure of the simulation experiments was based on the simulation environment that we proposed in Chapter 3. The performance of each approaches was measured under the extended accuracy–effort framework. Instead of generating product data randomly, here we utilize two existing real product datasets. The apartment data set used in (Pu & Kumar, 2004) contains 50 apartments with 6 attributes: *type*, *price*, *size*, *bathroom*, *kitchen*, and *distance*. The PC data set (Reilly et al., 2004a) contains 120 PCs with 8 different attributes. This data set is available at <http://www.cs.ucd.ie/staff/lmcginty/PCdataset.zip>. In each session of our experiments, a product from the dataset was appointed as the target choice, and the simulation process is to find this target product out from the dataset. Each product was appointed to be target choice for 10 times, and the number of interaction cycles for finding the target choice were recorded. We assume at the beginning the user will reveal several preferences to the system. According to our work on user’s behavior (Zhang, Pu, & Viappiani, 2006b), we observed that

CHAPTER 4. USER-CENTRIC ALGORITHM FOR COMPOUND CRITIQUING GENERATION

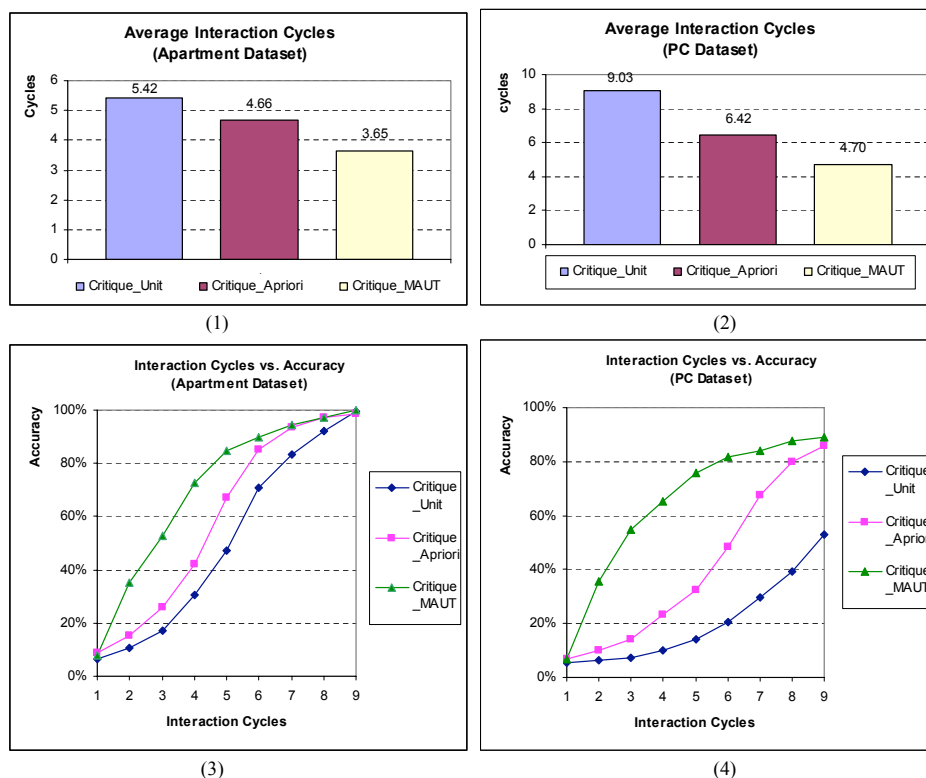


Figure 4.5: The results of the simulation experiments with the PC data set and the apartment data set. (1)The average interaction cycles for the apartment data set; (2)The average interaction cycles for the PC data set; (3) the accuracy of finding the target choice within given number of interaction cycles for the apartment data set; (4) the accuracy of finding the target choice within given number of interaction cycles for the PC data set.

an average user states about 3 initial preferences. Thus we randomly determined the number of the initial preferences from 1 to 5 in each session.

In each interaction cycle we assume that both the Critique_Apriori and the Critique_MAUT approaches generate 5 different compound critiques for user to choose. The Critique_Apriori approach adopts the *lowest support* (LS) strategy with a minimum support threshold of 0.25 to generate compound critiques.

Figure 4.5 (1) and (2) show the average interaction cycles of different approaches. Compared to the baseline Critique_Unit approach, the Critique_Apriori approach can reduce the average interaction cycles by 15% (for apartment data set) and 28% (for PC data set) respectively. This validates earlier research that the interaction

cycles can be reduced substantially by utilizing compound critiques. Moreover, the results show that the proposed Critique_MAUT approach can reduce the interaction cycles over 20% compared to the Critique_Apriori approach (significant difference, $p < 0.01$).

We define the *accuracy* as the percentage of finding the target product successfully within a certain number of interaction cycles. As shown in Figure 4.5 (3) and (4), the Critique_MAUT approach has a much higher accuracy than both the Critique_Unit and the Critique_Apriori approach when the number of interaction cycles is small. For example, in the apartment data set, when the user is assumed to make a maximum of 4 interaction cycles, the Critique_MAUT approach enables the user to reach the target product successfully 85% of the time, which is 38% higher than the Critique_Unit approach, and 18% higher than the Critique_Apriori approach.

Compound critiques allow users to specify their preferences on two or more attributes simultaneously thus they are more efficient than unit critiques. When the compound critiques are shown to the user, it is interesting to know how often they are applied during the interaction process. Here we also compared the application frequency of compound critiques generated by MAUT and the Apriori algorithm in our experiments. As shown in Figure 4.6, the application frequency of compound critiques generated by the Critique_MAUT method are much higher than those generated by the Critique_Apriori method for both the PC data set (29% higher) and the Apartment Data set (13% higher). We believe this result offers an explanation of why the Critique_MAUT method can achieve fewer interaction cycles than the Critique_Apriori method.

4.6 Discussions

The key improvement of the proposed Critique_MAUT approach is that the compound critiques are determined through their utility values given by MAUT instead of their support values given by the Apriori algorithm. Since a utility value measures the product's attractiveness according to a user's stated preferences, our approach has the potential to help the user find the target choice in an earlier stage. The simulation experiment results verified this advantage of the Critique_MAUT approach.

Modeling user's preferences based on MAUT is not a new idea. In fact, MAUT approach can enable users to make tradeoff among different attributes of the product space. For example, Stolze has proposed the scoring tree method for building

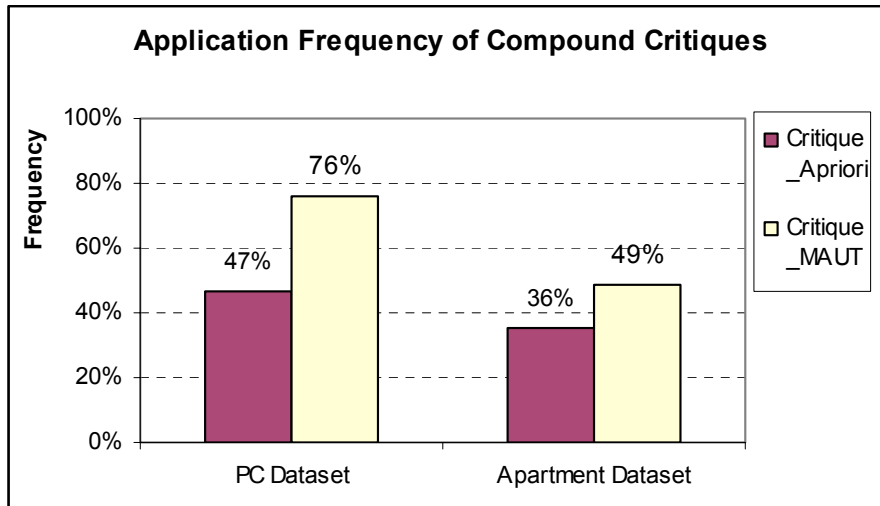


Figure 4.6: Application frequency of compound critiques generated by MAUT and the Apriori algorithm

interactive e-commerce systems based on MAUT (Stolze, 2000). However, in our approach we designed an automatic manner to gradually update the user's preference model according to the critique actions. The users are not obliged to state their preference value or to adjust the weight value on each attribute explicitly thus the interaction effort can be substantially reduced.

There are several limitations in our current work. The user's preference model is based on the weighted additive form of the MAUT approach, which might lead to some decision errors when the attributes of the products are not mutually preferentially independent (MPI). If some attributes are preferentially dependent, our approach is still able to generate the compound critiques. However, the user needs to spend some extra effort to determine the utility function which is more complicated than equation (4.1). Furthermore, currently the experiments are based on artificial users with simulated interaction processes. We assume that the artificial user has a clear and firm target in mind during the interaction process. In reality this assumption is not always true because the user may change his or her mind during the interaction process. Moreover, our approach determines the compound critiques only based on utility values. Some researchers have pointed out that the approach of combining similarity and diversity may provide better performance (Smyth & McClave, 2001). So far we haven't compared the Critique_MAUT approach with the approach based on similarity and diversity.

4.7 Summary

Generating high quality compound critiques is essential in designing critique-based conversational recommender systems. The main contribution of this work is that we propose a new approach in generating compound critiques based on the multi-attribute utility theory. Unlike the popular method of generating compound critiques directly by the Apriori algorithm, our approach adaptively maintains the user's preference model based on MAUT during the interaction process, and the compound critiques are determined according to the utility values. Our simulation experiments show that our approach can reduce the number of interaction cycles substantially compared to the unit critiques and the compound critiques generated by the Apriori algorithm. Especially when the user is willing to make only a few interactions with the system, our approach enables the user with a much higher chance in finding the final target product. In the next chapter, we organize a set of real-user studies to compare the performance of these critiquing approaches in terms of the actual number of interaction cycles, decision accuracy and the degree of users' satisfaction.

Real-User Evaluations of Critiquing-based Search Tools

5.1 Introduction

In this chapter we will compare two approaches to the dynamic generation of compound critiques through real-user studies. The first approach, Apriori, uses a data-mining algorithm to discover patterns in the types of products remaining in the system, then converts these patterns into compound critiques. The second approach, MAUT, takes a utility-based decision theory approach to identify the most suitable products for users and converts these into a compound critique representation. In Chapter 4 both these two approaches have been introduced and compared in a simulated environment. The simulation results show that the MAUT approach can reduce the number of interaction cycles substantially compared to the Apriori approach. However, this simulation process has some drawbacks in modeling some of the different characteristics that real consumers may have. The simulation method only simulated a very simple interaction process between the artificial user and the system. It may be not very consistent to the interaction process in real situations. Also, we cannot obtain users' subjective opinions on these two approaches through simulation method. To make the results be more convincing, it is better to evaluate systems by real-user studies. A direct comparison of these techniques in a real-user evaluation setting is needed to fully understand their relative pros and cons.

CHAPTER 5. REAL-USER EVALUATIONS OF CRITIQUING-BASED SEARCH TOOLS

To that end, two research groups¹ have come together to carry out this comparison for the approaches we each take. We set out to design a suitable evaluation platform, called *CritiqueShop*, to comparatively evaluate these techniques in a realistic product search situation. Ideally, this work would allow us to learn how to improve and/or look at ways of marrying ideas from both approaches.

In this chapter we report two trials of real-user studies for the comparison of these two different approaches. In the next section, we introduce the evaluation platform that we have developed for carrying out various real-user studies. Then we introduce the setup of these two trials of user studies. Next we report the experimental results that we obtained during these user studies. Finally we summarize our work at the end of this chapter.

5.2 The CritiqueShop Evaluation Platform

To carry out real-user studies, we must implement a product search tool so that both approaches can be reached by end-users. We have two main concerns about the real-user studies before we start to implement the product search tool. The first concern is that this tool should be accessible online directly through users' browsers, and users should be able to participate our user studies at any where and at any time without supervision. This demands us to implement this tool as a web-based system. And more importantly, the procedure of the user study should be self-explanatory. The second concern is that the search tool should works like a real online product search tool. It should be easily used and all the product information in this tool should be the latest information from the market.

We developed the *CritiqueShop* system to meet the above requirement. It was implemented as a web-based system by using the Google Web Toolkit (GWT),² which enables developers to build AJAX applications with Java language. CritiqueShop provides a wizard-like procedure so that end-users can easily follow the procedure of the user study. For each web page, the current task is highlighted at the left side and an instruction description is given on the top side. Figures 5.11 to 5.15 show some screen-shots of the general evaluation procedure based on the CritiqueShop evaluation platform. The CritiqueShop platform is available online and can be accessed by the following URL: <http://www.critiqueshop.com>. The system that

¹The two research groups are: the Human Computer Interaction group from EPFL, and the Adaptive Information Cluster group from University College Dublin.

²Please visit <http://code.google.com/webtoolkit/> to see more introduction and download this tool.

Table 5.1: Design of Trial 1 (Sept. 2006)
Dataset: Laptop

Group	Stage 1		Stage 2	
	Approach	Interface	Approach	Interface
A (37 users)	MAUT	Detailed	Apriori	Simplified
B (46 users)	Apriori	Simplified	MAUT	Detailed

we implemented for the real-user studies in this chapter can be accessed from the URL <http://gwtui.critiqueshop.com>.

5.3 Real-User Evaluation Trial 1

Accordingly, we designed a trial that asks users to compare two systems; one implementing the Apriori approach, and one implementing the MAUT approach. For this trial (referred to as *Trial 1*), we gathered a dataset of 400 laptop computers. A total of 83 users separately evaluated both systems by using each system to find a laptop that they would be willing to purchase. The order in which the different systems were presented was randomized and at the start of the trial they were provided with a brief description of the basic product search interface to explain the use of unit and compound critiques and basic system operation.

However, this trial was limited in two important ways. Firstly, the interface used to present the MAUT-generated compound critiques was different to the interface used to present the Apriori-generated compound critiques; each conveyed different types and amounts of information. These interfaces were selected as they had been used in prior evaluations of the respective approaches and Figures 5.9 (*simplified*) and 5.10 (*detailed*) illustrate the differences between the two interfaces. The *simplified* interface was used to display Apriori-generated compound critiques, translating them into one line of descriptive text. The MAUT compound critiques were displayed in the more informative *detailed* interface. Each MAUT compound critique was separated into two parts, highlighting the attributes that will be improved if the critique is chosen, as well as the compromises that will have to be made. In addition, the user is given the opportunity to examine the product that will be recommended on the next cycle if the compound critique is chosen. We believe that in this trial, the interface for presenting the compound critiques was having a greater influence than the compound critiques themselves on individual

Table 5.2: The datasets used in the online evaluation of the dynamic critiquing product search tools.

	Laptop	Camera
# Products	403	103
# Ordinal Attributes	7	7
# Nominal Attributes	3	1

users. Hence it was not possible to attribute the observed performance difference to the difference in critique-generation strategy since the relative importance of the interface differences was unclear.

The second limitation was that it was performed on one dataset only – the laptop dataset. In reality, an e-commerce product search tool may be used for many different types of products. It maybe reasonable to assume that the results from a real-user evaluation on one dataset may not be the same on other datasets. For example, we may find that a system employing Apriori-generated critiques performs better on one dataset, and MAUT-generated critiques perform better on another. Also, as some of our peers have suggested, asking users to perform the evaluation on the same dataset twice with different product search tools might bias the results towards the second system, as users will have become more familiar with the product domain.

5.4 Real-User Evaluation Trial 2

To address the limitations highlighted in *Trial 1*, we commissioned a second trial (referred to as *Trial 2*). For this trial we decided to homogenize the interfaces used by both techniques by using the detailed interface style for both the Apriori and MAUT-generated compound critiques. In this way we can better evaluate the impact of the different critique-generation strategies. In addition, we also used another dataset (containing 103 digital cameras) in order to thwart a domain learning effect. Table 5.2 lists the characteristics of the two datasets used in this trial. The attributes used to describe the digital camera dataset can be seen in Figure 5.8, and the attributes for the laptop dataset are shown in Figure 5.10.

For *Trial 2* we used a within-subjects design. Each participant evaluated the two critiquing-based product search tools in sequence. In order to avoid any carryover effect, we developed four (2×2) experiment conditions. The manipulated factors

5.4. REAL-USER EVALUATION TRIAL 2

Table 5.3: Design of Trial 2 (Nov. 2006)
Interface: Detailed

Group	Stage 1		Stage 2	
	Approach	Dataset	Approach	Dataset
C (19 users)	MAUT	Laptop	Apriori	Camera
D (23 users)	MAUT	Camera	Apriori	Laptop
E (22 users)	Apriori	Laptop	MAUT	Camera
F (19 users)	Apriori	Camera	MAUT	Laptop

Table 5.4: Demographic characteristics of participants

Characteristics		Trial 1 (83 users)	Trial 2 (85 users)
Country	Ireland	55	51
	Switzerland	26	31
	Other Countries	2	3
Age	<20	3	26
	20-24	28	38
	25-29	44	16
	≥30	8	5
Online Shopping Experience	Never	26	31
	≤ 5 times	55	51
	>5 times	2	3

are the approach order (MAUT first vs. Apriori first) and product dataset order (digital camera first vs. laptop first). Participants were evenly assigned to one of the four experiment conditions, resulting in a sample size of roughly 20 subjects per condition cell. Table 5.3 shows the details of the user-study design.

The real-user studies were carried out based on the *CritiqueShop* platform, containing all instructions, interfaces and questionnaires to end-users. The wizard-like trial procedure was easy to follow and all user actions were automatically recorded in a log file. During the first stage, users were instructed to find a product (laptop or camera) they would be willing to purchase if given the opportunity. After making a product selection, they were asked to fill in a post-stage questionnaire to evaluate their view of the effort involved, their decision confidence, and their level of

trust in the product search tool. Next, decision accuracy was estimated by asking each participant to compare their chosen product to the full list of products to determine whether or not they preferred another product. The second stage of the trial was almost identical, except that this time the users were evaluation a different approach/dataset combination. Finally, after completing both stages, participants were presented with a final questionnaire which asked them to compare both product search tools. Figures 5.7 to 5.10 present some screenshots of the systems we developed for these real-user trials.

5.5 Evaluation Results

5.5.1 Interaction Efficiency

To be successful, product search tools must be able to efficiently guide a user through a product-space and, in general, short interaction sessions are to be preferred. For this evaluation, we measure the length of a session in terms of interaction cycles, i.e. the number of products viewed by users before they accepted the system's recommendation. For each approach/dataset combination we averaged the session-lengths across all users. It is important to remember that any sequencing bias was eliminated by randomizing the presentation order in terms of critiquing technique and dataset: Sometimes users evaluated the Apriori-based approach first and other times they used the MAUT-based approach first. Similarly, sometimes users operated on the camera dataset first and other times on the laptop dataset first (for *Trial 2*).

In the real-user evaluation *Trial 1*, we only used the laptop dataset for both systems. Figure 5.1 shows the results of the average session length on both systems. As we can see, for new users (try the system first), the Apriori-based system can gain shorter interaction cycles (7.7 for Apriori vs. 8.9 for the MAUT). However, for return users (try the system second time), the MAUT-based system appears to be more efficient (10.2 for Apriori vs. 8.7 for the MAUT).

Figure 5.2 presents the results of the evaluation on the laptop dataset showing the average number of cycles for Apriori and MAUT based product search tools according to whether users used the Apriori or the MAUT-based system first or second in *Trial 2*. The results presented for the Laptop/MAUT combination are consistent with the results from *Trial 1*. Users need 10.1 (for new users) or 9.2 (for return users) cycles to reach their target product. The Laptop/Apriori system can achieve an average session-length around 7, better than the performance in *Trial 1*.

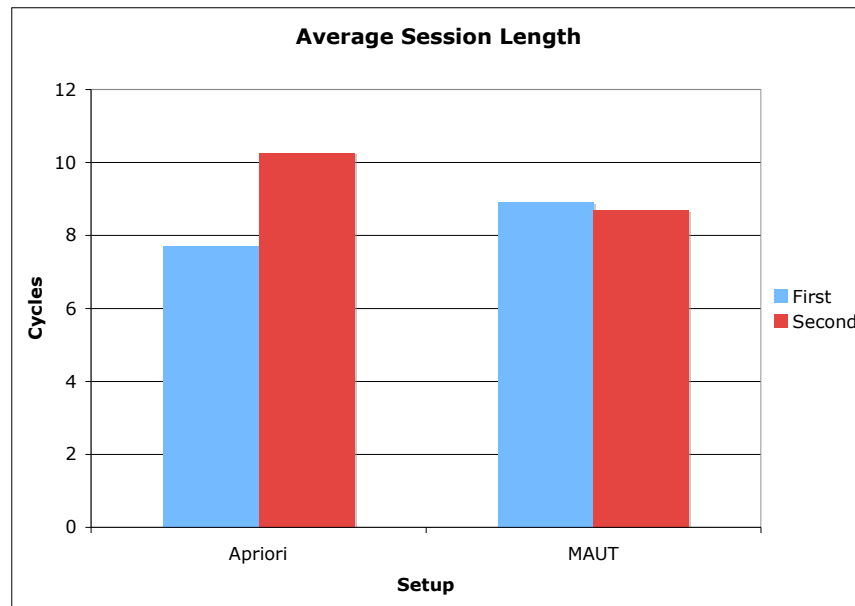


Figure 5.1: Average session lengths for both approaches on the laptop dataset (Trial 1).

The only difference of the Apriori-based system in two Trials is the user interfaces. In *Trial 1*, the Apriori-based system adopts a simplified interface; in *Trial 2*, it adopts the detailed interface.

We also measured the average session length on the digital camera dataset for both two systems in *Trial 2*. The results for this dataset are presented in Figure 5.3, and show a benefit for the MAUT-based approach to critique generation, which enjoyed an average session length of 4.1 cycles, compared to 8.5 cycles for the Apriori-based approach (significantly different, $p = 0.016$).

Dataset complexity is likely to be a factor when it comes to explaining this difference in performance. For example, the increased complexity of the laptop dataset (403 products or 10 attributes) compared to camera dataset (103 products of 8 attributes) suggests that the Apriori approach may offer improvements over MAUT in more complex product spaces. Overall, both product search tools are quite efficient. From a database of over 100 digital cameras, both are able to recommend cameras that users are willing to purchase in 10 cycles or less, on average. The results indicate that both tools are also very scalable. For instance, the laptop database contains over 400 laptop computers and yet users still find suitable laptops in just over 10 cycles. Although the product catalogue size has increased four-fold, session-lengths have increased by just 30% on average.

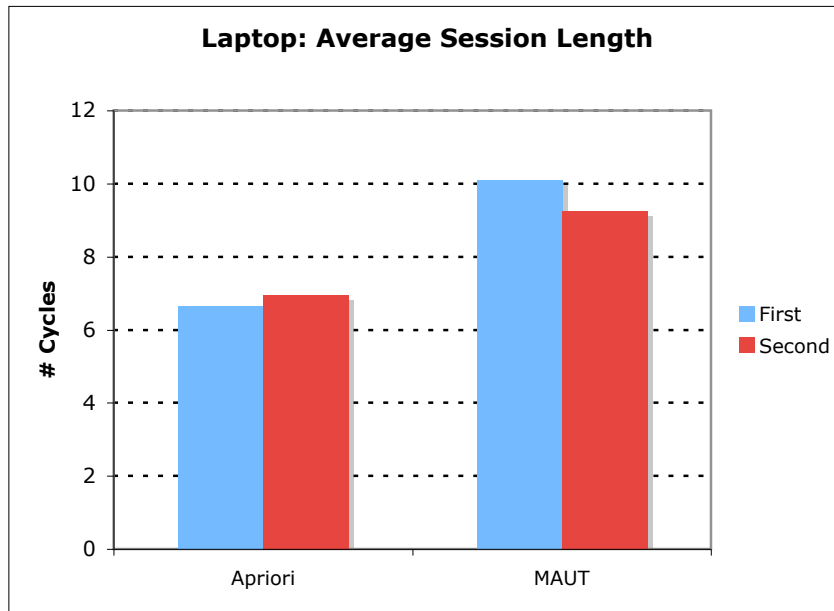


Figure 5.2: Average session lengths for both approaches on the laptop dataset (Trial 2).

5.5.2 Recommendation Accuracy

Session-length is just one performance metric for an interactive product search tool. The search tools should also be measured by the *quality* of the search made to users over the course of a session (McSherry, 2003). One way to estimate search quality is to ask users to review their final selection with reference to the full set of products (see (Pu & Chen, 2005)). Accordingly the quality or *accuracy* of the search tool can be evaluated in terms of percentage of times that the user chooses to stick with their selected product. If users consistently select a different product the search tool is judged to be not very accurate. If they usually stick with their selected product then the search tool is considered to be accurate.

The real-world datasets in our real-user studies are relatively large compared to datasets used in other real-user trials and the amount of products contained in these datasets presented us with some interface problems. For example, the laptop dataset contains over 400 products. Revealing all of these products to the users at once would lead user confusion. Also, presenting large numbers of products makes it very difficult for users to locate the actual product they desire. To deal with this, we designed the interface to show 20 products at a time while also providing the users with the facility to sort the products by attribute. Such an interface is called

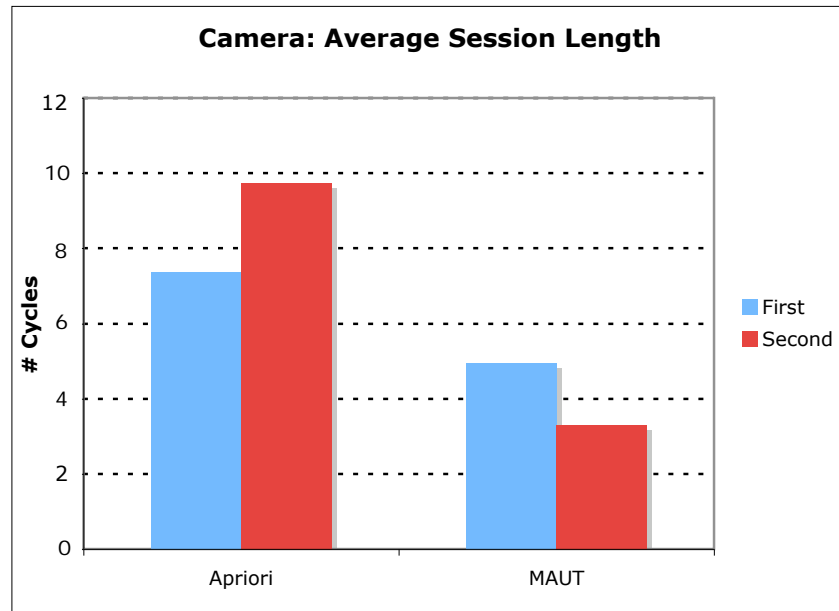


Figure 5.3: Average session lengths for both approaches on the camera dataset (Trial 2).

ranked-list and had been used as baseline in earlier research (Pu & Kumar, 2004). The bottom half of the interface showed the product they originally accepted and allowed them to select that if they so wished.

Figure 5.4 presents the average accuracy results for both approaches on both datasets in Trial 2. Interestingly it appears that the MAUT approach produces more accurate search results. For example, it achieves 68.4% accuracy on the laptop dataset and 82.5% on the camera dataset. This means that, on average, 4 out of 5 users didn't find a better camera when the entire dataset of cameras was revealed to them. The Apriori approach performed reasonably well, achieving an accuracy of 57.9% and 64.6% on the camera and laptop datasets respectively. The difference in accuracy between the two approaches on camera dataset is significant (82.5% vs 57.9%, $p = 0.015$). However, the difference in accuracy on laptop dataset is not significant (68.4% vs. 64.6%, $p = 0.70$).

Thus, despite the fact that users seemed to enjoy shorter sessions using the Apriori-based approach on the laptop dataset, they turned out to be selecting less optimal products as a result of these sessions. Users were significantly more likely to stick with their chosen laptop when using the MAUT-based product search tool.

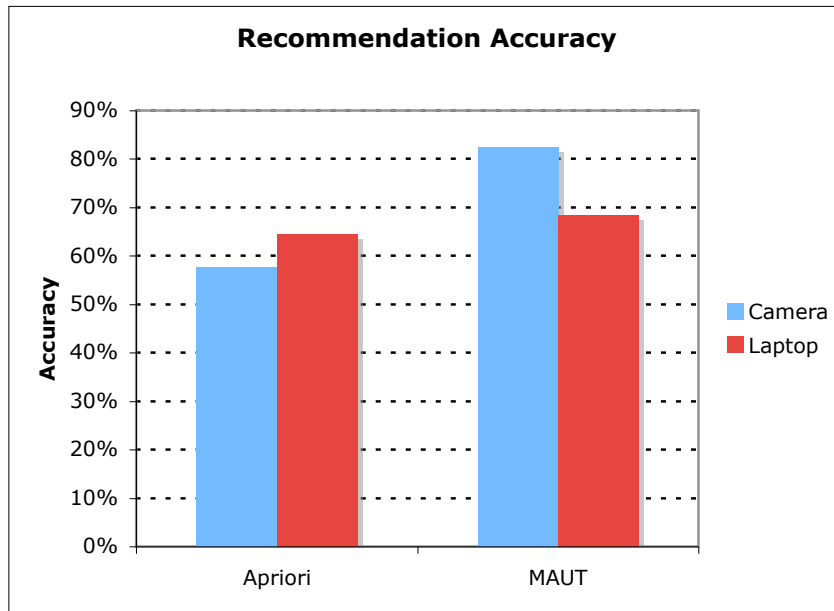


Figure 5.4: Average search accuracy of both approaches on both datasets (Trial 2).

5.5.3 User Experience

In addition to the above performance-based evaluation we were also interested in understanding the quality of the user experience afforded by the different critique generation strategies. To test this we designed two questionnaires to evaluate the response of users to the product search tools. The first (post-stage questionnaire) was presented to the users twice: once after they evaluated the first system and again after they evaluated the second system. This questionnaire asked users about their experience using the system. After the users had completed both stages and both questionnaires, they were presented with a final questionnaire that asked them to compare both systems directly to indicate which they preferred.

Post-Stage Questionnaires

Following the evaluation we presented users with a post-study questionnaire in order to gauge their level of satisfaction with the system. For each of 11 statements (see Table 5.5). The agreement level ranked from -2 to 2, where -2 is strongly disagree, and 2 is strongly agree. We were careful to provide a balanced coverage of both positive and negative statements so that the answers are not biased by the user's expression style. A summary of the responses is shown in Figure 5.5.

Table 5.5: Evaluation Questionnaire

ID	Statement
S1	I found the compound critiques easy to understand.
S2	I didn't like this system, and I would never use it again.
S3	I did not find the compound critiques informative.
S4	I found the unit-critiques better at searching for laptops (or digital cameras).
S5	Overall, it required too much effort to find my desired laptop (or digital camera).
S6	The compound critiques were relevant to my preferences.
S7	I am not satisfied with the laptop (or digital camera) I settled on.
S8	I would buy the selected laptop (or digital camera), given the opportunity.
S9	I found it easy to find my desired laptop (or digital camera).
S10	I would use this system in the future to buy other products.
S11	I did not find the compound critiques useful when searching for laptops (or digital cameras).

From the results, both systems received positive feedback from users in terms of their ease of understanding, usability and interfacing characteristics. Users were generally satisfied with the search results retrieved by both approaches (see *S2* and *S7*) and found the compound critiques efficient (see *S5*). The results generally show that compound critiquing is a promising approach for providing product search information to users, and most indicated that they would be willing to use the system to buy products (see *S2* and *S10*).

Some interesting results can be found if we compare the average ranking level of both systems. In the first trial of the user study, participants indicated on average a higher level of understanding in MAUT approach (see *S1*, 1.18 vs. 0.86, $p = 0.006$), which shows that compound critiques provided by the MAUT approach are easier to understand. Also, on average users ranked the MAUT approach more informative (see *S3*, -0.59 vs. -0.18 , $p = 0.009$). Moreover, users are more likely to agree with the statement that the unit-critiques are better at searching for laptops with Apriori approach than the MAUT approach (see *S4*, 0.82 vs. 0.41, $p = 0.01$). In Trial 2 however, these differences were no longer significant. As we can see, the MAUT approach acquires similar scores in both trials but now the Apriori approach scores much better in the second trial when using the same interface as the MAUT approach. This would seem to support our hypothesis that the compound critique presentation mechanism has a significant role in influencing users' opinions on the compound critiques approaches.

Final Questionnaires

The final questionnaire simply asked each user to vote on which system (Apriori or MAUT) performed better in terms of various criteria such as overall preference, informativeness, interface etc. The results are presented in Figure 5.6, showing the original feedback obtained during the earlier *Trial 1* evaluation (Reilly, Zhang, McGinty, Pu, & Smyth, 2007) (which used different interface styles for the Apriori and MAUT approaches) in comparison to the feedback obtained for the current *Trial 2* (in which such interface differences were removed). As previously reported (Reilly et al., 2007), users were strongly in favour of the MAUT-based approach. However, the results shown for *Trial 2* are consistent with the hypothesis that this preference was largely due to the more informative interface styles used during *Trial 1* by the MAUT-based product search tool. In *Trial 2*, for example, we see a much more balanced response by users that gives more or less equal preference to the MAUT and Apriori-based approaches and validate the benefit of the new more informative interface.

5.6 Summary

In this chapter carried out a series of comprehensive user studies to evaluate two product search tools that differ the way they generate compound critiques. We developed an online evaluation platform to evaluate both systems using a mixture of objective criteria (such as the interaction efficiency, recommendation quality/accuracy) and subjective criteria (such as a user's perceived satisfaction). Two trials of real-user studies were carried out to deeply compare the performance of both systems. Our findings show that both critique generation approaches are very effective for helping users navigate to suitable products. Both lead to efficient product search sessions. In some situations, the MAUT-based approach appears to lead to higher quality of search results. Overall, users responded positively to both systems in terms of the recommendation performance, accuracy and the interface style. Importantly, we discovered that the presentation mechanism is crucial to the users understanding and acceptance.

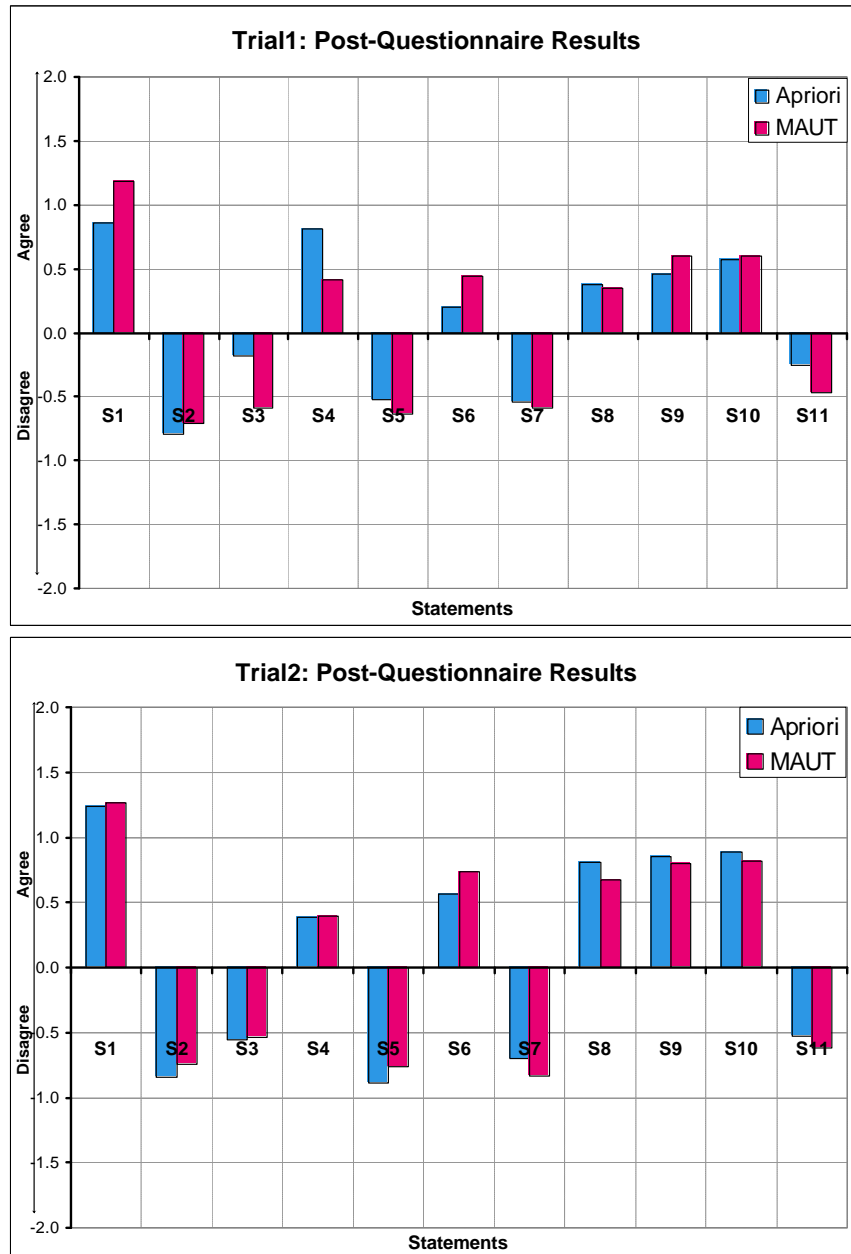


Figure 5.5: A comparison of the post-stage questionnaires from *Trial 1* and *Trial 2*.

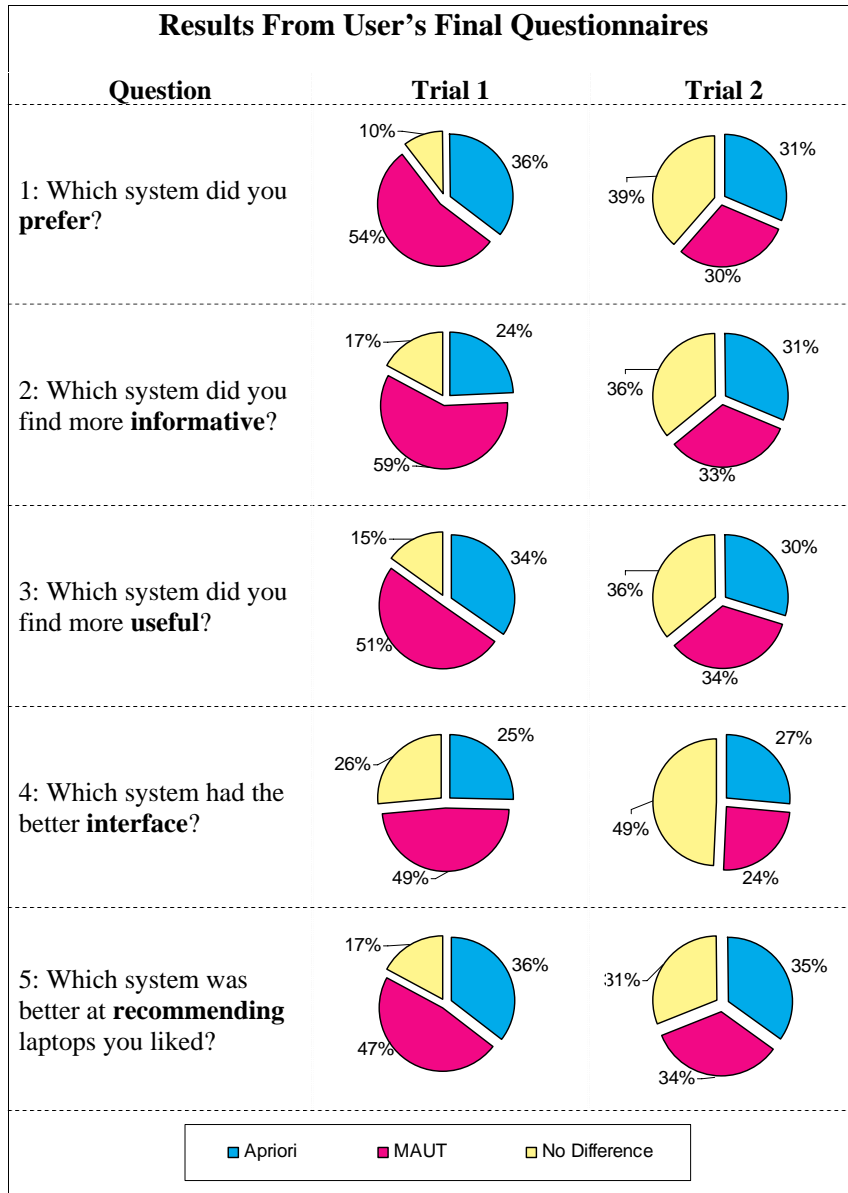


Figure 5.6: The final questionnaire results.






Instructions: Please use this system to find the laptop that you want to buy. You can either click the button for each feature on the left panel, or select one of the recommended products below. [Click here for more instructions...](#)

Laptop features

Brand: Apple

ProcessorType: Core Duo

ProcessorSpeed(GHz):

ScreenSize(inches):

Memory(MB):

HardDriveCapacity(GB):

Weight(lbs):

OperatingSystem: MacOS X 10

BatteryLife(hours):

Price(\$):

We recommend this laptop for you





Authorized Service Provider
Apple MacBook Pro

Price: 2199 USD

1759.2 EUR
2748.75 CHF

<p>Main Features:</p> <ul style="list-style-type: none"> ◆ ProcessorType: Core Duo ◆ ProcessorSpeed(GHz): 1.83 ◆ ScreenSize(inches): 15.4 ◆ Memory(MB): 1024 ◆ HardDriveCapacity(GB): 100 ◆ Weight: 5.5lbs (2.5kg) ◆ OperatingSystem: MacOS X 10.4 ◆ BatteryLife(hours): 5.6 	<p>Product Description:</p> <p>You've seen improvements in notebook performance before - but never on this scale. The Intel Core Duo powering MacBook Pro is actually two processors built into a single chip. This, combined with myriad other engineering leaps, boosts performance up to four times higher than the PowerBook G4. With this awesome power, it's a breeze to render complex 3D models, enjoy smooth playback of HD video, or host a four-way video conference.</p>
---	---

Not Satisfied with the result? you may select other recommendations listed below

- 1. Faster CPU.**
But with More Expensive.
[>>see product detail<<](#)
- 2. Faster CPU and Cheaper.**
But with Less Memory and Smaller Hard-Disk.
[>>see product detail<<](#)
- 3. Lighter and Cheaper.**
But with Different Type of CPU, Slower CPU, Smaller Screen, Less Memory, Smaller Hard-Disk and Shorter Battery Life.
[>>see product detail<<](#)
- 4. Larger Screen and Larger Hard-Disk.**
But with Different Type of CPU, Slower CPU, Less Memory, Heavier, Shorter Battery Life and More Expensive.
[>>see product detail<<](#)
- 5. Lighter and Longer Battery Life.**
But with Different Brand, Slower CPU, Smaller Screen, Different OS and More Expensive.
[>>see product detail<<](#)

Figure 5.7: Sample screen-shot of the evaluation platform (with detailed interface). Left: the unit critiquing panel; right bottom: the compound critiquing panel; center: the current recommended product panel.

CHAPTER 5. REAL-USER EVALUATIONS OF CRITIQUING-BASED SEARCH TOOLS

Initial Preferences			
Brand :	Kodak	✕★★★★★	a priority
Price(\$):	---	✕★★★★☆	important
Resolution(M Pixels):	5	✕★★★★☆	very important
OpticalZoom :	---	✕★★★★☆	important
FlashMemory(MB):	---	✕★★★★☆	a priority
ScreenSize(inches):	---	✕★★★★☆	important
Thickness(inches):	---	✕★★★★☆	important
Weight(lbs):	---	✕★★★★☆	important

Figure 5.8: Screen-shot of the initial preferences (digital cameras).

Not Satisfied with the result? you may select other recommendations listed below	
1. Different Type of CPU and Larger Screen.	I like this
2. More Memory, Heavier and More Expensive.	I like this
3. Slower CPU, Shorter Battery Life and Cheaper.	I like this
4. Smaller Screen and Lighter.	I like this
5. Faster CPU and Larger Hard-Disk.	I like this

Figure 5.9: Screen-shot of the simplified compound critiquing interface (laptop).

Not Satisfied with the result? you may select other recommendations listed below	
1. Faster CPU, More Memory and Larger Hard-Disk. But with More Expensive. >>see product detail<<	I like this
2. Larger Screen and Larger Hard-Disk. But with Different Type of CPU, Slower CPU, Heavier, Shorter Battery Life and More Expensive. >>see product detail<<	I like this
3. Faster CPU, More Memory, Larger Hard-Disk, Lighter and Longer Battery Life. But with Different Brand, Smaller Screen, Different OS and More Expensive. >>see product detail<<	I like this
4. Smaller Screen and Shorter Battery Life.	I like this
5. Cheaper. Shorter Battery Life.	I like this

SONY
Sony VAIO SZ280P/C

Main Features:

- ProcessorType:Core Duo
- ProcessorSpeed(GHz):2
- ScreenSize(inches):13.3
- Memory(MB):2048
- HardDriveCapacity(GB):120
- Weight(lbs):3.7
- OperatingSystem:WinXP Pro
- BatteryLife(hours):6
- Price(\$):2999.99

Figure 5.10: Screen-shot of the detailed compound critiquing interface (laptop).

Steps:

1. Welcome
2. Information
3. Instructions
4. Task A
5. Preferences A
6. System A
7. Questionnaire A
8. Verification A
9. Task B
10. Preferences B
11. System B
12. Questionnaire B
13. Verification B
14. Final Questionnaire
15. End

critique shop BETA

Instructions: Please read the following text carefully before you start the user study.

Thank you for participating in this user study for evaluating two different versions of our online recommender system. If you successfully complete the user study, you will be entered into a draw for a **100 EURO (or 150 CHF) voucher for Amazon.com!**

This system has a database of currently available laptops(610 cases) and digital cameras. Your task will be to use the system to discover your ideal products among them, i.e. the laptop or digital camera that you would be willing to buy. In order to be included in the draw, you must evaluate both recommenders and complete the questionnaires. The entire evaluation should not take longer than 20 minutes. Best of luck!

100 EURO

Be in with a chance to win!

When you are ready, please press the "next step" button below to start....

Next Step

Figure 5.11: Screen-shot of the CritiqueShop evaluation platform: the first welcome web page at the beginning.

Steps:

1. Welcome
2. Information
3. Instructions
4. Task A
5. Preferences A
6. System A
7. Questionnaire A
8. Verification A
9. Task B
10. Preferences B
11. System B
12. Questionnaire B
13. Verification B
14. Final Questionnaire
15. End

critique shop BETA

Instructions: Please fill in your personal details here. This information will be used for the purposes of this trial only. To be eligible for the prize, you must enter a valid email address

First Name:

Surname:

Profession:

Nationality:

Age:

Online shopping experience(frequency in one year):

Email Address:

Is this your first time to take this user study? yes no

Your personal information will only be used for this research.

Next Step

Figure 5.12: Screen-shot of the CritiqueShop evaluation platform: the web page of asking user's personal information.

CHAPTER 5. REAL-USER EVALUATIONS OF CRITIQUING-BASED SEARCH TOOLS

Steps:

1. Welcome
2. Information
3. Instructions
4. Task A
5. Preferences A
6. System A
7. Questionnaire A
8. Verification A
9. Task B
10. Preferences B
11. System B
12. Questionnaire B
13. Verification B
14. Final Questionnaire
15. End

critique shop BETA

Instructions: Please fill in this questionnaire

1: I found the compound critiques easy to understand

Strongly Disagree -2 -1 0 +1 +2 Strongly Agree

2: I didn't like this system, and I would never use it again.

Strongly Disagree -2 -1 0 +1 +2 Strongly Agree

3: I did not find the compound critiques informative.

Strongly Disagree -2 -1 0 +1 +2 Strongly Agree

4: I'm confident that I have found the laptop that I like.

Strongly Disagree -2 -1 0 +1 +2 Strongly Agree

5: Overall, it required too much effort to find my desired laptop.

Strongly Disagree -2 -1 0 +1 +2 Strongly Agree

6: The compound critiques were relevant to my preferences

Figure 5.13: Screen-shot of the CritiqueShop evaluation platform: the questionnaire web page of asking users to evaluate the system that they have just tried (post-questionnaire).

Steps:

1. Welcome
2. Information
3. Instructions
4. Task A
5. Preferences A
6. System A
7. Questionnaire A
8. Verification A
9. Task B
10. Preferences B
11. System B
12. Questionnaire B
13. Verification B
14. Final Questionnaire
15. End

critique shop BETA

Instructions: Please fill in this questionnaire based on the two systems that you have tried

1: Which system did you prefer?

First Second Neither Both

2: Which system did you find more informative?

First Second No difference

3: Which system did you find more useful?

First Second No difference

4: Which system had the better interface?

First Second No difference

5: Which system was better at recommending products (laptops or cameras) you liked?

First Second No difference

6: What are your reasons for preferring the first system?

User Interface Recommendations Interface and Recommendations I don't

7: What are your reasons for preferring the second system?

User Interface Recommendations Interface and Recommendations I don't

8: I understand the meaning of the different icons in the visual interface.

Strongly Disagree -2 -1 0 +1 +2 Strongly Agree

Figure 5.14: Screen-shot of the CritiqueShop evaluation platform: the questionnaire web page of asking users to compare two systems that they have tried (final-questionnaire).

5.6. SUMMARY

Steps:

1. Welcome
2. Information
3. Instructions
4. Task A
5. Preferences A
6. System A
7. Questionnaire A
8. Verification A
9. Task B
10. Preferences B
11. System B
12. Questionnaire B
13. Verification B
14. Final Questionnaire
15. End

critiqueshop^{BETA}

Instructions: Below is a table showing all the products available in our system. Please verify if you had found the most desired laptop in the previous step. You can click the title of each column to sort that attribute. You can either select a new product from the table if you want to change your selected product, or choose the option 'I haven't changed my mind' below.

First Page Previous Page Page 1 of 31 Next Page Last Page

select	ID	Brand	Processor Type	Processor Speed (GHz)	Screen Size (Inches)	Memory(MB)	Hard Drive Capacity (GB)	Weight(lbs)	Battery Life (hours)	Price(\$)	view
↻	52	Lenovo	Core 2 Duo	2	12.1	2048	160	1.75	7.8	1799.99	detail
↻	547	Lenovo	Core 2 Duo	2.2	15.4	2048	160	6.4	5	1807.99	detail
↻	284	Lenovo	Core 2 Duo	2	12.1	2048	160	3.5	7.8	2152.99	detail
↻	197	Lenovo	Core 2 Duo	2	12.1	2048	120	3.5	7.8	1837.99	detail
↻	58	Lenovo	Core 2 Duo	2	14.1	2048	160	5.3	3.8	1649.99	detail
↻	266	Lenovo	Core 2 Duo	2.2	15.4	2048	100	6	5.9	2287.99	detail
↻	192	Lenovo	Core 2 Duo	2.2	15.4	2048	100	6	5.9	2299.99	detail
↻	241	Lenovo	Core 2 Duo	2	15.4	2048	120	6.2	3.5	1499.99	detail
↻	99	Lenovo	Core 2 Duo	2	15.4	2048	100	6.2	5.5	1799.99	detail
↻	70	Lenovo	Core 2 Duo	2	12.1	1024	120	1.72	7.8	1599.99	detail
↻	128	Lenovo	Core 2 Duo	2	15.4	2048	100	6.2	6	2149.99	detail
↻	133	Lenovo	Core 2 Duo	2	15.4	2048	120	6.2	3.1	1599.99	detail
↻	69	Lenovo	Core 2 Duo	2.16	14.1	1024	120	5.5	9	2123.99	detail
↻	243	Lenovo	Core 2 Duo	2	12.1	2048	120	4.4	4	1499.99	detail
↻	190	Lenovo	Core 2 Duo	1.6	12.1	2048	160	4.2	7	2942.99	detail
↻	118	Lenovo	Core 2 Duo	1.6	12.1	2048	100	3.3	8.2	1999.99	detail
↻	103	Lenovo	Core 2 Duo	2	15.4	1024	160	6	3.9	1449.99	detail
↻	152	Lenovo	Core 2 Duo	2	15.4	1024	160	6	3.9	1499.99	detail
↻	293	Lenovo	Core 2 Duo	1.6	12.1	1024	160	3.3	9	2077.99	detail
↻	220	Lenovo	Core 2 Duo	2.2	15.4	1024	120	6	5.9	1952.99	detail

Figure 5.15: Screen-shot of the CritiqueShop evaluation platform: the web page of asking user to find out the product that he or she really wants from a list of all products in the dataset.

Visual Interface for Compound Critiquing

6.1 Introduction

In critiquing-based product search tools, it is important to encourage users to apply compound critiques frequently. In Chapter 5, it has been found that users often prefer the more detailed critiquing interface, rather than a simplified one. This result shows that the design of the user interface is an important factor for the overall performance of the search tool. However, to date there has been a lack of comprehensive investigation on the impact of interface design issues for critiquing-based product search tools.

In this chapter we are seeking ways to improve the performance of critiquing-based product search tools from the interface design level. Traditionally, compound critiques are represented textually with sentences (Reilly et al., 2004a, 2005). In our previous work (see Chapter 4 & 5) our online product search tool is also design in such a way, keeping displaying compound critiques with plain texts. If the product domain is complex and has many features, it often requires too much effort for users to read the whole sentence of each compound critique. We believe that such textual interfaces hamper the users' experience during the recommendation process. Aiming to solve this, here we propose a new *visual* design of the user interfaces, which represents compound critiques via a selection of value-augmented icons. Based on the *CritiqueShop* system developed in our earlier work, here we further develop

CHAPTER 6. VISUAL INTERFACE FOR COMPOUND CRITIQUING

an online shopping prototype system in both laptop and digital camera domains with visualized compound critiques, and carry out a real-user study to compare the performance of this design with the old one.

The rest of this chapter is organized as follows. We first provide a brief review of the related work about critiquing techniques. Then the two interface designs for critiquing-based product search tools are introduced. Next we describe the setup of the real-user study and report the evaluation results. Finally we present the discussion and summary of this work.

6.2 Related Work

Critiquing was first introduced as a form of feedback for product search interfaces as part of the FindMe systems (Burke et al., 1996, 1997), and is perhaps best known for the role it played in the Entrée restaurant recommender. During each cycle Entrée presents users with a fixed set of critiques to accompany a suggested restaurant case, allowing users to *tweak* or critique this case in a variety of directions; for example, the user may request another restaurant that is *cheaper* or *more formal*, for instance, by critiquing its *price* and *style* features.

The simplest form of critiquing is a *unit critique* which allows users to give feedback (eg. increase or decrease) on a single attribute or feature of the products at a time (Burke et al., 1997). It is a mechanism that gives direct control to each individual dimension. The unit critique can be readily presented as a button alongside the associated product feature value and it can be easily selected by the user. In addition, it can be used by users who have only limited understanding of the product domain. However, unit critiques are not very efficient: if a user wants to express preferences on two or more attributes, multiple interaction cycles between the user and the system are required and big jumps in the data space are not possible in one operation.

To make the critiquing process more efficient, an alternative strategy is to consider the use of what we call *compound critiques* (Burke et al., 1996; Reilly et al., 2004a). Compound critiques are collections of individual feature critiques and allow the user to indicate a richer form of feedback, but limited to the presented selection. For example, the user might indicate that they are interested in a digital camera with a higher resolution and a lower price than the current recommendation by selecting a *lower price, higher resolution* compound critique.

Obviously compound critiques have the potential to improve recommendation efficiency because they allow users to focus on multiple feature constraints within

a single cycle. Recently, several dynamic compound critique generation algorithms have been proposed. For example, the *Apriori* approach uses a data-mining algorithm to discover patterns in the types of products remaining, then converts these patterns into compound critiques (Reilly et al., 2004a). This is a *system-centric* approach, and may generate biased results when the underlying database is not well provided. In Chapter 4 we have introduced an alternative method called the *MAUT* approach to generate compound critiques. This approach takes the multi-attribute utility theory (MAUT) (Keeney & Raiffa, 1976) to model users' preferences, and then it identifies the most suitable products for users and converts them into compound critiques. The performance of this approach has been evaluated in Chapter 5.

Information visualization tools have been developed in past years to help users formulate their queries and understand the relationships between collection of information. In (Ahlberg & Shneiderman, 1994), the Starfield approach together with the dynamic query method allow users to explore information and data relationships in a large data collection. Users can manipulate attribute values using sliders, and once the values are changed, the display zooms in on a subspace, allowing information seeking at the detailed level. In (Cutting, Karger, Pedersen, & Tukey, 1992), a Scatter/Gather approach automatically clusters retrieved documents into categories and labels them with descriptive summaries. Kohonen maps cluster documents into regions of a 2-D map (Lin, Soergel, & Marchionini, 1991). Recently in (Pu & Janecek, 2003), a visual interface was implemented using semantic fish-eye views to expand search context and to allow users more opportunities to refine initial queries.

In this work we apply visualization technique on the critiquing-based product search tools. More specifically, we present the compound critiques with various meaningful icons, instead of descriptions of plain text. We believe that the visual interface can attract users to apply the compound critiques more frequently and reduce the users' interaction efforts substantially compared to the traditional textual interface.

6.3 Interface Design

One of the main focusses of this study is on the interface design for critiquing-based product search tools. In Chapter 5 we have implemented an online shopping system on the product domains of both digital cameras and laptops. It is designed in a way that allows users to concentrate on the utilization of both unit critiques and compound critiques as the feedback mechanism. The interface layout is composed

CHAPTER 6. VISUAL INTERFACE FOR COMPOUND CRITIQUING

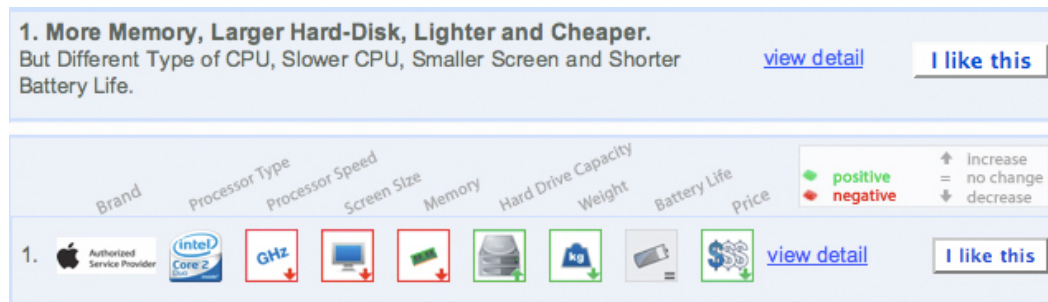


Figure 6.1: An illustrative example of the textual interface (above) and the visual interface (below).

of three main elements: a product panel, a *unit critique* panel and a *compound critique* panel. The product panel shows the current recommended product which best matches the user's preferences. In the unit critique panel, each feature is surrounded by two small buttons, which allow users to increase or decrease a value in the case of numeric features, and to change a value in categorical features such as the brand or processor type. In the compound critique panel, a list of compound critiques is shown (as textual sentences). Users can perform a compound critique by clicking the button "I like this" on its right-hand side. These three elements make up the main shopping interface and are always visible to end-users.

We are interested in getting a better perception of the role of the interface's design in the whole interaction process. We are in particular motivated by the frequent observation that people find the compound critiques too complex and admit to not actually reading all the information provided. In this context, we decide to create a visual representation of the compound critiques and to compare it with the traditional textual format through a real-user study. In the rest of this section, the design of the two interfaces are introduced in detail.

6.3.1 Textual Interface

The textual interface is the standard way to represent compound critiques and was used in our previous work (see Chapter 4 & 5). As an example, a typical compound critique will say that this product has "more memory, more disk space, but less battery" than the current best match item. A direct mapping is applied from the computed numerical values of the critique, to decide if there is more or less of each feature. Here we adopt the detailed interface where users are capable of seeing the product detail behind each compound critique. In addition, for each compound

critique, the positive critiques are listed in bold on the first line, while the negative ones follow on the second line in a normal font-weight. The Figure 5.10 shows an example of the textual interface for compound critiques.

6.3.2 Visual Interface

The visual interface used in this study was developed in several phases. The initial idea was to propose a graphical addition in order to complement the textual critiques, but this rapidly evolved into a complete alternative to a textual representation of the critiques. Three main solutions were considered: using icons, providing a graph of the different attributes or using text-effects such as tag-clouds. The first two solutions were kept and selected to build paper prototypes. The first test revealed that the icons were perceived as being closer in meaning to the textual representation, and they were hence chosen for this study.

Icons pose the known challenge that whilst being small they must be readable and sufficiently self-explanatory for users to be able to benefit from them. One difficult task was to create a set of clear icons for both datasets. We refined them twice after small pilot-studies to make them uniform and understandable. They were then *augmented* such as to represent the critiques: the icon *size* was chosen as a mechanism to represent the change of value of the considered parameter. For each parameter of a compound critique, we know if the raw value is bigger, equal or smaller. We used this to adapt the size of the iconized object thus creating an immediate visual impression of what were the features increasing or decreasing.

Whilst designing these icons we were concerned about two major issues. First of all, it rapidly appeared that changing the size of icons would be insufficiently clear or even confusing at times. This is due to a well known issue with icon design. Indicating an increase in value is not always a *positive* action: an increase in weight is a negative fact (for both cameras and laptops). Secondly we were convinced that all the icons would have to be displayed for each compound critique. The textual critiques only indicate the parameters that change, but doing so with the icons would have resulted in lines of different lengths, creating an alignment problem. These two potential issues lead us to further extend the icons with additional labelling.

Consequently we decided to add a token to the corner of each icon: an up arrow, a down arrow or an equal sign, to further indicate if the critique was respectively increasing, decreasing or equal to the current best match. At the same time we gave colors to the border and token of each icon such as to indicate if the change in value was positive, negative or equal. Green was chosen for positive, red for negative and

CHAPTER 6. VISUAL INTERFACE FOR COMPOUND CRITIQUING

grey for the status quo. For those features without value change, the corresponding icons were shown in light gray. Thus all compound critiques had an equal number of icons and the potential alignment problem was avoided. More importantly, these lines of aligned icons form a comparison matrix and they are decision supportive: a user can quickly decide which compound critique to apply by counting the number of positive or negative icons.

During our pilot user study we found that the visual interface required from users a learning effort. Two measures were taken to tune down this effect. Firstly, a miniature legend of the icons was included at the top of the compound critique panel. Secondly, in our user study we provided an instructions page to users with explanations of the meaning of icons and some icon examples.

In summary, our visual interface for compound critiques is designed as follows. We first choose meaningful icons to represent the product features in the datasets. These icons are listed in Figure 6.2. Each icon is then tagged with a color to describe the feature improvement:

- Green border: positive improvement;
- Red border: negative improvement;
- Gray icon : no difference

In addition, we also add some tokens at the right-bottom corner of each icon to represent the value increase or decrease:

- Up arrow (\uparrow): value increase;
- Down arrow (\downarrow): value decrease;
- Equal sign ($=$): no difference

For example, if the weight of a digital camera is smaller than the current product, then the corresponding icon will have an down arrow with green color, since lighter is a positive improvement for a digital camera. Figure 6.1 provides a quick comparison of the textual compound critiques and our visual design (including legend).

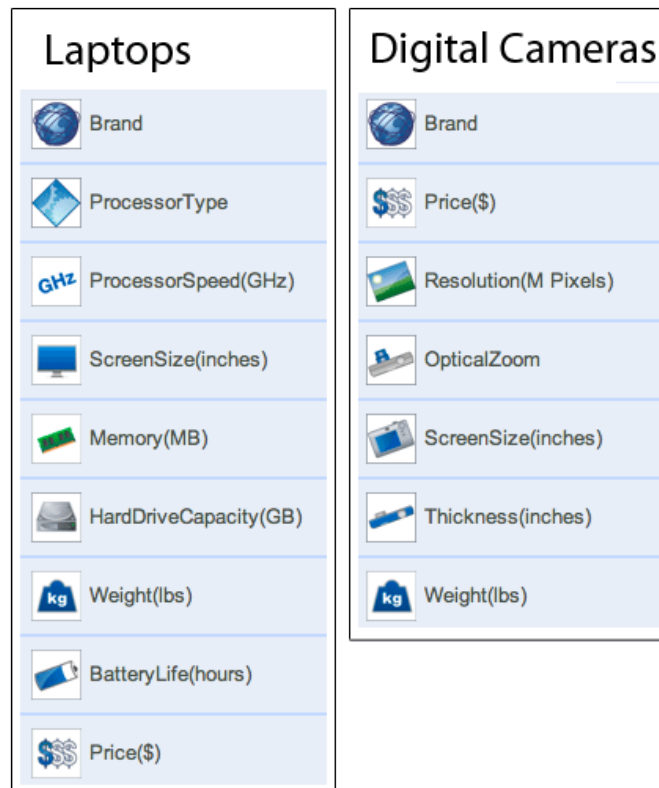


Figure 6.2: The icons that we designed for different features of the two datasets: laptops (left) and digital cameras (right).

6.4 Real-User Evaluation Trial 3

We conduct a new real-user evaluation (*Trial 3*) to compare the performance of the two interfaces in September 2007. In this section we first present the performance evaluation criteria, then we outline the setup of the evaluation and introduce the datasets and participants.

6.4.1 Evaluation Criteria

There are two types of criteria for measuring the performance of a critiquing-based recommender system: the objective criteria from the interaction logs and the subjective criteria from users' opinions. In this real-user evaluation we mainly concentrate on the following objective criteria: the average interaction length, the appli-

cation frequency of compound critiques, and the recommendation accuracy. Participants' subjective opinions include understandability, usability, confidence to choose, intention to purchase, etc. They are obtained through several questionnaires, which will be introduced later in this section.

6.4.2 Evaluation Setup

We extended the CritiqueShop evaluation platform so that it supports user studies for comparing various interface designs. The system that we used for this user study is available online through the URL <http://visual.critiqueshop.com>. In addition, the MAUT approach was applied to generate compound critiques dynamically in all situations. We adopted a within-subjects design of the real-user evaluation where each participant is asked to evaluate the two different interfaces in sequence and finally compare them directly. The interface order was randomly assigned so as to equilibrate any potential bias. To eliminate the learning effect that may occur when evaluating the second interface, we adopted two different datasets (laptops and digital cameras) so that the user was facing different domains each time. As a result, we had four (2×2) conditions in the experiment, depending on the factor of interface order (visual first vs. textual first) and product dataset order (digital camera first vs. laptop first). For each user, the second stage of evaluation is always the opposite of the first so that he or she may not take the same evaluation twice.

We implemented a wizard-like online web application containing all instructions, interfaces and questionnaires so that subjects could remotely participate in the evaluation. The general online evaluation procedure consists of the following steps.

Step 1. The participant is asked to input his/her background information.

Step 2. A brief explanation of the critiquing interface and how the system works is shown to the user.

Step 3. The user participates the first stage of the evaluation. The user is instructed to find a product (either laptop or camera, randomly determined) he/she would be willing to purchase if given the opportunity. The user is able to input his/her initial preferences to start the recommendation (see figure 6.8), and then he/she can play with both unit critiques and compound critiques to find a desired product to select. Figure 6.10 illustrates the online shopping system with the condition of visual interface and laptop dataset.

Table 6.1: Post-Stage Assessment Questionnaire

ID	Statement
S1	I found the compound critiques easy to understand.
S2	I didn't like this recommender, and I would never use it again.
S3	I did not find the compound critiques informative.
S4	I am confident that I have found the laptop (or digital camera) that I like.
S5	Overall, it required too much effort to find my desired laptop (or digital camera).
S6	The compound critiques were relevant to my preferences.
S7	I am not satisfied with the laptop (or digital camera) I found using this system.
S8	I would buy the selected laptop (or digital camera), given the opportunity.
S9	I found it easy to find my desired laptop (or digital camera).
S10	I would use this recommender in the future to buy other products.
S11	I did not find the compound critiques useful when searching for laptops (or digital cameras).
S12	Overall, this system made me feel happy during the online shopping process.

Step 4. The user is asked to fill in a post-stage assessment questionnaire to evaluate the system he/she has just tested. He/she can indicate the level of agreement for each statement on a five-point Likert scale, ranging from -2 to $+2$, where -2 means “strongly disagree” and $+2$ is “strongly agree”. We are careful to provide a balanced coverage of both positive and negative statements so that the answers are not biased by the user’s expression style. The post-stage questionnaire is composed of twelve statements as listed in table 6.1.

Step 5. Recommendation accuracy is estimated by asking the participant to compare his/her chosen product to the full list of products to determine whether or not he/she prefers another product. In our practice, the datasets are relatively large, and revealing all of these products to the user at once during the accuracy test would lead the user to confusion. To deal with this, we designed the accuracy test interface to show 20 products at a time while also providing the user with the facility to sort the products by attribute. Such interfaces are called *Rankedlists* and have been used as baseline in earlier research (Pu & Kumar, 2004).

Step 6 – 8. These are steps for the second stage of evaluation which are almost identical to the steps 3 – 5, except that this time the user is facing the system with

Table 6.2: Final Preference Questionnaire

ID	Questions
Q1	Which system did you prefer?
Q2	Which system did you find more informative?
Q3	Which system did you find more useful?
Q4	Which system had the better interface?
Q5	Which system was better at recommendaing products (laptops or cameras) you liked?
S13	I understand the meaning of the different icons in the visual interface.

a different interface/dataset combination.

Step 9. After completing both stages of evaluation, a final preference questionnaire is presented to the user to compare both systems he/she has evaluated. The user needs to indicate which interface (textual or visual) is preferred in terms of various criteria such as overall preference, informativeness, interface etc. The questions are listed in table 6.2. This final preference questionnaire also contains an extra statement (*S13*) to evaluate if the icons that we have designed are easy to understand.

6.4.3 Datasets and Participants

We noticed that the laptop and digital camera datasets used in Trial 1 & 2 are the products on the market one year ago and some information has already been out of date. This factor may influence the results of the real-user study. So we didn't use the old datasets in this experiment directly. Instead, we updated these two datasets one week before the beginning of the experiment, resulting in them containing the most recent products currently available on the market. The laptop dataset contains 610 different items. Each laptop product has 9 features: *brand, processor type, processor speed, screen size, memory, hard drive, weight, battery life, and price*. The second one is the digital camera dataset consisting of 96 cases. Each camera is represented by 7 features: *brand, price, resolution, optical zoom, screen size, thickness and weight*. Besides, each product has a picture and a detail description.

To attract users to participate in our user study, we set an incentive of 100 EURO and users were informed that one of those who had completed the user study will have a chance to win it. The user study was carried out over two weeks. Finally we obtained 83 users in total who completed the whole evaluation process. Their demographic information is shown in table 6.3. The participants were evenly assigned

Table 6.3: Demographic characteristics of participants (Trial 3)

Characteristics		Users (83 in total)
Nationality	Switzerland	36
	China	13
	France	12
	Ireland	6
	Italy	4
	Other Countries	12
Age	<20	6
	20-24	30
	25-29	40
	≥30	7
Gender	female	15
	male	68
Online Shopping Experience	Never	2
	≤ 5 times	38
	>5 times	43

to one of the four experiment conditions, resulting in a sample size of roughly 20 subjects per condition cell. Table 6.4 shows the details of the user study design.

6.5 Evaluation Results

6.5.1 Recommendation Efficiency

To be successful, a recommender system must be able to efficiently guide a user through a product-space and, in general, short recommendation sessions are to be preferred. For this evaluation, we measure the length of a session in terms of recommendation cycles, i.e. the number of products viewed by users before they accepted the system’s recommendation. For each recommendation interface and dataset combination we averaged the session lengths across all users. It is important to remember that any sequencing bias was eliminated by randomizing the presentation order in terms of interface type and dataset.

Figure 6.3 presents the results of the average session lengths with different interfaces. The visual interface appears to be more efficient than the baseline textual interface. For the laptop dataset, the visual interface can reduce the interaction cycles substantially from 11.7 to 5.5, a reduction of 53%. The difference between these two results is significant ($p = 0.03$, with ANOVA test in this chapter). For the

Table 6.4: Design of the real-user evaluation for Trial 3

Group	First stage		Second stage	
	Interface	Dataset	Interface	Dataset
I (20 users)	Textual	Camera	Visual	Laptop
II (20 users)	Textual	Laptop	Visual	Camera
III (23 users)	Visual	Camera	Textual	Laptop
VI (20 users)	Textual	Laptop	Textual	Camera

camera dataset, the visual interface can reduce the average interaction cycle from 9.7 to 7.3, a reduction of 25% (not significant, $p = 0.31$).

We also look into the detail of each interaction session to see how often the compound critiques had actually been applied. Previous studies have shown that frequent usage of compound critiques is correlated with shorter sessions. Higher application frequencies would indicate that users find the compound critiques more useful. Figure 6.4 shows application frequency of compound critiques for both systems. For the system with textual interface, the average application frequencies are respectively 7.0% (for laptops) and 9.0% (for cameras). For the system with visual interface, the average application frequency is nearly doubled to 13.6% for the laptop dataset (significant different, $p = 0.01$). For the camera dataset the application frequency is 9.9%, a 9.5% increase compared to the baseline textual interface (not significant, $p = 0.70$). Since for both systems we are using exactly the same algorithm to generate the compound critiques, the results shows that the visual interface can attract more users to choose the compound critiques during their decision process. Also, compared to the two systems with different datasets, it seems to show that the visual interface can be more effective when the domain is more complex.

6.5.2 Recommendation Accuracy

Recommenders should also be measured by the *quality* of the recommendations over the course of a session (McSherry, 2003). One factor for estimating recommendation quality is the recommendation accuracy, which can be measured by letting users to review their final selection with reference to the full set of products (see (Pu & Kumar, 2004)). Formally, here we define recommendation *accuracy* as the percentage

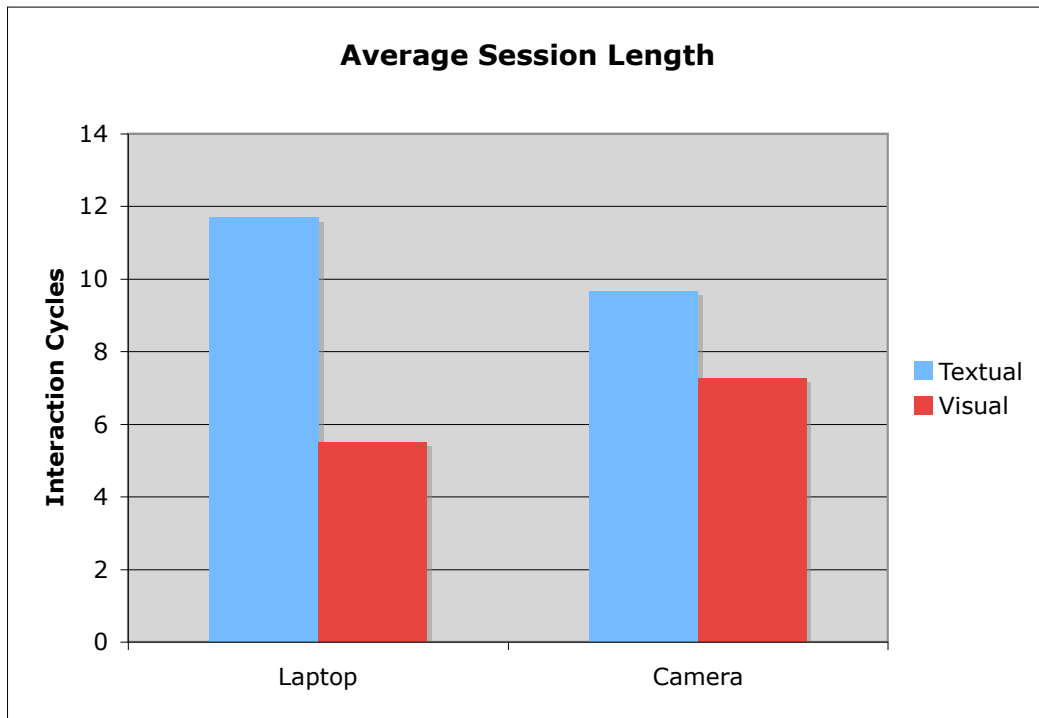


Figure 6.3: Average session lengths for both user interfaces

of times that users choose to stick with their selected product. If users consistently select a different product the recommender is judged to be not very accurate. The more people stick with their selected best-match product then the more accurate the recommender is considered to be.

Figure 6.5 presents the average accuracy results for both interfaces on both datasets. The system with textual interface performs reasonably well, achieving an accuracy of 74.4% and 65.0% on the laptop and camera datasets respectively. By comparison, the system with visual interface achieves 82.5% accuracy on the laptop dataset and 70.0% on the camera dataset. which have been increased 10% and 7% respectively. It appears that the visual interface produces more accurate recommendations. However, these improvements are not significant ($p = 0.378$ for laptop dataset, and $p = 0.648$ for camera dataset).

6.5.3 User Experience

In addition to the above objective evaluation results we were also interested in understanding the quality of the user experience afforded by the two interfaces. As

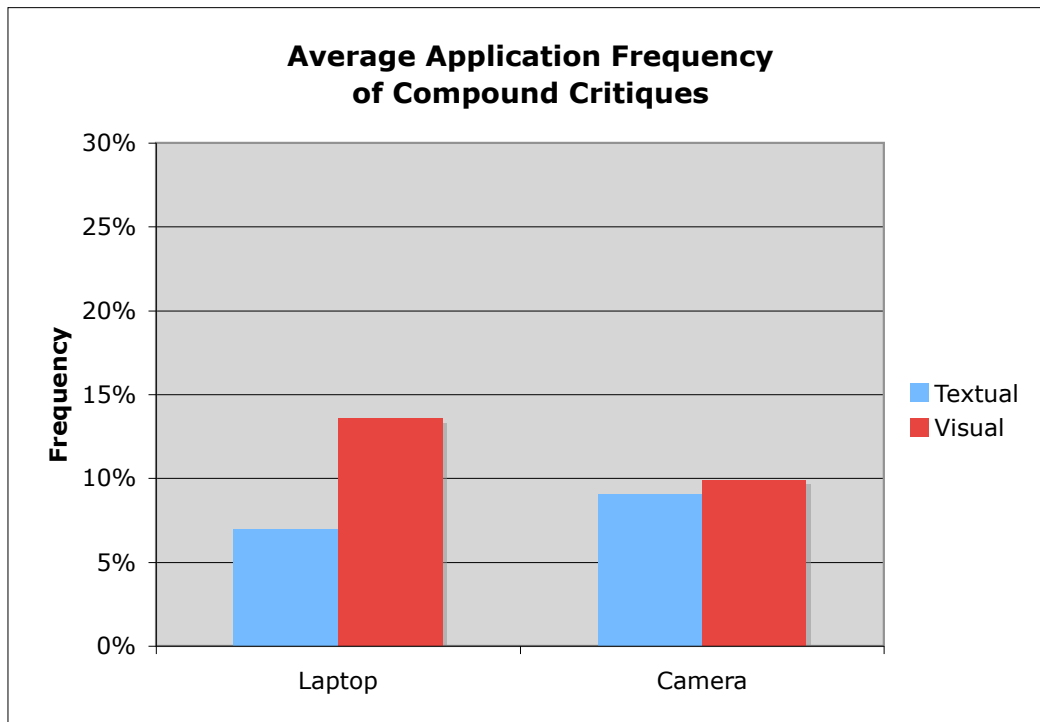


Figure 6.4: Average application frequency of the compound critiques for both user interfaces.

we have mentioned earlier, a post-stage assessment questionnaire was given when each system had been evaluated. The twelve statements are listed in table 6.1. A summary of the average responses from all users is shown in figure 6.6.

From the results we can see that both systems with different interfaces received positive feedback from users in terms of their ease of understanding, usability and interfacing characteristics. Users were generally satisfied with both systems (see *S2* and *S7*) and found them quite efficient (see *S5*). We also noticed that overall the visual interface has received higher absolute values than the baseline textual interface on all these statements. It is especially worthy to point out there are 3 statements that the visual interface has significant improvements: *S4* ($p = 0.001$), *S5* ($p < 0.01$) and *S9* ($p = 0.014$). These results show that the visual interface is significantly better than the textual interface in the criteria of efficiency, easy of usage and leading to a more confident shopping experience.

The final preference questionnaire asked each user to vote on which interface (textual or visual) had performed better. The detail of the final preference questionnaire is shown in table 6.2, and the results are shown in figure 6.7. The results show

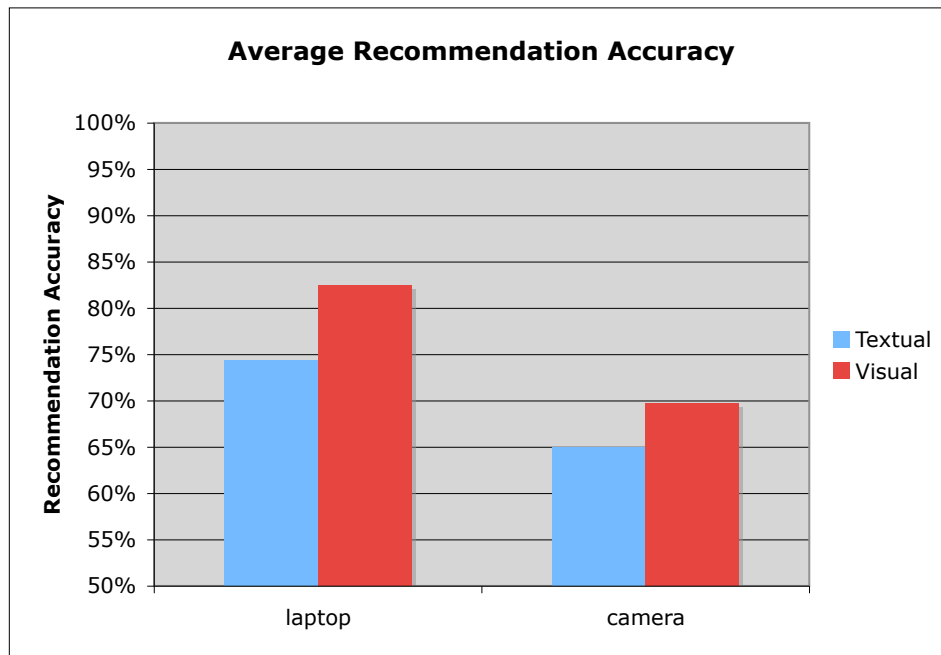


Figure 6.5: Average recommendation accuracy for both user interfaces

that overall users feel the visual interface is better than the textual interface in all given criteria. For instance, 51% of the whole users may prefer the visual interface compared to 25% of whom prefer the textual interface (see Q1). Also, more than 55% of users think the visual interface is better (see Q4). Furthermore, although the two systems have exactly the same algorithm to generate compound critiques, the visual interface can enhance users' perception on the recommendation quality (see Q5).

In the final questionnaire we provided one extra statement (S13) for users to evaluate if the icons in the visual interface are understandable. Again users were asked to score this statement from -2 (strongly disagree) to 2 (strongly agree). The overall average score is 1.23, which shows that the icons are quite understandable and have been well designed.

6.6 Discussion

It is interesting to notice that in the user study results, while the visual interface performed better than the textual interface with both laptop and camera datasets, the visual interface has achieved higher performance improvements in the situa-

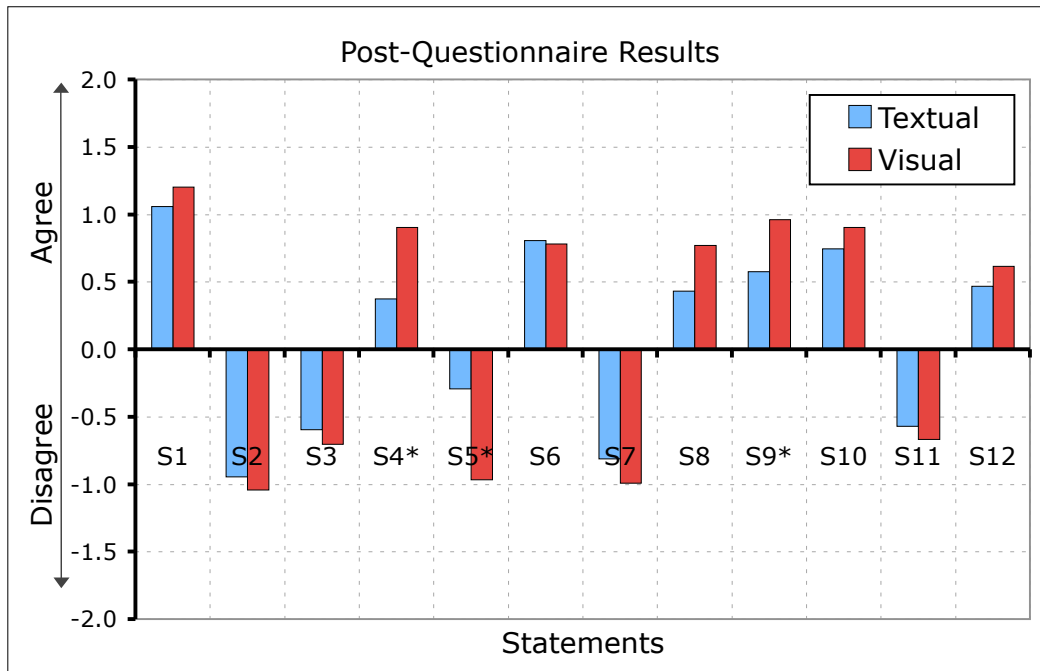


Figure 6.6: Results from the post-stage assessment questionnaire.

tion of laptop dataset than in the situation of camera dataset. The main difference between the two datasets is that the laptop assortment is more complex. It contains more products and each product has more features than the camera dataset. When the product domain is complex, the textual interface will generate very long strings of text to describe the compound critiques, which are not easy for users to read. By comparison, the visual interface could provide an intuitive and effective way for users to make decision by simply counting the number of positive and negative icons. As a result, we believe that the visual interface brings a tremendous advantage in complex domain situations.

While a large proportion of users prefer the visual interface for the critiquing-based recommender system, we also noticed that there is still a small number of users who insist on the textual interface. One reason for this phenomena is possibly that they aren't familiar with the visual interface. After all, it requires some additional learning effort to understand the meaning of various icons at the beginning. A few methods we could apply to satisfy this part of users in future include adding some detailed instructions and illustrative examples to educate new users, or in our system we could provide both textual and visual interfaces and let the users choose the preferred interfaces adaptively by themselves.

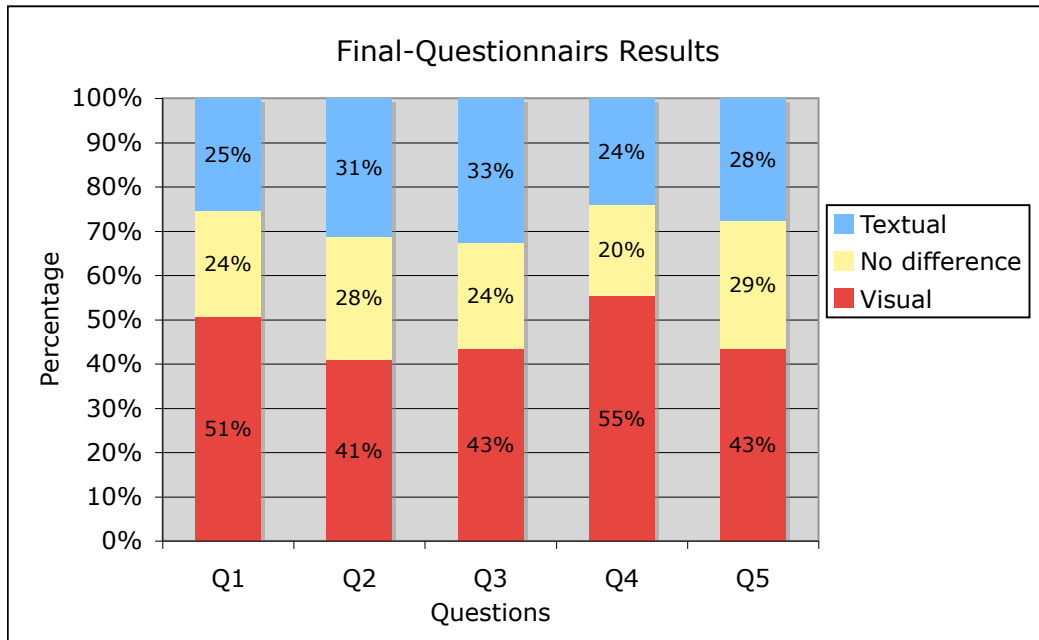


Figure 6.7: Results from the final preference questionnaire.

During the user study several users commented on the fact that our current system lacked some additional functions that currently exist in other normal websites. For example some users wanted to have a flexible search function by specifying preference values on multiple features during the interaction process. We do believe that by integrating such additional functions in the critiquing-based system, a higher overall satisfaction level can be reached. For example, it has been shown that a hybrid system is able to achieve higher overall performance (Chen & Pu, 2007). However, in this user study, the main purpose was to learn the performance of the critiquing techniques automatically generated by the system. Our current system was deliberately designed to exclude those functions in order to make sure the users would focus on the function of unit critiquing and compound critiquing that had been automatically recommended by the system. It will be our future work to find proper ways to integrate more functions into the current critiquing-based product search tools.

6.7 Summary

User interface design is an important issue for critiquing-based product search tools. Traditionally the interface is *textual*, which shows compound critiques as sentences in plain text. In this chapter we propose a new *visual* interface which represents various critiques by a set of meaningful icons. We developed an online web application to evaluate this new interface using a mixture of objective criteria and subjective criteria. Our real-user study showed that the visual interface is more effective than the textual interface. It can reduce user's interaction effort (up to 53% of reduction) and attract users to apply the compound critiques more frequently (10% of increase). Also, the system with visual interface is significantly more appreciated by users and could make users feel more confident in finding their desired products.

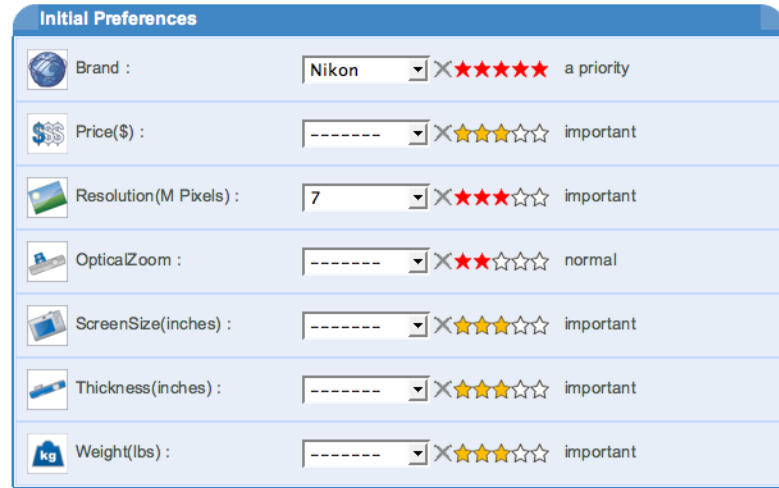


Figure 6.8: Screenshot of the interface for initial preferences (with digital camera dataset). Icons are added on the left side of features so that users could get familiar with the icon meanings.

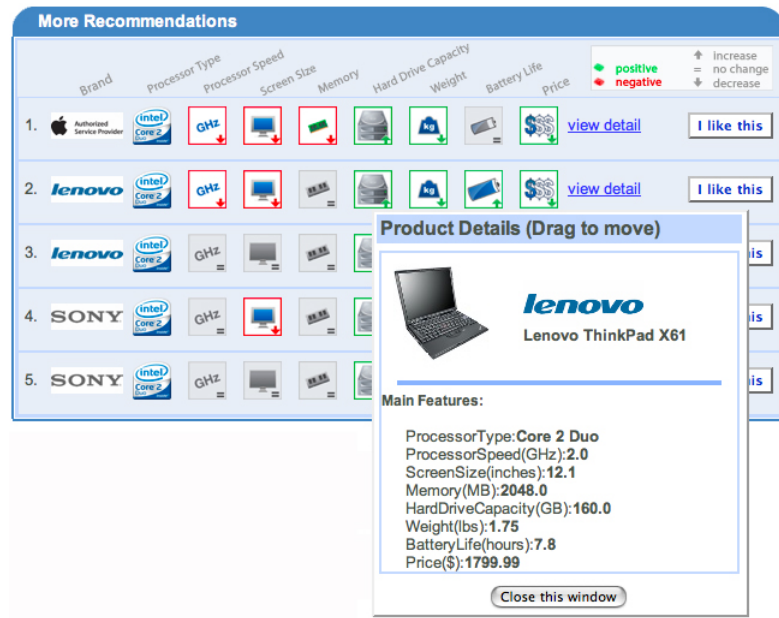


Figure 6.9: Screenshot of the interface for visual compound critiquing (with laptop dataset).

CHAPTER 6. VISUAL INTERFACE FOR COMPOUND CRITIQUING

critique BETA shop

Instructions: Please use this **visualization** interface to find the laptop that you want to buy. You can either click the button for each attribute on the left panel, or select one of the recommended products below. [Click here for more instructions...](#)

Refine Features

Brand:

ProcessorType:

ProcessorSpeed(GHz):

ScreenSize(inches):

Memory(MB):


HardDriveCapacity(GB):

Weight(lbs):

BatteryLife(hours):

Price(\$):

Our Recommendation










Authorized Service Provider
Apple MacBook Pro

Price: 1999.0 USD
1599.2 EUR
2498.75 CHF

[Buy now!](#)

Main Features:

-  ProcessorType: **Core 2 Duo**
-  ProcessorSpeed(GHz): **2.2**
-  ScreenSize(inches): **15.4**
-  Memory(MB): **2048.0**
-  HardDriveCapacity(GB): **120.0**
-  Weight: **5.5lbs (2.5kg)**
-  BatteryLife(hours): **6.0**

Product Description:

Powered by the most advanced mobile processors from Intel, the new Core 2 Duo MacBook Pro is over 50% faster than the original Core Duo MacBook Pro and now supports up to 4GB of RAM. The NVIDIA GeForce 8600M GT delivers exceptional graphics processing power. Featuring 802.11n wireless technology, the MacBook Pro delivers up to five times the performance and up to twice the range of previous-generation technologies. Quickly set up a videoconference with the built-in iSight camera. Control presentations and media from up to 30 feet away with the included Apple Remote. Connect to high-bandwidth peripherals with FireWire 800 and DVI. Innovations such as a magnetic power connection and an illuminated keyboard with ambient light sensor put the MacBook Pro in a class by itself.

More Recommendations

	Brand	Processor Type	Processor Speed	Screen Size	Memory	Hard Drive Capacity	Weight	Battery Life	Price	
1.	Apple	Intel Core 2	GHZ	15.4	2048.0	120.0	5.5	6.0	1999.0	view detail I like this
2.	lenovo	Intel Core 2	GHZ	15.4	2048.0	120.0	5.5	6.0	1999.0	view detail I like this
3.	lenovo	Intel Core 2	GHZ	15.4	2048.0	120.0	5.5	6.0	1999.0	view detail I like this
4.	SONY	Intel Core 2	GHZ	15.4	2048.0	120.0	5.5	6.0	1999.0	view detail I like this
5.	SONY	Intel Core 2	GHZ	15.4	2048.0	120.0	5.5	6.0	1999.0	view detail I like this

[Next Step](#)

Figure 6.10: Screenshot of the visual interface for the online shopping system (with laptop dataset).

Recursive Collaborative Filtering

7.1 Introduction

The collaborative filtering approach has been widely used to help people find low-risk products such as movies, books, etc (Resnick et al., 1994; Breese et al., 1998; Herlocker et al., 1999, 2002). The key idea of this approach is to infer the preference of an active user towards a given item based on the opinions of some similar-minded users in the system. Many popular e-commerce web sites – Amazon.com for example – have adopted this technique in making their online shopping system more efficient.

One of the most prevalent algorithms in collaborative filtering (CF) approach is based on the *nearest-neighbor* users (called user-based CF approach). To predict the rating value of a given item for an active user, a subset of neighbor users are chosen based on their similarity to the active user – called nearest-neighbor users – and their ratings of the given item are aggregated to generate the prediction value for it.

The conventional prediction process of the user-based CF approach selects neighbor users using two criteria: 1) they must have rated the given item; 2) they must be quite close to the active user (for instance, only the top K nearest-neighbor users are selected). However, in reality most users in recommender systems are unlikely to have rated many items before starting the recommendation process, making the

training data very sparse. As a result, the first criterion may cause a large proportion of users being filtered out from the prediction process even if they are very close to the active user. This in turn may aggravate the data sparseness problem.

To overcome the data sparseness problem and enable more users to contribute in the prediction process, here we propose a recursive prediction algorithm which relaxes the first criterion mentioned above. The key idea is the following: if a nearest-neighbor user hasn't rated the given item yet, we will first estimate the rating value for him or her *recursively* based on his or her own nearest-neighbors, and then we use the estimated rating value to join the prediction process for the final active user. In this way we have more information to contribute to the prediction process and it should be able to improve the prediction accuracy for collaborative filtering recommender systems. The main contribution of this work is that we relax the constraint that neighbor users must also have rated the given item. The proposed recursive prediction algorithm enables more flexibility in the prediction process of finding the useful neighbor users. One important issue is exactly how to select those effective nearest-neighbor users for the prediction process. In this work we will present the recursive prediction algorithm with various ways of determining the nearest-neighbor users and report their performances.

The rest of this chapter is organized as follows. The next section provides a brief review of the related work about collaborative filtering recommender systems. Section 3 recalls the general prediction process of the nearest-neighbor based collaborative filtering approach. In Section 4 we describe the recursive prediction algorithm in detail. Section 5 provides experimental results of evaluating the performance of the proposed approach. Finally we give discussions and summary in Section 6 and Section 7 respectively.

7.2 Related Work

The collaborative filtering approach has been developed in a long time ago. One of the earliest collaborative filtering recommender systems was implemented as an email filtering system called *Tapestry* (Goldberg et al., 1992). Later this technique was extended in several directions and was applied in various domains such as music recommendation (Shardanand & Maes, 1995) and video recommendation (Hill et al., 1995).

Generally speaking, collaborative filtering algorithms can be classified into 2 categories. One is *memory-based* CF algorithms, which predict the vote of a given item for the active user based on the votes from some other neighbor users. Memory-

based algorithms operate over the entire user voting database to make predictions on the fly. The most frequently used approach in this category is nearest-neighbor CF: the prediction is calculated based on the set of nearest-neighbor users for the active user (user-based CF approach) or, nearest-neighbor items of the given item (item-based CF approach). The second category of CF algorithms is *model-based*. It uses the user voting database to estimate or learn a probabilistic model (such as cluster models, or Bayesian network models, etc), and then uses the model for prediction. The detail of these methods and their respective performance have been reported in (Breese et al., 1998).

In this work we focus on the user-based CF approach (Resnick et al., 1994). The general prediction process is to select a set of nearest-neighbor users for the active user based on a certain criterion, and then aggregate their rating information to generate the prediction for the given item. More recently, an item-based CF approach has been proposed to improve the system scalability (Linden et al., 2001; Sarwar et al., 2001). The item-based CF approach explores the correlations or similarities between items. Since the relationships between items are relatively static, the item-based CF approach may be able to decrease the online computational cost without reducing the recommendation quality. The user-based and the item-based CF approaches are broadly similar, and it is not difficult to convert an implementation of the user-based CF approach into the item-based CF approach and vice versa.

The method of selecting nearest-neighbors is an important issue for the nearest-neighbor CF approach. In (Breese et al., 1998) it is found that highly correlated neighbors can be exceptionally more valuable to each other in the prediction process than low correlated neighbors. Herlocker et al. (Herlocker et al., 2002) have systematically studied various design issues for the nearest-neighbor CF approach. They have studied the correlation-thresholding technique in the neighbor selection process and found it does not give more accurate predictions than the plain non-thresholding method. Also, they have found that selecting a reasonable size of nearest-neighbors (usually range from 20 to 50) produced the lowest prediction error.

The above studies focused on selecting nearest-neighbor users from those who had also rated the given item. The prediction algorithm used in their work is direct and intuitive. However, this algorithm ignores the possibility that some users may be able to make a valuable contribution to the prediction process, despite not having explicitly rated the given item. By comparison, the recursive prediction algorithm proposed in this work relaxes such condition and could allow the nearest-neighbor users to be selected from a larger range of candidate users. Based on the recursive prediction algorithm, we propose new techniques for selecting nearest-neighbors to

improve the prediction accuracy.

7.3 Nearest-Neighbor based Collaborative Filtering

Herlocker et al. (Herlocker et al., 2002) have summarized the procedure of applying the nearest-neighbor based CF approach as the following three steps: 1) Measure all users with respect to the similarity with the active user; 2) Select a subset of users to use as a set of predictors for the given item; 3) Compute a prediction value for the given user based on the ratings from the selected neighbors. In this section we recall some of the most relevant prior work on each step and gives formal definition of the key techniques.

7.3.1 User Similarity

GroupLens (Resnick et al., 1994) introduced the Pearson correlations to measure similarity between users. Let I denotes the set of items which had been rated by both user x and y , the Pearson correlation similarity between user x and y is given by

$$sim(x, y) = \frac{\sum_{i \in I} (R_{x,i} - \bar{R}_x)(R_{y,i} - \bar{R}_y)}{\sqrt{\sum_{i \in I} (R_{x,i} - \bar{R}_x)^2} \sqrt{\sum_{i \in I} (R_{y,i} - \bar{R}_y)^2}} \quad (7.1)$$

Where $R_{x,i}$ represents user x 's rating of item i , and \bar{R}_x is the average rating value of user x .

In this work we choose the Pearson correlation as the metric for user similarity. There are several other ways of calculating the similarity among users such as the cosine based similarity. The comparison of the similarity metrics is out of the scope of this work. For more detailed information about this part of research please refer to (Breese et al., 1998).

7.3.2 Selecting Neighbors

Often CF recommenders can have a large number of users and it is infeasible to maintain real-time performance if the system adopts rating information from all users in the prediction process. One popular strategy is to choose the K nearest-neighbors of the active user as the subset predictors (Resnick et al., 1994). Another strategy is to set an absolute similarity threshold, where all neighbor users with absolute similarities greater than a given threshold are selected (Shardanand &

Maes, 1995). In this work we choose the K nearest-neighbor strategy as the baseline strategy of selecting neighbors.

7.3.3 Prediction Computation

Once the user similarity and the subset of neighbor users are determined, we need to aggregate their rating information to generate the prediction value. Resnick et al. (Resnick et al., 1994) have proposed a widely-used method for computing predicted ratings. Formally, for the active user x to the given item i , the predicted rating $\hat{R}_{x,i}$ can be calculated as following:

$$\hat{R}_{x,i} = \bar{R}_x + \frac{\sum_{y \in U_x} (R_{y,i} - \bar{R}_y) \text{sim}(x, y)}{\sum_{y \in U_x} |\text{sim}(x, y)|} \quad (7.2)$$

where U_x represents the subset of neighbor users selected in step 2 for the active user x . The similarity value $\text{sim}(x, y)$ can be calculated according to Equation 7.1 and it acts as a weight value on the normalized rating value $R_{y,i} - \bar{R}_y$. In the conventional CF approach, only those neighbor users who have rated the given item explicitly are selected, so the value $R_{y,i}$ can be fetched directly from the training dataset.

7.4 The Recursive Prediction Algorithm

In this section we illustrate the prediction problem further, using an example based on the popular MovieLens dataset. After that, we introduce the new neighbor selecting strategies and the recursive prediction algorithm.

7.4.1 An Illustrative Example

The MovieLens dataset was collected by the GroupLens research project at the University of Minnesota (project website: <http://movielens.umn.edu>). It consists of 100,000 ratings from 943 users on 1682 movies. This dataset has been cleaned up so that users who had less than 20 ratings or did not have complete demographic information were removed from this dataset. The sparsity level of the dataset is 0.9369 (Sarwar et al., 2001), which is quite high.

We partitioned the dataset into 2 parts: 80% of the ratings were used in the training set, and the remaining 20% was used as the testing set. Here we investigate one prediction task in the conventional CF approach: predicting the rating

Rank	Neighbor User (y)	Similarity ($sim(x, y)$)	Rating ($R_{y,i}$)
1	39	1.000	<i>n.a.</i>
2	571	1.000	<i>n.a.</i>
3	166	1.000	<i>n.a.</i>
4	384	1.000	<i>n.a.</i>
5	511	1.000	<i>n.a.</i>
6	531	1.000	<i>n.a.</i>
7	810	1.000	<i>n.a.</i>
8	812	1.000	<i>n.a.</i>
9	816	1.000	<i>n.a.</i>
10	861	0.968	<i>n.a.</i>
11	572	0.963	<i>n.a.</i>
12	520	0.919	<i>n.a.</i>
13	107	0.917	<i>n.a.</i>
14	599	0.905	<i>n.a.</i>
15	34	0.891	<i>n.a.</i>
16	691	0.890	<i>n.a.</i>
17	800	0.887	<i>n.a.</i>
18	803	0.883	<i>n.a.</i>
19	105	0.882	<i>n.a.</i>
20	923	0.872	4
21	900	0.868	<i>n.a.</i>
22	414	0.868	<i>n.a.</i>
23	702	0.866	<i>n.a.</i>
24	808	0.866	<i>n.a.</i>
25	485	0.866	<i>n.a.</i>

Table 7.1: The nearest-neighbor users for the active user $x = 1$ to predict the rating value of the item $i = 3$.

value of a given movie $i = 3$ for an active user $x = 1$. Table 7.1 lists the top 25 nearest-neighbors for the active user $x = 1$. It shows that most of those nearest-neighbors do not provide rating values for the given item $i = 3$. The only neighbor user who provides rating value for the item $i = 3$ in the list is the user with $id = 923$, which has a similarity value of 0.872 with the active user. Table 7.2 shows those nearest-neighbors that might be selected in the conventional CF approach. We can see that although the active user ($x = 1$) has many close neighbor users, most of them are filtered out from the prediction process because they haven't rated the given item ($i = 3$) yet. Among all the nearest-neighbor users listed in Table 7.1, only the user with $id = 923$ can be selected to join the conventional prediction

7.4. THE RECURSIVE PREDICTION ALGORITHM

Rank	Neighbor User (y)	Similarity ($sim(x, y)$)	Rating ($R_{y,i}$)
20	923	0.872	4
98	104	0.593	3
120	157	0.569	3
125	714	0.559	5
142	453	0.522	4
146	569	0.510	1
149	276	0.505	3
150	582	0.504	3
190	463	0.457	2
201	246	0.438	2
202	303	0.437	3
208	429	0.431	2
218	267	0.425	4
231	487	0.415	5
237	472	0.412	5
241	268	0.409	1
244	756	0.407	1
249	660	0.403	1
266	500	0.384	4
271	244	0.380	5

Table 7.2: The top 20 nearest-neighbors that will be selected in the conventional user-based CF approach for the active user $x = 1$ to predict the rating value of the item $i = 3$.

process and others are filtered out.

We believe that the MovieLens dataset is quite representative to most datasets used in collaborative filtering recommender systems. In another words, we believe sparsity is a common issue for collaborative filtering recommenders and the above illustrated problem exists commonly in the conventional prediction process.

7.4.2 The Strategies for Selecting Neighbors

The above example shows that the conventional CF approach excludes a large proportion of similar users from the set of nearest-neighbors just because they have not rated the given item. Here we propose a recursive prediction algorithm to relax this constraint. In addition, we also have observed that some neighbors might have only a few common ratings with the active user, but they could get very high similarity value with the active user by chance. This is because the calculation of the correlation value among users does not take into account the degree of overlaps between users. If two users have only few overlaps, it is unlikely that they are indeed

close-mined neighbors.

In summary, we propose the following five strategies for selecting the active user's nearest-neighbors:

1. Baseline strategy(*BS*): selects the top K nearest-neighbors who have rated the given item. This is the conventional strategy for selecting neighbors as illustrated in (Resnick et al., 1994);
2. Baseline strategy with overlap threshold(*BS+*): selects the top K nearest-neighbors who have rated the given item and have rated at least φ items that have also been rated by the active user(overlapped with the active user);
3. Similarity strategy(*SS*): selects the top K' nearest-neighbors purely according to their similarity with the active user;
4. Combination strategy(*CS*): combines top K nearest-neighbors selected by the baseline strategy(*BS*) and top K' nearest-neighbors selected by the similarity strategy(*SS*);
5. Combination strategy with overlap threshold (*CS+*): combines the top K nearest-neighbors selected by baseline Strategy(*BS*) and top K' nearest-neighbors selected by the similarity strategy (*SS*). Also, each user must have rated at least φ items overlapped with the active user.

Please keep in mind that for the *BS* and *BS+* strategy, since all those selected neighbors have provided rating values explicitly to the given item, the prediction value can be calculated straightforward without iteration. However, for the other three strategies(*SS*, *CS* and *CS+*), the recursive prediction algorithm must be applied to estimate the intermediate prediction values.

The *BS* strategy is an extreme case of selecting nearest-neighbor users. It only chooses users among those who have already explicitly rated the given item. The *SS* strategy is another extreme case, where only those top nearest-neighbors are considered to be useful for the prediction, no matter if they had rated the given item or not. The *CS* strategy is a compromise of the above two cases. *BS+* and *CS+* are the improved version of *BS* and *CS* respectively.

7.4.3 The Recursive Prediction Algorithm

The goal of the recursive prediction algorithm is to include nearest-neighbors who haven't rated the given item in the prediction process. When the process requires a

7.4. THE RECURSIVE PREDICTION ALGORITHM

```

Configuration Values:
   $\zeta$  — the threshold of recursive level;
   $\lambda$  — the combination weight threshold;
Function Parameters:
   $x$  — the active user;
   $i$  — the given item to be predicted;
   $level$  — the current recursive level;
return: the predicted rating value;

```

```

1. function RecursivePrediction ( $x, i, level$ )
2.   if  $level \geq \zeta$  then
3.     return BaselinePrediction( $x, i$ );
5.   endif
6.    $U \leftarrow$  SelectNeighbors( $x, i$ );
7.    $\alpha = 0.0$ ;
8.    $\beta = 0.0$ ;
9.   for each  $y$  in  $U$  do
10.    if  $R_{y,i}$  is given then
11.       $\alpha += (R_{y,i} - \bar{R}_y) sim(x, y)$ ;
12.       $\beta += |sim(x, y)|$ ;
13.    else
14.       $\hat{R}_{y,i} =$  RecursivePrediction( $y, i, level + 1$ );
15.       $\alpha += \lambda(\hat{R}_{y,i} - \bar{R}_y) sim(x, y)$ ;
16.       $\beta += \lambda|sim(x, y)|$ ;
17.    endif
18.  endfor
19.  return  $\bar{R}_x + \alpha/\beta$ ;

```

Figure 7.1: The recursive prediction algorithm.

rating value that doesn't exist in the dataset, we can estimate it recursively on the fly, and then use it in the prediction process. The estimated rating values may not be as accurate as those ratings explicitly given by the users. In our algorithm we specify a weight value to distinguish the different contribution of these two types of ratings.

Formally, our recursive prediction algorithm can be represented as the following:

$$\hat{R}_{x,i} = \bar{R}_x + \frac{\sum_{y \in U_x} w_{y,i} (R_{y,i} - \bar{R}_y) sim(x, y)}{\sum_{y \in U_x} w_{y,i} |sim(x, y)|} \quad (7.3)$$

where U_x is the set of neighbor users for the active user x determined by the respective strategy we have mentioned earlier. If $R_{y,i}$ is not given explicitly from

the training dataset, we will apply the recursively predicted value $\hat{R}_{y,i}$ instead. Also, we apply a weight value $w_{y,i}$ to the recursively predicted value. The $w_{y,i}$ is determined as the following:

$$w_{y,i} = \begin{cases} 1 & R_{y,i} \text{ is given} \\ \lambda & R_{y,i} \text{ is not given} \end{cases} \quad (7.4)$$

Here the weight threshold λ is a value between $[0, 1]$.

A detailed implementation of the recursive prediction algorithm is shown in Figure 7.1. The function “SelectNeighbors (x, i)” is the implementation of one of the 5 different selection strategies. Note that if the recursive algorithm has reached the pre-defined maximal recursive level, it will stop the recursive procedure and call the “BaselinePrediction” function instead. The *BaselinePrediction* function can be implemented by various strategies. But to make it simple, here we use the conventional baseline strategy (BS). Each of the five neighbor selection strategies can be applied with the recursive prediction algorithm to form a CF approach. In the rest of the chapter, we also use the neighbor selection strategy to represent the corresponding CF approach. For instance, the token *CS* will also represent the CF approach implemented with the recursive prediction algorithm together with the combination strategy for selecting neighbors.

The recursive prediction algorithm is an extension of the conventional nearest-neighbor based prediction algorithm in the CF approach. It is worth pointing out that the recursive prediction algorithm is equivalent to the conventional prediction algorithm if it is applied with the BS strategy. Therefore *BS* represents the conventional CF approach, which is the baseline CF approach in this work.

There are a number of important parameters to be decided before a real feasible CF system can be implemented based on the proposed recursive algorithm. One important parameter of the recursive algorithm is the recursive level. As the recursive level increases, so too does the computational cost. For the final level of the algorithm, we use the BS strategy to select nearest-neighbors so that the recursive algorithm could be terminated within a limited computation time. In addition, the neighbor size is an important factor for these strategies and can have a large impact on overall system performance. Furthermore, we must also set the combination weight threshold for the CS and CS+ strategies. Finally, we need to determine the overlap size for the nearest-neighbor users for both the BS+ and CS+ strategy.

7.5 Evaluation

7.5.1 Setup

Our experiments adopt the MoiveLens dataset that we have mentioned earlier (see Section 7.4.1). The full dataset is divided into a training set and a test set. In our experiments 80% of the data was used as training data and the other 20% for testing data.

Our experiments are executed on a ThinkPad Notebook (model T41P), which has 1GB memory and one CPU of 1.7GHZ under the Linux operating system. We implement both the baseline algorithm and our recursive prediction algorithm in Java based on the *Taste* collaborative filtering open source project.¹

7.5.2 Evaluation Metrics

Mean Absolute Error (MAE) is a widely used metric for measuring the prediction accuracy between ratings and predictions for collaborative filtering systems. For each rating–prediction pair $\langle R_t, \hat{R}_t \rangle$, the absolute error between them can be calculated as $|R_t - \hat{R}_t|$. The MAE is the average value of these absolute errors $|R_t - \hat{R}_t|$ for all items in the test dataset. Formally it is calculated as the following:

$$MAE = \frac{\sum_{t=1}^N |R_t - \hat{R}_t|}{N} \quad (7.5)$$

Where N is the size of the test dataset. The lower the MAE, the more accurate the recommendation system predicts the ratings. Compared to other metrics, MAE is easier to measure, and empirical experiments have shown that mean absolute error has high correlations with many other proposed metrics for collaborative filtering. Mean absolute error is the most frequently used metric among collaborative filtering researchers.

Here we adopt MAE as the metric to measure the prediction accuracy. Also, we use the task time to measure the computation cost of the each algorithm.

¹*Taste* is an open source collaborative filtering package in Java. The source code can be downloaded from <http://taste.sourceforge.net/>

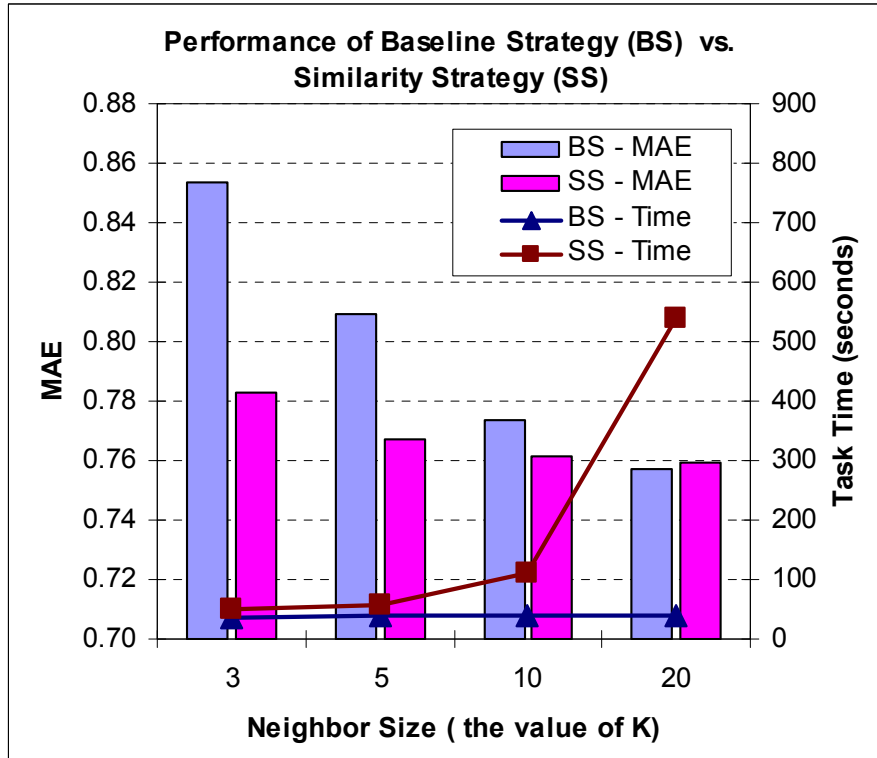


Figure 7.2: Performance results with various neighbor sizes.

7.5.3 Experimental Results

There are a number of parameters which can affect the performance of the recursive prediction algorithm. Here we outline these parameters:

1. the neighbor size (the values of K and K')
2. the recursive level (the value of ζ)
3. the combination weight threshold (the value of λ)
4. the overlap size threshold (the value of φ)

In this work, we first carry out experiments to determine the value of each parameter, and then we compare the overall performance of the CF approach that adopts the recursive prediction algorithm with the baseline CF approach.

Neighbor Size for the Similarity Strategy(SS)

Compared to the baseline strategy(BS) for selecting neighbors, the similarity strategy(SS) is able to adopt those highly correlated neighbor users who haven't rated the given item. Here we compare the prediction accuracy and the task time of completing all predictions in the testing dataset between the two strategies(BS vs SS). The results are shown in Figure 7.2. For this experiment the recursive level is set to 2.

From the results we can see that the accuracy for both strategies improves as the neighborhood size increases, from 3 to 20. One interesting result is that although the SS strategy includes users who may only have estimated rating values, it can obtain higher prediction accuracy than the BS strategy when the neighbor size is relatively small. For example, if both strategies take only 5 neighbors, the SS strategy reduces the MAE value by 5.2%.

We can also see that the SS strategy requires more computation resources, especially when the neighbor size is large. This is because the SS method needs to estimate the intermediate value recursively. This is a drawback of the SS strategy and it tell us that if we want to improve the prediction accuracy, it is infeasible to use the SS strategy alone. A better approach would be to combine the SS strategy with other neighbor selection techniques.

We noticed that when the neighbor size is 10, the SS strategy can produce low-error predictions, while not being very computationally expensive. In the following experiments we set the neighbor size for the SS strategy as 10.

Recursive Level

Recursive level is an important parameter for the recursive prediction algorithm. In this experiment we choose the SS strategy with a neighborhood size of 10. We explore the prediction performance and the algorithm complexity with various recursive levels. Please keep in mind that when the recursive level is zero, the Similarity strategy (SS) is equivalent to the baseline strategy (BS).

From the results shown in Figure 7.3 we can see that the prediction performance has an improvement of 1.4% from the non-recursive prediction process to the one with 1 level of recursion. We can also see that when the recursive level is larger than 2, the prediction accuracy doesn't improve significantly, but the task time increases does. To balance the prediction accuracy and the computational cost, we set the recursive level as 2 in our later experiments .

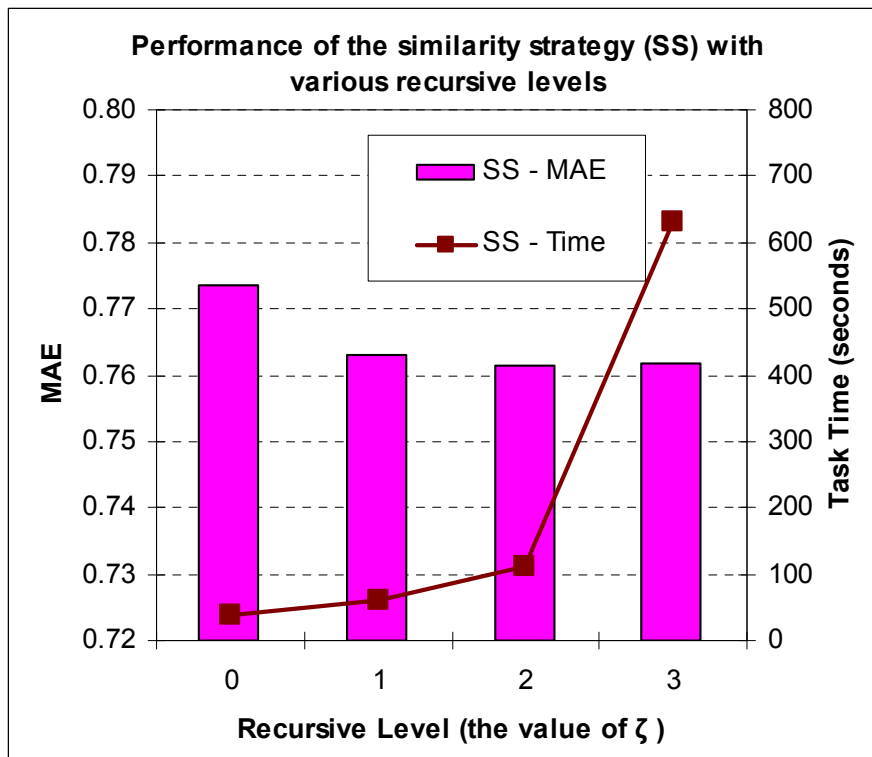


Figure 7.3: Performance results with various recursive levels.

Combination Weight Threshold

As mentioned earlier, the combination strategy (CS) is supposed to benefit from both the conventional baseline strategy (BS) and the similarity strategy (SS). One parameter that we need to determine is the combination weight threshold λ . In this experiment, we set the neighbor size for both the BS strategy and the SS strategy as 10 (i.e., $K = 10$ and $K' = 10$), the recursive level is set as 2 (i.e. $\zeta = 2$). Please also note that when $\lambda = 0$, the CS strategy is equivalent to the baseline BS strategy.

From Figure 7.4 we can see that the SS strategy can perform better than the BS strategy. Also, we can get even better performance if we combine them together. For instance, in the experiment when $\lambda = 0.5$, MAE can be reduced by around 1% (compared with the SS strategy) and 3% (compared with the BS strategy) respectively.

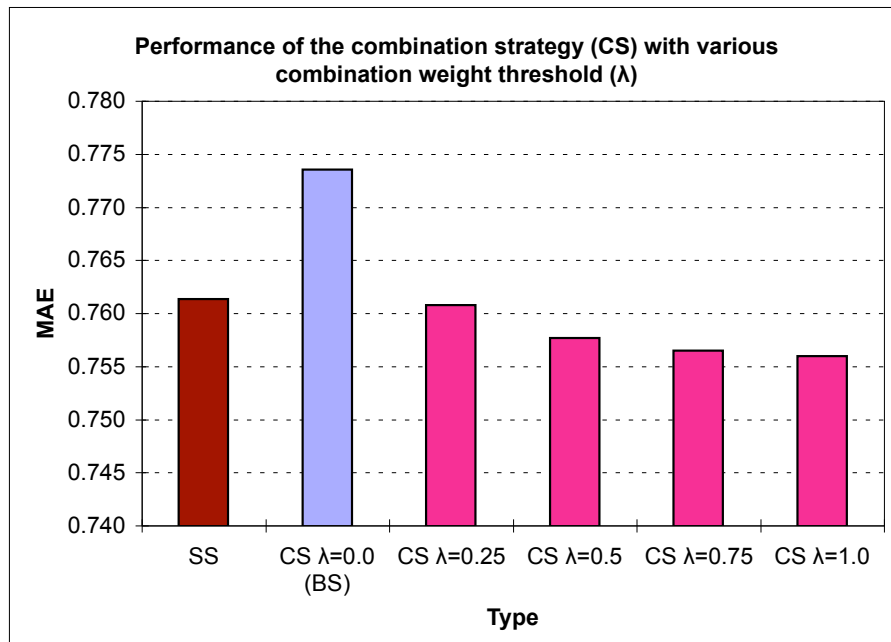


Figure 7.4: Performance results of the combination strategy(CS) with various combination weight thresholds.

Overlap Size Threshold

In this experiment we investigate the relationship between prediction performance and the overlap size threshold for the two strategies: BS+ and CS+. As we can see in Figure 7.5, increasing the overlap size threshold from 2 to 10 produces better prediction performance for both strategies. However, the prediction performance decreases gradually when the overlap size threshold increases from 10. From this result we can see that a good choice of the overlap size threshold would be around 10. We can also see that in all cases, the CS+ strategy is more accurate than the BS+ strategy given the same overlap size threshold. On average, the CS+ strategy can reduce the MAE by around 2.1% compared to the BS+ strategy when the overlap threshold is set as 10. In the following experiments, we set the overlap size threshold as 10.

Performance Comparison

It is important to know if the recursive prediction algorithm (adopting the SS, CS or CS+ strategy) can lead to substantial performance improvement compared with

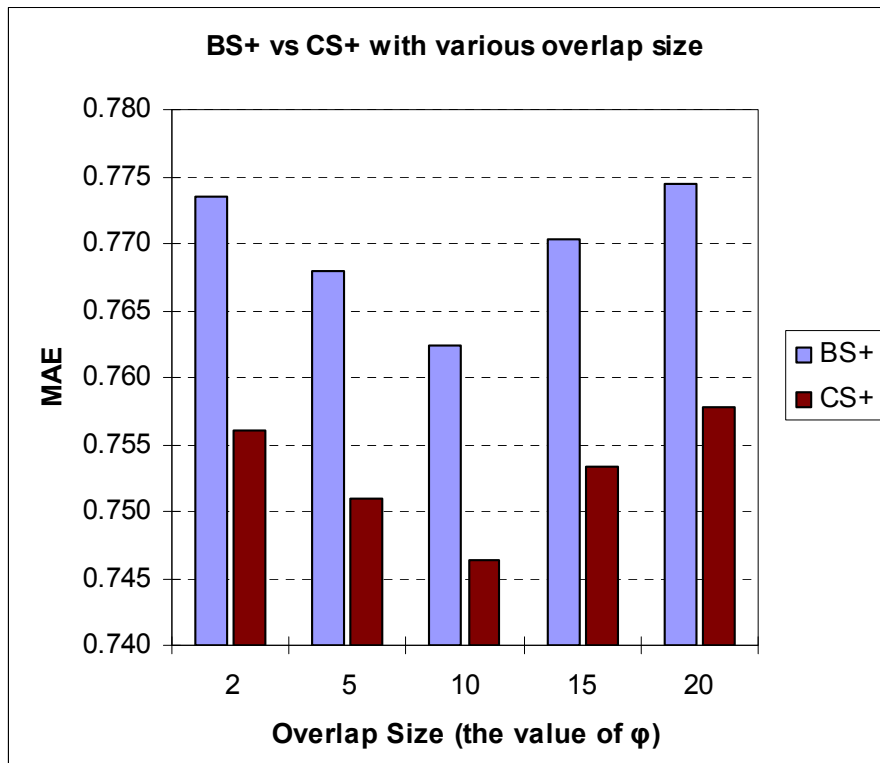


Figure 7.5: Performance results of the CS+ strategy with various overlap thresholds.

the traditional direct prediction algorithm (adopting the BS or BS+ strategy) for CF recommender systems. Here we carry out a set of experiments to compare the performance of these strategies for various neighborhood size K . In this experiment, we choose the following parameters for the recursive prediction algorithm: the recursive level ζ is set as 2, and the combination weight threshold λ is set as 0.5. Additionally, the neighborhood size K' is set as 10. The overlap threshold φ for both the BS+ and CS+ strategy is set as 10. The experimental result is shown in Figure 7.6.

From Figure 7.6 we can learn several interesting results. Firstly, prediction accuracy increases for all strategies, when the neighborhood size increases from 3 to 50. After that, it levels off. The performance of the baseline strategy (BS) is in line with the results in the earlier literature (Herlocker et al., 2002).

Additionally, the BS+ strategy performs better than the BS strategy, especially when the neighborhood size is relatively small. For example, when the neighbor size K is 10, BS+ strategy can reduce the MAE from 0.774 to 0.762 (by 1.4%). Since the

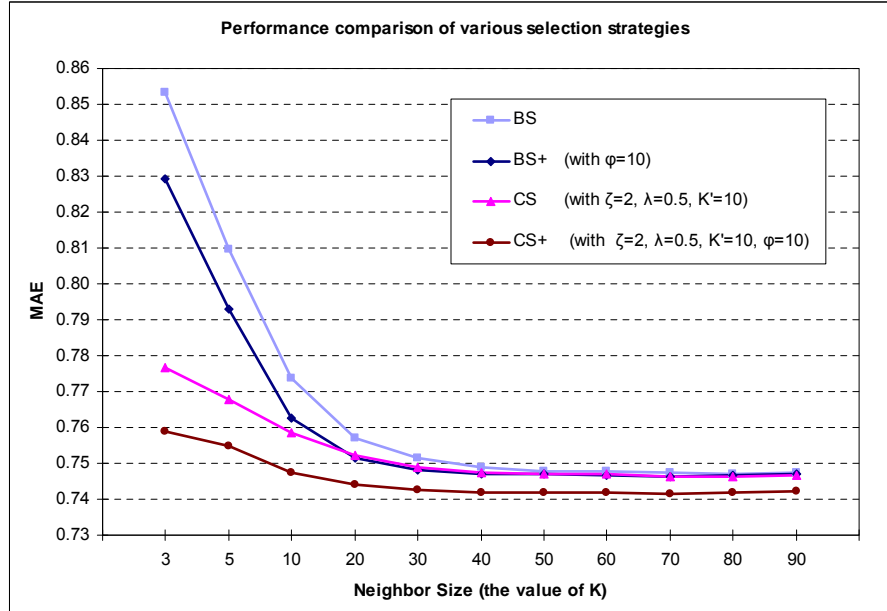


Figure 7.6: Overall performance comparison of the recursive prediction algorithm with various strategies

only difference between the BS and the BS+ strategy is that BS+ only selects those users with at least 10 overlap items to the active user, we can see that only keeping neighbor users with a high overlap size has a positive influence on the prediction power. However, such improvement is limited with the neighbor size: when the neighbor size is bigger than 50, the BS+ strategy doesn't make any substantial improvement compared with the BS strategy.

Moreover, the CS strategy performs better than the BS strategy, especially when the neighbor size is relatively small. For example, when $K=10$, the CS strategy can reduce the MAE by 1.9% compared with the BS strategy. This shows that the proposed recursive prediction algorithm can improve the prediction accuracy. Again, this improvement is not significant when the neighbor size is bigger than 50.

Finally, the results show that the CS+ strategy has the best performance among all strategies in all cases. For example, when the neighbor size is $K=10$, the CS strategy can reduce the MAE by 3.4% compared with the BS strategy. It is also important to notice that when the neighbor size is large, the CS+ strategy still improves on the BS strategy. For example, when the neighbor size is $K=60$, the BS strategy reaches its best performance (MAE=0.748). By comparison, the CS+ strategy can further reduce the MAE to 0.742 (0.8% lower than BS strategy).

7.6 Discussion

To predict the rating value of a given item for an active user, the conventional prediction algorithm in collaborative filtering recommender systems selects neighbor users only from those who have already rated the given item. Because of the dataset sparseness, a large proportion of *nearest-neighbor* users are filtered out without contributing to the prediction process. By comparison, our recursive prediction algorithm is able to keep those *nearest-neighbor* users in the prediction process even though they haven't given ratings to the given item. To the best of our knowledge, this is the first work of improving the prediction accuracy towards the direction of selecting more promising nearest-neighbor users.

The key contribution of this work is that the recursive prediction algorithm enables a larger range of neighbor users to be included in the prediction process of CF recommender systems. Our experimental results show that the proposed prediction algorithm can produce higher prediction accuracy than the conventional direct nearest-neighbor prediction algorithm. When selecting nearest-neighbors with at least a certain number of overlapped rating items with the active user (the CS+ strategy), the recursive prediction algorithm can reduce the prediction error (measured by MAE) by 0.8% compared to the best performance that can be achievable by the conventional user-based prediction algorithm for collaborative filtering recommender systems.

It is worthy to point out that the parameters for the recursive prediction algorithm in our experiments were determined empirically according to the feedback from performance test, and so values are dependent to the current underlying dataset. If the dataset changes, we may need to tune these parameters again to make the algorithm reach its best overall performance on the new dataset.

This work can be extended in several directions. Here we only described our algorithm for the user-based CF recommender systems. This algorithm can be easily applied to the item-based CF recommender systems (Linden et al., 2001; Sarwar et al., 2001). This change is intuitive: what we need to do is to change the nearest-neighbor users into nearest-neighbor items. The item-based version of the recursive prediction algorithm is left for future work. Furthermore, trust has been identified as a very useful information in recommender systems (O'Donovan & Smyth, 2005). We could also extract trust information from the training dataset and integrate it into our recursive algorithm to improve the prediction accuracy and robustness. For instance, we can apply the trust information as one of the criteria in selecting the nearest-neighbor users in our recursive prediction algorithm to further improve the

overall prediction performance. Moreover, recently O’Sullivan et al. (Sullivan, Wilson, & Smyth, 2002) has proposed a different approach to ameliorate the data sparseness problem. They use data mining technique to increase the similarity coverage among users and could make recommender systems perform better. This technique can also be integrated into the recursive prediction algorithm so to further improve the overall recommendation performance.

Singular Value Decomposition (SVD) is a matrix decomposition approach in linear algebra and some researchers (Berry, Dumais, & O’Brien, 1995; Sarwar, Karypis, Konstan, & Riedl, 2000; Goldberg, Roeder, Gupta, & Perkins, 2001) suggest it can be applied to improve the prediction accuracy of the classic CF approach. Given the user-item rating matrix R of size $m \times n$, SVD can decompose it into three ones $U(m \times m)$, $S(m \times n)$, and $V(n \times n)$ so that $R = U \times S \times V^T$, where matrix U and V are orthogonal and matrix S is diagonal containing the singular values. By only keeping the largest k singular values and setting others to 0 in matrix S , we can produce the matrix S_k . Similarly we can produce the matrix U_k and V_k . Then a rank- k approximation R_k of R can be obtained as $R_k = U_k \times S_k \times V_k^T$. Finally the prediction can be generated from it by computing the similarities between m pseudo-users $U_k \cdot \sqrt{S_k^T}$ and n pseudo-items $\sqrt{S_k} \cdot V_k^T$. Unfortunately one serious limitation of the SVD-based recommendation approach is that it is computationally very expensive ($O(\max(m^3, n^3))$). It is not suitable for large-scale deployment in E-commerce. By comparison, the recursive-CF approach that we proposed here is very flexible in balancing the computational complexity and the recommendation accuracy by tuning some parameters.

7.7 Summary

In this chapter we proposed a recursive prediction algorithm for CF recommender systems which can predict the missing rating values of the neighbor users, and then apply these missing values to the prediction process for the active user. Our studies show that the recursive prediction algorithm is a promising approach for achieving higher prediction accuracy than the conventional direct prediction algorithm in the nearest-neighbor based CF approach. Specifically, the recursive prediction algorithm together with the CS+ strategy achieved the best prediction performance. When the neighbor size is relatively small ($K=10$), it can reduce prediction error by 3.4%. When the neighbor size is large ($K=60$), it can reduce the prediction error by 0.8% than the best performance that the conventional nearest-neighbor based CF approach could achieve.

While the rapid growth of web technologies allows a massive amount of products to be sold through online e-commerce systems, it also brings the information overload problem to end-users. An e-commerce website can easily provide millions of products to end-users to choose from. However, it may not be feasible for users to navigate all these products manually to find the desired ones. People are inclined to rely on some well-designed online product search tools to help them find their desired products both accurately and efficiently.

This dissertation pursues the *user-centric* approach in designing online product search tools. A user-centric online product search tool should have the ability to recommend suitable products to meet the user's various preferences. In addition, it should be able to help the user navigate the product space and reach the final target product without expending too much effort. Furthermore, according to behavior decision theory, users are likely to construct their preferences during the decision process, so the tool should be designed in an interactive way to elicit users' preferences gradually.

In this thesis, we investigate both the design and evaluation issues of online product search from the user's perspective. For the design issue, we have proposed both algorithms and user interfaces to improve the performance of product search tools. For the evaluation issue, we have carried out both simulation evaluations and

real-user studies. In the rest of this chapter, we summarize the contributions and results from our research, and then outline the future research directions.

8.1 Contributions

8.1.1 Methodology for Performance Evaluation

We proposed a simulation environment, where a generic product search tool can be evaluated quickly. In this environment, we can simulate the underlying decision support approach of the system to generate performance evaluation results. Compared to the real-user study evaluation method, this simulation method is faster, cheaper, and scalable to different product domains. Based on this simulation environment, we carried out simulation experiments to analyze the performances for a set of various decision strategies.

In this simulation environment, the performance of a decision approach can be evaluated quantitatively in terms of decision accuracy, elicitation effort and cognitive effort under the extended effort–accuracy framework. The extended effort–accuracy framework provides a way for us to analyze the performance of a given product search tool from the user’s perspective. The simulation environment is a methodology to forecast the acceptance of online product search tools in the real world and curtail the evaluation of each tool’s performance from months of real-user studies to rapid simulation process. It allows system designers to evaluate more tools in various situations, and more importantly, discover design opportunities for new product search tools.

8.1.2 Algorithm for Generating Compound Critiques

Generating high quality compound critiques is essential in designing critique-based product search tools. One approach for dynamically generating compound critiques, called *dynamic critiquing* (Reilly et al., 2004a), discovers critique patterns that are common to the remaining products in each interaction cycle based on the *A priori* algorithm. Essentially this is a data-driven approach. However, this approach may not generate compound critiques that satisfy users’ real preferences, and users may be frustrated and give up the interaction process in some situations.

We proposed a user-centric algorithm to generate compound critiques through a preference model based on multi-attribute utility theory (MAUT) (Keeney & Raiffa, 1976) for online product search tools. In each interaction cycle our approach first

determines a list of products via the user's preference model, and then generates compound critiques by comparing them with the current reference product. In this approach, the user's preference model is maintained adaptively based on user's critique actions during the interaction process, and the compound critiques are determined according to the utilities they gained.

We developed an online evaluation platform called *CritiqueShop* so that real users can evaluate various online product search tools in a natural way. Particularly, two systems were implemented: the Apriori-base system and the MAUT-based system.

We further carried out large-scale comparative real-user studies to verify the performance of these two systems, using a mixture of objective criteria (such as the interaction efficiency, recommendation quality/accuracy) and subjective criteria (such as a user's perceived satisfaction). Our findings showed that the MAUT-based system is quite efficient and accurate. The real-user study results showed that the accuracy of the search results can reach up to 82%. Users' subjective feedback also showed that this system is easy to use, informative, and effective for search products. Most users are willing to use this system to purchase products in the future.

8.1.3 Visual Representation of Compound Critiques

To enhance the efficiency of critique-based product search tools, we not only need to generate high quality compound critiques, but also need to attract users' attention so that more compound critiques can be applied during the interaction process. Traditionally the interface is *textual*, which shows compound critiques as sentences in plain text. In this thesis we proposed a new *visual* interface which represents various compound critiques by a set of meaningful icons. We implemented this visual interface for the MAUT-based system in the CritiqueShop platform and made a comparative real-user study. The visual interface and the textual interface were compared under a mixture of objective criteria and subjective criteria.

Our real-user study showed that the visual interface is more effective than the textual interface. It can reduce user's interaction effort and attract users to apply the compound critiques more frequently. For instance, the length of interaction cycles can be reduced 53% and the compound critiquing frequency can be doubled for the laptop dataset in our real-user study. Also, we found that the system with visual interface is more valued by users in terms of efficiency, ease of use, and degree of satisfaction with respect to shopping experience.

To the best of our knowledge, this is the first work on visualizing compound critiques for critique-based product search tools, and we are the first to prove that visual interface is a promising method of displaying compound critiques for critique-based product search tools.

8.1.4 Improvement on Collaborative Filtering Approach

The collaborative filtering approach has been widely used to help people find products in low-risk domains such as movies and books (Resnick et al., 1994; Herlocker et al., 1999). To predict the rating value of a given item for an active user, the conventional prediction algorithm of the collaborative filtering approach is to select nearest-neighbor users from those who have already rated the given item. If the users' ratings are very sparse, the given item may only receive a small number of ratings from users. As a result, a large proportion of *nearest-neighbor* users of the active user are filtered out from the prediction process because they haven't rated the given item.

In this thesis we proposed a recursive prediction algorithm for CF recommender systems. This algorithm can predict the missing rating values of the nearest-neighbor users, and then apply these missing values to the prediction process for the active user. With this approach, we are able to alleviate the problem of data sparsity.

Our studies showed that the recursive prediction algorithm is a promising approach for achieving higher prediction accuracy than the conventional direct prediction algorithm in the nearest-neighbor based CF approach. Specifically, the recursive prediction algorithm together with the CS+ strategy achieved the best prediction performance. When the neighbor size is relatively small ($K=10$), it can reduce prediction error by 3.4%. When the neighbor size is large ($K=60$), it can still reduce the prediction error by 0.8% than the best performance that the conventional nearest-neighbor based CF approach could achieve.

8.2 Limitations

In this section we quickly summarize the limitations of our work. A detailed discussion of these limitations has already been provided in previous chapters.

In our work on the simulation environment for performance evaluation, the interaction effort and cognitive effort are measured by approximation, and the results might therefore contain some inaccuracies. So far, it is unknown how accurate the

simulation results would be by comparing with the results obtained from real-user studies. In addition, some criteria – such as a user’s degree of satisfaction, confidence or trust – are also very important for measuring the performance of a given product search tool. But the simulation environment is unable to generate direct measurement on these criteria.

When we use MAUT to model users’ preferences, we adopt the utility function with the weighted additive form, assuming that attributes are mutually preferentially independent (MPI) to each other. However, MPI is a very strong condition, and in reality this assumption may be violated. As a result, the user’s preference model based on this form of utility function might be not very accurate. To achieve better performance, we need to find some more sophisticated forms of utility functions.

The recursive collaborative filtering approach that we proposed in Chapter 7 has a limitation in terms of computational complexity. Currently we haven’t optimized the implementation of this algorithm, and we only recursively predict the missing rating values in two levels. We noticed that in the recursive prediction process there are many computational redundancies. For example, those intermediate prediction results can be calculated offline, or can be calculated only once and saved for later use. Thus the online computation complexity could be significantly reduced.

8.3 Future Research Directions

In this section, we outline several promising research directions which might be helpful for improving the performance of the online product search tools that we have developed. We also briefly analyze the challenges that each research direction may have.

8.3.1 Generating Diverse Compound Critiques

In this thesis we proposed the approach of dynamically generating compound critiques through a utility-based preference model. Some researchers have pointed out that increasing a certain amount of diversity in the product search results has the potential to make the interaction more efficient (Smyth & McClave, 2001; McGinty & Smyth, 2003; McCarthy et al., 2005). For example, in (McCarthy et al., 2005) diversity is enhanced by reducing the overlap among those compound critiques generated by the dynamic critiquing approach. In our current approach of generating compound critiques, the weight of each product attribute is revised adaptively

CHAPTER 8. CONCLUSIONS

during the critiquing process; thus, we believe that there is a certain degree of diversity among the compound critiques generated by our approach. However, this is not enough and we need to further study the relationship between the diversity and the performance of product search tools in detail. One challenging research problem that we are facing here is how to integrate the diversity effectively into the existing utility-based decision approaches to determine the final search results.

8.3.2 Collaborative Critiquing

The critique-based product search tools that we have discussed so far are based on the preferences elicited from each individual. There is no mechanism for users to share their preferences between each other. When two or more similar-minded users are looking for products through the same product search tool, each of them has to follow the preference elicitation process.

The collaborative filtering approach provides a mechanism for users to share their tastes on products based on their ratings. Motivated by this idea, we could identify some similar-minded users based on their critiquing history, and then predict which critiques is most likely to be selected by a given user according to the critiquing history from his or her similar-minded users. In this way, we believe the system is able to generate compound critiques for users more effectively. One challenge of this research is how to gather information from users' critique history to build the preference model effectively.

8.3.3 Adaptive Interfaces for Preference Elicitation

Currently most product search tools are designed in a way that users' preferences are elicited in one of the four styles: (1) Value elicitation, (2) Item-based, (3) Ratings-based, and (4) Critiquing. In reality, different users may prefer different styles of preference elicitation. Even for the same user, he or she may like to input preferences in different styles in different situations. For example, if a user is using a desktop computer to search products, maybe he or she is willing to input some values as the preferences. But for a mobile user, most likely he or she will try to avoid typing values because it is not easy to do so on small devices. Instead he or she may prefer critiquing or rating-based preferences. As a result, here we believe that an adaptive interface that allows users to reveal preferences in different styles would be very helpful for online product search. One challenge to this research direction is the mechanism to change the interface dynamically according to users' needs. An-

other challenge is how to utilize various styles of preferences together to generate search results accurately.

8.3.4 Handling Implicit Preferences

The critiquing-based product search tool that we have investigated in this thesis is based on preferences explicitly elicited from users. That is, users are obliged to specify their preferences through the given user interfaces provided by the system. This inevitably requires some effort from users each time. In some situations, the system may capture some implicit preferences from the user. For example, if a music search tool has detected that a user skipped a specific song several times, it can guess that the user does not like this song and will not show it to him or her in the future. Understanding the implicit preferences is necessary to help users find products more efficiently.

The main challenges of this research are the following: 1) How to gather users' implicit preferences; 2) How to apply the implicit preferences to improve the quality of the search results. For the first challenge, we need to identify those implicit preferences that are stable for making decisions and representative to users' true preferences. For the second challenge, the Bayesian filtering technique can be used as a way to improve the quality of the search results. Bayesian filtering is a well-known technique in classifying data into different classes and has been successfully used in some email filtering and anti-spam software (Sahami, Dumais, Heckerman, & Horvitz, 1998). In our work (Zhang & Pu, 2007) we have proposed an approach of refining the search results through Bayesian filtering and the preliminary experiment shows that it is promising in generating more accurate search results and saving users' interaction effort.

8.4 Summary

In this thesis, we are aiming at designing and evaluating user-centric online product search tools to help end-users find desired products effectively in e-commerce environment. Our main work is the implementation of a critique-based product search tool with two novel improvements: a user-centric algorithm to automatically generate compound critiques, and a visual interface to represent compound critiques effectively. Evaluation results from both simulation and real-user studies confirm the positive benefits of these improvements. Overall, we believe our work is helpful for designing the next generation of product search tools in the e-commerce environment.

- Agrawal, R., Imielinski, T., & Swami, A. (1993). Mining association rules between sets of items in large databases. In Buneman, P., & Jajodia, S. (Eds.), *Proceedings of the ACM SIGMOD International Conference on Management of Data*, pp. 207–216. ACM Press. Washington, D.C., May 26-28, 1993.
- Agrawal, R., Mannila, H., Srikant, R., Toivonen, H., & Verkamo, A. I. (1996). Fast discovery of association rules. In *Advances in Knowledge Discovery and Data Mining*, pp. 307–328. AAAI/MIT Press.
- Agrawal, R., & Srikant, R. (1994). Fast algorithms for mining association rules. In Bocca, J. B., Jarke, M., & Zaniolo, C. (Eds.), *Proceedings of the 20th International Conference Very Large Data Bases(VLDB)*, pp. 487–499. Morgan Kaufmann.
- Ahlberg, C., & Shneiderman, B. (1994). Visual information seeking: tight coupling of dynamic query filters with starfield displays. In *CHI '94: Proceedings of the SIGCHI conference on Human factors in computing systems*, pp. 313–317, New York, NY, USA. ACM.
- Alba, J., & Marmorstein, H. (1987). The effects of frequency knowledge on consumer decision making. *Journal of Consumer Research*, 14, 14–25.
- Allen, J. F., Schubert, L. K., Ferguson, G. M., Heeman, P. A., Hwang, C. H., Kato, T., Light, M. N., Martin, N. G., Miller, B. W., Poesio, M., , & Traum, D. R. (1994). The TRAINS project: A case study in building a conversational planning agent. In *TRAINS Technical Note 94-3*, University of Rochester, Department of Computer Science.

Bibliography

- Anderson, C. (2006). *The Long Tail: How Endless Choice Is Creating Unlimited Demand*. Random House Business Books.
- Baeza-Yates, R., & Ribeiro-Neto, B. (1999). *Modern Information Retrieval*. Addison Wesley.
- Bernoulli, D. (1954). Exposition of a new theory on the measurement of risk (original 1738). *Econometrica*, 22, 23–36.
- Berry, M. W., Dumais, S. T., & O'Brien, G. W. (1995). Using linear algebra for intelligent information retrieval. *SIAM Review*, 37(4), 573–595.
- Billsus, D., & Pazzani, M. J. (2000). User modeling for adaptive news access. *User Modeling and User-Adapted Interaction*, 10(2-3), 147–180.
- Bistarelli, S. (2004). Semirings for soft constraint solving and programming. In *Springer Lecture Notes in Computer Science, Vol. 2962*, p. 279. Springer.
- Bistarelli, S., Montanari, U., & Rossi, F. (1997). Semiring-based constraint satisfaction and optimization. *Journal of ACM*, 44(2), 201–236.
- Boutilier, C., Brafman, R., Domshlak, C., Hoos, H., & Poole, D. (2004). CP-nets: A tool for representing and reasoning with conditional ceteris paribus preference statements. *Journal of Artificial Intelligence Research*, 21, 135–191.
- Boutilier, C., Bacchus, F., & Brafman, R. I. (2001). UCP-networks: A directed graphical representation of conditional utilities. In *UAI '01: Proceedings of the 17th Conference in Uncertainty in Artificial Intelligence*, pp. 56–64.
- Boutilier, C., Brafman, R., Geib, C., & Poole, D. (1997). A constraint-based approach to preference elicitation and decision making. In Doyle, J., & Thomason, R. H. (Eds.), *Working Papers of the AAAI Spring Symposium on Qualitative Preferences in Deliberation and Practical Reasoning*, pp. 19–28, Menlo Park, California. American Association for Artificial Intelligence.
- Boutilier, C., Brafman, R. I., Hoos, H. H., & Poole, D. (1999). Reasoning With Conditional Ceteris Paribus Preference Statements. In *Proc. Fifteenth Conference on Uncertainty in Artificial Intelligence (UAI-99)*.
- Breese, J. S., Heckerman, D., & Kadie, C. (1998). Empirical analysis of predictive algorithms for collaborative filtering. In *Proceedings of the Fourteenth Conference on Uncertainty in Artificial Intelligence (UAI-98)*, pp. 43–52, San Francisco. Morgan Kaufmann.
- Burke, R. (2002). Interactive critiquing for catalog navigation in e-commerce. *Artificial Intelligence Review*, 18(3-4), 245–267.

- Burke, R., Hammond, K., & Young, B. (1996). Knowledge-based navigation of complex information spaces. In *Proceedings of the Thirteenth National Conference on Artificial Intelligence*, pp. 462–468. AAAI Press. Portland, OR.
- Burke, R. D., Hammond, K. J., & Young, B. C. (1997). The FindMe approach to assisted browsing. *IEEE Expert*, 12(4), 32–40.
- Carenini, G., & Poole, D. (2002). Constructed preferences and value-focused thinking: Implications for AI research on preference elicitation. In *Proceedings of the AAAI workshop on Preferences in AI and CP: Symbolic Approaches*, Edmonton, Canada. AAAI.
- Chen, L., & Pu, P. (2007). Hybrid critiquing-based recommender systems. In *IUI '07: Proceedings of the 12th international conference on Intelligent user interfaces*, pp. 22–31, New York, NY, USA. ACM.
- Cutting, D. R., Karger, D. R., Pedersen, J. O., & Tukey, J. W. (1992). Scatter/gather: a cluster-based approach to browsing large document collections. In *SIGIR '92: Proceedings of the 15th annual international ACM SIGIR conference on Research and development in information retrieval*, pp. 318–329, New York, NY, USA. ACM.
- Faltings, B., Pu, P., Torrens, M., & Viappiani, P. (2004a). Designing example-critiquing interaction. In Vanderdonckt, J., Nunes, N., & Rich, C. (Eds.), *Proceedings of the International Conference on Intelligent User Interfaces, (IUI-2004)*, pp. 22–29. ACM. Funchal, Madeira, Portugal.
- Faltings, B., Torrens, M., & Pu, P. (2004b). Solution generation with qualitative models of preferences. *Computational Intelligence*, 20(2), 246–263.
- Fargier, H., Hang, J., & Schiex, T. (1993). Selecting preferred solutions in fuzzy constraint satisfaction problems. In *Proceedings of the First European Congress on Fuzzy and Intelligent Technologies*, pp. 1128–1134.
- Fargier, H., & Lang, J. (1993). Uncertainty in constraint satisfaction problems: a probabilistic approach. In Clarke, M., Kruse, R., & Moral, S. (Eds.), *ECSQARU*, Vol. 747 of *Lecture Notes in Computer Science*, pp. 97–104. Springer.
- Goldberg, D., Nichols, D., Oki, B. M., & Terry, D. (1992). Using collaborative filtering to weave an information tapestry. *Commun. ACM*, 35(12), 61–70.
- Goldberg, K., Roeder, T., Gupta, D., & Perkins, C. (2001). Eigentaste: A constant time collaborative filtering algorithm. *Information Retrieval*, 4(2), 133–151.
- Grether, D. M., & Wilde, L. L. (1983). Consumer choice and information: New experimental evidence. *Information Economics and Policy*, 1, 115–144.

Bibliography

- Herlocker, J., Konstan, J., & Riedl, J. (2000). Explaining collaborative filtering recommendations. In *Proceeding on the ACM 2000 Conference on Computer Supported Cooperative Work, (CSCW 2000)*, pp. 241–250. Philadelphia, PA.
- Herlocker, J., Konstan, J. A., & Riedl, J. (2002). An empirical analysis of design choices in neighborhood-based collaborative filtering algorithms. *Information Retrieval*, 5(4), 287–310.
- Herlocker, J. L., Konstan, J. A., Borchers, A., & Riedl, J. (1999). An algorithmic framework for performing collaborative filtering. In *SIGIR '99: Proceedings of the 22nd annual international ACM SIGIR conference on Research and development in information retrieval*, pp. 230–237, New York, NY, USA. ACM Press.
- Herstein, I. N., & Milnor, John (1953). An axiomatic approach to measurable utility. *Econometrica*, 21(2), 291–297.
- Hill, W. C., Stead, L., Rosenstein, M., & Furnas, G. W. (1995). Recommending and evaluating choices in a virtual community of use.. In *CHI*, pp. 194–201.
- Horvitz, E. (1999). Principles of mixed-initiative user interfaces. In *Proceedings of the SIGCHI conference on Human factors in computing systems(CHI'99)*, pp. 159–166. ACM Press.
- Keeney, R. (1992). *Value-Focused Thinking: A Path to Creative Decision Making*. Harvard University Press, Cambridge.
- Keeney, R., & Raiffa, H. (1976). *Decisions with Multiple Objectives: Preferences and Value Tradeoffs*. John Wiley and Sons, New York.
- Kendall, M. (1948). *Rank Correlation Methods*. Charles Griffin Company Limited.
- Lawler, E., & Wood, D. (1966). Branch-and-bound methods: A survey. *Operational Research*, 14(4), 699–719.
- Lin, X., Soergel, D., & Marchionini, G. (1991). A self-organizing semantic map for information retrieval. In *SIGIR '91: Proceedings of the 14th annual international ACM SIGIR conference on Research and development in information retrieval*, pp. 262–269, New York, NY, USA. ACM.
- Linden, G., Hanks, S., & Lesh, N. (1997). Interactive assessment of user preference models: The automated travel assistant. In *Proceedings of the 6th International Conference on User Modeling (UM97)*.
- Linden, G. D., Jacobi, J. A., & Benson, E. A. (2001). Collaborative recommendations using item-to-item similarity mappings. *US Patent 6,266,649 (to Amazon.com)*.

-
- Luce, M., Payne, J., & Bettman, J. (1999). Emotional trade-off difficulty and choice. *Journal of Marketing Research*, 36, 143–159.
- Mackworth, A. (1988). Constraint satisfaction. *Encyclopedia of Artificial Intelligence*, 205–211.
- Marshall, J. (1950). Rational behavior, uncertain prospects, and measurable utility. *Econometrica*, 18, 111–141.
- McCarthy, K., Reilly, J., McGinty, L., & Smyth, B. (2004). On the dynamic generation of compound critiques in conversational recommender systems. In Bra, P. D., & Nejdil, W. (Eds.), *Proceedings of the Third International Conference on Adaptive Hypermedia and Adaptive Web-Based Systems (AH-2004)*, Vol. 3137 of *Lecture Notes in Computer Science*, pp. 176–184. Springer. Eindhoven, The Netherlands.
- McCarthy, K., Reilly, J., Smyth, B., & McGinty, L. (2005). Generating diverse compound critiques. *Artificial Intelligence Review*, 24(3-4), 339–357.
- McGinty, L., & Smyth, B. (2003). On the role of diversity in conversational recommender systems. In Ashley, K., & Bridge, D. (Eds.), *Proceedings of the 5th International Conference on Case-Based Reasoning (ICCBR 2003)*, Vol. 2689, pp. 276–290. Springer. Trondheim, Norway.
- McSherry, D. (2003). Similarity and compromise. In Ashley, K., & Bridge, D. (Eds.), *Proceedings of the 5th International Conference on Case-Based Reasoning (ICCBR 2003)*, Vol. 2689 of *Lecture Notes in Computer Science*, pp. 291–305. Springer. Trondheim, Norway, June 23-26, 2003.
- Montaner, M., López, B., & Rosa, J. L. D. L. (2003). A taxonomy of recommender agents on the internet. *Artificial Intelligence Review*, 19(4), 285–330.
- Newell, A., & Simon, H. (1972). *Human Problem Solving*. Prentice-Hall, Englewood Cliffs, NJ.
- Nguyen, Q. N., & Ricci, F. (2007). Replaying live-user interactions in the off-line evaluation of critique-based mobile recommendations. In *RecSys '07: Proceedings of the 2007 ACM conference on Recommender systems*, pp. 81–88, New York, NY, USA. ACM.
- O'Donovan, J., & Smyth, B. (2005). Trust in recommender systems. In *IUI '05: Proceedings of the 10th international conference on Intelligent user interfaces*, pp. 167–174, New York, NY, USA. ACM Press.
- O'Sullivan, B., Papadopoulos, A., Faltings, B., & Pu, P. (2007). Representative explanations for over-constrained problems. In *AAAI*, pp. 323–328. AAAI Press.

Bibliography

- Palmer, J. W. (1997). Retailing on the WWW: the use of electronic product catalogs. *International Journal of Electronic Markets*, 7(3), 6–9.
- Payne, J., Bettman, J., & Johnson, E. (1993). *The Adaptive Decision Maker*. Cambridge University Press.
- Pu, P., & Chen, L. (2005). Integrating tradeoff support in product search tools for e-commerce sites. In *Proceedings of the ACM Conference on Electronic Commerce (EC'05)*, pp. 269–278, Vancouver, Canada.
- Pu, P., & Faltings, B. (2000). Enriching buyers' experiences: the smartclient approach. In *SIGCHI conference on Human factors in computing systems*, pp. 289–296. ACM Press New York, NY, USA.
- Pu, P., & Faltings, B. (2002). Personalized navigation of heterogeneous product spaces using smartclient. In *International Conference on Intelligent User Interfaces*. ACM Press.
- Pu, P., & Faltings, B. (2004). Decision tradeoff using example-critiquing and constraint programming. *Constraints: An International Journal*, 9(4).
- Pu, P., Faltings, B., & Torrens, M. (2004). Effective interaction principles for online product search environments. In *Proceedings of the 3rd ACM/IEEE International Conference on Web Intelligence*. IEEE Press.
- Pu, P., & Janecek, P. (2003). Visual interfaces for opportunistic information seeking. In *the 10th International Conference on Human - Computer Interaction (HCII'03)*, pp. 1131–1135.
- Pu, P., & Kumar, P. (2004). Evaluating example-based search tools. In *Proceedings of the ACM Conference on Electronic Commerce (EC'04)*, pp. 208–217, New York, USA.
- Reilly, J., McCarthy, K., McGinty, L., & Smyth, B. (2004a). Dynamic critiquing. In Funk, P., & González-Calero, P. (Eds.), *Proceedings of the Seventh European Conference on Case-Based Reasoning (ECCBR-04)*, Vol. 3155 of *Lecture Notes in Computer Science*, pp. 763–777. Springer. Madrid, Spain.
- Reilly, J., McCarthy, K., McGinty, L., & Smyth, B. (2004b). Explaining compound critiquing. In Lees, B. (Ed.), *Proceedings of the 9th UK Workshop on Case-Based Reasoning*, pp. 12–20. Cambridge, UK.
- Reilly, J., McCarthy, K., McGinty, L., & Smyth, B. (2004c). Incremental critiquing. In Bramer, M., Coenen, F., & Allen, T. (Eds.), *Research and Development in Intelligent Systems XXI. Proceedings of AI-2004*, pp. 101–114. Springer. Cambridge, UK.

- Reilly, J., Zhang, J., McGinty, L., Pu, P., & Smyth, B. (2007). A comparison of two compound critiquing systems. In *Proceedings of the 12th International Conference on Intelligent User Interfaces (IUI '07)*. ACM Press.
- Reilly, J. (2007). *Critique-based Recommendation*. Ph.D. thesis, UCD, Ireland.
- Reilly, J., McCarthy, K., McGinty, L., & Smyth, B. (2005). Incremental critiquing. *Knowledge Based Systems*, 18(4-5), 143–151.
- Reingold, E., Nievergelt, J., & Deo, N. (1977). *Combinatorial Algorithm: Theory and Practice*. Prentice-Hall, Englewood Cliffs, NJ, USA.
- Resnick, P., Iacovou, N., Suchak, M., Bergstorm, P., & Riedl, J. (1994). Grouplens: An open architecture for collaborative filtering of netnews. In *Proceedings of ACM 1994 Conference on Computer Supported Cooperative Work*, pp. 175–186, Chapel Hill, North Carolina. ACM.
- Ricci, F., & Nguyen, Q. N. (2005). Critique-based mobile recommender systems. *ÖGAI Journal*, 24(4).
- Ricci, F., & Nguyen, Q. N. (2006). Mobyrek: A conversational recommender system for on-the-move travelers. In Fesenmaier, D. R., Werthner, H., & Wober, K. W. (Eds.), *Destination Recommendation Systems: Behavioural Foundations and Applications*, pp. 281–294. CABI Publishig.
- Ricci, F., & Nguyen, Q. N. (2007). Acquiring and revising preferences in a critique-based mobile recommender system. *IEEE Intelligent Systems*, 22(3), 22–29.
- Rossi, F., Petrie, C., & Dhar, V. (1990). On the equivalence of constraint satisfaction problems. In Aiello, L. C. (Ed.), *ECAI'90: Proceedings of the 9th European Conference on Artificial Intelligence*, pp. 550–556, Stockholm. Pitman.
- Russo, J., & Doshier, B. (1983). Strategies for multiattribute binary choice. *Journal of Experimental Psychology: Learning, Memory, and Cognition*, 9, 676–696.
- Ruttkay, Z. (1994). Fuzzy constraint satisfaction. In *Proceedings of the Third IEEE International Conference on Fuzzy Systems*, pp. 1263–1268.
- Saaty, T. (1980). *The Analytic Hierarchy Process*. New York: McGraw-Hill, Inc.
- Sahami, M., Dumais, S., Heckerman, D., & Horvitz, E. (1998). A bayesian approach to filtering junk E-mail. In *Learning for Text Categorization: Papers from the 1998 Workshop*, Madison, Wisconsin. AAAI Technical Report WS-98-05.
- Sarwar, B., Karypis, G., Konstan, J., & Reidl, J. (2001). Item-based collaborative filtering recommendation algorithms. In *WWW '01: Proceedings of the 10th international conference on World Wide Web*, pp. 285–295, New York, NY, USA. ACM Press.

Bibliography

- Sarwar, B., Karypis, G., Konstan, J., & Riedl, J. (2000). Analysis of recommendation algorithms for e-commerce. In *EC '00: Proceedings of the 2nd ACM conference on Electronic commerce*, pp. 158–167, New York, NY, USA. ACM.
- Schafer, J. B., Konstan, J., & Riedl, J. (2001). E-commerce recommendation applications. *Data Mining and Knowledge Discovery*, 5(1/2), 115–153.
- Schiex, T., Fargier, H., & Verfaillie, G. (1995). Valued constraint satisfaction problems: Hard and easy problems. In *Proceedings International Joint Conference on Artificial Intelligence(IJCAI'95):*, pp. 631–639.
- Shardanand, U., & Maes, P. (1995). Social information filtering: Algorithms for automating "word of mouth".. In *CHI*, pp. 210–217.
- Shearin, S., & Lieberman, H. (2001). Intelligent profiling by example. In *Proceedings of the Conference on Intelligent User Interfaces*, pp. 145–151. ACM Press New York, NY, USA.
- Shimazu, H., Shibata, A., & Nihei, K. (2001). Expertguide: A conversational case-based reasoning tool for developing mentors in knowledge spaces. *Applied Intelligence*, 14(1), 33–48.
- Shimazu, H. (2001). Expertclerk: Navigating shoppers buying process with the combination of asking and proposing. In *Proceedings of the 17 International Joint Conference on Artificial Intelligence (IJCAI'01)*, Vol. 2, pp. 1443–1448, Seattle, Washington, USA.
- Simon, H. A. (1955). A behavioral model of rational choice. *Quarterly Journal of Economics*, 69, 99–118.
- Smyth, B., & Cotter, P. (2000). A personalised tv listings service for the digital tv age. *Knowledge-Based Systems*, 13(2-3), 53–59.
- Smyth, B., & McClave, P. (2001). Similarity vs. diversity. In Aha, D., & Watson, I. (Eds.), *Proceedings of the Fourth International Conference on Case-Based Reasoning, (ICCBR '01)*, Vol. 2080 of *Lecture Notes In Computer Science*, pp. 347–361. Springer-Verlag.
- Smyth, B., McGinty, L., Reilly, J., & McCarthy, K. (2004). Compound critiques for conversational recommender systems. In *Proceedings of the IEEE/WIC/ACM International Conference on Web Intelligence (WI 2004)*, pp. 145–151. IEEE Computer Society.
- Smyth, B., & McGinty, L. (2003). An analysis of feedback strategies in conversational recommenders. In *AICS '03: Proceedings of the 14th Irish International Conference on Artificial Intelligence and Cognitive Science*.

- Spearman, C. (1906). 'Footrule' for measuring correlation. *British Journal of Psychology*.
- Stolze, M. (1999). Comparative study of analytical product selection support mechanisms. In *Proceedings of INTERACT 99*, pp. 45–53.
- Stolze, M. (2000). Soft navigation in electronic product catalogs. *International Journal on Digital Libraries*, 3(1), 60–66.
- Sullivan, D. O., Wilson, D., & Smyth, B. (2002). Improving case-based recommendation: A collaborative filtering approach. In *ECCBR '02: Proceedings of the 6th European Conference on Advances in Case-Based Reasoning*, pp. 278–291, London, UK. Springer-Verlag.
- Torrens, M. (2002). *Scalable Intelligent Electronic Catalogs*. Phd. thesis no. 2690, Swiss Federal Institute of Technology (EPFL), Lausanne (Switzerland).
- Torrens, M., Faltings, B., & Pu, P. (2002). Smartclients: Constraint satisfaction as a paradigm for scaleable intelligent information systems. *CONSTRAINTS: an International Journal*. Kluwer Academic Publishers, 7(1), 49–69.
- Tsang, E. (1993). *Foundations of Constraint Satisfaction*. Academic Press, London, UK.
- Tversky, A. (1972). Elimination by aspects: A theory of choice. *Psychological Review*, 79, 281–299.
- Tversky, A., & Simonson, I. (1993). Context-dependent preferences. *Management Science*, 39(10), 1179–1189.
- Viappiani, P. (2007). *Preference-based Search with Suggestions*. Ph.D. thesis, EPFL.
- Viappiani, P., Faltings, B., & Pu, P. (2006a). Evaluating preference-based search tools: a tale of two approaches. In *Proceedings of the Twenty-first National Conference on Artificial Intelligence (AAAI-06)*, pp. 205–211, Boston, MA, USA. AAAI press.
- Viappiani, P., Faltings, B., & Pu, P. (2006b). Preference-based search using example-critiquing with suggestions. *Journal of Artificial Intelligence Research (JAIR)*, 27, 465–503.
- Viappiani, P., Faltings, B., Schickel-Zuber, V., & Pu, P. (2005). Stimulating preference expression using suggestions. In *IJCAI-05 Multidisciplinary Workshop on Advances in Preference*, pp. 186–191, Edinburgh, UK.
- von Neumann, J., & Morgenstern, O. (1944). *Theory of Games and Economic Behavior*. Princeton University Press, Princeton.

Bibliography

- Yager, R., & Pasi, G. (2002). A consumer decision support system for internet shopping. In *Proceedings of the 2002 IEEE International Conference on FUZZ-IEEE'02*, pp. 1286–1291.
- Zhang, J., & Pu, P. (2007). Refining preference-based search results through bayesian filtering. In *IUI '07: Proceedings of the 12th international conference on Intelligent user interfaces*, pp. 294–297, New York, NY, USA. ACM.
- Zhang, J., Pu, P., & Faltings, B. (2006a). Agile decision agent for service-oriented e-commerce systems. In *CEC-EEE '06: Proceedings of the The 8th IEEE International Conference on E-Commerce Technology and The 3rd IEEE International Conference on Enterprise Computing, E-Commerce, and E-Services*, p. 24, Washington, DC, USA. IEEE Computer Society.
- Zhang, J., Pu, P., & Viappiani, P. (2006b). A study of user's online decision making behavior.. Tech. rep., Swiss Federal Institute of Technology (EPFL), Lausanne (Switzerland).

Publications during Ph.D. study:

- Jiyong Zhang, Nicolas Jones and Pearl Pu. A Visual Interface for Critiquing-based Recommender Systems. *Technical report, Swiss Federal Institute of Technology (EPFL), Lausanne, Switzerland, 2007.*
- Jiyong Zhang and Pearl Pu. A Recursive Prediction Algorithm for Collaborative Filtering Recommender Systems. In *Proceedings of ACM Recommender Systems*, pages 57–64, Minneapolis, Minnesota, USA, October 19-20, 2007.
- James Reilly, Jiyong Zhang, Lorraine McGinty, Pearl Pu and Barry Smyth. Evaluating Compound Critiquing Recommenders: A Real-User Study. In *Proceedings of ACM Conference on Electronic Commerce (EC'07)*, pages 114–123, San Diego, USA, June 11-15, 2007.
- Jiyong Zhang and Pearl Pu. Refining Preference-Based Search Results Through Bayesian Filtering. In *Proceedings of The International Conference on Intelligent User Interfaces(IUI2007)*, pages 294–297, Hawaii, USA, January 2007.
- James Reilly, Jiyong Zhang, Lorraine McGinty, Pearl Pu and Barry Smyth. A Comparison of Two Compound Critiquing Systems. In *Proceedings of The International Conference on Intelligent User Interfaces(IUI2007)*, pages 317–320, Hawaii, USA, January 2007.

APPENDIX A. PUBLICATION LIST

- Jiyong Zhang and Pearl Pu. Performance evaluation of consumer decision support systems. *International Journal of E-Business Research*, 2(3):28–45, 2006.
- Jiyong Zhang, Pearl Pu, and Boi Faltings. Agile decision agent for service-oriented e-commerce systems. In *Proceedings of The 8th IEEE International Conference on E-Commerce Technology and The 3rd IEEE International Conference on Enterprise Computing, E-Commerce, and E-Services (CEC/EEE'06)*, pages 182-189, Palo Alto, California, USA, IEEE Computer Society, June 26-29, 2006.
- Jiyong Zhang and Pearl Pu. A comparative study of compound critique generation in conversational recommender systems. In *Proceedings of the 4th International Conference on Adaptive Hypermedia and Adaptive Web-Based Systems (AH2006)*, volume 4018 of *Lecture Notes in Computer Science*, pages 234–243, Dublin, Ireland, June 2006. Springer.
- Jiyong Zhang, Pearl Pu, and Paolo Viappiani. A study of user's online decision making behavior. *Technical report, Swiss Federal Institute of Technology (EPFL)*, Lausanne, Switzerland, 2006.
- Boi Faltings, Pearl Pu, and Jiyong Zhang. Agile preference models based on soft constraints. In *Challenges to Decision Support in a Changing World, AAAI Spring Symposium*. AAAI Press, March, 2005.
- Jiyong Zhang and Pearl Pu. Effort and accuracy analysis of choice strategies for electronic product catalogs. In Hisham Haddad, Lorie M. Liebrock, Andrea Omicini, and Roger L. Wainwright, editors, *Proceedings of The 20th ACM Symposium on Applied Computing (SAC-2005)*, pages 808–814, Santa Fe, New Mexico, USA, March 2005. ACM.
- Jiyong Zhang and Pearl Pu. Survey of solving multi-attribute decision problems. *Technical Report No. IC/2004/54, Swiss Federal Institute of Technology (EPFL)*, Lausanne, Switzerland, 2004.

Previous publications:

- Jing Li, Fang Zheng, Jiyong Zhang, and Wenhui Wu. Context Dependent Initial/Final Acoustic Modeling for Chinese Continuous Speech Recognition. *Journal of Tsinghua Univ (Sci. & Tech.)*, 24(1): 61–64, January, 2004

-
- Jing Li, Fang Zheng, Jiyong Zhang, and Wenhui Wu. The Extension and Refinement of the Question Set for Decision Tree Based State Tying in Chinese Speech Recognition. In *Proceedings of the International Conference on Chinese Computing (ICCC2001)*, pages 106–110, Singapore, Nov. 27-29, 2001.
 - Jiyong Zhang, Fang Zheng, Jing Li, Chunhua Luo, and Guoliang Zhang. Improved Context-Dependent Acoustic Modeling for Continuous Chinese Speech Recognition. In *Proceedings of the 7th European Conference on Speech Communication and Technology (EuroSpeech'2001)*, 3:1617–1620, Aalborg, Denmark, Sept. 3-7, 2001.
 - Jiyong Zhang, Fang Zheng, Mingxing Xu, and Ditang Fang. Semi-continuous segmental probability modeling for continuous speech recognition. In *Proceedings of the International Conference on Spoken Language Processing (ICSLP'00)*, pages 278–281, Beijing, China. Oct. 16-20, 2000.
 - Jiyong Zhang, Fang Zheng, Mingxing Xu, and Shuqing Li. Intra-syllable dependent phonetic modeling for Chinese speech recognition. In *Proceedings of the International Symposium on Chinese Spoken Language Processing (ISCSLP'00)*, pages 73–76, Beijing, China, Oct. 13-15, 2000.
 - Jiyong Zhang, Fang Zheng, Shu Du, Zhanjiang Song, and Mingxing Xu. Merging-based Syllable Detection Automaton in Continuous Speech Recognition. *Journal of software*, 10(11): 1212–1215, 1999.

Jiyong ZHANG

Human Computer Interaction Group (HCIG),
School of Computer and Communications Sciences,
Ecole Polytechnique Fédérale de Lausanne (EPFL)

Tel: +41 21 693 1246

Fax: +41 21 693 6490

Email: jiyong.zhang@epfl.ch *Homepage:* <http://hci.epfl.ch/members/jiyong/>

Mailing Address: GRPU-IIF-IC, station 14, EPFL, CH-1015, Lausanne

Research Directions

Human computer interaction, intelligent user interface, automated decision making, e-commerce technologies, online product search, recommender systems.

Education

2004–present: Swiss Federal Institute of Technology in Lausanne (EPFL)
Ph.D. in Computer Science

1999–2001: Tsinghua University, China
Master in Computer Science

APPENDIX B. CURRICULUM VITAE

1994–1999: Tsinghua University, China
Bachelor in Computer Science

Work Experience

2004–2007: Computer Science Department, EPFL, Switzerland
Research Assistant, Webmaster at
Human Computer Interaction Group (HCIG)

2003–2004: Research Institute, Lenovo, China
Project Manager

2002–2003: Research Institute, Lenovo, China
R&D Engineer

1998–2001: Computer Science Department, Tsinghua University, China
Research Assistant at Center of Speech Technologies (CST)

Honors and Awards

Best student paper award, the 4th International Conference on Adaptive Hypermedia and Adaptive Web-Based Systems (AH2006), 2006.

Outstanding new employee award, Lenovo, 2003.

Motorola outstanding undergraduate student scholarship, Tsinghua University, 1998.

First prize winner, the National Mathematical Olympic Game, China, 1994.

Professional Activities

ACM student membership, 2005–2008

Reviewer, the 11th International Conference on User Modeling (UM2007)